MCIMX31 and MCIMX31L Applications Processors

Reference Manual

MCIMX31RM
Rev. 2.3
1/2007

ARM POWERED®

freescale™ semiconductor

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064, Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

***For Literature Requests Only*:**
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-521-6274 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Contents

About This Book

Book I: i.MX31 and i.MX31L Integration and Description

Chapter 1 Introduction to the i.MX31 and i.MX31L Multimedia Applications Processors

Chapter 2
System Memory Map, Interrupts, and SDMA Events

Chapter 3
Clocks, Power Management and Reset (AP Clock Controller Module)

Chapter 4
Signal Multiplexing

Chapter 5
General Purpose Input/Output (GPIO)

Chapter 6
Debugging the i.MX31 and i.MX31L

Chapter 7
i.MX31 and i.MX31L Boot

Book II: Applications Processors' Core and Peripherals

Book II, Part 1: ARM11 Core and Interrupts

Chapter 8 ARM11 Platform

Chapter 9
ARM1136JF-S Vectored Interrupt Controller (AVIC)

Book II, Part 2:
Security

Chapter 10
Security Controller (SCC)

Chapter 11
Security Random Number Generator Accelerator (RNGA)

Chapter 12
Run-Time Integrity Checker (RTIC)

Chapter 13
IC Identification (IIM)

Book II, Part 3:
Memory Systems

Chapter 14
L2 Cache Controller (L2CC)

Chapter 15
ARM11 Event Monitor (EVTMON)

Book II, Part 4:
External Interfaces

Chapter 16
External Memory Interface (EMI)

Chapter 17
Multi-Master Memory Interface (M3IF)

Chapter 18
Wireless External Interface Module (WEIM)

Chapter 19
Enhanced SDRAM Controller (ESDCTL)

Chapter 20
NAND Flash Controller (NANDFC)

Chapter 21
Personal Computer Memory Card International Association (PCMCIA) Controller

Book II, Part 5:
Connectivity Peripherals

Chapter 22
1-Wire Interface (1-Wire)

Chapter 23
Advanced Technology Attachment (ATA)

Chapter 24
Configurable Serial Peripheral Interface (CSPI)

Chapter 25
Fast Infrared Interface (FIR)

Chapter 26
Inter-Integrated Circuit (I^2C)

Chapter 27
Keypad Port (KPP)

Chapter 28
Memory Stick Host Controller (MSHC)

Chapter 29
Secured Digital Host Controller (SDHC)

Chapter 30
Subscriber Identification Module (SIM)

Chapter 31
Universal Asynchronous Receiver/Transmitter (UART)

Chapter 32
Universal Serial Bus, On-The-Go (USBOTG)

Book II, Part 6: Timer Peripherals

Chapter 33 Enhanced Periodic Interrupt Timer (EPIT 1, 2)

Chapter 34
General Purpose Timer (GPT)

Chapter 35
Pulse-Width Modulator (PWM)

Chapter 36
Real Time Clock (RTC)

Chapter 37
Watchdog Timer (WDOG)

Book II, Part 7:
System Control Peripherals

Chapter 38
AHB-Lite 2.v6 to IP Bus Interface (AIPS)

Chapter 39
Multi-Layer AHB Crossbar Switch (MAX)

Chapter 40
Smart Direct Memory Access (SDMA)

Chapter 41
Shared Peripheral Bus Arbiter (SPBA)

Book II, Part 8:
Multimedia Peripherals

Chapter 42
Digital Audio Multiplexer (AUDMUX)

Chapter 43
Moving Pictures Experts Group-4 (MPEG-4) Encoder

Chapter 44
Image Processing Unit (IPU)

Chapter 45
Synchronous Serial Interface (SSI)

Chapter 46 Graphics Accelerator (MBX R-S)

About This Book

The *MCIMX31 and MCIMX31L Multimedia Applications Processors Reference Manual* describes the features and operation of the i.MX31 and i.MX31L Multimedia Applications Processors. This manual provides details on how to initialize, configure, and program the ICs. It is assumed that the reader has a good working knowledge of the ARM1136JF-S architecture. For programming information about the ARM1136JF-S processor, see the documents listed in the Suggested Reading section of this preface.

Revision History

The following table summarizes revisions made to this document since the previous release (Rev. 2.2). Each change contains a cross-reference from this table to the location in the book where the actual change was made.

Revision History

Book	Location (Chapter)	Revision
"Book I: i.MX31 and i.MX31L Integration and Description"	Chapter 3 "Clocks, Power Management and Reset (AP Clock Controller Module)"	• An HSP_PODF warning note was added. • Section 3.4.4.3.1, "USB Clock Domain Switch Unit" was updated to reflect clock source switching. • Content referring to silicon integration of the module was removed. A typical case use for clock settings was added. • SIR maximum data rate was increased. • A section describing the FIRI clock path was added. • DMA values in the transmit and receive programming scenario sections were updated.
"Book I: i.MX31 and i.MX31L Integration and Description"	Chapter 4 "Signal Multiplexing"	**Note:** This chapter was named "*Signal Descriptions and Pin Assignments*" in previous versions of the i.MX31 reference manual. **Note:** The contents in this chapter (Rev. 2.3) replace the information in the i.MX31 RM addendum (Rev. 2) that had been posted on www.freescale.com/imx. • Usability for the IOMUX tables was improved. • Implementation modes/graphics and OBS content was deleted.
"Book II, Part 1: ARM11 Core and Interrupts"	Chapter 9 "ARM1136JF-S Vectored Interrupt Controller (AVIC)"	• The high-level interrupt generation examples were updated. • Figures that explain the interrupt generation process were added.
"Book II, Part 2: Security"	Chapter 13 "IC Identification (IIM)"	The SILICON_REV Settings table (Table 13-2) was updated.

Revision History (continued)

Book	Location (Chapter)	Revision
"Book II, Part 4: External Interfaces"	Chapter 17 "Multi-Master Memory Interface (M3IF)"	**Note:** The updated contents in this chapter (Rev. 2.3) replace the information in the i.MX31 RM addendum (Rev. 2) that had been posted on www.freescale.com/imx. • ESDCTL references that referred to an on-chip MDDRC module in iMX31 were replaced—the i.MX31 device interfaces to an *external* MDDRC module. • The description of the SWBA, SSS0, and SSS1 registers was expanded. • A third bullet to the Snooping Overview was added. • In Section 17.4.1, "M3IF in a System," "32-bit flash" was replaced with "16-bit flash" per M3IF. • Snooping Windows settings were added.
"Book II, Part 4: External Interfaces"	Chapter 19 "Enhanced SDRAM Controller (ESDCTL)"	**Note:** The updated contents in this chapter (Rev. 2.3) replace the information in the i.MX31 RM addendum (Rev. 2) that had been posted on www.freescale.com/imx. • The value of one cell in Table 19-32 was changed from X to H to ensure compliance with the naming conventions of the JEDEC specification. Nothing else has changed, nor has any setting in the silicon changed.
"Book II, Part 5: Connectivity Peripherals"	Chapter 24 "Configurable Serial Peripheral Interface (CSPI)"	• Three new tables (Table 24-13, Table 24-14, and Table 24-15) were added to provide additional information about configuring and multiplexing the CSPI pins.
"Book II, Part 8: Multimedia Peripherals"	Chapter 42 "Digital Audio Multiplexer (AUDMUX)"	• When referenced as a mode, "CE_Bus" was updated to "Bottom Connector." • Internal signal names were updated in figures.
"Book II, Part 8: Multimedia Peripherals"	Chapter 45 "Synchronous Serial Interface (SSI)"	• Module operation, signal, and register definitions were expanded throughout the chapter. Technical data remains unchanged.

Audience

The *MCIMX31 and MCIMX31L Multimedia Applications Processors Reference Manual* provides to the design engineer the necessary data to successfully integrate the applications processor ICs into a wide variety of applications.

The intended audience for this document includes system architects, system modeling teams, IC designers, software architects/designers, and the platform integration and testing teams. The level of detail in this document is intended to provide the reader with sufficient information to validate the capabilities of the applications processor ICs in the targeted applications. This document is supplemented with users' manuals for hardware design and software development.

Organization

This reference manual is organized into chapters that describe the operation and programming of the i.MX31 and i.MX31L processors. It includes brief summaries of the major components of the i.MX31 and i.MX31L, as well as a complete listing of the memory maps for the applications processor ICs and shared memories. This book also describes the generation and distribution of clocks.

A brief summary of the ARM11 Platform and its operational features is covered in this document, as well as an extensive overview of the components including the features, addressing, and operational modes of the ARM11 Platform. This manual also provides detailed information about the ARM11 core and the relationship and prioritization of the interrupts, and describes how the Security Controller provides a way to securely store sensitive information in on-chip RAM, as well as in off-chip non-volatile memory to prevent un-authorized access. It also describes how data is stored in encrypted form, using an encryption key that is unique to each device and accessible only to the Secure RAM module.

This reference manual contains chapters that describe the operations and configuration of all of the peripherals, including the modules that provide security, memory, and connectivity. The chapters in this book are as follows.

Book I, i.MX31 and i.MX31L Integration and Description

Device Introduction and Memory Map

Chapter 1 “Introduction to the i.MX31 and i.MX31L Multimedia Applications Processors,” on page 1-1

Chapter 2 “System Memory Map, Interrupts, and SDMA Events,” on page 2-1

Clocks, Power Management and Reset

Chapter 3 “Clocks, Power Management and Reset (AP Clock Controller Module),” on page 3-1

Pins

Chapter 4 “Signal Multiplexing,” on page 4-1

Chapter 5 “General Purpose Input/Output (GPIO),” on page 5-1

Debug

Chapter 6 “Debugging the i.MX31 and i.MX31L,” on page 6-1

Boot

Chapter 7 “i.MX31 and i.MX31L Boot,” on page 7-1

Book II, Applications Processors’ Core and Peripherals

ARM11 Core and Interrupts

Chapter 8 “ARM11 Platform,” on page 8-1

Security

Memory Systems

External Interfaces

Connectivity Peripherals

Timer Peripherals

System Control Peripherals

Multimedia Peripherals

Suggested Reading

The following documents are suggested for a complete description of the i.MX31 and i.MX31L, and are necessary to design with the device. Especially for those not familiar with the ARM1136JF-S processor, these manuals and documents will be helpful when used in conjunction with this reference manual.

These manuals can be found at the ARM Ltd. World Wide Web site at http://www.arm.com, and at the Freescale Semiconductor, Inc. World Wide Web site at http://www.freescale.com. They can be downloaded directly from the Web sites, or printed versions can be ordered.

1. *AMBA AHB specifications*
2. *ARMv6 AMBA Extensions*
3. *ARM1136JF-S Platform specifications*
4. *Hip7a KiloBit Single Port HP SRAM Compiler*
5. *Hip7A SAMI ROM Compiler*
6. *Hip7A KiloBit HD VIA ROM Compiler*
7. *ARM1136JF-S Platform Test Guide*
8. *ARM Architecture Reference Manual*

9. *ARM11DT1 Data Sheet Manual*
10. *ARM Technical Reference Manual*
11. IP Interface, Semiconductor Reuse Standard, Freescale Semiconductor
12. *ARM1136 Technical Reference Manual*

Conventions

This reference manual uses the following conventions:

- $\overline{\text{OVERBAR}}$ is used to indicate a signal that is active when pulled low: for example, $\overline{\text{RESET}}$.
- *Logic level one* is a voltage that corresponds to Boolean true (1) state.
- *Logic level zero* is a voltage that corresponds to Boolean false (0) state.
- To *set* a bit or bits means to establish logic level one.
- To *clear* a bit or bits means to establish logic level zero.
- A *signal* is an electronic construct whose state conveys or changes in state convey information.
- A *pin* is an external physical connection. The same pin can be used to connect a number of signals.
- *Asserted* means that a discrete signal is in active logic state.
  - *Active low* signals change from logic level one to logic level zero.
  - *Active high* signals change from logic level zero to logic level one.
- *Negated* means that an asserted discrete signal changes logic state.
  - *Active low* signals change from logic level zero to logic level one.
  - *Active high* signals change from logic level one to logic level zero.
- LSB means *least significant bit* or *bits*, and MSB means *most significant bit* or *bits*. References to low and high bytes or words are spelled out.
- Numbers preceded by a percent sign (%) are binary. Numbers preceded by a *0x* are hexadecimal.
- Courier monospaced type indicate commands, command parameters, code examples, expressions, data types, and directives.
- Italic type indicates replaceable command parameters.
- All source code examples are in C.

Definitions, Acronyms, and Abbreviations

The following table defines the acronyms and abbreviations used in this document.

Term	Definition
1-Wire	An interface providing a single serial communication line to a 1 K-bit EPROM.
ABIST	Array built-in self test —A hardware unit that autonomously performs memory array testing when in test mode.
ADC	Analog-to-digital converter
ADC	Asynchronous display controller

Term	Definition
address translation	Address conversion from virtual domain to physical domain
AFE	Analog front end
AHB	AMBA high-performance bus
AIPS	AHB-Lite 2.v6 to IP bus interface
AMBA	AMBA is an open standard, on-chip bus specification that details a strategy for the interconnection and management of functional blocks that makes up a System-on-Chip (SoC).
API	Application programming interface
ARM	Advanced RISC machines processor architecture
associativity	Refers to the number of sets in the cache.
atomic operation	An ARM11 operation in which the processor reads data from a memory location, modifies it and immediately writes it back. A successful operation means that there was no other write operation to the same location by another bus master between the read and the write.
AUDMUX	Digital audio MUX—provides a programmable interconnection for voice, audio, and synchronous data routing between host serial interfaces and peripheral serial interfaces.
BCD	Binary coded decimal
BDMA	Baseband direct memory access—the BDMA controls the DMA accesses of StarCore peripherals to the DMA bus shared by the DMA channels, and the DMA channels' accesses to M1 memory and MAX slaves (M2 memory, external memory and AIPS peripherals) through the AHB bus.
be/le	Big Endian/Little Endian
beat	A bus transaction that is part of a burst.
BER	Bit error ratio
BIST	Built in self-test
burst	A sequence of transactions (beats) on the bus that the bus controller cannot interrupt.
bus	A path between several devices via data lines.
bus load	The percentage of time a bus is busy.
cache coherency	A state where data images in a cache match the data in other memories or caches.
cache hit	A request for data that is in the cache, which the cache can fulfill with minimum latency.
cache miss	A request for data from the cache that cannot be fulfilled without some latency because the data is not present in the cache.
cache synchronization	A procedure to reach cache coherency by writing back to the L2/M2 data that was written only to the cache.
CCM	Clock control module
CF	CompactFlash—a small form factor card standard that encompasses CF data storage cards, magnetic disk cards and I/O cards.
channel	Refers to a defined path for flow of data between a source and sink. A channel is uni-directional. Example source and sink nodes are AP Processor or Peripherals. Up to 32 channels can be defined and operating at any one time.

Term	Definition
CMOS	Complimentary metal-oxide semiconductor
CODEC	Coder/decoder or compression/decompression algorithm—Used to encode and decode (or compress and decompress) various types of data.
contention	A situation in which multiple simultaneous accesses cannot be served by memory in the same cycle.
context switch	Saving micro RISC (for example, SDMA core) registers by copying their contents into internal SRAM space followed by a load of the registers from a different channel. Essentially, a context switch means that the current state of the SDMA micro-RISC engine is saved, and the new state of the core is determined by loading the core registers with the state of a different channel.
CPU	Central processing unit—generic term used to describe a processing core.
CRC	Cyclic redundancy check—Bit error protection method for data communication.
CSF	Command sequence file—a script loaded together with an application to direct HAB verification
CSI	Camera sensor interface
CSIC	Complex instruction set computer
CTL	Control
DAC	Digital-to-analog converter
DDR RAM	Double data rate RAM
degradation	Refers to the increase of execution time of an a program due to the use of cache. The reference execution time is that obtained by running the program with infinite M1 memory (without cache). The degradation is measured in percentage.
DFD	Digital frequency doubler—produces an output clock that is twice the frequency of the input clock.
dirty	An attribute of a VBR indicating that its content was modified without being updated in M2 memory.
DMA	Direct memory access—an independent block that can initiate memory-to-memory data transfers.
DPLL	Digital phase locked loop—provides clock generation in digital and mixed analog/digital chips designed for wireless communication and other applications. The DPLL produces a high-frequency chip clock with a low frequency and phase jitter.
DRAM	Dynamic random access memory
DRM	Data read messaging
DSM	Deep sleep module—handles the transition to and from Deep Sleep power saving mode.
DSP	Digital signal processor—a special-purpose CPU used for digital signal processing. It provides ultra-fast instruction sequences, such as shift and add, and multiply and add, which are commonly used in math-intensive signal processing applications.
DWM	Data write messaging
EDIO	External interrupt module—recognizes external (to the IC) asynchronous signals as interrupt sources.
EDO RAM	Extended data out DRAM
e-Fuse	Electrically-programmable poly fuses—a fusible element that may be blown under software or JTAG control during IC final test, at the customer factory or in the field.
EMI	External memory interface—controls all IC external memory accesses (read/write/erase/program) from all the masters in the system.

Term	Definition
Endian	Refers to byte ordering of data in memory. Little Endian means that the least significant byte of the data is stored in a lower address than the most significant byte. In Big Endian, the order of the bytes is reversed.
EOnCE	Enhanced on-chip emulation—a unit that supports core-level debugging and profiling.
EPIT	enhanced periodic interrupt timer—a 32-bit set and forget timer capable of providing precise interrupts at regular intervals with minimal processor intervention.
FCS	Frame checker sequence
FEC	Forward error correction
fetch	The process of retrieving information (program and data) initiated by a miss in the cache. The process is comprised of a mandatory fetch and a speculated fetch.
FIFO	First in first out
FIPS	Federal information processing standards—United States Government technical standards published by the National Institute of Standards and Technology (NIST). NIST develops FIPS when there are compelling Federal government requirements such as for security and interoperability but no acceptable industry standards or solutions
FIPS-140	Security requirements for cryptographic modules—Federal Information Processing Standard 140-2(FIPS 140-2) is a standard that describes US Federal government requirements that IT products should meet for Sensitive, but Unclassified (SBU) use.
FIR	Fast infrared—high speed IrDA protocol that supports speeds up to 4M baud.
FIR	See FIR.
Flash	A non-volatile storage device similar to EEPROM, but where erasing can only be done in blocks or the entire chip.
Flash path	Path within ROM bootstrap pointing to an executable Flash application.
Flush	A procedure to reach cache coherency. Refers to removing a data line from cache. This process includes cleaning the line, invalidating its VBR and resetting the tag valid indicator. The flush is triggered by a software command.
FPM	Full port mode
GEM	GPRS encryption module—The module is a hardware accelerator that assists in the processing of general packet radio service (GPRS) data packets.
GPCR	Global peripheral control registry
GPIO	General purpose input/output
GPS	Global positioning system—GPS module within BP domain that can process satellite signals and compute position coordinates.
H.264	A high compression digital video codec standard
HAB	High assurance boot—Used to prevent hackers from bypassing device security during the boot process. The high assurance boot permits flashed code to be verified for integrity before being loaded by a mobile device.
HAC	Hash accelerator controller—a security hardware module that is also called HACC
hash	Hash values are produced to access secure data. A hash value (or simply hash), also called a message digest, is a number generated from a string of text. The hash is substantially smaller than the text itself, and is generated by a formula in such a way that it is extremely unlikely that some other text will produce the same hash value.

Term	Definition
hit	See cache hit.
hw	Hardware
i/o	Input/output
ICACHE	Instruction cache—A cache memory dedicated to program only; a component in the SC140e Platform 2002
ICE	In-circuit emulation
IEEE 1149.1	The IEEE standard defining JTAG, see JTAG
IIM	IC identification module
Indigo	An IP interrupt request standard based on the SRS standard.
IP	intellectual property
IPU	Image Processing Unit —supports video and graphics processing functions and provides an interface to video/still image sensors and displays.
iRAM	Processor-internal RAM
IrDa	Infrared Data Association—A nonprofit organization whose goal is to develop globally adopted specifications for infrared wireless communication.
iROM	Processor-internal ROM
ISR	Interrupt service routine
JTAG	JTAG (IEEE Standard 1149.1) A standard specifying how to control and monitor the pins of compliant devices on a printed circuit board.
Kill	To stop a memory access
KPP	Keypad port—a 16-bit peripheral that can be used as a keypad matrix interface or as general purpose input/output (I/O).
l1 memory	Level 1 cache memory—It is closest to the core and serves the core directly.
l2 memory	Level 2 cache memory—a cache memory that serves the Level 1 cache
L-Fuse	Laser fuse—a fusible connection that can only be during chip manufacturing (at the wafer level).
line	Refers to a unit of information in the cache that is associated with a tag.
lock	The state of a line or group of lines in the cache that prevents them from being replaced in case of a miss.
LRU	Least recently used—a policy for line replacement in the cache
LSP	Least significant part.
LV	Low voltage
LWB	Late-write buffer
m1 memory	Level 1 RAM connected directly to the core.
m2 memory	Level 2 RAM connected to the core via L1 cache memory.
MAP	Mold array process
MAPBGA	Mold array process ball grid array
MCTL	Memory controller

Term	Definition
MDO	Message data out
MEMCTC	Embedded Memory Core Technology Center (Freescale, Austin, TX)
memory mapping	Translation of address referenced by the core to a physical memory address
MIPS	Million instructions-per-second
MISO	Master in slave out—supplies the output data from a slave to the input of the master. There can be no more than one slave that is transmitting data during any particular transfer. *See SPI*.
miss	See cache miss.
miss rate	The ratio between the number of memory misses and the total memory accesses.
MMC	MultiMedia card—a memory storage card. See SD.
MMU	Memory management unit—a component responsible for memory protection and address translation.
MOSI	Master out slave in—supplies the output data from the master to the inputs of the slaves. *See SPI*
MPEG	Moving Picture Experts Group—an ISO committee that generates standards for digital video compression and audio. It is also the name of the algorithms used to compress moving pictures and video.
MPEG standards	There are several standards of compression for moving pictures and video. • MPEG-1 is optimized for CD-ROM and is the basis for MP3. • MPEG-2 is defined for broadcast quality video in applications such as digital television set-top boxes and DVD. • MPEG-3 was merged into MPEG-2. • MPEG-4 is a standard for low-bandwidth video telephony and multimedia on the World-Wide Web.
MQSPI	Multiple queue serial peripheral interface—used to perform serial programming operations necessary to configure radio subsystems and selected peripherals.
MSHC	Memory stick host controller
MSP	Most significant part
MUX	Multiplexing
NAND Flash	Flash ROM technology—NAND Flash architecture is one of two flash technologies (the other being NOR) used in memory cards such as the CompactFlash cards. NAND is best suited to flash devices requiring high capacity data storage. NAND flash devices offer storage space up to 512-Mbyte and offers faster erase, write, and read capabilities over NOR architecture.
NANDFC	NAND flash controller—Provides a glueless interface to a Flash memory device.
NC	Not connected—designates unassigned connections in connection diagrams.
Nexus	A standard providing real-time trace capabilities in compliance with the IEEE-ISTO 5001-2003 standard. The Nexus standard defines an Auxiliary port used in conjunction with the IEEE 1149.1 JTAG port to provide development support capabilities without requiring address and data pins for internal visibility.
NOR Flash	See NAND Flash.
O-Wire	See 1-Wire.
OnCE™	On-Chip Emulation—circuitry that provides convenient and inexpensive debug facilities normally available only using expensive external hardware.
One-wire	See 1-Wire.

Term	Definition
OS	operating system.
OWire	See 1-Wire.
P2002	Platform 2002–The platform of a StarCore and other modules that provide the BP portion of the IC, see also DSP, SC140e and StarCore.
Parity	Check bit for detection of errors during data transfers
PC card	See PCMCIA.
PCMCIA	Personal Computer Memory Card International Association—a multi-company organization that has developed a standard for small, credit card-sized devices, called PC Cards. There are three types of PCMCIA cards that have the same rectangular size (85.6 by 54 millimeters), but different widths.
physical address	The address by which the memory in the system is physically accessed.
PLL	Phase locked loop—an electronic circuit controlling an oscillator so that it maintains a constant phase angle (a lock) on the frequency of an input, or reference, signal.
POR	Power on reset
pre-fetch	Refers to the speculated fetch process, up to the end of the cache line (*see fetch)*.
pre-fetch hit	A match between the required address and a speculated access before it is written to the cache.
pre-fetch line	A pre-fetch mode supported by the caches, where pre-fetching is done up to the end of the cache line.
privileged mode	Core mode that enables execution of certain instructions, usually used by the operating system. user application is executed in non-privileged (User) mode.
PWM	Pulse-width modulator—Using a PWM waveforms are synthesized by a sequence of pulses with a progressive width adjustment to produce a range of frequencies.
R-AHB bus	Reduced advanced high-performance bus (AHB), related to ARM bus architecture
RAM	Random access memory
RAMC	Random Access Memory (RAM) Controller
RAM path	Path within ROM bootstrap leading to the downloading and the execution of a RAM application
RGB	The RGB color model is based on the additive model in which red, green, and blue light are combined in various ways to create other colors. The abbreviation RGB come from the three primary colors in additive light models.
RGBA	RGBA color space stands for red green blue alpha. The alpha channel is the transparency channel, and is unique to this color space. RGBA, like RGB, is an additive color space, so the more of a color you place, the lighter the picture gets. PNG is the best known image format that uses the RGBA color space.
RNGA	Random number generator accelerator—a security hardware module that produces 32-bit pseudo random numbers as part of the security module.
ROM	Read-only memory
ROMC	Read-only memory (ROM) controller
ROM bootstrap	Internal boot code encompassing main boot flow as well as exception vectors.
RPM	Reduced port mode
RTC	Real-time clock

Term	Definition
RTIC	Real-time integrity checker—a security hardware module
RTOS	Real-time operating system—any operating system where interrupts are guaranteed to be handled within a certain specified maximum time, thereby making it suitable for control of hardware in embedded systems and other time-critical applications. RTOS is not a specific product but a class of operating systems.
sample	The result of measuring the amplitude of an analog signal at a specified time. In digital signal processing a sample is a signed or unsigned number and the number of samples per second is called the sample rate.
SCC	Security controller—a security hardware module
SCLK	Serial clock—a control line driven by the master, regulating the flow of data bits. *See SPI.*
script	An assembly language program executed by the SDMA core. Scripts reside in the ROM or may be pre-loaded into the SDMA SRAM.
SD	Secure digital card—based on the MMC specification, the SD is a highly secure small flash memory card that measures 32 x 24 x 2.1 millimeters that provide high-capacity memory storage in capacities between 16 Megabytes to beyond 1 Gigabyte. SD cards provide encryption capabilities for protected content to ensure secure distribution of copyrighted material.
SDC	Synchronous display controller
SDMA	Smart direct memory access
SDRAM	Synchronous dynamic random access memory
SHA-1	The Secure Hash Algorithm (SHA-1), National Institute of Standards and Technology, NIST FIPS PUB 180-1, “Secure Hash Standard,” U.S. Department of Commerce, April 1995
shared peripherals	Shared peripherals domain—one of three major domains of the IC
SHW	Security hardware
SHWI	Security hardware interface
SIM	Subscriber identification module—This module is designed to facilitate communication between the IC and SIM cards or Eurochip pre-paid phone cards.
SMIF	StarCore memory interface
SoC	System on a chip
SPBA	Shared peripheral bus arbiter—a three-to-one IP-Bus arbiter, with a resource-locking mechanism.
SPI	Serial peripheral interface—a full-duplex synchronous serial interface for connecting low-/medium-bandwidth external devices using four wires. SPI devices communicate using a master/slave relationship over two data lines and two control lines: *Also see SS, SCLK, MISO, and MOSI.*
SRAM	Static random access memory
SRS	Semiconductor Reuse Standards
SS	Slave select—a control line that enables slaves to be turned on and off with hardware control. *See SPI.*
SSI	Synchronous-serial interface—standardized interface for serial data transfer
Supervisor Level	A core privilege level that enables the execution of all instructions and access to all registers (as opposed to user level).
Sync Flash	An obsolete Flash ROM technology that was discontinued in 2003.

Term	Definition
Task	A program unit with a set of attributes that is controlled by the operating system. Task attributes can include such elements as the execution priority, data area protection, program protection, and so on.
TBD	To be determined
TQFP	Thin quad flat pack
tristate	Input control that switches outputs either to active or to high impedance.
true IDE	A CompactFlash (CF) storage card mode of operation. When the CF card runs in True IDE mode it is electrically compatible with an IDE disk drive.
TWB	Trace write buffer—a component of the SC140e that temporarily holds trace data before the debugging and profiling unit (DPU) saves it into main memory.
TYPE	Identifier that distinguishes a production, engineering, or HAB-disabled device.
UART	Universal asynchronous receiver/transmitter—this module provides asynchronous serial communication to external devices.
UID	Unique ID–a field in the processor and CSF identifying a device or group of devices
USB	Universal serial bus—an external bus standard that supports high speed data transfers. The USB 1.1 specification supports data transfer rates of up to 12Mb/s and USB 2.0 has a maximum transfer rate of 480 Mbps. A single USB port can be used to connect up to 127 peripheral devices, such as mice, modems, and keyboards. USB also supports Plug-and-Play installation and hot plugging.
USBOTG	USB on the go—an extension of the USB 2.0 specification for connecting peripheral devices to each other. USBOTG devices, also known as dual-role peripherals, can act as limited hosts or peripherals themselves depending on how the cables are connected to the devices, and they also can connect to a host PC.
user level	A core privilege level with restricted access to some core registers and no access to some instructions. It is also called user mode.
valid bit	An attribute of a VBR, indicating its contents is valid, and can be used.
VBR	Valid bit resolution—a unit of data that is treated as a whole regarding its validity.
VBR miss	A miss reported when the Tag of the data is in the cache, but the VBR is not.
VC	Virtual component.
VDD	Virtual data descriptor.
Virtual DMA	DMA function executed as observed from the outside. However, the function is actually executed by a script in the micro-RISC core.
WBB	Write back buffer
word	A group of bits comprising 32 bits
write miss	A miss caused when trying to write data that is not in the cache (either the entire line is missing, or the specific VBR is not valid).
write-allocate	A writing scheme in which a write miss first reads the data from the L2/M2 before executing a write in the cache.
write-back	A write scheme in which data is written only to the cache. The main memory is updated when the data in the cache is replaced.
write-through	A write scheme in which the data is written simultaneously to the cache and to memory.

Term	Definition
WTB	Write-through buffer—buffer that temporarily saves the data written into main memory in a write-through mode.
WTLS	Wireless transport layer security—a part of the wireless application protocol
XTAL	Crystal
YUV	YUV is the color space used in the PAL system of television broadcasting that is the standard in most of Europe and other areas of the world. Y stands for the luminance component (the brightness) and U and V are the chrominance (color) components.
YUV 4:2:2	YUV 4:2:2 is a specific encoding for digital representation of the YUV color space. In YUV 4:2:2, the basic unit is composed of two pixels, and occupies four bytes of space. Each pixel has an individual 8 bit Y channel. Then, the first pixel specifies an 8 bit U channel, and the second pixel an 8 bit V channel. Both pixels use the same U and V channels.
YUV 4:4:4	A specific encoding for digital representation of the YUV color space. Each of the Y,U and V channels are expressed with 8 bits, and have therefore 256 (28) possible levels. It is in this respect similar to RGB 24-bit, and uses the same amount of space (3 bytes per pixel). YUV 4:4:4 is the highest-quality digital YUV standard available.

USBOTG Glossary of Terms and Abbreviations

This section lists and defines terms and abbreviations used in Chapter 32 “Universal Serial Bus, On-The-Go (USBOTG).”

Term	Definition
ACK	Handshake packet indicating a positive acknowledgment.
Active Device	A device that is powered and is not in the Suspend state.
Asynchronous Data	Data transferred at irregular intervals with relaxed latency requirements.
Asynchronous RA	The incoming data rate, $F^{S}i$, and the outgoing data rate, $F^{S}o$, of the RA process are independent (for example, there is no shared master clock). See also rate adaptation.
Asynchronous SRC	The incoming sample rate, $F^{S}i$, and outgoing sample rate, $F^{S}o$, of the SRC process are independent (for example,, there is no shared master clock). See also sample rate conversion.
Audio Device	A device that sources or sinks sampled analog data.
AWG#	The measurement of a wire’s cross-section, as defined by the American Wire Gauge standard.
b/s	Transmission rate expressed in bits per second.
B/s	Transmission rate expressed in bytes per second.
Babble	Unexpected bus activity that persists beyond a specified point in a (micro) frame.
Bandwidth	The amount of data transmitted per unit of time, typically bits per second (b/s) or bytes per second (B/s).

Term	Definition
Big Endian	A method of storing data that places the most significant byte of multiple-byte values at a lower storage address. For example, a 16-bit integer stored in Big Endian format places the least significant byte at the higher address and the most significant byte at the lower address. See also Little Endian.
Bit	A unit of information used by digital computers. Represents the smallest piece of addressable memory within a computer. A bit expresses the choice between two possibilities and is typically represented by a logical one (1) or zero (0).
Bit Stuffing	Insertion of a “0” bit into a data stream to cause an electrical transition on the data wires, enabling a PLL to remain locked.
Buffer	Storage used to compensate for a difference in data rates or time of occurrence of events, when transmitting data from one device to another.
Bulk Transfer	One of the four USB transfer types. Bulk transfers are non-periodic, large burst communication typically used for a transfer that can use any available bandwidth and can be delayed until bandwidth is available. See also transfer type.
Bus Enumeration	Detecting and identifying USB devices.
Byte	A data element that is eight bits in size.
Capabilities	Those attributes of a USB device that are administrated by the host.
Characteristics	Those qualities of a USB device that are unchangeable; for example, the device class is a device characteristic.
Client	Software resident on the host that interacts with the USB System Software to arrange data transfer between a function and the host. The client is often the data provider and consumer for transferred data.
Configuring Software	Software resident on the host software that is responsible for configuring a USB device. This may be a system configuration or software specific to the device.
Control Endpoint	A pair of device endpoints with the same endpoint number that are used by a control pipe. Control endpoints transfer data in both directions and, therefore, use both endpoint directions of a device address and endpoint number combination. Thus, each control endpoint consumes two endpoint addresses.
Control Pipe	Same as a message pipe.
Control Transfer	One of the four USB transfer types. Control transfers support configuration/command/status type communications between client and function. See also transfer type.
CRC	See Cyclic Redundancy Check.
CTI	Computer Telephony Integration.
Cyclic Redundancy Check (CRC)	A check performed on data to see if an error has occurred in transmitting, reading, or writing the data. The result of a CRC is typically stored or transmitted with the checked data. The stored or transmitted result is compared to a CRC calculated for the data to determine if an error has occurred.
Default Address	An address defined by the USB Specification and used by a USB device when it is first powered or reset. The default address is 00H.
Default Pipe	The message pipe created by the USB System Software to pass control and status information between the host and a USB device’s endpoint zero.

Term	Definition
Device	A logical or physical entity that performs a function. The actual entity described depends on the context of the reference. At the lowest level, device may refer to a single hardware component, as in a memory device. At a higher level, it may refer to a collection of hardware components that perform a particular function, such as a USB interface device. At an even higher level, device may refer to the function performed by an entity attached to the USB; for example, a data/FAX modem device. Devices may be physical, electrical, addressable, and logical. When used as a non-specific reference, a USB device is either a hub or a function.
Device Address	A seven-bit value representing the address of a device on the USB. The device address is the default address (00H) when the USB device is first powered or the device is reset. Devices are assigned a unique device address by the USB System Software.
Device Endpoint	A uniquely addressable portion of a USB device that is the source or sink of information in a communication flow between the host and device. See also endpoint address.
Device Resources	Resources provided by USB devices, such as buffer space and endpoints. See also Host Resources and Universal Serial Bus Resources.
Device Software	Software that is responsible for using a USB device. This software may or may not also be responsible for configuring the device for use.
Downstream	The direction of data flow from the host or away from the host. A downstream port is the port on a hub electrically farthest from the host that generates downstream data traffic from the hub. Downstream ports receive upstream data traffic.
Driver	When referring to hardware, an I/O pad that drives an external load. When referring to software, a program responsible for interfacing to a hardware device, that is, a device driver.
DWord	Double word. A data element that is two words (for example, four bytes or 32 bits) in size.
Dynamic Insertion and Removal	The ability to attach and remove devices while the host is in operation.
E^2PROM	See Electrically Erasable Programmable Read Only Memory.
EEPROM	See Electrically Erasable Programmable Read Only Memory.
Electrically Erasable Programmable Read Only Memory (EEPROM)	Non-volatile re-writeable memory storage technology
End User	The user of a host.
Endpoint	See device endpoint.
Endpoint Address	The combination of an endpoint number and an endpoint direction on a USB device. Each endpoint address supports data transfer in one direction.
Endpoint Direction	The direction of data transfer on the USB. The direction can be either IN or OUT. IN refers to transfers to the host; OUT refers to transfers from the host.
Endpoint Number	A four-bit value between 0H and FH, inclusive, associated with an endpoint on a USB device.
Envelope detector	An electronic circuit inside a USB device that monitors the USB data lines and detects certain voltage related signal characteristics.

Term	Definition
EOF	End-of- (micro) Frame.
EOP	End-of-Packet.
External Port	See port.
Eye pattern	A representation of USB signaling that provides minimum and maximum voltage levels as well as signal jitter.
False EOP	A spurious, usually noise-induced event that is interpreted by a packet receiver as an EOP.
Flush (Endpoint)	A term used in this device controller implementation to describe the action of clearing an endpoint ready status.
Frame	A 1-millisecond time base established on full-/low-speed buses.
Frame Pattern	A sequence of frames that exhibit a repeating pattern in the number of samples transmitted per frame. For a 44.1 kHz audio transfer, the frame pattern could be nine frames containing 44 samples followed by one-frame containing 45 samples.
Fs	See sample rate.
Full-duplex	Computer data transmission occurring in both directions simultaneously.
Full-speed	USB operation at 12 Mb/s. See also low-speed and high-speed
Function	A USB device that provides a capability to the host, such as an ISDN connection, a digital microphone, or speakers.
Handshake Packet	A packet that acknowledges or rejects a specific condition. For example, see ACK and NAK.
High-bandwidth endpoint	A high-speed device endpoint that transfers more than 1024 bytes and less than3073 bytes per microframe.
High-speed	USB operation at 480 Mb/s. See also low-speed and full-speed
Host	The host computer system where the USB Host Controller is installed. This includes the host hardware platform (CPU, bus, among others) and the operating system in use.
Host Controller	The host's USB interface.
Host Controller Driver (HCD)	The USB software layer that abstracts the Host Controller hardware. The Host Controller Driver provides an SPI for interaction with a Host Controller. The Host Controller Driver hides the specifics of the Host Controller hardware implementation.
Host Resources	Resources provided by the host, such as buffer space and interrupts. See also Device Resources and Universal Serial Bus Resources.
Hub	A USB device that provides additional connections to the USB.
Hub Tier	One plus the number of USB links in a communication path between the host and a function.
I/O Request Packet	An identifiable request by a software client to move data between itself (on the host) and an endpoint of a device in an appropriate direction.
Interrupt Request (IRQ)	A hardware signal that enables a device to request attention from a host. The host typically invokes an interrupt service routine to handle the condition that caused the request.
Interrupt Transfer	One of the four USB transfer types. Interrupt transfer characteristics are small data, non-periodic, low frequency, and bounded-latency. Interrupt transfers are typically used to handle service needs. See also transfer type.
IRP	See I/O Request Packet.

Term	Definition
IRQ	See Interrupt Request.
Isochronous Data	A stream of data whose timing is implied by its delivery rate
Isochronous Device	An entity with isochronous endpoints, as defined in the USB Specification, that sources or sinks sampled analog streams or synchronous data streams.
Isochronous Sink Endpoint	An endpoint that is capable of consuming an isochronous data stream that is sent by the host.
Isochronous Source Endpoint	An endpoint that is capable of producing an isochronous data stream and sending it to the host.
Isochronous Transfer	One of the four USB transfer types. Isochronous transfers are used when working with isochronous data. Isochronous transfers provide periodic, continuous communication between host and device. See also transfer type.
Jitter	A tendency toward lack of synchronization caused by mechanical or electrical changes. More specifically, the phase shift of digital pulses over a transmission medium.
kb/s	Transmission rate expressed in kilobits per second.
kB/s	Transmission rate expressed in kilobytes per second.
Little Endian	Method of storing data that places the least significant byte of multiple-byte values at lower storage addresses. For example, a 16-bit integer stored in Little Endian format places the least significant byte at the lower address and the most significant byte at the next address. See also Big Endian.
LOA	Loss of bus activity characterized by an SOP without a corresponding EOP.
Low-speed	USB operation at 1.5 Mb/s. See also full-speed and high-speed.
LSb	Least significant bit.
LSB	Least significant byte.
Mb/s	Transmission rate expressed in megabits per second.
MB/s	Transmission rate expressed in megabytes per second.
Message Pipe	A bidirectional pipe that transfers data using a request/data/status paradigm. The data has an imposed structure that enables requests to be reliably identified and communicated.
Microframe	A 125-microsecond time base established on high-speed buses.
MSB	Most significant byte.
NAK	Handshake packet indicating a negative acknowledgment.
Non Return to Zero Invert (NRZI)	A method of encoding serial data in which ones and zeroes are represented by opposite and alternating high and low voltages where there is no return to zero (reference) voltage between encoded bits. Eliminates the need for clock pulses.
NRZI	See Non Return to Zero Invert.
Object	Host software or data structure representing a USB entity.
Packet	A bundle of data organized in a group for transmission. Packets typically contain three elements: control information (for example, source, destination, and length), the data to be transferred, and error detection and correction bits.
Packet Buffer	The logical buffer used by a USB device for sending or receiving a single packet. This determines the maximum packet size the device can send or receive.

Term	Definition
Packet ID (PID)	A field in a USB packet that indicates the type of packet and by inference, the format of the packet and the type of error detection applied to the packet.
Phase	A token, data, or handshake packet. A transaction has three phases
Phase Locked Loop (PLL)	A circuit that acts as a phase detector to keep an oscillator in phase with an incoming frequency.
Physical Device	A device that has a physical implementation; for example, speakers, microphones, and CD players.
PID	See Packet ID.
Pipe	A logical abstraction representing the association between an endpoint on a device and software on the host. A pipe has several attributes; for example, a pipe may transfer data as streams (stream pipe) or messages (message pipe) See also stream pipe and message pipe.
PLL	See Phase Locked Loop.
Polling	Asking multiple devices, one at a time, if they have any data to transmit.
POR	See Power On Reset.
Port	Point of access to or from a system or circuit. For the USB, the point where a USB device is attached.
Power On Reset (POR)	Restoring a storage device, register, or memory to a predetermined state when power is applied.
Prime (Endpoint)	A term used in this device controller implementation to describe the action of readying an endpoint to transmit or receive data.
Programmable Data Rate	A fixed data rate (single-frequency endpoints), a limited number of data rates (32 kHz, 44.1 kHz, 48 kHz, …), or a continuously programmable data rate. The exact programming capabilities of an endpoint must be reported in the appropriate class-specific endpoint descriptors.
Protocol	A specific set of rules, procedures, or conventions relating to format and timing of data transmission between two devices.
RA	See rate adaptation.
Rate Adaptation	The process by which an incoming data stream, sampled at F^Si, is converted to an outgoing data stream, sampled at F^So, with a certain loss of quality, determined by the rate adaptation algorithm. Error control mechanisms are required for the process. F^Si and F^So can be different and asynchronous. F^Si is the input data rate of the RA; F^So is the output data rate of the RA.
Request	A request made to a USB device contained within the data portion of a SETUP packet.
Retire	The action of completing service for a transfer and notifying the appropriate software client of the completion.
Root Hub	A USB hub directly attached to the Host Controller. This hub (tier 1) is attached to the host.
Root Port	The downstream port on a Root Hub.
Sample	The smallest unit of data on which an endpoint operates; a property of an endpoint.
Sample Rate (Fs)	The number of samples per second, expressed in Hertz (Hz).
Sample Rate Conversion (SRC)	A dedicated implementation of the RA process for use on sampled analog data streams. The error control mechanism is replaced by interpolating techniques.
Service	A procedure provided by a System Programming Interface (SPI).
Service Interval	The period between consecutive requests to a USB endpoint to send or receive data.

Term	Definition
Service Jitter	The deviation of service delivery from its scheduled delivery time.
Service Rate	The number of services to a given endpoint per unit time.
SOF	See Start-of-Frame.
SOP	Start-of-Packet
SPI	See System Programming Interface.
Split transaction	A transaction type supported by host controllers and hubs. This transaction type enables full- and low-speed devices to be attached to hubs operating at high-speed.
SRC	See Sample Rate Conversion.
Stage	One part of the sequence composing a control transfer; stages include the Setup stage, the Data stage, and the Status stage.
Start-of-Frame (SOF)	The first transaction in each (micro) frame. An SOF enables endpoints to identify the start of the (micro) frame and synchronize internal endpoint clocks to the host.
Stream Pipe	A pipe that transfers data as a stream of samples with no defined USB structure.
Synchronization Type	A classification that characterizes an isochronous endpoint's capability to connect to other isochronous endpoints.
Synchronous RA	The incoming data rate, F^Si, and the outgoing data rate, F^So, of the RA process, are derived from the same master clock. There is a fixed relation between F^Si and F^So.
Synchronous SRC	The incoming sample rate, F^Si, and outgoing sample rate, F^So, of the SRC process are derived from the same master clock. There is a fixed relation between F^Si and F^So.
System Programming Interface (SPI)	A defined interface to services provided by system software.
TDM	See Time Division Multiplexing.
TDR	See Time Domain Reflectometer.
Termination	Passive components attached at the end of cables to prevent signals from being reflected or echoed.
Time Division Multiplexing (TDM)	A method of transmitting multiple signals (data, voice, and/or video) simultaneously over one communications medium by interleaving a piece of each signal one after another.
Time Domain Reflectometer (TDR)	An instrument capable of measuring impedance characteristics of the USB signal lines.
Timeout	The detection of a lack of bus activity for some predetermined interval.
Token Packet	A type of packet that identifies what transaction is to be performed on the bus.
Transaction	The delivery of service to an endpoint; consists of a token packet, optional data packet, and optional handshake packet. Specific packets are enabled/required based on the transaction type.
Companion Controller	A functional component of a USB hub. The Companion Controller responds to special high-speed transactions and translates them to full/low-speed transactions with full/low-speed devices attached on downstream facing ports.
Transfer	One or more bus transactions to move information between a software client and its function.
Transfer Type	Determines the characteristics of the data flow between a software client and its function. Four standard transfer types are defined: control, interrupt, bulk, and isochronous.

Term	Definition
Turn-around Time	The time a device needs to wait to begin transmitting a packet after a packet has been received to prevent collisions on the USB. This time is based on the length and propagation delay characteristics of the cable and the location of the transmitting device in relation to other devices on the USB.
Universal Serial Bus Driver (USBD)	The host resident software entity responsible for providing common services to clients that are manipulating one or more functions on one or more Host Controllers.
Universal Serial Bus Resources	Resources provided by the USB, such as bandwidth and power. See also Device Resources and Host Resources.
Upstream	The direction of data flow towards the host. An upstream port is the port on a device electrically closest to the host that generates upstream data traffic from the hub. Upstream ports receive downstream data traffic.
USBD	See Universal Serial Bus Driver.
USB-IF	USB Implementers Forum, Inc. is a nonprofit corporation formed to facilitate the development of USB compliant products and promote the technology.
Virtual Device	A device that is represented by a software interface layer. An example of a virtual device is a hard disk with its associated device driver and client software that makes it able to reproduce an audio .WAV file.
Word	A data element that is two bytes (16 bits) in size.

Book I:
i.MX31 and i.MX31L Integration and Description

Introduction

Book I comprises detailed descriptions and information on the integration of the i.MX31 and i.MX31L ICs. Book I includes the following chapters.

Device Introduction and Memory Map

Clocks, Power Management and Reset

Pins

Debug

Boot

Chapter 1
Introduction to the i.MX31 and i.MX31L Multimedia Applications Processors

The i.MX31 and i.MX31L Multimedia Applications Processors are designed for high-tier and mid-tier smart phone markets. They are a model solution for multimedia and graphics applications that require low power combined with high performance.

The i.MX31 and i.MX31L processors are built around an ARM1136JF-S™ processor core and are implemented using 90 nm technology.

NOTE

The MBX R-S graphics accelerator is not available in the i.MX31L Multimedia Applications Processor.

The system includes the following features:

- Multimedia and floating point hardware acceleration that supports:
  - MPEG-4 real-time encoding of up to 30 fps VGA
  - MPEG-4 real-time video post processing of up to 30 fps VGA
  - Video conference call of up to 30 fps QCIF (decoder in software), 128 Kbps
  - Video streaming playback of up to 30 fps VGA, 384 Kbps
  - 3D graphics and other application acceleration with the ARM tightly coupled Vector Floating Point (VFP) coprocessor
  - On-the-fly video processing reducing system memory load (for example, power-efficient Viewfinder application with no involvement of either the memory system or the ARM CPU)
- Advanced power management that includes:
  - Dynamic voltage and frequency scaling
  - Multiple clock and power domains
  - Independent gating of power domains
- Multiple communication and expansion ports, including a fast parallel interface to an external graphic accelerator (with support for major graphic accelerator vendors)

1.1 Architectural Overview

With their ARM11 microprocessor core, the i.MX31 and i.MX31L processors provide new milestones in applications processors, delivering the high performance and low-power consumption demanded by modern digital devices, such as the following:

- Feature-rich cellular phones

- Portable media players and mobile gaming machines
- Personal digital assistants (PDAs) and wireless PDAs
- Portable DVD players
- Digital cameras

The heart of the i.MX31 and i.MX31L is an ARM1136JF-S core, which can run at speeds up to 665 MHz and is optimized for minimal power consumption, using the most advanced techniques for power saving (including DPTC, DVFS, power gating, and clock gating). With 90 nm technology and dual VT, the i.MX31 and i.MX31L offer an optimum balance of performance versus current leakage.

Performance is boosted by a multi-level cache system, and the i.MX31 and i.MX31L feature peripheral devices, such as an MPEG-4 Hardware Encoder (VGA, 30 fps), an autonomous Image Processing Unit, a Vector Floating Point (VFP11) coprocessor, and a Smart Direct Memory Access (SDMA)/RISC-based DMA controller.

The i.MX31 and i.MX31L support connections to various types of external memory, such as 266 MHz DDR, NAND Flash, NOR Flash, SDRAM, and SRAM.

The i.MX31 and i.MX31L can be connected to a variety of external devices using technologies, such as Universal Serial Bus, On-The-Go, High-Speed (USB 2.0, OTG), ATA-4, MMC/SDIO, and CompactFlash.

1.1.1 High-Level Block Diagram

Figure 1-1 is a high-level block diagram of the i.MX31 and i.MX31L Multimedia Applications Processors. Figure 1-2 shows an example scenario of external connections for the i.MX31.

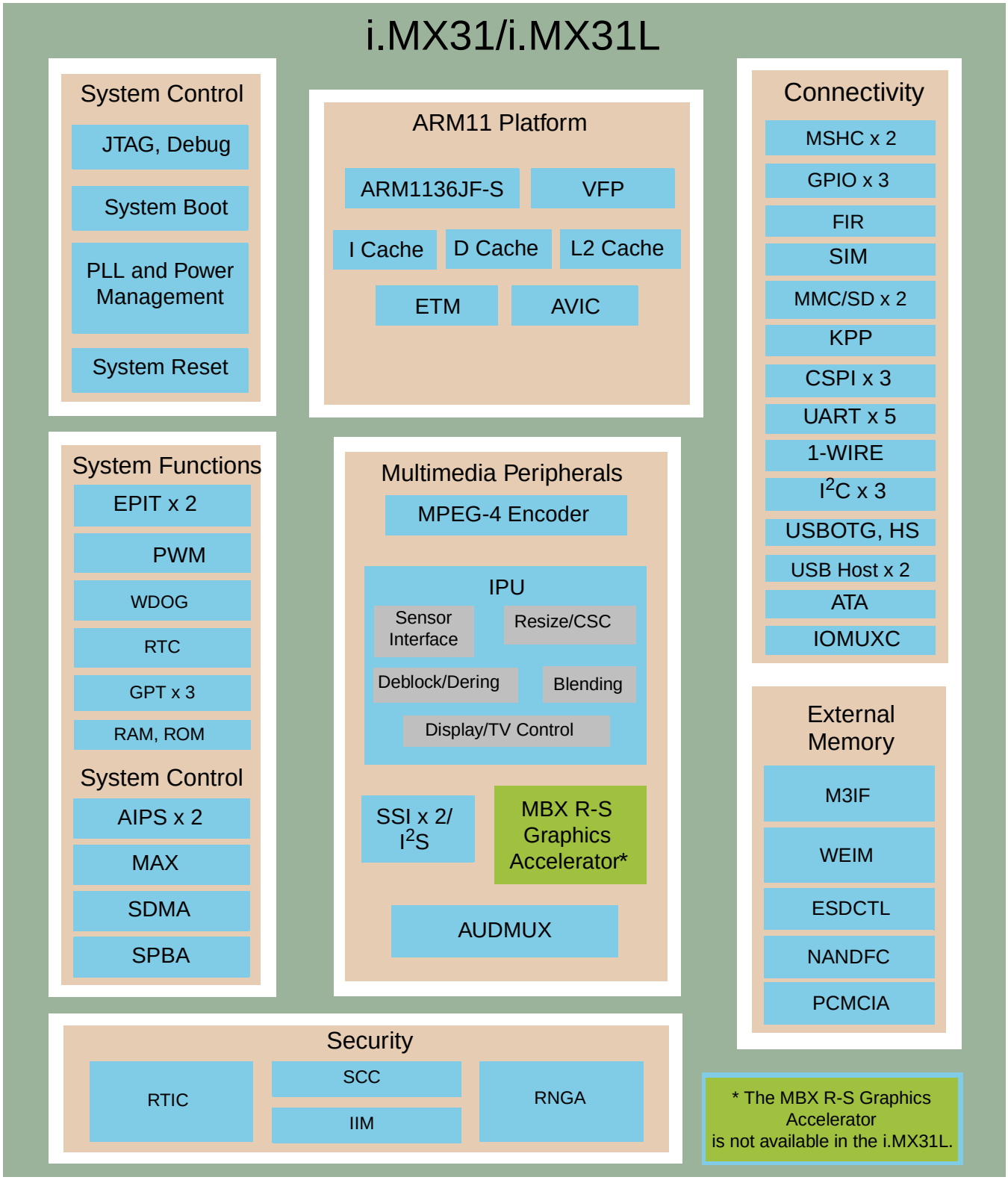


Figure 1-1. i.MX31 and i.MX31L Simplified Block Diagram

i.MX31

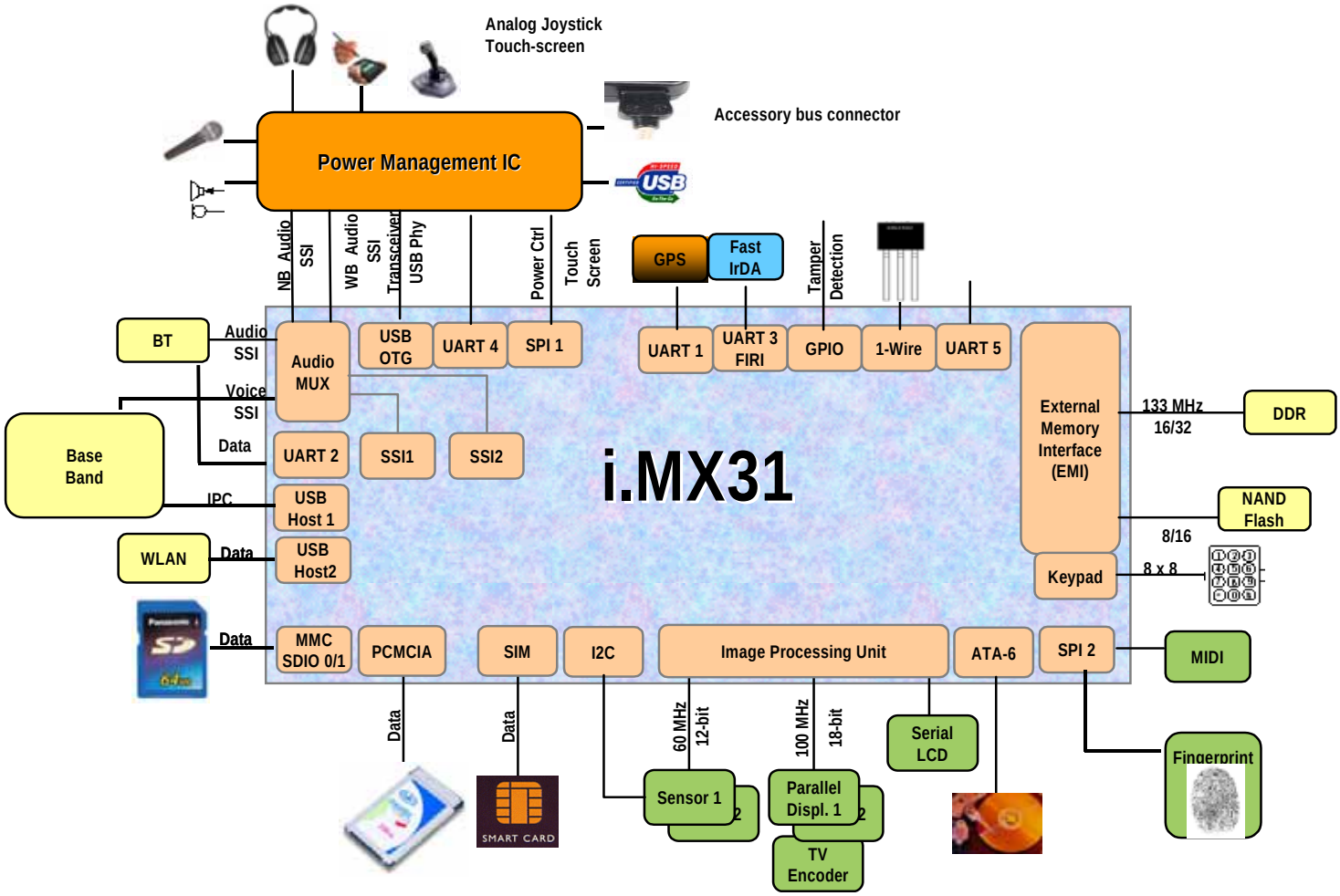


Figure 1-2. i.MX31 External Connections Diagram

1.2 Hardware Modules

The hardware modules for the i.MX31 and i.MX31L can be divided into six areas:

- System control
- ARM11 Platform
- Standard system functional elements
- Multimedia and human interface
- Peripherals
- Special functional blocks

The specific modules that make up each of these areas are listed in the following subsections.

1.2.1 System Control

System control modules provide features such as clocks, debugging, and a wake-up sequence:

- System JTAG Controller (SJC)
- Debug features
- Bootstrap
- Clocks, power management, and reset (AP clock controller module)
- System reset

1.2.2 ARM11 Platform

The ARM11 Platform is based on the ARM11 core, and provides basic processing for the device, as follows:

- ARM11 core
- VFP
- I-Cache
- D-Cache
- L2 Cache
- ETM
- MAX

1.2.3 Standard System Functional Elements

The modules in this area provide resources and services for the operating system:

- Timers
- PWM
- Watchdog
- RTC
- GPIOs
- RAM and ROM
- Smart Direct Memory Access (SDMA)

1.2.4 Multimedia and Human Interface

The modules in this area provide state-of-the-art multimedia and human interface capabilities:

- MPEG-4 Encoder
- GPU (Graphics Processing Unit, based on the MBX R-S)

NOTE

The MBX R-S graphics accelerator is not available in the i.MX31L.

- IPU:
  - Sensor interface
  - Resize/CSC
  - Deblock/Dering
  - Blending
  - Display and TV control
- Keypad
- Audio MUX
- SSI/I^2S

1.2.5 Peripherals

The peripheral devices provide connection capabilities to standard interfaces:

- Fast IrDA
- MMC/SD units
- CSPI units
- UARTs
- 1-Wire
- USBOTG, High-Speed
- USB host units
- SIM
- ATA
- PCMCIA/CF
- External Memory Interface (EMI):
  - SDRAM/DDR
  - VSync
  - PSRAM
  - NAND/NOR Flash
  - SmartMedia

1.2.6 Special Functional Blocks

The special functional blocks are a group of modules and features that provide a complete solution for system security, including security hardware.

1.2.7 Detailed Block Diagram

Figure 1-3 is a detailed block diagram of the i.MX31 and i.MX31L.

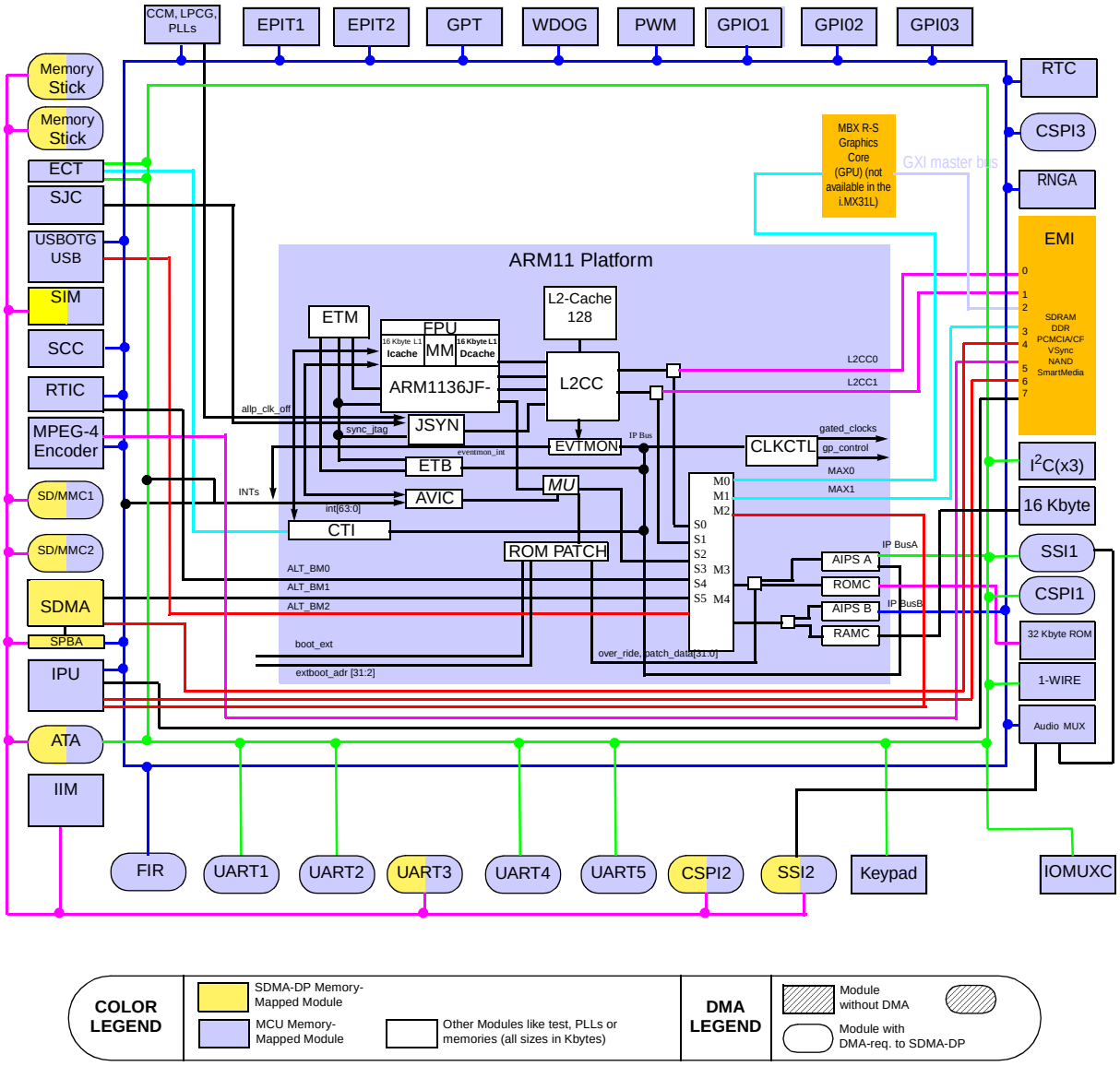


Figure 1-3. i.MX31 and i.MX31L Detailed Block Diagram

NOTE

The MBX R-S graphics accelerator is not available in the i.MX31L.

1.2.8 Applications Processor Core (ARM11 Core)

The core of the i.MX31 and i.MX31L Multimedia Applications Processors is the ARM1136JF-S, which is based on the innovative ARM v6 architecture. It supports ARM Thumb® instruction sets, and features Jazelle® technology (which enables direct execution of Java™ bytecodes), as well as a range of SIMD DSP instructions that operate on 16-bit or 8-bit data values in 32-bit registers.

The ARM1136JF-S processor core features:

- Integer unit with integral EmbeddedICE® logic
- Eight-stage pipeline
- Branch prediction with return stack
- Low-interrupt latency
- Instruction and Data Memory Management units (MMUs), managed by using micro TLB structures backed by a unified main TLB
- Instruction and Data L1 Caches, including a non-blocking Data cache with Hit-Under-Miss
- Virtually indexed/physically addressed L1 caches
- 64-bit interface to both L1 caches
- Write buffer (bypassable)
- High-Speed Advanced Micro Bus architecture (AMBA) L2 interface
- Vector Floating Point coprocessor (VFP) for hardware acceleration of 3D graphics and other floating-point applications
- ETM™ and JTAG-based debug support

1.2.8.1 Memory System

The ARM1136JF-S complex includes a 16-Kbyte instruction cache and a 16-Kbyte data L1 cache. The complex connects to the i.MX31 128 Kbyte L2 unified cache through a read-only 64-bit instruction interface, a bidirectional 64-bit data read/write interface, and a 64-bit data write interface.

The embedded SRAM (16 Kbyte) can be used for Audio Streaming data to avoid external memory accesses for applications such as low-power audio playback, security, or other applications. There is also a ROM (32 Kbyte) for bootstrap code and other frequently used code and data.

1.2.8.1.1 Internal RAM

There are 16 Kbytes of internal SRAM.

1.2.8.1.2 Internal ROM

The ROM is partitioned into two parts (ROM is not aliased):

- The location of the first 16 Kbytes of Secure ROM starts at the lowest part of the memory map, at address 0×0. After reset, the ARM processor starts running code from this location.
- The remaining 16 Kbytes are mapped at the starting address 0×00404000 (4 Mbytes +16 Kbytes).

1.2.8.1.3 Internal Registers

The various categories of internal registers are all decoded by the two AIPS modules—AIPS A and AIPS B. Many registers belong to the ARM11 Platform; others belong to the various off-platform modules.

- Any write access by the ARM1136JF-S core to off-platform modules encounters two wait states; that is, any write access lasts for three cycles.
- Any read access by the ARM1136JF-S core from off-platform modules has one wait state; that is, any read access lasts for two cycles.
- Any number of architected registers can be defined in each peripheral space. Software must explicitly address the registers without making assumptions regarding multiple mapping.

1.2.9 Interrupts

The i.MX31 and i.MX31L have a dedicated Programmable Interrupt controller for the AP. The controller handles interrupts generated by associated peripherals and can handle multiple interrupts with varying priority levels. These characteristics make the controller an excellent solution for administering interrupt requirements at both the RTOS and SoC level. The controller also has advanced features to reduce module power consumption.

1.2.9.1 ARM11 Platform Vectored Interrupt Controller (AVIC)

The ARM11 Platform Vectored Interrupt controller (AVIC) provides interrupt support for the ARM-based peripherals.

The AVIC is a 32-bit peripheral that collects interrupt requests from a maximum of 64 sources and provides an interface to the ARM1136JF-S core. The AVIC includes hardware acceleration and software-controlled priority levels for normal interrupts.

The AVIC has hardware for prioritizing interrupts, and it can supply the interrupt vector address of the highest priority interrupt to the ARM1136JF-S core automatically via an interrupt sideband bus. The module has an AHB-Lite interface, and can be programmed by the ARM1136JF-S peripheral Advanced High-Performance Bus (AHB). The AVIC contains a 30×64 memory array to store the vector table.

The following list summarizes AVIC features:

- Supports a maximum of 64 interrupt sources
- Handles fast and normal interrupts
- Selects normal or fast interrupt request for any interrupt source
- Supports hardware accelerated vectoring to service routines for all normal interrupts
- Indicates pending interrupt sources via a register for normal and fast interrupts
- Identifies the highest priority interrupt number via a register, which can be used as a table index
- Can independently enable or disable any interrupt source
- Provides a mechanism for software to schedule an interrupt
- Has an integrated vector table for hardware acceleration of normal interrupt service routine entries

- Supports a maximum of 16 software-controlled priority levels for normal interrupts and priority masking
- Supports single bit disabling

1.2.10 External Memory Interface (EMI)

The following list describes the main features of the External Memory Interface (EMI):

- Programmable 16-bit or 32-bit wide external data bus
- Address interleaving
- Four input ports with arbitration
- External Bus Alternative Master support (for example, external graphics accelerator IC)
- External Memory Sharing (for example, with a cellular baseband IC)
- Support for standard SRAM and Flash device
- SDRAM controller (up to 133 MHz)
- PSRAM support (up to 133 MHz)
- DDR support with a data rate up to 266 MHz
- NAND Flash support
  - Address space to a maximum of 2 Gbyte
  - One Chip Select
  - Support for 8-bit and 16-bit devices, with dedicated IO pins for the 8-bit interface
  - Little Endian addressing
  - Internal buffer of 2 Kbyte RAM for boot (at start-up) and as a transfer buffer during normal operation
  - Internal automatic Bootloader
  - Data protection for buffer RAM and NAND Flash
- SmartMedia card support
- PCMCIA release 2.1 support
  - PC Card
  - Compact Flash
  - TrueIDE mode
- Memory snooping mechanism for minimizing memory traffic when using Smart Displays (For details, see Section 1.2.17.4, “Image Processing Unit.”)

1.2.11 Clock Power Management and Reset

1.2.11.1 Clocking and Synchronization

The i.MX31 and i.MX31L have several frequency domains. The frequency ratios and absolute values are flexible, and provide the performance needed to run applications while keeping the operating frequency as low as possible to save power. Table 1-1 shows some of the fixed frequency ratios.

Table 1-1. Frequency Domains

F_{ARM}	F_{SL2}	F_{ML2}	F_{XBAR}	F_{MEM}
F	F	F/2 or F/4	F/2 or F/4	F/4
532	532	133	133	133
432	432	108	108	108
266	266	133	133	133
134	134	67	67	67

1.2.11.2 Power Management

The i.MX31 and i.MX31L use the following advanced power management features:

- Dynamic frequency and voltage scaling (DVFS) with one voltage domain
- Active Well-Biasing (AWB)
- Power gating in standby
- Three power domains (for power gating)
- Independent low power modes for different power domains

The i.MX31's and i.MX31L's power domains and their operating modes are as follows:

- ARM11 Platform domain (ARM11 Core + MMU + Caches)
  - — ARM_Active mode:
    The ARM clock frequency can vary between fmax and fmin. The voltage can vary between Vmax and Vmin.
  - — ARM_Stop mode:
    The mode is a result of STANDBY instruction execution. In this mode, the supply voltage can be reduced to $Vcc_{min.}$
  - — ARM_Standby mode:
    This mode is similar to ARM_Stop mode, except the supply voltage is reduced to a minimum value for data retention and well-biasing is on. ARM is not functional in this mode. Before attempting to perform any function, increase the supply voltage and disable well-biasing.
  - — ARM_Shut-down mode:
    The ARM power domain supply is off. If ARM state retention is required, save the data from critical registers before entering Standby mode.
- Peripheral Domain, which includes all the peripherals except the DPLL
  - — Peripheral_On mode:
    Supply of the domain is on.
  - — Peripheral_Standby mode:
    Supply voltage is reduced to minimum for data retention. Well-biasing is on.
  - — Peripheral_Off mode:
    Supply of the domain is off.

- DPLL, included in a separate domain because an unknown influence on operating frequency can occur while changing voltage in DVFS modes
  - — DPLL_On mode:
    Supply of the domain is on.
  - — DPLL_Standby mode:
    DPLL is in reset. Supply voltage can be reduced to minimum. Well-biasing is on. Restart DPLL to get the generated output clock.
- DPLL_Off mode:
  Supply of the domain is off.

1.2.11.3 Reset Module

The reset module controls or distributes all of the system reset signals used by the i.MX31. The reset module generates seven distinct events—a global reset and a processor reset.

- $\overline{\text{mcu\_reset\_out}}$ signal is connected to ARM11P and RTIC.
- $\overline{\text{reset\_fuse}}$ signal is connected to laser fuses in L2 cache data array.
- $\overline{\text{ccm\_por\_reset}}$ signal is connected to JTAG.
- $\overline{\text{periph\_reset\_out}}$ signal is connected to all peripherals except IIM, EMI, WDOG.
- $\overline{\text{ccm\_pll\_reset}}$ signal is connected to three PLLs and FPM.
- $\overline{\text{global\_reset}}$ signal is connected to modules engaged in boot and security (IIM, EMI, MGA, RTC).
- $\overline{\text{ect\_reset}}$ signal is connected to ECT module.

1.2.12 Pins

The external pins in the i.MX31 and i.MX31L can be configured for various functions, according to system use. The memory interface pins are controlled by the EMI, but the rest of the pins are controlled by I/O MUX (IOMUX) units.

1.2.12.1 Multiplexing, GPIO, and Pad Control Architecture

Peripheral signals are mapped (or multiplexed) to give the end user a great deal of flexibility for determining the external connections planned for the i.MX31 and i.MX31L implementation. The user can use software to configure the signals associated with each pin.

The muxing hardware is composed of the following:

- IOMUX—Combinational logic that does the muxing
- IOMUX Controller (IOMUXC)—An IP that consists of memory-mapped registers designed to:
  - — Control the IOMUX
  - — Handle interrupt observation
  - — Control pad settings, for example pull-up, pull-down, hysteresis, and keeper.
- GPIO—An IP that consists of memory-mapped registers used for handling GPIO capabilities—for example, using software to apply a value on a pad, capturing a value from a pad, or generating interrupts from pads.

1.2.13 Security

Tamper detect logic is used to issue a security violation. This logic is activated if the tamper detect pin is asserted. The tamper detect logic is disabled after reset. After enabling the logic, it is impossible to disable it until the next reset. For a schematic diagram of the tamper detect logic, refer to Chapter 4, "Signal Multiplexing."

1.2.14 Connectivity

1.2.14.1 Wired Connectivity

1.2.14.1.1 UART x 5

Five UART modules in the i.MX31 and i.MX31L support the following serial data transmit/receive protocols and configurations:

- Data words of 7-bit or 8-bit length, 1 or 2 stop bits, programmable parity (even, odd, or none)
- Programmable baud rates to a maximum of 1.875 Mbit/s
- 32-byte FIFO on Tx and 32 halfword FIFO on Rx supporting auto-baud
- IrDA 1.0 support (to a maximum SIR speed of 115200 bps)

1.2.14.2 USB Module

The USB module in the i.MX31 and i.MX31L is used for inter-processor communication with the onboard Cellular Modem baseband processor (Host Port #1), for connection to a WLAN/Bluetooth and other onboard peripherals (Host Port #2), and for communication with external USB devices (OTG host/device port) via a transceiver IC.

The USB has the following main features:

- EHCI compatibility
- Power-Saving mode for hosts and Suspend mode for other functions
- Transaction scheduling and transfer level protocol implemented in hardware

1.2.14.2.1 USB Host Port 1

USB Host 1 has the following capabilities and features:

- Is compliant with USB 2.0 specification for operation at high speed (480 Mbit/s), full speed (12 Mbit/s), and low speed (1.5 Mbit/s).
- Features built-in switching logic to support bypass mode, so an external host can communicate directly with the Cellular Modem baseband processor.
- Is designed to support transceiver-less connection to the onboard peripherals in low speed and full speed mode, and connection to ULPI (UTMI+ low pin count) and legacy full speed transceivers.

1.2.14.2.2 USB Host Port 2

USB Host 2 has the following capabilities and features:

- Is compliant with USB 2.0 specification for operation at full speed (12 Mbit/s) and low speed (1.5 Mbit/s)
- Is designed to support transceiver-less connection to the Cellular Modem baseband processor

1.2.14.2.3 USBOTG Port

The USBOTG port has the following capabilities and features:

- Is compliant with the OTG Supplement to the USB 2.0 specification
- As a host, operates at high speed, full speed, and low speed; as a device, operates at high speed and full speed
- Includes Host Negotiation Protocol (HNP) and Session Request Protocol (SRP) implemented in the hardware with software support. (These protocols are also full controllable by software.)
- Includes built-in switching logic to support bypass mode, so an external host can communicate directly to the Cellular Modem baseband processor in full speed or low speed
- Is designed to interface with ULPI transceivers (Low Pin Count Supplement to the UTMI+ specification), and legacy full speed transceivers

1.2.14.3 PCMCIA Port

The i.MX31 and i.MX31L PCMCIA Rel.2.1 port is part of the EMI module (described in Section 1.2.10, “External Memory Interface (EMI)”). The PCMCIA port provides a high data rate interface to external peripherals (such as WLAN 802.11b) and Compact Flash cards.

1.2.14.4 Wireless Connectivity

1.2.14.4.1 Fast Infrared Interface (FIR)

The Fast Infrared interface (FIR) module supports infrared communication, including the following features:

- Is compliant with IrDA 1.1 for MIR and FIR. (The IrDA 1.0 Serial Infrared (SIR) protocol can be supported by one of the UART modules.)
- Has a full physical layer implementation
- Supports 0.56 Mbit/s and 1.152 Mbit/s Medium Infrared (MIR) physical layer protocol
- Supports 4 Mbit/s FIR physical layer protocol defined by IrDA version 1.4
- Generates 16-bit and 32-bit CRC for error detection

1.2.14.4.2 Bluetooth

Connection to high-bit rate communication devices that use Bluetooth wireless technology is supported.

1.2.14.4.3 Wireless LAN 802.11a/b

Connection to high-bit rate communication devices that use WLAN 802.11 a/b technology is supported.

1.2.15 Timers

1.2.15.1 General Timers

The i.MX31 and i.MX31L include several timers:

- One General Purpose Timer (GPT) for Operating System Real-Time scheduling and waveform sampling/generation functions. (This timer measures intervals or generates periodic outputs.)
- Two Enhanced Periodic Interrupt Timers (EPIT1,EPIT2) for Operating System Real-Time scheduling. (This timer provides precise interrupts at regular intervals with minimal processor intervention.)

1.2.15.2 Watchdog Timer (WDOG)

The Watchdog Timer provides a time-out notification if the system ceases activity for a user-specified period of time. Both the period of time and the time-out action are programmable.

The Watchdog Timer has the following features:

- Time-out, which is programmable to a value between 0.5–64 s
- Resolution of 0.5 s

In case of a time-out, the following actions can be taken:

- Interrupt to the MPU, which is programmed in software
- Internal reset, which is programmed in software and can follow the interrupt after a predefined time-out
- External pin toggle for resetting external devices, issued together with the internal reset

1.2.16 System Resources

1.2.16.1 AIPS

The AIPS acts as an interface between the system bus (AHB-Lite 2.v6) and lower bandwidth peripherals that conform to the IP Bus Specification Rev 3.0 SkyBlue line interface (IPS).[1]

The following list summarizes the key features of the AIPS:

- Supports the IPS slave bus (SkyBlue) signals. This interface is only meant for slave mode peripherals.
- Supports 32-bit IPS peripherals. (AIPS supports byte, halfword, word, and double-word read and write operations for each peripheral.)
- Supports two global external IPS peripheral spaces (32 Mbyte and 31 Mbyte each)

1. To acquire a copy of this specification, contact Freescale Semiconductors, Inc.

- Supports a pair of IPS accesses for 64-bit transfers and certain misaligned AHB transfers
- AIPS A directly supports a maximum of 16 on-platform IPS peripherals, with 16 Kbytes of address space for each one
- AIPS B directly supports a maximum of 32 on-platform IPS peripherals, with 16 Kbytes of address space for each one
- Supports configurable per-module write buffering
- Provides configurable per-module and per-master access protections
- Handles peripheral read transactions that require a minimum of two HCLK clocks, and unbuffered write transactions that require a minimum of three HCLK clocks
- Uses one single asynchronous reset and one global clock
- Provides Secure Restricted Access Control for different masters to selected peripherals

1.2.16.2 Smart Direct Memory Access Controller (SDMA)

The Smart Direct Memory Access (SDMA) controller maximizes system performance by relieving the ARM core of the task of the bulk transfer of data from memory to memory or between memory and on-chip peripherals. The advantage of the SDMA is its dynamic routing capability and its ability to perform numerous tasks simultaneously based on the DMA channel's descriptors. Application processor OS software drivers can make extensive use of DMA channels to minimize software overhead and transfer latencies.

Each of the 32 DMA channels support linear memory, 2D memory, buffer chaining, FIFO, and Enable FIFO for both source and destination.

The SDMA controller has the following features:

- Three independent AHB buses, typically used for transferring data between the applications processor, DSP, and EMI domains
- Dedicated IP SkyBlue peripheral bus
- Daisy chain
- Multi-channel DMA with virtual support of up to 32 simultaneous DMA channels
- Very fast context-switching with two-level priority-based preemptive multi-tasking
- DMA units with flush and pre-fetch capability, flexible Endianness, and word sizes that range from 8, 16, or 32 bits in length
- Flexible address management for DMA transfers (increment, decrement, and no address changes on source and destination address)
- Support for byte swapping
- Buffer for configurable burst transfer (buffer size is a maximum of 16 words long)
- Capacity to be configured to respond to any of the 32 DMA request signals
- Bus utilization control that can be handled by script if needed
- Burst time-out error for terminating the DMA cycle if the burst cannot be completed in the user-specified length of time
- Buffer overflow—SDMA stops reading if FIFO is full (64 bytes of data)

- Transfer error to terminate the DMA cycle if a transfer error is detected during the DMA burst
- Interrupts provided to interrupt handler, then to the core for bulk data transfer completion or transfer error events
- Capability for each peripheral that supports DMA transfer to generate a request signal to the DMA controller. Each FIFO has a unique system address and can generate a dedicated request signal to the DMA.
- Support for single interface peripheral
- Small footprint and power-efficient architecture
- Power supplied by a 16-bit Instruction Set microRISC engine
- Access to a library of scripts and API

1.2.16.3 ATA Controller

The ATA block is an AT attachment host interface. Its main function is to interface with IDE hard disc drives and ATAPI optical disc drives. It interfaces with the ATA device over a number of ATA signals.

The ATA interface is compliant to the ATA-6 standard, and supports the following protocols:

- PIO mode 0, 1, 2, 3, and 4
- Multiword DMA mode 0, 1, and 2
- Ultra DMA modes 0, 1, 2, 3, and 4 with bus clock of 50 MHz or higher
- Ultra DMA modes 5 with bus clock of 80 MHz or higher

1.2.17 Image, Video and Graphics

Visual data—video and graphics—is handled in the i.MX31 and i.MX31L with the aid of the Image Processing Unit (IPU), the MPEG-4 Video Encoder module, and software.

1.2.17.1 Video Processing

The i.MX31 processor supports the following video-related activities:

- Capturing video input from an image sensor (support for a maximum of two sensors, no concurrency)
- Displaying a still image or moving video from a stored file
- A video call

The video processing chain is illustrated in Figure 1-4. The IPU performs the steps displayed in the purple box. The remaining steps are performed as follows:

- The camera typically performs format conversion and quality enhancement operations after the image is captured. (This processing chain is typically called post-image processing or pre-processing.) Any processing the camera does not perform can be handled by ARM processor software.
- Any controls the camera requires from the applications processor are performed by ARM software. These controls are transferred to the sensor through an I^2C interface, so the IPU is not involved in

the transfer. (The I^2C interface is a two-wire, bidirectional serial bus that provides a simple standard interface to peripherals.)

- ARM software performs other processing. The software implementation provides the flexibility needed to support a variety of compression algorithms and to adapt as compression standards evolve.

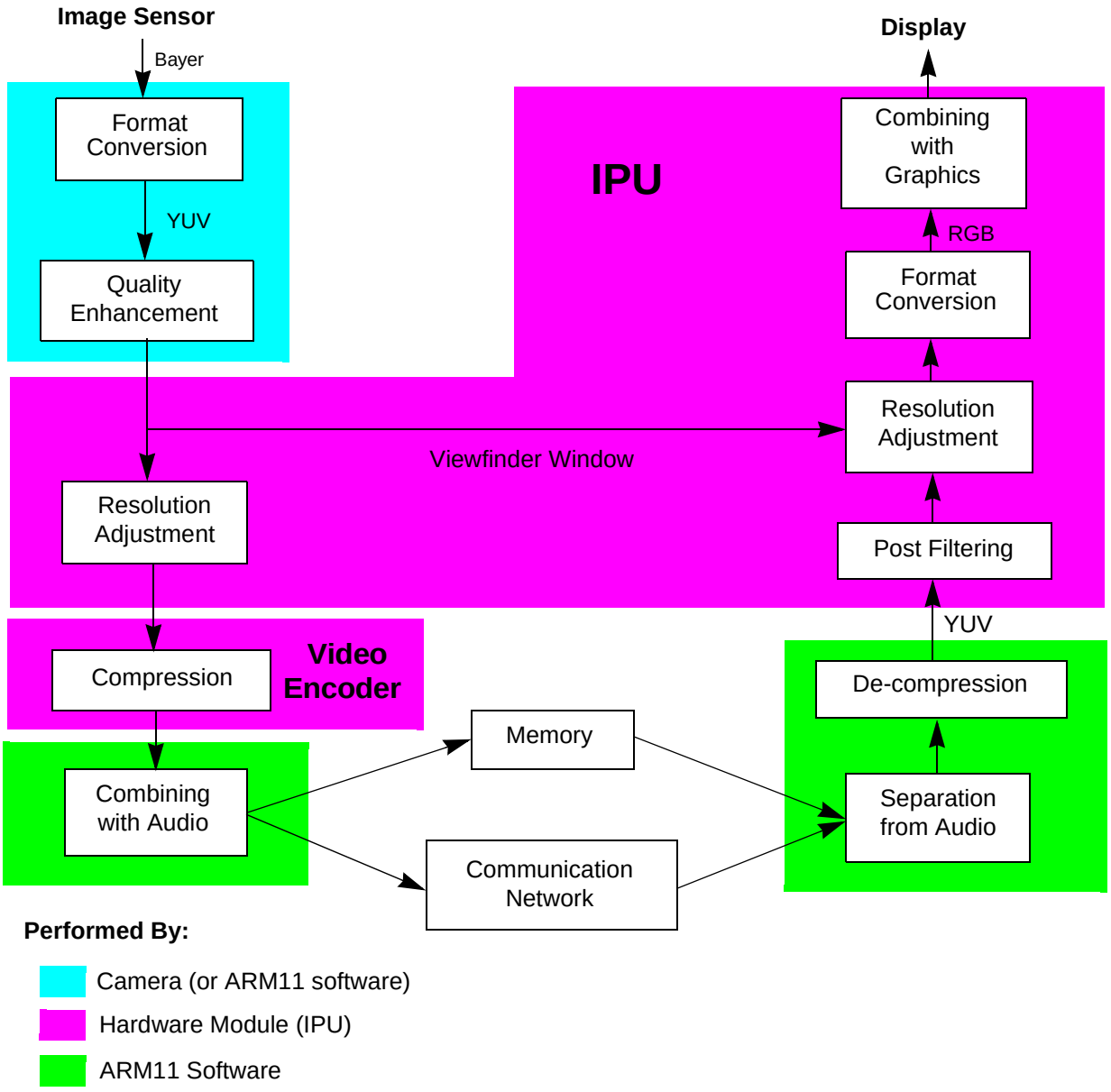


Figure 1-4. Video Processing Chain in the i.MX31

1.2.17.2 Graphics Processing Unit

NOTE

The MBX R-S graphics accelerator is not available in the i.MX31L.

The Graphics Processing Unit (GPU) provides hardware acceleration for 2D and 3D graphics algorithms. GPU acceleration is sufficient to run desktop-quality interactive graphic applications on displays with a screen resolution equivalent to VGA and above and with color representation up to 32 bits per pixel. The i.MX31 GPU uses an ARM MBX R-S graphics accelerator. Figure 1-5 shows the GPU high-level block diagram.

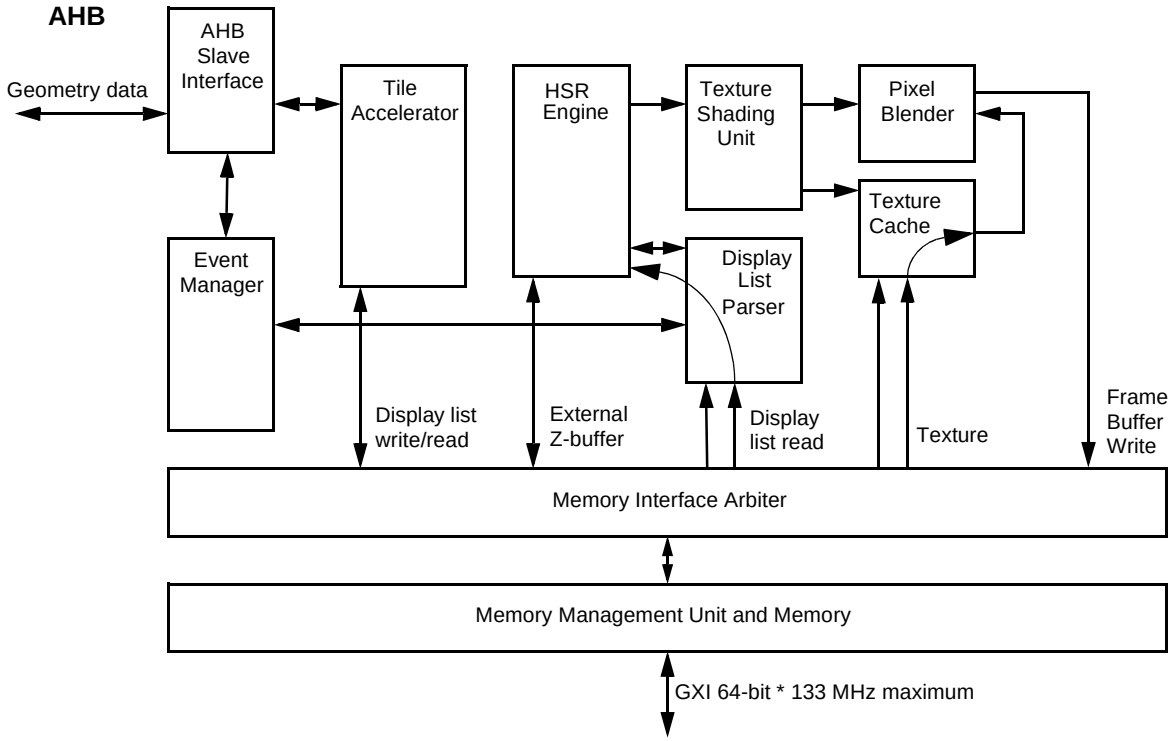


Figure 1-5. GPU Block Diagram

1.2.17.2.1 Graphics Processing Unit Overview

The GPU consists of the following modules:

- Tile Accelerator (TA)
- Event Manager
- Vertex Geometry processor (VGP)
- Display List Parser
- Hidden Surface Removal (HSR) engine
- Texture Shading unit
- Texture cache
- Pixel Blender
- Memory Interface Arbiter
- Memory Management unit
- GPI (Graphics Port interface) to MBX—master gasket

The GPU operates on 3D scene data, sent as batches of triangles, then transformed and lit by the VGP. Triangles are written directly to the TA on a First In First Out (FIFO) basis so the CPU does not stall. The SDMA can also be used to perform batch transfers with minimal CPU involvement.

The TA performs advanced culling of the triangle data by writing the tiled non-culled triangles to external memory.

The Event Manager uses SmartBuffer technology for control, so any amount of scene complexity is handled in a fixed display list buffer size.

The HSR engine reads the tiled data and implements per-pixel HSR with full Z-accuracy. The resulting visible pixels are textured and shaded in 24 bit Internal True Color (ITC) before the final image is rendered for the display buffer.

1.2.17.2.2 Graphics Processing Unit Features

The GPU includes the following features:

- Deferred texturing
- Screen tiling
- Flat and Gouraud shading
- Perspective-correct texturing
- Specular highlights
- Floating-point Z-buffer
- 32-bit ARGB internal rendering and layer buffering
- Full tile blend buffer
- Z-load and store mode
- Per-vertex fog
- 16-bit RGB textures, 1555, 565, 4444, 8332, 88
- 32-bit RGB textures, 8888
- YUV 422 textures
- PVR-TC compressed textures
- One-bit textures for text acceleration
- Point, bilinear, trilinear, and anisotropic filtering
- Full range of OpenGL and Direct3D (D3D) blend modes
- Dot3 bump mapping
- Alpha test
- Zero-cost full-scene anti-aliasing
- 2D-via-3D 2D graphics acceleration.

1.2.17.3 Display Management

The i.MX31 controls the transfer of visual data (video and graphics) to the display. The transfer is performed in the following way:

- Before it is sent to the screen, the data is arranged and stored in a memory buffer called a display buffer. The data sources update the display buffer whenever required.
- Some applications generate content in a separate background buffer and the data is transferred periodically to the display buffer. This is necessary, for example, to avoid visual tearing while displaying frequently changing data. The IPU handles this type of block transfer.
- The data from the display buffer is transferred to the display screen at a rate determined by the display's needs (as described in Table 1-2). This process is called screen refresh. Screen refresh is performed by a dedicated controller on the device in which the i.MX31 is installed, typically called a display controller.

Table 1-2. Refresh Rates and Pixel Formats

	Resolution [pixels]	Refresh Rate	Format
Display	up to 640 × 480 (VGA)	up to 100 Hz (typically 70)	RGB, up to 18 BPP progressive
TV–PAL	704 × 576	25 Hz	YCC 4:2:2 8 bit color component progressive or interlaced
TV–NTSC	704 × 480	30 Hz	

Some display devices, called smart displays, refresh the screen internally. To do this, they include an integrated controller and memory for the display buffer. The advantage of such an implementation is low power consumption. However, the data access rate from the applications processor to the display memory is typically limited by the smart display's data port bandwidth.

Other display devices, called memory-less displays, have no control capabilities. For such displays, the screen refresh is performed from a display buffer in the i.MX31's system memory and is controlled by a display controller in the i.MX31's IPU. A TV screen driven by a Digital Video Encoder or DVE also belongs to this category.

The display devices are typically connected to the i.MX31 through a dedicated display port, controlled by the IPU.

Smart displays can also be connected to the host through the External Memory Interface (EMI) port. In this case, they are managed directly by the processor, without involving the IPU.

1.2.17.4 Image Processing Unit

The role of the Image Processing Unit (IPU) is to provide hardware acceleration and control for processing and displaying visual data (video and graphics) in the i.MX31 and i.MX31L. A schematic description of the IPU internal structure is illustrated in Figure 1-6.

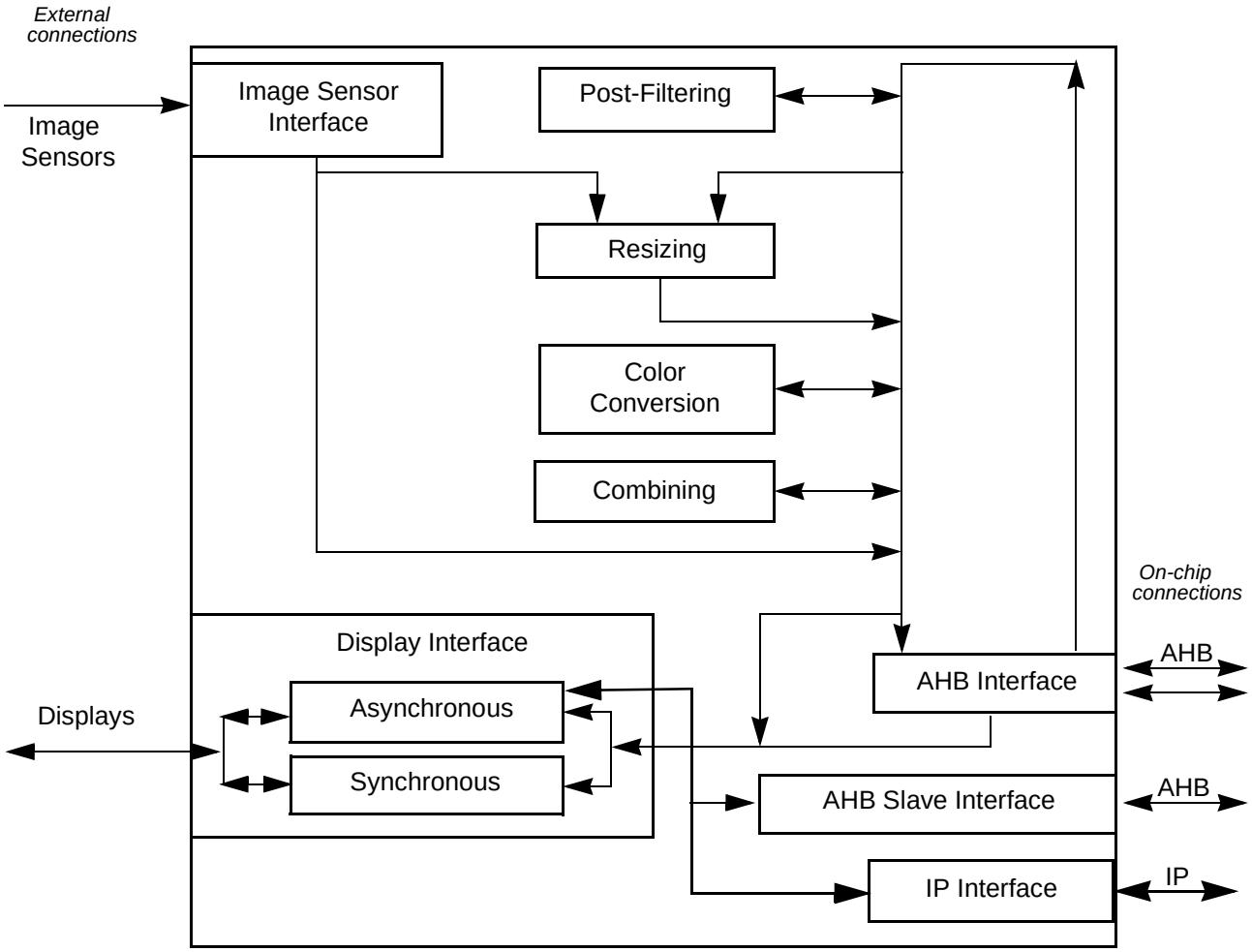


Figure 1-6. Image Processing Unit

1.2.17.4.1 External Ports

The IPU receives input from the following sources:

- An image sensor (support for two image sensors, no concurrency)
- System memory

IPU output can be sent to the following destinations:

- System memory
- External display controller or graphics accelerator
- A screen—either an LCD or a TV via a digital video encoder

The IPU also provides a communication channel between the processor and an external display controller or graphics accelerator. The IPU uses the following ports for communication:

- Image sensor port connected to one or two sensors
- Display port connected to one or two display devices and, optionally, also to a digital video encoder shared with the External Graphics Accelerator port

- AHB master and slave ports connected to the ARM cross-bar switch
- IP port

1.2.17.4.2 Connectivity to Displays

To handle the diversity of displays supported, the IPU uses two types of display interface: synchronous and asynchronous.

1.2.17.4.3 Synchronous Interface

The synchronous interface is used to transfer a two-dimensional block of pixels to the display, either to display memory or directly to the screen. When the IPU uses this interface, it also sends the display vertical and horizontal synchronization signals to make sure the screen refresh cycle and block transfers are synchronized.

- For a memory-less display (to an LCD device or TV screen), the synchronous interface is used to perform screen refreshes from a display buffer in system memory.
- For a smart display with an integrated controller, the synchronous interface provides a high-speed channel to transfer a rectangular block of pixels to the display and still avoid tearing effects.
- The supported display types are summarized in Table 1-3.

Table 1-3. Display Types Supported by the Synchronous Interface

Display Type	Bus Width [Bits]	Monochrome/Color Mode	Notes
TFT LCD (or smart)	8	up to 256 colors in gray scale	1 bus cycle per pixel
	9	up to 512 colors	
	12	up to 4096 colors	
	16	up to 64 Kbyte colors	
	18	up to 256 Kbyte colors	
	6	up to 256 Kbyte colors	3 bus cycles per pixel
	8	up to 16M colors	
TV encoder	8	up to 16M colors	Format: YUV 4:2:2 Encoder is synchronized as a slave.

1.2.17.4.4 Asynchronous Interface

The asynchronous interface is used to communicate with the integrated controller of a smart display or a graphics accelerator. Communication includes updating the on-display buffer. Unlike the synchronous interface, the asynchronous interface handles these updates without synchronizing them with the screen refresh cycle.

The main features of the asynchronous interface are as follows:

- Interfaces: parallel (18-bit data) and serial
- Operation modes:

— Data transfer (DMA)—Read and write operations between the host system's memory and the external device. These operations can be performed while either the host or the device are bus masters.

— Direct access—Read and write operations of the processor to the external device. The IPU provides the processor with an emulation of the device as a memory-mapped peripheral.

- The serial interface pins can be set to tri-state mode to enable the sharing the display device control with the cellular baseband IC.

1.2.17.4.5 Simultaneous Connectivity

The display port can provide simultaneous connectivity to the following display devices:

- Primary display device—smart or memory-less display or a graphics accelerator
- Secondary display device—a smart display
- Digital video encoder

These devices share most of the port pins as well as parts of the internal interface, so simultaneous use of the devices is limited. Table 1-4 describes availability for each device.

Table 1-4. Simultaneous Functionality in the Dedicated Port

Primary Display Type	Secondary Display Type		
	Smart Display Serial Interface	**Smart Display Asynchronous Parallel Interface**	**TV Screen**
Smart Display Serial Interface	Yes	Yes	Yes
Smart Display Asynchronous Parallel Interface	Yes	Yes	Yes, but access to the smart display is restricted to blanking intervals.
Graphics Accelerator	Yes	Yes, if the accelerator supports a chip select functionality.	NA
Dual Port Smart Display Synchronous + Asynchronous Parallel Interface	Yes	Yes	NA
TV Screen	Yes (No Vsync for smart display)	Yes, but access to the smart display is restricted to blanking intervals.	NA
TFT Memory-Less Display Generic	Yes	Yes, but access to the smart display is restricted to blanking intervals.	NA
TFT Memory-Less Display Sharp HR	NA	Yes, but access to the smart display is restricted to blanking intervals.	NA

1.2.17.4.6 IPU Processing

The IPU processes rectangular blocks of pixels. The block must be part of a progressive (non-interlaced) video or graphics stream. Its format can be either RGB or YUV. The IPU has two processing chains: post-processing and video capturing.

1.2.17.4.7 Post-Processing

- Input from system memory:
  - Frame size is a maximum of 1024 × 1024 pixels.
  - Input rate is a maximum of 9M pixels/s (for example, VGA at 30 fps).
- Processing:
  - Post-filtering
  - Resizing
  - Color space conversion
  - Combining with a graphics plane
  - Inversion and rotation
- Outputs:
  - For a reference frame in H.264 decoding, output is after post-filtering.
  - For display, output is after combining.
- Post-processing is mainly used for a video stream after decompression. Post-processing can also be applied without post-filtering to a graphics block, however.

1.2.17.4.8 Video Capturing

- Input is from an image sensor. (The i.MX31 supports a maximum of two sensors.)
  - Frame size is a maximum of 4096 × 4096 pixels.
  - Input rate is a maximum of 30M pixels/s (for example, UXGA at 15 fps).
- Processing:
  - Resizing
  - Color space conversion
  - Combining with a graphics plane
  - Inversion and rotation
- Outputs:
  - For encoding, output is in YUV format and occurs after resizing.
  - For display, output is in RGB format and occurs after combining, with independent resizing.

The display output is sent either to system memory through the AHB master port or to display memory by using the asynchronous display interface. The preferred destination is display memory, which produces minimal power consumption.

1.2.17.4.9 Processing Stages

The processing stages have the following features:

Post-Filtering

- De-blocking and de-ringing for YUV 4:2:0 format

Resizing

- Fully flexible resizing ratio that is independent for horizontal and vertical resizing
- Maximum downsizing is 16:1 during video recapturing and 8:1 during post-processing.
- Subject to this limitation, any N->M resizing can be performed

Color Space Conversion

- Flexible conversion using a configurable conversion matrix and offsets, in particular the following conversions:
  - YUV <-> RGB, YUV<->YUV conversions
  - Color adjustment

Combining

- Two planes can be combined: for example, you can add a graphics plane on top of a video plane.
- You can specify the transparency of the upper (graphics) plane by using either a key color, a global alpha parameter, or an individual alpha parameter for each pixel.

Inversion and Rotation

- Horizontal and vertical rotation
- Rotation at a 90 degree angle
  - Rotation performed during post-processing, by reading and writing 2D blocks and rotating them internally.
  - If a video frame is received from the sensor, rotation can be performed by first writing it to external memory, then reading it in back in blocks.

1.2.17.4.10 Automatic Procedures

The IPU is equipped with all the control capabilities needed to perform its tasks with minimal ARM involvement and minimal memory use. In particular, the IPU uses the following controls:

- A master AHB port and DMA control functionality and enable the IPU to access system memory autonomously.
- An integrated display controller enables the IPU to synchronize the screen refresh cycle with the internal processing chains and to transfer processed video and graphics data directly to the display. This synchronization helps avoid tearing effects.
- The IPU also features internal synchronization between sensor input and display output.
- The EMI sends the IPU direct indications about changes in system memory buffers (snooping).

The IPU can perform complex procedures automatically without involving the processor—thus saving power. The following list describes some important examples of this type of procedure:

- Screen refresh for memory-less displays.
- Periodic update of the display buffer in a smart display. This can be done conditionally (only when needed) by using the snooping mechanism mentioned previously.
- Viewfinder (a video stream from the image sensor to the display).
- Video capturing (a video stream from the image sensor to memory, for compression).

1.2.17.5 MPEG-4 Video Encoder

The MPEG-4 encoder accelerates video compression, following the MPEG-4 standard.

The encoder has the following main features:

- Compression formats: MPEG-4 simple profile (all levels), H.263 baseline
- Pixel rate: a maximum of VGA at 30 fps
- Compressed bit-rate: a maximum of 4 Mbps
- Essentially performs the complete video processing chain, generating a Huffman-coded stream. Only the formation of the final MPEG-4 stream is performed by the processor.
- Additional processing:
  - Picture smoothening (low-pass filter)
  - Camera movement stabilization
  - Enhanced conference call format, which inserts additional information in the MPEG stream. An MPEG-4 decoder uses the additional information to improve performance.

1.2.18 Audio Interfaces

The i.MX31 and i.MX31L processors have a flexible audio architecture with multiple possibilities for routing narrowband audio (voice) and wideband audio (Hi-Fi).

1.2.18.1 Synchronous Serial Interface or Inter-IC Sound (SSI/I^2S) Module

Depending on system programming, the same module can provide Synchronous Serial interface or Inter-IC Sound (SSI/I^2S) capabilities.

The SSI/I^2S features:

- Generic SSI interface support for an external audio processor
- Support for Philips standard Inter-IC Sound (I^2S) bus for external digital audio processor interface
- Non-integer clock divider for bit rate generation
- Independent transmit and receive sections that operate in master or slave mode
- Operation in normal and network mode
- Support for AC-97 standard

1.2.18.2 Digital Audio MUX

The Digital Audio MUX (AUDMUX) supports various audio applications. The Digital Audio MUX is configured by software. Figure 1-7 shows a schematic diagram of the AUDMUX.

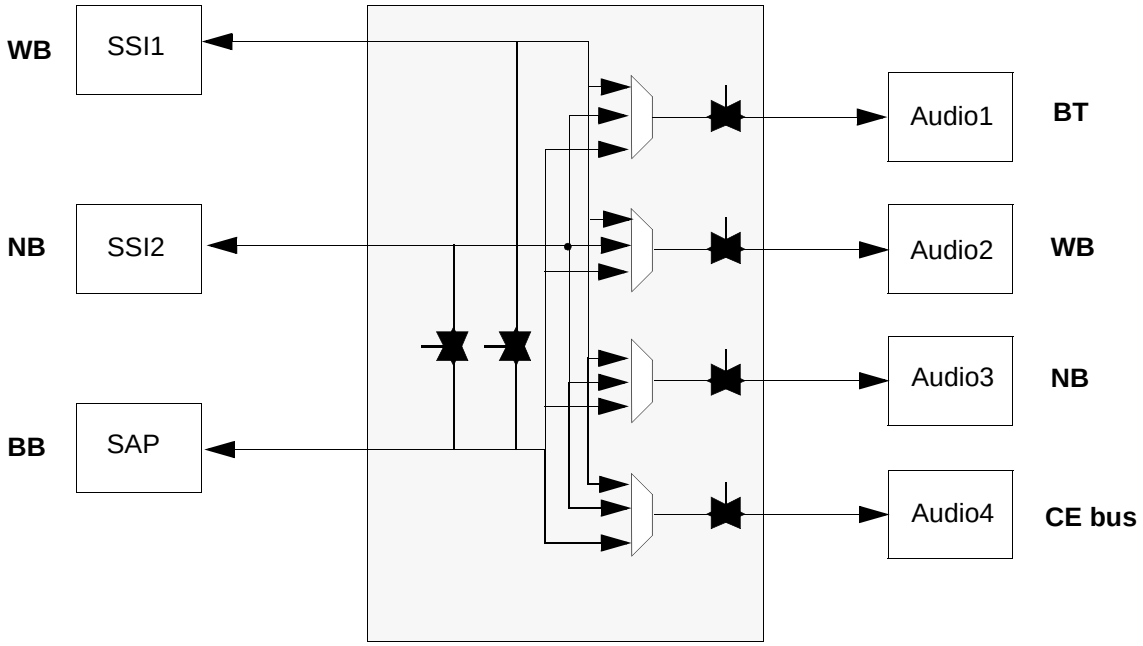


Figure 1-7. Digital Audio MUX (AUDMUX)

1.2.19 Debug Features

The i.MX31 and i.MX31L debug features handle hardware and software debugging and silicon validation, either on evaluation boards, on customer application boards, or even on a closed or opened radio device. Debugging helps identify and isolate causes of failure when you run hardware and software in real applications. The failure source could be in the software or hardware (for example, a race condition).

The i.MX31 and i.MX31L debug hardware also supports system profiling. You can use system profiling to improve overall system performance by identifying optimal system configurations.

Because of the multi-core nature of the i.MX31 and i.MX31L, all internal cores have their own dedicated debug features and ports so you can perform parallel debugging on the processor (ARM11) core and peripherals along with the Smart Direct Memory Access (SDMA) core. The debug architecture of the i.MX31 and i.MX31L encompasses the individual cores' debug components and shared debug components. The individual cores share the following resources:

- The JTAG controller port, which is used to communicate with each of the multiple cores
- The ECT module, which is used to control cross trigger events among the multiple cores

In addition, secure JTAG options are provided to protect debug resources from attacks by unauthorized users. The secure JTAG design prevents the debug architecture from compromising security.

Figure 1-8 shows a block diagram of the i.MX31 and i.MX31L along with the debug-related I/O.

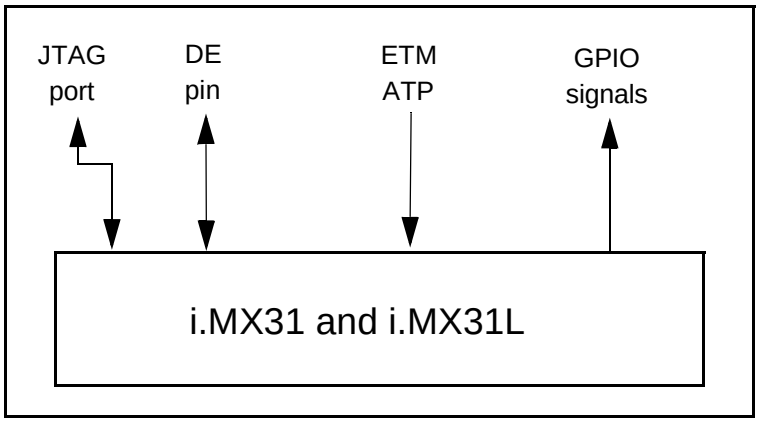


Figure 1-8. i.MX31 and i.MX31L Debug Port Scheme

1.2.19.1 Features

The i.MX31 and i.MX31L provide a full set of features for multi-core debugging. The overall debug system for the i.MX31 and i.MX31L consists of the following elements and features:

- Multi-core debug support is provided by multi-core debuggers via the System JTAG Controller (SJC) and the extensive cross trigger support of the Embedded Cross Trigger (ECT) module.
- Static debug support is provided via the System JTAG Controller (SJC) and appropriate accesses to ICE/OnCE resources on the cores. Support is provided for debug start, stop, single-step, break points, and access to CPU and system resources.
- Non-intrusive real-time instruction and data tracing is supported on the (ETM11) processor.
- ROM patching is supported on the processor. Patching is used to effectively replace data in a ROM memory location. Patching can also enable the software to execute different instructions from the instructions that reside in ROM.
- It is possible to MUX internal signals by using the IOMUX to processor IOs. Critical signals can be routed to the top level SoC pads for external visibility.
- Limited time stamping support in the processor domain is facilitated by using three counters in the PMU of the ARM11 processor, in conjunction with other resources such as the ETM and the ECT module.
- Performance profiling is supported for processor domains: for example, it is possible to count the number of processor stalls, L1 cache hits, L2 cache misses, or external memory accesses that have occurred. Profiling data is accessible by the software and can be used to optimize system configurations for optimal performance.

1.2.20 Boot

The i.MX31 and i.MX31L system boot up is designed according to the configuration of the boot fuses in the IIM and the external BOOT pins. The sequence is therefore divided into Boot-external, Boot-internal, Bootloader, and In-Factory security test modes.

The system boot flow is defined in the i.MX31 processor. The boot flow controls booting, ensures that the system boots up in a predefined secure path, and ensures that the software in the Flash ROM runs on the correct system. This process helps prevent unauthorized modifications to the trusted OS. The process also turns the Flash ROM into tamper-evident memory, protecting it against unauthorized key extraction, hacking program downloads, and viruses.

The boot flow uses security hardware and software components to protect the Flash image and to create a well-bonded secure-based platform for the core and other functional modules. The boot components are shown in Figure 1-9.

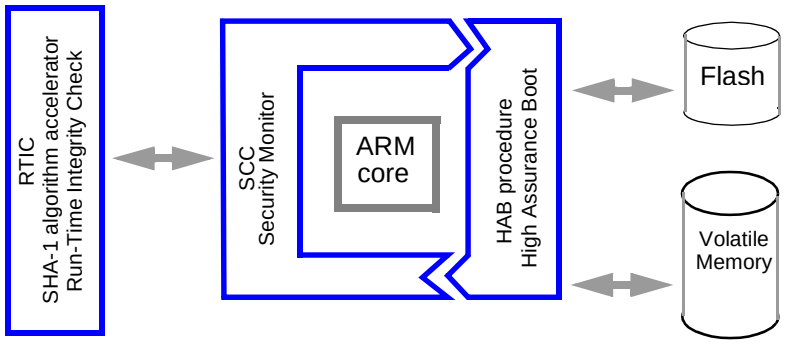


Figure 1-9. Boot Components

The ARM core interfaces with the SCC and operates the HAB procedure. The HAB procedure is stored in Flash memory and uses volatile memory for its operation. During boot, the RTIC is in operation and performs its tasks.

1.2.20.1 Boot Features

The boot flow has the following features:

- The system boots up in a predefined secure path.
- The boot flow prevents unauthorized modifications to the trusted OS by ensuring that software in Flash ROM runs on the correct system.
- Boot action effectively makes Flash ROM tamper-evident memory.
- The boot flow protects against unauthorized access, hacking, and viruses.
- Boot modes are divided into Boot-external, Boot-internal, Bootloader, and In-Factory security test

Chapter 2
System Memory Map, Interrupts, and SDMA Events

This chapter provides information on the memory map, internal ROM and RAM, internal register space, peripheral access types, external memory, interrupts, and SDMA events for the i.MX31 and i.MX31L applications processors.

2.1 Memory Map

The ARM1136JF-S processor's physical memory is mapped according to the addresses shown in Table 2-1.

Table 2-1. Memory Map

Start Address	End Address	Size	Name
0x000 00000	0x0000 3FFF	16 Kbyte	Secure ROM
0x0000 0000	0x0040 3FFF	4 Mbyte–16 Kbyte	Reserved
0x0040 4000	0x0040 7FFF	16 Kbyte	ROM
0x0040 8000	0x0FFF FFFF	252 Mbyte–32 Kbyte	Reserved
0x1000 0000	0x1FFF BFFF	256 Mbyte–16 Kbyte	Reserved for RAM aliasing
0x1FFF C000	0x1FFF FFFF	16 Kbyte	RAM
0x2000 0000	0x2FFF FFFF	256 Mbyte	Reserved
0x3000 0000	0x3FFF FFFF	256 Mbyte	L2CC configuration Registers
0x4000 0000	0x41FF FFFF	32 Mbyte	AIPS A off platform global module enable #0
0x4200 0000	0x43EF FFFF	31 Mbyte	AIPS B off platform global module enable #1
0x43F0 0000	0x43F0 3FFF	16 Kbyte	AIPS A Control registers on platform slot 0
0x43F0 4000	0x43F0 7FFF	16 Kbyte	AIPS A MAX on platform slot 1
0x43F0 8000	0x43F 0BFFF	16 Kbyte	AIPS A EVTMON on platform slot 2
0x43F0 C000	0x43F0 FFFF	16 Kbyte	AIPS A CLKCTL on platform slot 3
0x43F1 0000	0x43F1 3FFF	16 Kbyte	AIPS A ETB registers on platform slot 4
0x43F1 4000	0x43F1 7FFF	16 Kbyte	AIPS A ETB Memory on platform slot 5
0x43F1 8000	0x43F1 BFFF	16 Kbyte	AIPS A ECT CTIO on platform slot 6
0x43F1 C000	0x43F7 FFFF	400 Kbyte	Reserved AIPS A on platform slots
0x43F8 0000	0x43F8 3FFF	16 Kbyte	I2C
0x43F8 4000	0x43F8 7FFF	16 Kbyte	I2C3

Table 2-1. Memory Map (continued)

Start Address	End Address	Size	Name
0x43F8 8000	0x43F8 BFFF	16 Kbyte	USBOTG
0x43F8 C000	0x43F8 FFFF	16 Kbyte	ATA (control port)
0x43F9 0000	0x43F9 3FFF	16 Kbyte	UART1
0x43F9 4000	0x43F9 7FFF	16 Kbyte	UART2
0x43F9 8000	0x43F9 BFFF	16 Kbyte	I2C2
0x43F9 C000	0x43F9 FFFF	16 Kbyte	1-WIRE
0x43FA 0000	0x43FA 3FFF	16 Kbyte	SSI1
0x43FA 4000	0x43FA 7FFF	16 Kbyte	CSPI1
0x43FA 8000	0x43FA BFFF	16 Kbyte	KPP
0x43FA C000	0x43FA FFFF	16 Kbyte	IOMUXC
0x43FB 0000	0x43FB 3FFF	16 Kbyte	UART4
0x43FB 4000	0x43FB 7FFF	16 Kbyte	UART5
0x43FB 8000	0x43FBBFFF	16 Kbyte	ECT (IP BUS 1)
0x43FB C000	0x43FBFFFF	16 Kbyte	ECT (IP BUS 2)
0x43FC 0000	0x43FFFFFF	256 Kbyte	Reserved AIPS A off platform slots
0x44000 000	0x4FFFFFFF	192 Mbyte	Reserved (aliased AIPS A slots)
0x5000 0000	0x5000 3FFF	16 Kbyte	SPBA base address
0x5000 4000	0x5000 7FFF	16 Kbyte	MMC/SDHC1
0x5000 8000	0x5000 BFFF	16 Kbyte	MMC/SDHC2
0x5000 C000	0x5000 FFFF	16 Kbyte	UART3
0x5001 0000	0x500 13FFF	16 Kbyte	CSPI2
0x5001 4000	0x500 17FFF	16 Kbyte	SSI2
0x5001 8000	0x500 1BFFF	16 Kbyte	SIM
0x5001 C000	0x500 1FFFF	16 Kbyte	IIM
0x5002 0000	0x5002 3FFF	16 Kbyte	ATA (DMA port)
0x5002 4000	0x5002 7FFF	16 Kbyte	MSHC1
0x5002 8000	0x5002 BFFF	16 Kbyte	MSHC2
0x5002 C000	0x5002 FFFF	16 Kbyte	Reserved
0x5003 0000	0x5003 3FFF	16 Kbyte	Reserved
0x5003 4000	0x5003 7FFF	16 Kbyte	Reserved
0x5003 8000	0x5003 BFFF	16 Kbyte	Reserved
0x5003 C000	0x5003 FFFF	16 Kbyte	SPBA Registers

Table 2-1. Memory Map (continued)

Start Address	End Address	Size	Name
0x5004 0000	0x51FF FFFF	32 Mbyte-256 Kbyte	Reserved AIPS B off platform global module enable #0
0x5200 0000	0x53EF FFFF	31 Mbyte	AIPS B off platform global module enable #1
0x53F0 0000	0x53F0 3FFF	16 Kbyte	AIPS B Control registers on platform slot 0
0x53F0 4000	0x53F7 FFFF	496 Kbyte	Reserved AIPS B on platform slots
0x53F8 0000	0x53F8 3FFF	16 Kbyte	CCM
0x53F8 4000	0x53F8 7FFF	16 Kbyte	CSPI3
0x53F8 8000	0x53F8 BFFF	16 Kbyte	Reserved
0x53F8 C000	0x53F8 FFFF	16 Kbyte	FIR
0x53F9 0000	0x53F9 3FFF	16 Kbyte	GPT
0x53F9 4000	0x53F9 7FFF	16 Kbyte	EPIT1
0x53F9 8000	0x53F9 BFFF	16 Kbyte	EPIT2
0x53F9 C000	0x53F9 FFFF	16 Kbyte	Reserved
0x53F A0000	0x53FA 3FFF	16 Kbyte	Reserved
0x53FA 4000	0x53FA 7FFF	16 Kbyte	GPIO3
0x53FA 8000	0x53FA BFFF	16 Kbyte	Reserved
0x53FA C000	0x53FA FFFF	16 Kbyte	SCC
0x53FB 0000	0x53FB 3FFF	16 Kbyte	RNGA
0x53FB 4000	0x53FB 7FFF	16 Kbyte	Reserved
0x53FB 8000	0x53FB BFFF	16 Kbyte	Reserved
0x53FB C000	0x53FB FFFF	16 Kbyte	Reserved
0x53FC 0000	0x53FC 3FFF	16 Kbyte	IPU
0x53FC 4000	0x53FC 7FFF	16 Kbyte	AUDMUX
0x53FC 8000	0x53FC BFFF	16 Kbyte	MPEG4_Encoder
0x53FC C000	0x53FC FFFF	16 Kbyte	GPIO1
0x53FD 0000	0x53FD 3FFF	16 Kbyte	GPIO2
0x53FD 4000	0x53FD 7FFF	16 Kbyte	SDMA
0x53FD 8000	0x53FD BFFF	16 Kbyte	RTC
0x53FD C000	0x53FD FFFF	16 Kbyte	WDOG
0x53FE 0000	0x53FE 3FFF	16 Kbyte	PWM
0x53FE 4000	0x53FE 7FFF	16 Kbyte	Reserved
0x53FE 8000	0x53FE BFFF	16 Kbyte	Reserved
0x53FE C000	0x53FE FFFF	16 Kbyte	RTIC

Table 2-1. Memory Map (continued)

Start Address	End Address	Size	Name
0x53FF 0000	0x53FF 3FFF	16 Kbyte	Reserved
0x53FF 4000	0x53FF 7FFF	16 Kbyte	Reserved
0x53FF 8000	0x53FF BFFF	16 Kbyte	Reserved
0x53FF C000	0x53FF FFFF	16 Kbyte	Reserved
0x5400 0000	0x5FFF FFFF	192 Mbyte	Reserved (aliased AIPS B slots)
0x6000 0000	0x67FF FFFF	128 Mbyte	ROMPATCH
0x6800 0000	0x6BFF FFFF	128 Mbyte	AVIC
0x7000 0000	0x7FFF FFFF	256 Mbyte	IPU (MAX M2)
0x8000 0000	0x8FFF FFFF	256 Mbyte	CSD0 SDRAM/DDR
0x9000 0000	0x9FFF FFFF	256 Mbyte	CSD1 SDRAM/DDR
0xA000 0000	0xA7FF FFFF	128 Mbyte	CS0 (Flash) 128 Mbyte
0xA800 0000	0xAFFF FFFF	128 Mbyte	CS1 (Flash) 64 Mbyte
0xB000 0000	0xB1FF FFFF	32 Mbyte	CS2 (SRAM)
0xB200 0000	0xB3FF FFFF	32 Mbyte	CS3 (Spare)
0xB400 0000	0xB5FF FFFF	32 Mbyte	CS4 (Spare)
0xB600 0000	0xB7FF FFFF	32 Mbyte	CS5 (spare)
0xB800 0000	0xB800 0FFF	4 Kbyte	NAND Flash
0xB800 1000	0xB800 1FFF	4 Kbyte	ESDCTL registers
0xB800 2000	0xB800 2FFF	4 Kbyte	WEIM registers
0xB800 3000	0xB800 3FFF	4 Kbyte	M3IF Registers
0xB800 4000	0xB800 4FFF	4 Kbyte	pcmcia_if registers
0xB800 5000	0xBFFF FFFF	128 Mbyte–20 Kbyte	Reserved
0xC000 0000	0xC3FF FFFF	64 Mbyte	PCMCIA/CF
0xC400 0000	0xFFFF FFFF	960 Mbyte	Reserved

2.1.1 Internal RAM

The Internal RAM base address is 0x1FFF C000.

2.1.2 Internal ROM

The Internal ROM is partitioned into two parts, as follows (ROM is not aliased):

- The first 16 Kbytes of Secure ROM is located starting at the lowest part of the memory map, from address 0x0. As such, the ARM processor will start running code from this location after reset.
- The remaining 16 Kbytes are mapped starting at 0x00404000 (4 Mbyte +16 Kbyte).

2.1.3 Internal Register Space

There are various categories of internal registers, all of which are decoded by the two AIPS modules, AIPS A and AIPS B (see Chapter 38, "AHB-Lite 2.v6 to IP Bus Interface (AIPS)").

- Many registers belong to the ARM11 Platform, and others belong to the various off-platform modules.
- Any ARM1136JF-S core write access to off-platform modules will experience two wait states (any write access will last for three cycles).
- Any ARM1136JF-S core read access from these modules will have one wait state (any read access will last for two cycles).

Within each peripheral space, any number of architected registers may be defined (as outlined in the chapter for each peripheral), and software should explicitly address them making no assumptions regarding multiple mapping.

2.1.4 Peripheral Access Types

Table 2-2 shows how different types of accesses are treated by each module.

- Y—The module can be accessed in that access type
- N—The module cannot be accessed in that access type
- O—It is possible to access the peripheral in that access type but there may be unwanted side affects. So, it should be treated as unsupported.

For example, the 1-Wire module has Read/Write 8-bit capability, but does not have Read/Write 16-bit capability; the SSI module has capability for all of these types, but there may be unwanted results.

Table 2-2. Peripheral Access Type

Name	Access Type ->	8 Bit-Access	8 Bit-Access	16-Bit Access	16-Bit Access
	Access Width \/	Read	Write	Read	Write
I2C	16-bit	Y	Y	Y	Y
SIM	16-bit	Y	N	Y	N
UART	32-bit	O	O	Y	Y
1-WIRE	8-bit	Y	Y	N	N
SSI	32-bit	O	O	O	O
CSPI	32-bit	O	O	O	O
FIR	32-bit	N	N	N	N
GPT	32-bit	O	O	O	O
SCC	32-bit	N	N	N	N
RNGA	32-bit	N	N	N	N

Table 2-2. Peripheral Access Type (continued)

Name	Access Type ->	8 Bit-Access	8 Bit-Access	16-Bit Access	16-Bit Access
	Access Width \/	Read	Write	Read	Write
IPU	32-bit	Y	Y	Y	Y
Digital Audio MUX	32-bit	O	O	O	O
GPIO	32-bit	O	O	O	O
RTC	32-bit	Y	Y	Y	Y
Watchdog	16-bit	Y	Y	Y	Y
PWM	32-bit	O	O	O	O
Keypad	16-bit	Y	Y	O	O
USBOTG	32-bit	N	N	N	N

2.1.5 External Memory

There are 896 Mbytes of the memory map allocated for external chip access.

There are nine external chip selects, which are allocated as follows:

- 256 Mbyte for each $\overline{\text{CSD1}}$–$\overline{\text{CSD0}}$
- 32 Mbyte for each $\overline{\text{CS5}}$–$\overline{\text{CS2}}$
- 128 Mbyte for each $\overline{\text{CS1}}$–$\overline{\text{CS2}}$ and
- 64 Mbyte for the PCMCIA/CF

2.1.6 Misaligned Accesses

- The i.MX31supports misaligned accesses to external memories supported by the ESDCTL and WEIM modules.
- The i.MX31 does not support misaligned accesses to NAND Flash and PCMCIA_IF.
- The i.MX31 does not support misaligned access to peripherals residing on the AIPS buses.
- The IPU slave AHB bus supports misaligned accesses.

2.2 Interrupts

2.2.1 Interrupt Operation

The description of the ARM11 Platform Vectored Interrupt Controller can be found in Chapter 9, “ARM1136JF-S Vectored Interrupt Controller (AVIC).”

Each module will send one or more interrupt request to the AVIC.

The existence of an interrupt is reflected in the value of a bit in the Interrupt Pending register. It is the responsibility of the software to determine the underlying reason of the interrupt when servicing it. This is done by reading the status register of the module whose interrupt is being serviced. Once the interrupt has been serviced, and at the end of the interrupt service routine, software explicitly clears the interrupt in accordance with the appropriate value in the status register of the module in question.

2.2.2 Interrupt Summary Table

Table 2-3 shows the interrupts in the i.MX31 and i.MX31L Multimedia Applications Processors.

Each module can send one or more interrupt requests to the ARM11 Platform Vectored Interrupt Controller (AVIC).

While servicing an interrupt, it is the software's responsibility to read the peripheral module's status/interrupt register and determine which event from the peripheral module generated the interrupt.

Table 2-3. Interrupt Summary

Interrupt	Source	Description	Interrupt	Source	Description
0	Reserved	—	32	UART2	OR'ed (rx,tx,mint)
1	Reserved	—	33	NANDFC	NAND Flash Controller
2	Reserved	—	34	SDMA	Smart Direct Memory Access
3	I^2C3	Inter-Integrated Circuit 3	35	USB	Host 1
4	I^2C2	Inter-Integrated Circuit 2	36	USB	Host2
5	MPEG4_Encoder	MPEG-4 Encoder	37	USB	OTG
6	RTIC	Depending on the mode an Interrupt indicates that a HASH error has occurred, or the RTIC has completed hashing.	38	Reserved	—
7	FIR	Fast Infrared Controller	39	MSHC1	Memory Stick Host Controller 1
8	MMC/SDHC2 module	MultiMedia/Secure Data Host Controller 2	40	MSHC2	Memory Stick Host Controller 2
9	MMC/SDHC1	MultiMedia/Secure Data Host Controller 1	41	IPU	Image Processing Unit error
10	I^2C module	MultiMedia/Secure Data Host Controller	42	IPU	IPU general interrupt
11	SSI2 module	Synchronous Serial Interface 1	43	Reserved	
12	SSI1 module	Synchronous Serial Interface 2	44	Reserved	
13	CSPI2 module	Configurable Serial Peripheral Interface 2	45	UART1 Module	OR'ed (rx,tx,mint)

Table 2-3. Interrupt Summary (continued)

Interrupt	Source	Description	Interrupt	Source	Description
14	CSPI1 module	Configurable Serial Peripheral Interface 1	46	UART4 module	OR'ed (rx,tx,mint)
15	ATA controller	HArd Drive (ATA) Controller	47	UART5 module	OR'ed (rx,tx,mint)
16	MBX R-S	Graphic accelerator	48	ect_irq	AND of oct_irq_b[1:0]
17	CSPI3 module	Configurable Serial Peripheral Interface 3	49	SCC module	SCM interrupt
18	UART3 module	OR'ed (rx,tx,mint)	50	SCC module	SMN interrupt
19	IIM module	IC Identification	51	GPIO2 module	General Purpose I/O 2
20	SIM module	Subscriber Identification Module	52	GPIO1 module	General Purpose I/O 1
21	SIM module	Subscriber Identification Module	53	CCM	Clock controller
22	RNGA module	Random Number Generator Accelerator	54	PCMCIA module	
23	EVTMON module	OR of $\overline{\text{evtmon\_interrupt}}$, $\overline{\text{pmu\_irq}}$	55	WDOG module	Watch Dog Timer
24	KPP module	Keyboard Pad Port	56	GPIO3 module	General Purpose I/O 3
25	RTC module	Real Time Clock	57	Reserved	
26	PWM module	Pulse Width Modulator	58	External (power management)	
27	EPIT2 module	Enhanced Periodic Timer 2	59	External (Temper)	
28	EPIT1 module	Enhanced Periodic Timer 1	60	External (sensor)	
29	GPT module	General Purpose Timer	61	External (sensor)	
30	Power fail		62	External (WDOG)	
31	CCM (DVFS)		63	External (TV)	

Table 2-4 lists the interrupt sources that connect directly to the CCM (dsm_wakeup_int[31:0] input). These sources are used to wakeup the i.MX31 and i.MX31L from DSM mode. Sources can be masked in the Wake-Up Interrupt Mask Register (WIMR0) shown on page 3-27.

If the WAMO (bit 10) in CCM Control Register (CCMR) shown on page 3-9 is set, then this mechanism is also used for State Retention (SR) mode. When enabled, interrupts from ARM are ignored.

Table 2-4. Interrupt Sources

Mask bit in WIMRO	Interrupt Source
WIM0	GPIO 3
WIM1	GPIO 2
WIM2	GPIO 1

Table 2-4. Interrupt Sources (continued)

Mask bit in WIMRO	Interrupt Source
WIM3	PCMCIA
WIM4	Watch Dog Timer
WIM5	USB On the Go
WIM6	ipi_int_uh2
WIM7	ipi_int_uh1
WIM8	ipi_int_uart5_anded
WIM9	ipi_int_uart4_anded
WIM10	ipi_int_uart3_anded
WIM11	ipi_int_uart2_anded
WIM12	ipi_int_uart1_anded
WIM13	ipi_int_sim_data_irq
WIM14	ipi_int_sdhc2
WIM15	ipi_int_sdhc1
WIM16	ipi_int_rtc
WIM17	ipi_int_pwm
WIM18	ipi_int_kpp
WIM19	ipi_int_iim
WIM20	ipi_int_gpt
WIM21	ipi_int_firi
WIM22	ipi_int_epit2
WIM23	ipi_int_epit1
WIM24	ipi_int_cspi2
WIM25	ipi_int_cspi1
WIM26	ipp_ind_power_fail
WIM27	ipi_int_cspi3
WIM28	Reserved
WIM29	Reserved
WIM30	Reserved
WIM31	Reserved

2.3 Smart Direct Memory Access (SDMA) Events

Table 2-5 provides a summary of the Smart Direct Memory Access (SDMA) events.

Table 2-5. SDMA Events Summary

Event Number	Module	Description
31	IPU OR ECT	IPU or ECT sources, defaults to IPU at reset.
30	NAND Flash	
29	SSI1 Module	SSI #1 transmit 1 DMA request
28	SSI1 Module	SSI #1 receive 1 DMA request
27	SSI1 Module	SSI #1 transmit 2 DMA request
26	SSI1 Module	SSI #1 receive 2 DMA request
25	SSI2 Module	SSI #2 transmit 1 DMA request
24	SSI2 Module	SSI #2 receive 1 DMA request
23	SSI2 Module	SSI #2 transmit 2 DMA request
22	SSI2 Module	SSI #2 receive 2 DMA request
21	MMC/SDHC2/MSHC 2	MMC/SDHC2 DMA request OR MSHC2
20	MMC/SDHC1/MSHC 1	MMC/SDHC1 DMA request OR MSHC1
19	UART1 Module	TxFIFO
18	UART1 Module	RxFIFO
17	UART2/FIR	TxFIFO of UART2 or DMA request of FIR's transmitter FIFO controlled by the pgp_firi signal from the IOMUXC PGP register.
16	UART2/FIR	RxFIFO of UART2 or DMA request of FIR's receiver FIFO controlled by the pgp_firi signal from the IOMUXC PGP register.
15	EXTDMAREQ1	External DMA request from GPIO1_1
14	EXTDMAREQ2	External DMA request from GPIO1_2 or from MBX (Graphic accelerator)
13	UART4 Module	TxFIFO
12	UART4 Module	RxFIFO
11	UART5 Module/CSPI3	TxFIFO or CSPI3 Tx request
10	UART5 Module/CSPI3	RxFIFOor CSPI3 Rx request
9	CSPI1 Module/UART3 Module	DMA Tx request of CSPI/TxFIFO of UART3
8	CSPI1 Module/UART3 Module	DMA Rx request of CSPI/RxFIFO of UART3

Table 2-5. SDMA Events Summary (continued)

Event Number	Module	Description
7	CSPI2 Module	DMA Tx request
6	CSPI2 Module	DMA Rx request
5	SIM module	
4	ATA	ata_rcv_fifo_alarm
3	ATA	ata_tx_fifo_alarm
2	ATA	ata_txfer_end_alarm
1	CCM	DVFS/DPTC event (ccm_dvfs_sdma_int)
0	EXTDMAREQ0	External DMA request from GPIO1_0

Chapter 3
Clocks, Power Management and Reset (AP Clock Controller Module)

The Clock Controller Module (CCM) controls the system frequency, distributes clocks to various parts of the chip, controls the reset mechanism of the chip, and provides an advanced low-power management capability for the i.MX31 and i.MX31L processors.

3.1 Overview

The CCM includes these distinctive features:

- Frequency Pre-Multiplier (FPM) and PLLs control
- Clock distributions—division of PLLs output clock and clock source selectors
- Reset Controller—generate reset signals to the core and to the peripherals
- CCM registers are accessible via IP bus
- Power manager controls power modes and special techniques, such as AWB, power gating, DPTC.

3.2 PLLs

The i.MX31 and i.MX31L processors have three Digital PLLs (DPLLs) in the system that generate three separate clock frequencies from the PLL reference clock. The PLL reference clock, in turn, can be generated either from an external high frequency source (CKIH), or from a low frequency source that has been passed through a Frequency Pre-Multiplier (FPM).

The DPLLs and the choice of specific clock source is accomplished by programming the PLL registers, which are part of the CCM registers.

3.2.1 PLL Reference Clock Sources

The PLL can have one of two clock sources: an external high frequency source (CKIH), or a low frequency source that is passed through a Frequency Pre-Multiplier (FPM).

3.2.1.1 External High Frequency Clock—CKIH

One of the potential input clocks to the CCM is an external high frequency clock that can be connected to the CKIH pin. It can be used as one of sources for the PLL reference clock.

3.2.1.2 Frequency Pre-Multiplier (FPM)

In case of a low-frequency input clock, the required Multiplication Factor (MF) becomes extremely large (~3000). Such requirement is non-trivial for the PLL and the modern Digital PLL does not support it. Configuring the Analog PLL with such a large MF may result in unstable output clocks, very large frequency jitters (up to 5–10%), and very long lock time (~2000 CKIL clock periods).

The Digital PLL will be used with an additional fully digital Pre-Multiplier (FPM). The FPM creates a relatively stable output clock with constant frequency that equals 1024*(CKIL frequency).

The FPM therefore can be used as one of the sources for the PLL reference clock.

3.2.1.3 PLL Reference Clock Switch Unit

The PLL reference clock is pll_ref_clk and is generated by the PLL reference clock switch unit. See Figure 3-1 for a schematic diagram of this unit. Section 3.2.1.1, “External High Frequency Clock—CKIH” and Section 3.2.1.2, “Frequency Pre-Multiplier (FPM)” provide information on two clocks (CKIH and FPM) that function as possible clock sources. CKIH is always the selected clock source in PLL bypass test mode. Selection is accomplished between CKIH and FPM by external pin signal ipp_clkss during reset.

Selection between the two sources can be done in S/W by setting the PRCS bits in the Control Register (CCMR). When a different clock source is selected, the new clock source will be automatically enabled. Only after it becomes available will switching be done. If CKIH is selected, the value of the OSCNT bits may be updated with the delay time that CKIH will be available. After switching is done, the previously selected clock source will be automatically disabled.

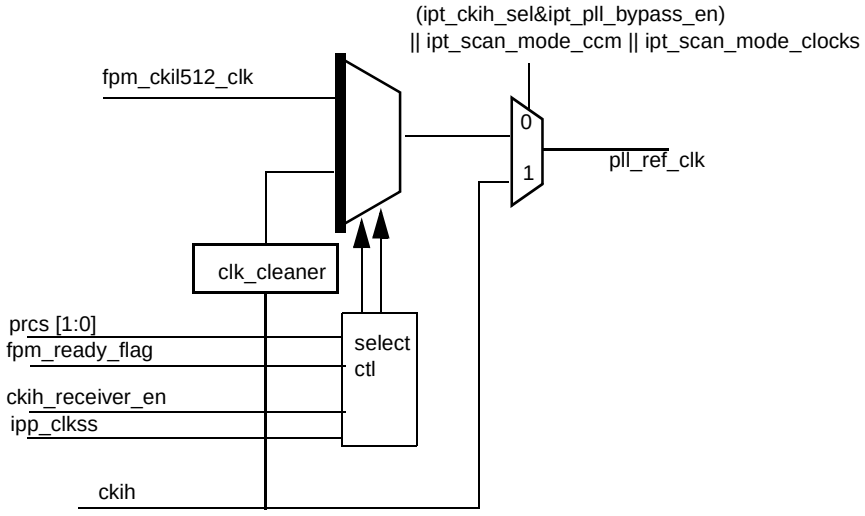


Figure 3-1. PLL Reference Clock Switch Unit

3.2.2 High Frequency Clock Source

There are three DPLLs in the system that generate three separate clock frequencies from the PLL reference clock.

The MCU PLL, configured by the MPCTL register, produces the mcu_main _clk clock. The MCU clock sub-domain is generated from that clock.

The USB PLL, configured by the UPCTL register, produces the clock for the USB circuitry (60 MHz). This clock can be used for FIR. When the clock is used for FIR it can be programmed to be 46 MHz (MIR mode) or 60 MHz (FIR mode). Since the USB requires a 60-MHz clock and FIR (when in MIR mode) requires a 46 MHz clock, it is impossible to use the USB_PLL for MIR and the USB at the same time.

3.3 CCM

The CCM controls the system frequency, distributes clocks to various parts of the chip, controls the reset mechanism of the chip, and provides an advanced low-power management capability for the i.MX31 and i.MX31L.

3.3.1 Features

The CCM includes the following distinctive features:

- Frequency Pre-Multiplier (FPM) and PLLs control
- Clocks' distributions—Division of PLLs output clock and clock source selectors
- Reset Controller—Generate reset signals to the core and to the peripherals
- CCM registers are accessible via the IP bus.
- Power manager controls power modes and special techniques, such as AWB, power gating, DPTC.

3.3.2 External Signal Description

The CKO pin can be used to output internal clock signals. It is controlled by the COSR register in the CCM. See Table 3-11 for a detailed description of the COSR settings.

3.4 Register Definition and Memory Map

The CCM has 28 user accessible 32-bit registers used to configure, operate and monitor the state of the CCM. Section 3.4.3, "Register Descriptions" provides the detailed descriptions for all of the CCM registers.

3.4.1 Memory Map

Table 3-1 shows the CCM memory map.

Table 3-1. CCM Memory Map

Address	Register	Access	Reset Value	Section/Page
0x53F8_0000 (CCMR)	Control Register (CCMR)	R/W	0x074B_0B79	3.4.3.1/3-9
0x53F8_0004 (PDR0)	Post Divider Register 0 (PDR0)	R/W	0xFF87_0B48	3.4.3.2/3-12

Table 3-1. CCM Memory Map (continued)

Address	Register	Access	Reset Value	Section/Page
0x53F8_0008 (PDR1)	Post Divider Register 1 (PDR1)	R/W	0x49FC_FE7F	3.4.3.3/3-13
0x53F8_ 000C (RCSR)	Reset Control and Source Register (RCSR)	R/W	0x007F_0000	3.4.3.4/3-15
0x53F8_ 0010 (MPCTL)	MCU PLL Control Register (MPCTL)	R/W	0x0400_1800	3.4.3.5/3-16
0x53F8_ 0014 (UPCTL)	USB PLL Control Register (UPCTL)	R/W	0x0405_1C03	3.4.3.6/3-18
0x53F8_ 0018 (SPCTL)	Serial PLL Control Register (SPCTL)	R/W	0x0404_3001	3.4.3.7/3-20
0x53F8_ 001C (COSR)	Clock Out Source Register (COSR)	R/W	0x0000_0280	3.4.3.8/3-22
0x53F8_ 0020 (CGR0)	Clock Gating Register 0 (CGR0)	R/W	0xFFFF_FFFF	3.4.3.9/3-24
0x53F8_ 0024 (CGR1)	Clock Gating Register 1 (CGR1)	R/W	0xFFFF_FFFF	3.4.3.9/3-24
0x53F8_ 0028 (CGR2)	Clock Gating Register 2 (CGR2)	R/W	0xFFFF_FFFF	3.4.3.9/3-24
0x53F8_ 002C (WIMR0)	Wake-up Interrupt Mask Register (WIMR)	R/W	0xFFFF_FFFF	3.4.3.10/3-27
0x53F8_ 0030 (LDC)	Latch Divergence Counter Register (LDC)	R/W	0x0000_0000	3.4.3.11/3-27
0x53F8_ 0034 (DCVR0)	DPTC Comparator Value Register 0 (DCVR0)	R/W	0x0000_0000	3.4.3.12/3-28
0x53F8_ 0038 (DCVR1)	DPTC Comparator Value Register 1 (DCVR1)	R/W	0x0000_0000	3.4.3.12/3-28
0x53F8_ 003C (DCVR2)	DPTC Comparator Value Register 2 (DCVR2)	R/W	0x0000_0000	3.4.3.12/3-28
0x53F8_ 0040 (DCVR3)	DPTC Comparator Value Register 3 (DCVR3)	R/W	0x0000_0000	3.4.3.12/3-28
0x53F8_ 0044 (LTR0)	Load Tracking Register 0 (LTR0)	R/W	0x0000_0000	3.4.3.13/3-29
0x53F8_ 0048 (LTR1)	Load Tracking Register 1 (LTR1)	R/W	0x0000_4040	3.4.3.14/3-30
0x53F8_ 004C (LTR2)	Load Tracking Register 2 (LTR2)	R/W	0x0000_0000	3.4.3.15/3-31
0x53F8_ 0050 (LTR3)	Load Tracking Register 3 (LTR3)	R/W	0x0000_0000	3.4.3.16/3-32
0x53F8_ 0054 (LTBR0)	Load Tracking Buffer Register 0 (LTBR0)	R	0x0000_0000	3.4.3.17/3-33

Table 3-1. CCM Memory Map (continued)

Address	Register	Access	Reset Value	Section/Page
0x53F8_ 0058 (LTBR1)	Load Tracking Buffer Register 1 (LTBR1)	R	0x0000_0000	3.4.3.18/3-34
0x53F8_ 005C (PMCR0)	Power Management Control Register 0 (PMCR0)	R/W	0x8020_9828	3.4.3.19/3-35
0x53F8_ 0060 (PMCR1)	Power Management Control Register 1 (PMCR1)	R/W	0x00AA_0000	3.4.3.20/3-38
0x53F8_ 0064 (PDR2)	Post Divider Register 2 (PDR2)	R/W	0x0000_0285	3.4.3.21/3-39

3.4.2 Register Summary

Figure 3-2 shows the key to the register fields and Table 3-2 shows the register figure conventions.

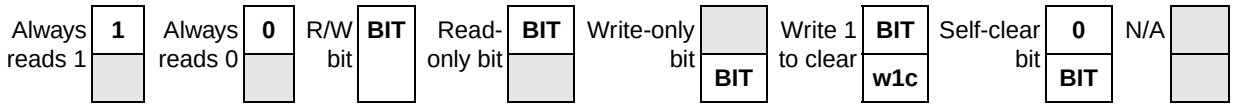


Figure 3-2. Key to Register Fields

Table 3-2. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 3-3 shows the CCM register summary.

Table 3-3. CCM Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x53F8_0000 (CCMR)	R			L2PG	VSTBY	WBEN	FPMF	CSCS	PERCS	0	SSI2S		0	SSI1S		RAMW	
	W																
	R	LPM		0	FIRS		WAMO	UPE	SPE	MDS	ROMW		SBYCS	MPE	PRCS		FPME
	W																
0x53F8_0004 (PDR0)	R	CSI_PODF									0	0	PER_PODF				
	W																
	R	0	0	HSP_PODF			NFC_PODF			IPG_PODF		MAX_PODF		MCU_PODF			
	W																
0x53F8_0008 (PDR1)	R	USB_PRDF		USB_PODF			FIRI_PRE_PODF			FIRI_PODF						SSI2_PRE_PODF[2:1]	
	W																
	R	SSI2_PRE_PODF[0]	SSI2_PODF					SSI1_PRE_PODF		SSI1_PODF							
	W																
0x53F8_ 000C (RCSR)	R	NF16B	NFMS	0	0	BTP4	BTP3	BTP2	BTP1	BTP0	OSCNT						
	W																
	R	PERES	0	SDM		0	0	0	0	GPF			WFIS	0	REST		
	W																
0x53F8_ 0010 (MPCTL)	R	BRMO	0	PD				MFD									
	W																
	R	0	0	MFI				MFN									
	W																
0x53F8_ 0014 (UPCTL)	R	BRMO	0	PD				MFD									
	W																
	R	0	0	MFI				MFN									
	W																
0x53F8_ 0018 (SPCTL)	R	BRMO	0	PD				MFD									
	W																
	R	0	0	MFI				MFN									
	W																

Table 3-3. CCM Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x53F8_ 001C (COSR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	CLK OEN	CLKOUTDIV					CLKOSEL			
	W																
0x53F8_ 0020 (CGR0)	R	CG15		CG14		CG13		CG12		CG11		CG10		CG9		CG8	
	W																
	R	CG7		CG6		CG5		CG4		CG3		CG2		CG1		CG0	
	W																
0x53F8_ 0024 (CGR1)	R	CG15		CG14		CG13		CG12		CG11		CG10		CG9		CG8	
	W																
	R	CG7		CG6		CG5		CG4		CG3		CG2		CG1		CG0	
	W																
0x53F8_ 0028 (CGR2)	R	CG15		CG14		CG13		CG12		CG11		CG10		CG9		CG8	
	W																
	R	CG7		CG6		CG5		CG4		CG3		CG2		CG1		CG0	
	W																
0x53F8_ 002C (WIMR0)	R	WIM 31	WIM 30	WIM 29	WIM 28	WIM 27	WIM 26	WIM 25	WIM 24	WIM 23	WIM 22	WIM 21	WIM 20	WIM 19	WIM 18	WIM 17	WIM 16
	W																
	R	WIM 15	WIM 14	WIM 13	WIM 12	WIM 11	WIM 10	WIM 9	WIM 8	WIM 7	WIM 6	WIM 5	WIM 4	WIM 3	WIM 2	WIM 1	WIM 0
	W																
0x53F8_ 0030 (LDC)	R	LDC 31	LDC 30	LDC 29	LDC 28	LDC 27	LDC 26	LDC 25	LDC 24	LDC 23	LDC 22	LDC 21	LDC 20	LDC 19	LDC 18	LDC 17	LDC 16
	W																
	R	LDC 15	LDC 14	LDC 13	LDC 12	LDC 11	LDC 10	LDC 9	LDC 8	LDC 7	LDC 6	LDC 5	LDC 4	LDC 3	LDC 2	LDC 1	LDC 0
	W																
0x53F8_ 0034 (DCVR0)	R	ULV										LLV					
	W																
	R	LLV				ELV											
	W																
0x53F8_ 0038 (DCVR1)	R	ULV										LLV					
	W																
	R	LLV				ELV											
	W																

Table 3-3. CCM Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x53F8_ 003C (DCVR2)	R	ULV										LLV					
	W																
	R	LLV				ELV											
	W																
0x53F8_ 0040 (DCVR3)	R	ULV										LLV					
	W																
	R	LLV				ELV											
	W																
0x53F8_ 0044 (LTR0)	R	SIGD 15	SIGD 14	SIG D13	0	UPTHR						DNTHR					
	W																
	R	SIGD 12	SIGD 11	SIG D10	SIG D9	SIG D8	SIG D7	SIG D6	SIG D5	SIG D4	SIG D3	SIG D2	SIG D1	SIG D0	DIV3CK		0
	W																
0x53F8_ 0048 (LTR1)	R	SIGD 15	SIGD 14	SIG D13	0	UPTHR						DNTHR					
	W																
	R	SIGD 12	SIGD 11	SIG D10	SIG D9	SIG D8	SIG D7	SIG D6	SIG D5	SIG D4	SIG D3	SIG D2	SIG D1	SIG D0	DIV3CK		0
	W																
0x53F8_ 004C (LTR2)	R	SIGD 15	SIGD 14	SIG D13	0	UPTHR						DNTHR					
	W																
	R	SIGD 12	SIGD 11	SIG D10	SIG D9	SIG D8	SIG D7	SIG D6	SIG D5	SIG D4	SIG D3	SIG D2	SIG D1	SIG D0	DIV3CK		0
	W																
0x53F8_ 0050 (LTR3)	R	SIGD 15	SIGD 14	SIG D13	0	UPTHR						DNTHR					
	W																
	R	SIGD 12	SIGD 11	SIG D10	SIG D9	SIG D8	SIG D7	SIG D6	SIG D5	SIG D4	SIG D3	SIG D2	SIG D1	SIG D0	DIV3CK		0
	W																
0x53F8_ 0054 (LTBR0)	R	LTS7				LTS6				LTS5				LTS4			
	W																
	R	LTS3				LTS2				LTS1				LTS0			
	W																
0x53F8_ 0058 (LTBR1)	R	LTS15				LTS14				LTS13				LTS12			
	W																
	R	LTS11				LTS10				LTS9				LTS8			
	W																

Table 3-3. CCM Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x53F8_ 005C (PMCR0)	R	DFSUP		DVSUP		UDSC	VSCNT			DVFEV	DVFIS	LBMI	LBFL	LBCF		PTVIS	UPDTEN
	W																
	R	FSVAIM	FSVAI		DPVCR	DPVV	WFIM	DRCE3	DRCE2	DRCE1	DRCE0	DCR	DVFEN	PTVAIM	PTVAI		DPTEN
	W																
0x53F8_ 0060 (PMCR1)	R	0	0	0	0	0	0	0	0	WBCN							
	W																
	R	0	0	0	CPSPA				PWTS	NWTS	CPFA	0	0	DVGP			
	W																
0x53F8_ 0064 (PDR2)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	MST2_PDF							MST1_PDF					
	W																

3.4.3 Register Descriptions

This section contains the detailed register descriptions for the CCM registers.

3.4.3.1 Control Register (CCMR)

Figure 3-3 shows the register; Table 3-4 provides the register's field descriptions.

0x53F8_0000 (CCMR) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			L2PG	VSTBY	WBEN	FPMF	CSCS	PERCS	0	SSI2S		0	SSI1S		RAMW	
W																
Reset	0	0	0	0	0	1	1	1	0	1	0	0	1	0	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LPM		0	FIRS		WAMO	UPE	SPE	MDS	ROMW		SBYCS	MPE	PRCS[1]		FPME
W																
Reset	0	0	0	0	1	0	1	1	0	1	1	1	1	0	0	1

Figure 3-3. Control Register (CCMR)

Table 3-4. CCMR Field Descriptions

Field	Description
31–30	Reserved
29 L2PG	L2 cache power gating enable. The MCU writes to this bit to indicate that supply of L2 cache should be disabled when the MCU enters State Retention mode. 0 L2 cache supply is not to be shutdown when MCU enters State Retention mode. 1 L2 cache supply is to be shutdown when MCU enters State Retention mode.
28 VSTBY	Supply regulator standby mode enable. The MCU writes to this bit to indicate that the supply regulator should be placed in Standby mode when the MCU enters State Retention or Deep Sleep modes. 0 Supply regulator will not be put in standby mode when the MCU enters State Retention or Deep Sleep modes. 1 Supply regulator will be put in Standby mode when the MCU enters State Retention or Deep Sleep modes.
27 WBEN	ARM domain well-bias enable bit. This bit enables the well-biasing of modules in the ARM11P. Well-biasing can be activated only after the MCU has entered Standby mode and max_halted signal is asserted. If the MCU entered Standby mode and this bit is set, the well-biasing mechanism is enabled. 0 Well-bias is disabled. 1 Well-bias is enabled.
26 FPMF	FPM multiplying factor. 1 = 1024
25 CSCS	ipg_clk_csi_baud clock generation source. This clock can be generated from SRPLL or USB clock domain sources 0 USB clock domain source is selected 1 SRPLL clock domain source is selected
24 PERCS	ipg_per_clk clock generation source. This clock can be generated from ipg_clk or USB clock domain source 0 USB clock domain source is selected 1 ipg_clk is selected
23	Reserved
22–21 SSI2S	SSI2 post divider clock source select 00 MCU CLK 01 usb_clk 10 serial_clk (default) 11 reserved
20	Reserved
19–18 SSI1S	SSI1 post divider clock source select 00 mcu clk 01 usb_clk 10 serial_clk (default) 11 Reserved
17–16 RAMW	wait state control bits for ARM RAM 00 0 Wait states for both ARM and alternate masters 01 0 Wait states and 1 wait state for alternate masters—not recommended for i.MX31 uses. 10 1 Wait state for ARM and 0 wait states for alternate masters—not recommended for the i.MX31 and i.MX31L uses. 11 1 Wait state for both ARM and alternate masters.

Table 3-4. CCMR Field Descriptions (continued)

Field	Description
15–14 LPM	These bits define which Low Power Mode the i.MX31 and i.MX31L will enter when the WFI command is next executed by the MCU. 00 Wait mode 01 Doze mode 10 State retention mode 11 Deep sleep mode
13	Reserved
12–11 FIRS	FIR post divider clock source select 00 mcu clk 01 usb_clk 10 serial_clk 11 Reserved
10 WAMO	Wakeup interrupt mask. 0 Masked in all modes but DSM 1 Masked in all modes but SR or DSM
9 UPE	USB PLL enable bit. This bit enables/disables the USB PLL. This bit cannot be set when selected clock source is disabled. 0 USB PLL disabled. 1 USB PLL enabled.
8 SPE	Serial PLL enable bit. This bit enables/disables the Serial PLL. This bit cannot be set when selected clock source is disabled. 0 Serial PLL disabled. 1 Serial PLL enabled.
7 MDS	MCU clock Domain source select. This bit selects the source of the MCU domain post divider. If the MCU PLL is disabled in run mode, the MCU PLL cannot be the MCU clock domain source. The default MCU clock domain source is MCU PLL. 0 MCU PLL is the MCU clock domain source 1 Reference clock is the MCU clock domain source (bypass)
6–5 ROMW	Wait state control bit from ARM ROM 00 0 Wait states for both ARM and alternate masters 01 0 Wait states and 1 wait state for alternate masters—not recommended for the i.MX31 and i.MX31L uses. 10 1 Wait state for ARM and 0 wait states for alternate masters—not recommended for the i.MX31 and i.MX31L uses. 11 1 Wait state for both ARM and alternate masters.
4 SBYCS	Clock source enable in Standby mode. This bit determines whether or not the selected clock source for the MCU clock domain will be disabled in Standby mode. This is applied to all reference clocks involved in the specific clock generation. To enter Deep Sleep Mode (see Section 3.5.2, “Power Modes”), all other clock sources should be disabled by software before the system enters Standby mode. This ensures proper interrupt command execution. 0 Clock source is disabled in Standby mode. 1 Clock source is enabled in Standby mode.
3 MPE	MCU PLL enable. This bit enables/disables the MCU PLL. If MCU PLL is disabled, MCU clock domain source will be automatically switched to pll_ref_clk. When MCU PLL is re-enabled, the MCU clock domain source will be automatically switched to the MCU PLL output, after which MCU PLL will be re-locked. 0 MCU PLL is disabled 1 MCU PLL is enabled.

Table 3-4. CCMR Field Descriptions (continued)

Field	Description
2–1 PRCS	PLL reference clock select bits. These bits select the reference clock of all PLLs. This bit can be modified only when the MCU PLL is disabled. When a reference clock source is changed, the relevant clock source will be automatically enabled, and only after it is available, clock sources will be switched to the new source. After that the other source will be disabled. The default value of these bits is defined by external signal ipp_clkss. 00 Reserved 01 FPM 10 CKIH 11 Reserved
0 FPME	FPM enable bit. This bit defines if the FPM will be enabled. This applies even if it was not selected as reference clock source. 0 FPM is disabled. 1 FPM is enabled.

3.4.3.2 Post Divider Register 0 (PDR0)

Figure 3-4 shows the register; Table 3-5 provides the register's field descriptions.

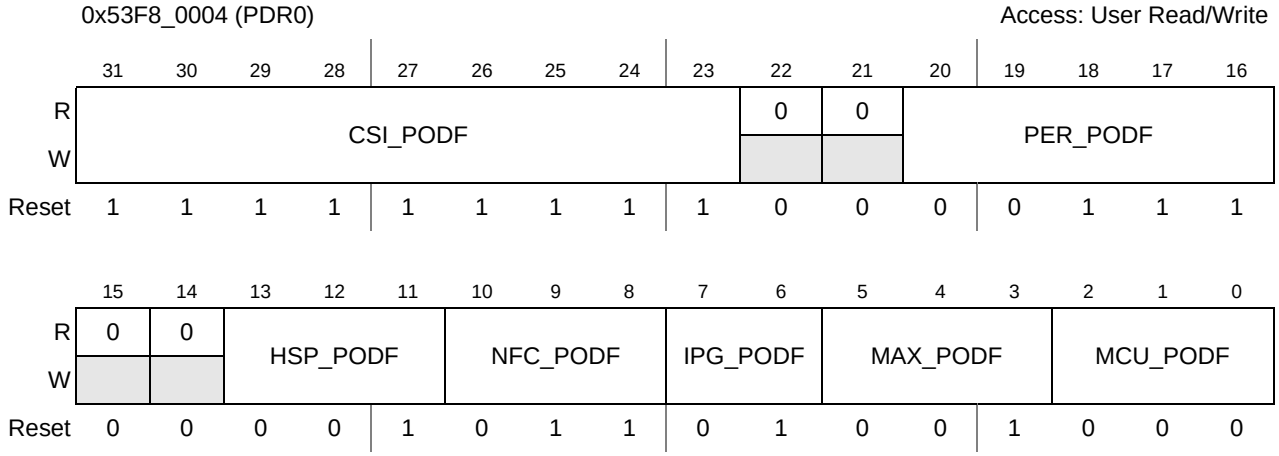


Figure 3-4. Post Divider Register 0 (PDR0)

Table 3-5. PDR0 Field Descriptions

Field	Description
31–23 CSI_PODF	These bits control the CSI post divider. See Figure 3-24, which shows a diagram of the role the CSI post divider plays. 000000000 Divide by 1 000000001 Divide by 2 —— 111111111 Divide by 512
22–21	Reserved

Table 3-5. PDR0 Field Descriptions (continued)

Field	Description
20–16 PER_PODF	These bits control the per post-divider. See Figure 3-24, which shows a diagram of the role the per post divider plays 00000Divide by 1 00001Divide by 2 —— 11111Divide by 32
15–14	Reserved
13–11 HSP_PODF	These bits control the hsp post divider, which plays a part in generating the IPU high speed clock. See Figure 3-24. **Warning**: Before changing the post divider of the IPU high speed clock, first ensure that the IPU itself has been enabled because the Clock Controller Module waits for an acknowledge from the IPU. To enable the IPU, simply set the DI_EN bit (bit 6) in the IPU_CONF register. Refer to Chapter 44, “Image Processing Unit (IPU) for more details. 000 Divide by 1 001 Divide by 2 —— 111 Divide by 8
10–8 NFC_PODF	These bits control the NFC post divider of the nfc_clk (nfc divider). See Figure 3-24. 000 Divide by 1 001 Divide by 2 —— 111 Divide by 8
7–6 IPG_PODF	These bits control the peripheral clock post divider. See Figure 3-24. 00 Divide by 1 01 Divide by 2 10 Divide by 3 11 Divide by 4
5–3 MAX_PODF	These bits control the hclk post divider. See Figure 3-24. 000 Divide by 1 001 Divide by 2 —— 111 Divide by 8
2–0 MCU_PODF	These bits control the MCU post divider. See Figure 3-24. 000 Divide by 1 001 Divide by 2 —— 111 Divide by 8

3.4.3.3 Post Divider Register 1 (PDR1)

Figure 3-5 shows the register; Table 3-6 provides the register’s field descriptions.

0x53F8_0008 (PDR1) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	USB_PRDF		USB_PODF			FIRI_PRE_PODF			FIRI_PODF						SSI2_PRE_PODF[2:1]	
Reset	0	1	0	0	1	0	0	1	1	1	1	1	1	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SSI2_PRE_PODF[0]		SSI2_PODF						SSI1_PRE_PODF		SSI1_PODF					
Reset	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1

Figure 3-5. Post Divider Register 1 (PDR1)

Table 3-6. PDR1 Field Descriptions

Field	Description
31–30 USB_PRDF	These bits control the USB pre divider. See Figure 3-24. 00 Divide by 1 —— 11 Divide by 4
29–27 USB_PODF	These bits control the USB post divider. See Figure 3-24. 000 Divide by 1 —— 111 Divide by 8
26–24 FIRI_PRE_PODF	These bits control the FIR pre divider. See Figure 3-24. 000 Divide by 1 —— 111 Divide by 8
23–18 FIRI_PODF	These bits control the FIR post divider. See Figure 3-24. 000000 Divide by 1 —— 111111 Divide by 64
17–15 SSI2_PRE_PODF	These bits control the SSI2 pre divider. See Figure 3-24. 000 Divide by 1 —— 111 Divide by 8
14–9 SSI2_PODF	These bits control the SSI2 post divider. See Figure 3-24. 000000 Divide by 1 —— 111111 Divide by 64

Table 3-6. PDR1 Field Descriptions (continued)

Field	Description
8–6 SSI1_PRE_PODF	These bits control the SSI1 pre divider. See Figure 3-24. 000 Divide by 1 —— 111 Divide by 8
5–0 SSI1_PODF	These bits control the SSI2 post divider. See Figure 3-24. 000000 Divide by 1 —— 111111 Divide by 64

3.4.3.4 Reset Control and Source Register (RCSR)

Figure 3-6 shows the register; Table 3-7 provides the register's field descriptions.

0x53F8_ 000C (RCSR) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	NF16B	NFMS	0	0	BTP4	BTP3	BTP2	BTP1	BTP0	OSCNT						
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PERES	0	SDM		0	0	0	0	GPF			WFIS	0	REST		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 3-6. Reset Control and Source Register (RCSR)

Table 3-7. RCSR Field Descriptions

Field	Description
31 NF16B	This bit defines bit select for NAND Flash device on the EMI. 0 8 bit 1 16 bit
30 NFMS	This bit defines page size for NAND Flash device. Its default (reset) value is defined by decoding the boot mode pins. 0 512-byte page size 1 2 Kbyte page size
29–28	Reserved
27–23 BTPn [4–0]	These read only bits reflect the value of the boot pins, and are sampled upon reset negation. See Chapter 7, "i.MX31 and i.MX31L Boot."

Table 3-7. RCSR Field Descriptions (continued)

Field	Description
22–16 OSCNT	Oscillator ready counter value. These bits define the value of 32 KHz counter. This counter serves as the counter for the external high speed clock oscillator lock time. 0000000Count 1 cycle —— 1111111Count 128 cycles
15 PERES	Peripheral S/W reset bit. Writing “1” to this bit will result in a reset of all peripherals on MCU side, except CCM and PLLs. 0 MCU peripherals are not in the process of being reset. 1 MCU peripherals are in the process of being reset.
14	Reserved
13–12 SDM	Scan divergence mode bits. 00 Disable scan divergence mode 01 Clocks to the MCU will be stopped one cycle after the counter finishes counting. The LDC register value will not change 10 Clocks to the MCU will be stopped after counter finishes counting and value of LDC register will be decremented by 1. 11 Clocks to the MCU will be stopped two cycles after the counter finishes counting and value of LDC register will be incremented by 1.
11–8	Reserved
7–5 GPF	General purpose Flag Bits. This bits are reset by the $\overline{\text{reset\_in\_por}}$ input pin. This bits can be used as warm start parameters or for debug purposes. For example, the user can write a value to these bits before turning off the voltage. This value can then be used by the wake up routine for the wake up scenario purposes.
4 WFIS	WFI Pending SW control bit 0 No software control 1 SW control to emulate WFI pending behavior.
3	Reserved
2–0 REST	Reset status bits. Shows what caused the most recent reset to the system. If several sources’ signals overlap and if the signals are released during the same CLK32 cycle (which also causes the assertion of the RESET_OUT signal), only the highest–priority event is registered by the REST using the following priority order: • POR external reset signal • Qualified external reset signal • Watchdog signal • S/W reset Otherwise, the last signal that is released is honored. 000 POR external reset 001 Qualified external reset 010 Watchdog timeout 011 Reserved 100 Reserved 101 Reserved 110 JTAG (s/w or IEEE) reset 111 ARM11P power gating

3.4.3.5 MCU PLL Control Register (MPCTL)

Figure 3-7 shows the register; Table 3-8 provides the register’s field descriptions.

0x53F8_ 0010 (MPCTL) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BRMO	0	PD				MFD									
W																
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	MFI				MFN									
W																
Reset	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0

Figure 3-7. MCU PLL Control Register (MPCTL)

Table 3-8. MPCTL Field Descriptions

Field	Description
31 BRMO	BRM Order bit. Determines if the BRM order is first order or second order. The first order BRM is used if a MF fractional part is both more than 1/10 and less than 9/10. In other cases, the second order BRM is used. The BRMO bit is cleared by a hardware reset. 0 BRM order is first order. 1 BRM order is second order.
30	Reserved
29–26 PD	Pre-divider Factor bits define the pre-divider factor (PD) applied to the PLL input frequency. See Equation 3-1. PD is an integer between 1and 16 (inclusive). PD is chosen to ensure that the resulting output frequency remains within the specified range. When a new value is written into PD bits, the PLL loses its lock; after a freq. lock time delay (see DPLL specification), the PLL re-locks. 0000 = 1 0001 = 2 —— 1111 = 16
25–16 MFD	Multiplication Factor (Denominator Part). Defines the denominator part of the BRM value for the MF. See Equation 3-1. When a new value is written into the MFD bits, the PLL loses its lock; after a freq. lock time delay, the PLL re-locks. 0000000000 = 1 0000000001 = 2 —— 1111111111 = 1024
15–14	Reserved

Table 3-8. MPCTL Field Descriptions (continued)

Field	Description
13–10 MFI	Multiplication Factor (Integer part). Defines the integer part of the BRM value for the MF. See Equation 3-1. The MFI is encoded so that MFI < 5 results in MFI = 5. When a new value is written into the MFI bits, the PLL loses its lock: after a freq. lock time delay, the PLL re-locks. 0000–0101 = 5 0110 = 6 —— 1111 = 15
9–0 MFN	Multiplication Factor (Numerator part). Defines the numerator of the BRM value for the MF. See Equation 3-1. When a new value is written into the MFN bits, the PLL loses its lock; after a freq. lock time delay, the PLL re-locks. This value is a 2's complements number. 0000000000 = 0 0000000001 = 1 —— 0111111111 = 511 1000000000 = -512 —— 1111111111 = -1

Restriction: The absolute value of MFn/MFd must be smaller than 1.

3.4.3.5.1 Calculating MPLL's Output Frequency

The MPLL's output frequency (F_{VCO}) oscillates at a value determined by Equation 3-1, where:

F_{ref} =Reference clock (input frequency) of MPLL

MFx =Various multiplication factors, whose value is set using the MPCTL register (see bit descriptions below)

PD =Pre-Divider factor, whose value is also set in the MPCTL register

$$F_{ref} \times 2\frac{MF_I + \frac{MF_N}{MF_D}}{PD} = Fvco$$

MPLL Output Frequency ***Eqn. 3-1***

3.4.3.6 USB PLL Control Register (UPCTL)

Figure 3-8 shows the register; Table 3-9 provides the register's field descriptions.

0x53F8_ 0014 (UPCTL) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BRMO	0	PD				MFD									
W																
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	MFI				MFN									
W																
Reset	0	0	0	1	1	1	0	0	0	0	0	0	0	0	1	1

Figure 3-8. USB PLL Control Register—UPCTL

Table 3-9. UPCTL Field Descriptions

Field	Description
31 BRMO	BRM Order bit. Determines if the BRM order is first order or second order. The first order BRM is used if a MF fractional part is both more than 1/10 and less than 9/10. In other cases, the second order BRM is used. The BRMO bit is cleared by a hardware reset. 0 BRM order is first order. 1 BRM order is second order.
30	Reserved
29–26 PD	Pre-divider Factor bits define the pre-divider factor (PD) applied to the PLL input frequency. See Equation 3-1. PD is an integer between 1and 16 (inclusive). PD is chosen to ensure that the resulting output frequency remains within the specified range. When a new value is written into PD bits, the PLL loses its lock; after a freq. lock time delay (see DPLL specification), the PLL re-locks. 0000 = 1 0001 = 2 —— 1111 = 16
25–16 MFD	Multiplication Factor (Denominator Part). Defines the denominator part of the BRM value for the MF. See Equation 3-1. When a new value is written into the MFD bits, the PLL loses its lock; after a freq. lock time delay, the PLL re-locks. 000000000 = 1 000000001 = 2 —— 1111111111 = 1024
15–14	Reserved

Table 3-9. UPCTL Field Descriptions (continued)

Field	Description
13–10 MFI	Multiplication Factor (Integer part). Defines the integer part of the BRM value for the MF. See Equation 3-1. The MFI is encoded so that MFI < 5 results in MFI = 5. When a new value is written into the MFI bits, the PLL loses its lock: after a freq. lock time delay, the PLL re-locks. 0000–0101 = 5 0110 = 6 —— 1111 = 15
9–0 MFN	Multiplication Factor (Numerator part). Defines the numerator of the BRM value for the MF. See Equation 3-2. When a new value is written into the MFN bits, the PLL loses its lock; after a freq. lock time delay, the PLL re-locks. This value is a 2's complements number. 0000000000 = 0 0000000001 = 1 —— 0111111111 = 511 1000000000 = –512 —— 1111111111 = –1

Restriction: The absolute value of MFn/MFd must be smaller than 1.

3.4.3.6.1 Calculating USB PLL Output Frequency

The UPLL's output frequency (F_{VCO}) oscillates at a value determined by Equation 3-2, where:

F_{ref} =Reference clock (input frequency) of UPLL

MFx =Various multiplication factors, whose value is set using the UPCTL register (see bit descriptions)

PD =Pre-Divider factor, whose value is also set in the UPCTL register

$$F_{ref} \times 2\frac{MF_I + \frac{MF_N}{MF_D}}{PD} = Fvco$$

UPLL Output Frequency ***Eqn. 3-2***

3.4.3.7 SR PLL Control Register (SPCTL)

Figure 3-9 shows the register; Table 3-10 provides the register's field descriptions.

0x53F8_ 0018 (SPCTL) Access: User Read/Write

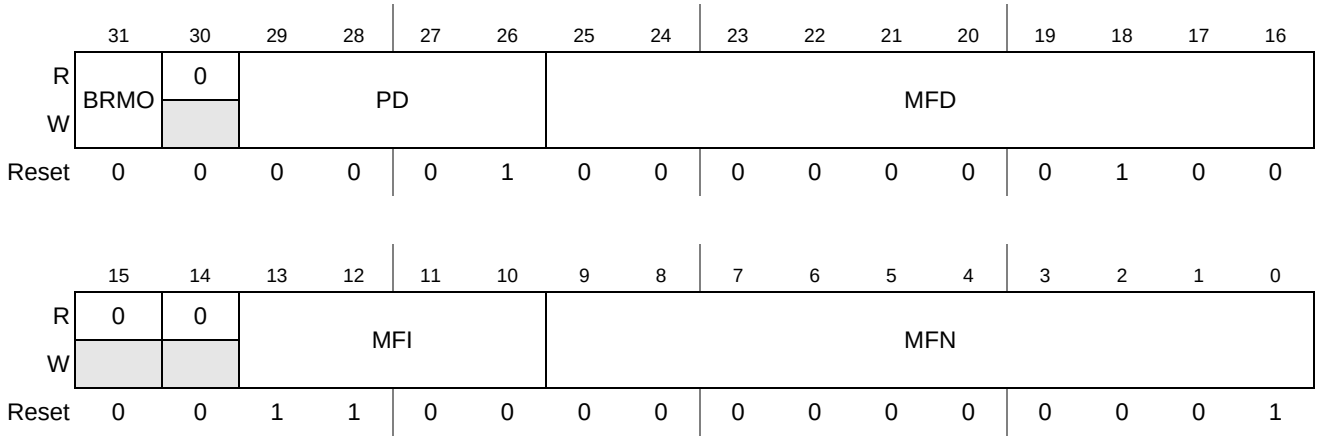


Figure 3-9. SR PLL Control Register (SPCTL)

Table 3-10. SPCTL Field Descriptions

Field	Description
31 BRMO	BRM Order bit. Determines if the BRM order is first order or second order. The first order BRM is used if a MF fractional part is both more than 1/10 and less than 9/10. In other cases, the second order BRM is used. The BRMO bit is cleared by a hardware reset. 0 BRM order is first order. 1 BRM order is second order.
30	Reserved
29–26 PD	Pre-divider Factor bits define the pre-divider factor (PD) applied to the PLL input frequency. See Equation 3-1. PD is an integer between 1and 16 (inclusive). PD is chosen to ensure that the resulting output frequency remains within the specified range. When a new value is written into PD bits, the PLL loses its lock; after a freq. lock time delay (see DPLL specification), the PLL re-locks. 0000 = 1 0001 = 2 —— 1111 = 16
25–16 MFD	Multiplication Factor (Denominator Part). Defines the denominator part of the BRM value for the MF. See Equation 3-1. When a new value is written into the MFD bits, the PLL loses its lock; after a freq. lock time delay, the PLL re-locks. 0000000000 = 1 0000000001 = 2 —— 1111111111 = 1024
15–14	Reserved

Table 3-10. SPCTL Field Descriptions (continued)

Field	Description
13–10 MFI	Multiplication Factor (Integer part). Defines the integer part of the BRM value for the MF. See Equation 3-1. The MFI is encoded so that MFI < 5 results in MFI = 5. When a new value is written into the MFI bits, the PLL loses its lock: after a freq. lock time delay, the PLL re-locks. 0000–0101 = 5 0110 = 6 —— 1111 = 15
9–0 MFN	Multiplication Factor (Numerator part). Defines the numerator of the BRM value for the MF. See Equation 3-1. When a new value is written into the MFN bits, the PLL loses its lock; after a freq. lock time delay, the PLL re-locks. This value is a 2's complements number. 0000000000 = 0 0000000001 = 1 —— 0111111111 = 511 1000000000 = -512 —— 1111111111 = -1

Restriction: The absolute value of MFn/MFd must be smaller than 1.

3.4.3.7.1 Calculating SRPLL Output Frequency

The SRPLL's output frequency (FVCO) Oscillates at a value determined by Equation 3-3, Where:

F_{ref} = Reference clock (input frequency) of SRPLL

MFx = Various multiplication factors, whose value is set using the SPCTL register (see bit descriptions below)

PD = Pre-Divider factor, whose value is also set in the SPCTL register

$$F_{ref} \times 2 \frac{MF_I + \frac{MF_N}{MF_D}}{PDF} = Fvco$$

SRPLL Output Frequency ***Eqn. 3-3***

3.4.3.8 Clock Out Source Register (COSR)

Figure 3-10 shows the register; Table 3-11 provides the register's field descriptions.

0x53F8_ 001C (COSR) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	CLK OEN	CLKOUTDIV			0	0	CLKOSEL			
W																
Reset	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0

Figure 3-10. Clock Out Source Register (COSR)

Table 3-11. COSR Field Descriptions

Field	Description
31–10	Reserved
9 CLKOEN	Clock output enable bit 1 clock output IO pin is enabled 0 clock output IO pin disabled
8–6 CLKOUTDIV	Clock output divide factor 000 2^0 001 2^1 010 2^2 011 2^3 100 2^4 101 Reserved 110 Reserved 111 Reserved

Table 3-11. COSR Field Descriptions (continued)

Field	Description
5–4	Reserved
3–0 CLKOSEL	These bits select which clock is to be reflected on the clock output CKO: 0000 mpl_dpdgck_clk 0001 ipg_clk_ccm 0010 upl_dpdgck_clk 0011 pll_ref_clk 0100 fpm_ckil512_clk 0101 ipg_clk_ahb_arm 0110 ipg_clk_arm 0111 spl_dpdgck_clk 1000 ckih 1001 ipg_clk_ahb _emi_clk 1010 ipg_clk_ipu_hsp 1011 ipg_clk_nfc_20m 1100 ipg_clk_perclk_uart1 1101 ref_cir1 (ref_cir_gateload) 1110 ref_cir2 (ref_cir_intrcload) 1111 ref_cir3 (ref_cir_path)

3.4.3.9 Clock Gating Registers (CGR0–CGR2)

Figure 3-11 shows the register; Table 3-12 provides the register's field descriptions.

0x53F8_ 0020 (CGR0)
0x53F8_ 0024 (CGR1)
0x53F8_ 0028 (CGR2)

Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CG15		CG14		CG13		CG12		CG11		CG10		CG9		CG8	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CG7		CG6		CG5		CG4		CG3		CG2		CG1		CG0	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 3-11. Clock Gating Registers (CGR0–CGR2)

Table 3-12. GR0–CGR2 Field Descriptions

Field	Description
31–30 CG15	The clock gating registers define the clock gating for power reduction of each clock (CG(i) bits). There are *n* CGR registers. The number of registers required is according to the number of peripherals (*n*) in the system. CG(i) bits These bits are used to turn on/off the clock to each module independently. The following list details the possible clock activity conditions for each module. 00 clock is off during all modes. 01 clock is on in run mode, but off in wait and doze modes 10 clock is on in run and wait modes, but off in doze mode 11 clock is on during all modes, except when PLL clock is off. **Note:** Before writing 00 to these bits (thus shutting off the clock to the module), the module to which it is connected must first be stopped. If this is not done, the module's behavior can be adversely affected. RTIC, SDMA, IPU, and EMI clock gating is not possible during run mode.
—— (2i+1)–2i CG(i)	——

Table 3-13, Table 3-14, and Table 3-15 provide information on the mapping of i.MX31 modules and the CGR registers bits.

Table 3-13. CGR0 Register Mapping

Module Name	CG(i) Bits Index
sd_mmc1	0
sd_mmc2	1
GPT	2
EPIT1	3
EPIT2	4
IIM	5
ATA	6
SDMA	7
CSPI3	8
RNG	9
UART1	10
UART2	11
SSI1	12
I2C1	13
I2C2	14
I2C3	15

Table 3-14. CGR1 Register Mapping

Module Name	CG(i) Bits Index
HANTRO	0
MEMSTICK1	1
MEMSTICK2	2
CSI	3
RTC	4
WDOG	5
PWM	6
SIM	7
ECT	8
USBOTG	9
KPP	10
IPU	11
UART3	12
UART4	13
UART5	14
1-WIRE	15

Table 3-15. CGR2 Register Mapping

Module or Clock Name	CG Bits Index
SSI2	0
CSPI1	1
CSPI2	2
GACC	3
EMI	4
RTIC	5
FIR	6
Reserved	7
Reserved	8

Table 3-15. CGR2 Register Mapping (continued)

Module or Clock Name	CG Bits Index
Reserved	9
Reserved	10
Reserved	11
Reserved	12
Reserved	13
Reserved	14
Reserved	15

3.4.3.10 Wake-Up Interrupt Mask Register (WIMR0)

Figure 3-12 shows the register; Table 3-16 provides the register's field descriptions.

0x53F8_ 002C (WIMR0) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	WIM31	WIM30	WIM29	WIM28	WIM27	WIM26	WIM25	WIM24	WIM23	WIM22	WIM21	WIM20	WIM19	WIM18	WIM17	WIM16
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	WIM15	WIM14	WIM13	WIM12	WIM11	WIM10	WIM9	WIM8	WIM7	WIM6	WIM5	WIM4	WIM3	WIM2	WIM1	WIM0
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 3-12. Wake-up Interrupt Mask Register 0 (WIMR0)

Table 3-16. WIMR0 Field Descriptions

Field	Description
31–0 WIMn	WIM[31:0]. These bits are interrupt mask bits. They are applicable for all interrupts that can be connected to the CCM interrupt controller, and can be used, for instance, for wake-up interrupt generation in MCU power gating mode. 0 Interrupt is enabled. 1 Interrupt is masked.

3.4.3.11 Latch Divergence Counter Register (LDC)

Figure 3-13 shows the register; Table 3-17 provides the register's field descriptions.

0x53F8_ 0030 (LDC) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	LDC31	LDC30	LDC29	LDC28	LDC27	LDC26	LDC25	LDC24	LDC23	LDC22	LDC21	LDC20	LDC19	LDC18	LDC17	LDC16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	LDC15	LDC14	LDC13	LDC12	LDC11	LDC10	LDC9	LDC8	LDC7	LDC6	LDC5	LDC4	LDC3	LDC2	LDC1	LDC0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Figure 3-13. Latch Divergence Counter Register (LDC)

Table 3-17. LDC Field Descriptions

Field	Description
31–0 LDCn	LDC[31:0]. These bits contain the latch divergence counter value. Returns to default value only after POR_B‘ef.

3.4.3.12 DPTC Comparator Value Registers (DCVR0–DCVR3)

Figure 3-14 shows the register; Table 3-18 provides the register’s field descriptions.

0x53F8_ 0034 (DCVR0)
0x53F8_ 0038 (DCVR1)
0x53F8_ 003C (DCVR2)
0x53F8_ 0040 (DCVR3)
Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ULV										LLV					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	LLV				ELV											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 3-14. DPTC Comparator Value Register 0:3 (DCVR0:3)

These registers contain relevant DPTC look-up table values.

Table 3-18. DPTC Field Descriptions

Field	Description
31–22 ULV	Upper Limit-value for the upper performance limit of the reference circuit clock counter.
21–12 LLV	Lower Limit-value for the lower performance limit of the reference circuit clock counter.
11–2 ELV	Emergency Limit -value for the lower performance limit of the reference circuit clock counter. This serves as an “emergency” lower limit, which indicates a critical value.
1–0	Reserved

3.4.3.13 Load Tracking Register (LTR0)

Figure 3-15 shows the register; Table 3-19 provides the register’s field descriptions.

0x53F8_ 0044 (LTR0) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SIGD15	SIGD14	SIGD13	0	UPTHR						DNTHR					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SIGD12	SIGD11	SIGD10	SIGD9	SIGD8	SIGD7	SIGD6	SIGD5	SIGD4	SIGD3	SIGD2	SIGD1	SIGD0	DIV3CK		0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 3-15. Load Tracking Register—LTR0

Table 3-19. LTR0 Field Descriptions

Field	Description
31–29	SIGD15–13. These bits define whether the dvfs_w_sig[15:0] signals are detected using level or edge detection. 0 Level detection 1 Edge detection
27–22 UPTHR	Upper threshold for load tracking
21–16	Lower threshold for load tracking
15–3 SIGD*n*	SIGD12–0. These bits define whether the dvfs_w_sig[15:0] signals are detected using level or edge detection. 0 Level detection 1 Edge detection

Table 3-19. LTR0 Field Descriptions (continued)

Field	Description
2–1 DIV3CK	Defines the division value of div_3_clk
0	Reserved

3.4.3.14 Load Tracking Register (LTR1)

Figure 3-16 shows the register; Table 3-20 provides the register's field descriptions.

0x53F8_ 0048 (LTR1) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	LTBRSH	LTBRSR	DNCNT					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DNCNT		UPCNT								PNCTHR					
W																
Reset	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Figure 3-16. Load Tracking Register (LTR1)

Table 3-20. LTR1 Field Descriptions

Field	Description
31–22	Reserved
23 LTBRSH	Load tracking buffer shift 0 takes the original MSB of ld_add–ld_add[5:2] 1 takes a shift of ld_add–ld_add[4:1]
22 LTBRSR	Load tracking buffer source 0 pre_ld_add 1 ld_add
21–14 DNCNT	These bits define the number of consecutive times the lower frequency threshold is undershot (that is, the effective frequency is lower than this threshold) to generate a dvfs_fdw signal. This signal causes a decrease of the system frequency. 00000001=2 00000010=3 —— **Note:** The value 00000000 is not enabled.

Table 3-20. LTR1 Field Descriptions (continued)

Field	Description
13–6 UPCNT	These bits define the number of consecutive times the upper frequency threshold must be exceeded (threshold overcomes) to generate a dvfs_fup signal. This signal causes an increase of the system frequency. 00000001=2 00000010=3 —— **Note:** The value 00000000 is not enabled.
5–0 PNCTHR	These bits define panic mode level threshold for load tracking

3.4.3.15 Load Tracking Register (LTR2)

Figure 3-17 shows the register; Table 3-21 provides the register's field descriptions.

0x53F8_ 004C (LTR2) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	WSW15				WSW14				WSW13				WSW12			WSW11				WSW10
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	WSW10			WSW9			0	0	EMAC							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 3-17. Load Tracking Register (LTR2)

Table 3-21. LTR Field Descriptions

Field	Description
31–29 WSW15	These bits define the general purpose load tracking signals dvfs_w_sig[15:9]. For more details about how these values are defined, see Table 3-29.
28–26 WSW14	
25–23 WSW13	
22–20 WSW12	
19–17 WSW11	
16–14 WSW10	
13–11 WSW9	
10–9	Reserved
8–0 EMAC	These bits define the EMA configuration.

3.4.3.16 Load Tracking Register (LTR3)

Figure 3-18 shows the register; Table 3-22 provides the register's field descriptions.

0x53F8_ 0050 (LTR3) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	WSW8				WSW7				WSW6				WSW5			WSW4				WSW3
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	WSW3				WSW2				WSW1				WSW0				0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 3-18. Load Tracking Register—LTR3

Table 3-22. LTR3 Field Descriptions

Field	Description
31–29 WSW8	These bit fields (each one in turn) define the weight of each of the general purpose load tracking signals (dvfs_w_sig[8:0]). The total CPU load as a result of the dvfs_w_sig signal is defined as a sum of each of the individual signal's dvfs_w_sig[n] value multiplied by its weight, as defined by the corresponding WSWn bit field.
28–26 WSW7	
25–23 WSW6	
22–20 WSW5	
19–17 WSW4	
16–14 WSW3	
13–11 WSW2	
10–8 WSW1	
7–5 WSW0	

3.4.3.17 Load Tracking Buffer Register (LTBR0)

Figure 3-19 shows the register; Table 3-23 provides the register's field descriptions.

0x53F8_ 0054 (LTBR0) Access: User Read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	LTS7				LTS6				LTS5				LTS4			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LTS3				LTS2				LTS1				LTS0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 3-19. Load Tracking Register—LTBR0

Table 3-23. LTBR0 Field Descriptions

Field	Description
31–28 LTS7	These bits contain data of the last eight samples of load tracking.
27–24 LTS6	
23–20 LTS5	
19–16 LTS4	
15–12 LTS3	
11–8 LTS2	
7–4 LTS1	
3–0 LTS0	

3.4.3.18 Load Tracking Buffer Register (LTBR1)

Figure 3-20 shows the register; Table 3-24 provides the register's field descriptions.

0x53F8_ 0058 (LTBR1) Access: User Read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	LTS15				LTS14				LTS13				LTS12			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LTS11				LTS10				LTS9				LTS8			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 3-20. Load Tracking Register (LTBR1)

Table 3-24. LTBR1 Field Descriptions

Field	Description
31–28 LTS15	These bits contain data of the first eight samples of load tracking.
27–24 LTS14	
23–20 LTS13	
19–16 LTS12	
15–12 LTS11	
11–8 LTS10	
7–4 LTS9	
3–0 LTS8	

3.4.3.19 Power Management Control Register 0 (PMCR0)

Figure 3-21 shows the register; Table 3-25 provides the register's field descriptions.

0x53F8_ 005C (PMCR0) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DFSUP		DVSUP		UDSC	VSCNT			DVFEV	DVFIS	LBMI	LBFL	LBCF		PTVIS	UPDTEN
W																
Reset	1	0	0	0	0	0	0	0	0	0	1	0	1	1	0	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FSVAIM	FSVAI		DPVCR	DPVV	WFIM	DRCE3	DRCE2	DRCE1	DRCE0	DCR	DVFEN	PTVAIM	PTVAI		DPTEN
W																
Reset	1	0	0	1	1	0	0	0	0	0	1	0	1	0	0	0

Figure 3-21. Power Management Control Register 0 (PMCR0)

Table 3-25. PMCR0 Field Descriptions

Field	Description
31 DFSUP[1]	DFS update. These bits define which aspect of the MCU frequency will be updated, as follows: 0 SRPLL update 1 MCUPLL update
30 DFSUP[0]	DFS update. These bits define which aspect of the MCU frequency will be updated, as follows: 0 pll and post-dividers update 1 post-dividers update only
29–28 DVSUP	DVS update. These bits, connected to external I/O pins, define in what manner the DVS0 and DVS1 voltages are updated, if at all. **Note:** These bits merely set a value, and it is incumbent upon software external to the i.MX31 and i.MX31L to use this information and act upon it. 00 DVS1=0 DVS0=0—highest frequency/voltage level 01 DVS1=0 DVS0=1 10 DVS1=1 DVS0=0 11 DVS1=1 DVS0=1—lowest frequency/voltage level
27 UDSC	Up-down scaling. This bit indicates the direction of frequency scaling. 0 Frequency is decreased. 1 Frequency is increased.
26–24 VSCNT	Voltage scaling counter. These bits define the number of CKIL cycles required to carry out voltage scaling. This counter is initialized when the UDSC bit is set to “1”. After the counter arrives at the value indicated in the VSCNT bits, the frequency is then scaled. (No. of CKIL cycles) 000 No delay 001 1-0 CKIL cycle delay 010 2-1 CKIL cycle delay (1 CKIL cycle guaranteed). 011 3-2 CKIL cycle delay (1 CKIL cycle guaranteed). ... 111 7-6 CKIL cycle delay **Note:** The value 0x2 should be used to ensure a minimal delay of one CKIL cycle. (Higher values can be used where needed, depending on system requirement).
23 DVFEV	Always give a DVFS event. 0 Do not give an event always. 1 Always give event.
22 DVFIS	DVFS Interrupt select. These bits define destination of DVFS interrupts. 0 SDMA interrupt will be generated for DVFS events. 1 MCU interrupt will be generated for DVFS events.
21 LBMI	Load buffer full mask interrupt. This bit masks the generation of this interrupt. 0 Load buffer full interrupt is enabled. 1 Load buffer full interrupt is masked.
20 LBFL	Load buffer full status bit. This bit indicates that log buffer registers are full. An interrupt will be generated if LBMI bit is set to “0”. 0 Load buffer is not full. 1 Load buffer is full. **Note:** SW can write only “0” into this bit.

Table 3-25. PMCR0 Field Descriptions (continued)

Field	Description
19–18 LBCF	DVFS load buffer programmable size 00 Load buffer size is 4. 01 Load buffer size is 8. 10 Load buffer size is 12. 11 Load buffer size is 16.
17 PTVIS	DPTC Interrupt select. These bits define destination of DPTC interrupts. 0 SDMA interrupts will be generated for DPTC events. 1 MCU interrupts will be generated for DPTC events
16 UPDTEN	DVFS update enable of new value 0 SW is not enabled to write new setting of frequency change (Not able to update DFSUP, DVSUP, UDSC, VSCNT) because PLL is not locked yet. 1 SW is enabled to write new setting of frequency change.
15 FSVAIM	DVFS Frequency adjustment interrupt mask. This bit masks the DVFS frequency adjustment interrupt. FSVAI status bits will be still asserted in relevant cases. 0 Interrupt is enabled. 1 Interrupt is masked.
14–13 FSVAI[1:0]	DVFS Frequency adjustment interrupt. These status bits indicate that the system frequency should be changed. 00 no interrupt 01 frequency should be increased. Low priority interrupt. Interrupt is asserted, if FSVAIM=0. Interrupt is masked if DVSUP = 00 (highest frequency). 10 frequency should be decreased. Interrupt is asserted, if FSVAIM=0. Interrupt is masked if DVSUP = 11 (lowest frequency). 11 frequency should be increased immediately. High priority interrupt. Interrupt is asserted, if FSVAIM=0. Interrupt is masked if DVSUP = 00 (highest frequency).
12 DPVCR	DPTC voltage change request 0 Disabled 1 Enabled
11 DPVV	DPTC voltage valid edge detect. Can be updated by SW. 0 Voltage is not valid 1 Received a voltage valid acknowledge **Note:** If written by SW, it is possible to assert it only after DPVCR is asserted.
10 WFIM	DVFS Wait for Interrupt mask bit 0 Wait for interrupt is not masked. 1 Wait for interrupt is masked.
9 DRCE3	DPTC reference circuit3 enable bit. This bit defines if reference circuit3 is enabled during DPTC operation. 0 DPTC reference circuit3 is disabled. 1 DPTC reference circuit3 is enabled.
8 DRCE2	DPTC reference circuit2 enable bit. This bit defines if reference circuit2 is enabled during DPTC operation. 0 DPTC reference circuit2 is disabled. 1 DPTC reference circuit2 is enabled.
7 DRCE1	DPTC reference circuit1 enable bit. This bit defines if reference circuit1 is enabled during DPTC operation. 0 DPTC reference circuit1 is disabled. 1 DPTC reference circuit1 is enabled.
6 DRCE0	DPTC reference circuit0 enable bits. This bit defines if reference circuit0 is enabled during DPTC operation. 0 DPTC reference circuit0 is disabled. 1 DPTC reference circuit0 is enabled.

Table 3-25. PMCR0 Field Descriptions (continued)

Field	Description
5 DCR	DPTC counting range. This bit sets how many times the system clock may increment and the reference circuits remain active (and their output signals will be counted). Value of “1” causes 512 system clock count. Value of “0” causes 256 system clock count. 0 256 system clock count 1 512 system clock count
4 DVFEN	DVFS enable. This bit enables the DVFS block. 1 DVFS is enabled. 0 DVFS is disabled. **Note:** Between disable and enable there has to be at least 3 cycles of div_3_clk.
3 PTVAIM	DPTC Voltage adjustment interrupt mask. This bit masks the DPTC voltage adjustment interrupt. PTVAI status bits will be still asserted in relevant case. 0 Interrupt is enabled. 1 Interrupt is masked.
2–1 PTVAI[1:0]	DPTC Voltage adjustment interrupt. These status bits indicate that the supply voltage should be changed. 00 no interrupt 01 voltage should be decreased. Interrupt is asserted, if PTVAIM=0 10 voltage should be increased. Low priority interrupt. Interrupt is asserted, if PTVAIM=0 11 voltage should be increased immediately. High priority interrupt. Interrupt is asserted, if PTVAIM=0
0 DPTEN	DPTC enable. This bit enables the DPTC block and starts the reference circuit clock counting and compares this to look-up table values. 0 DPTC is disabled. 1 DPTC is enabled.

3.4.3.20 Power Management Control Register 1 (PMCR1)

Figure 3-22 shows the register; Table 3-26 provides the register’s field descriptions.

0x53F8_ 0060 (PMCR1) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	WBCN							
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	CPSPA				PWTS	NWTS	CPFA	0	0	DVGP			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 3-22. Power Management Control Register 1 (PMCR1)

Table 3-26. PMCR1 Field Descriptions

Field	Description
31–24	Reserved
23–16 WBCN	8 bit programmable well-bias counter configuration bits
15–13	Reserved
12–9 CPSPA	WBCP adjustment. These signals enable adjustment of well-bias function parameters. The value of CPSPA will be inverted to match the WBCP expectation as stated in the WBCP specification.
8 PWTS	WBCP pwell pump test. This signal enables the pwell charge pump in test mode. 0 pwell charge pump is disabled. 1 pwell charge pump is enabled.
7 NWTS	WBCP nwell pump test. This signal enables the nwell charge pump in test mode. 0 nwell charge pump is disabled. 1 nwell charge pump is enabled.
6 CPFA	WBCP Frequency Adjustment. This bit changes the frequency of the internal ring oscillator in the well-bias charge pump block. 0 Normal frequency set 1 Lower frequency set
5–4	Reserved
3–0 DVGP	DVFS general purpose registers.

3.4.3.21 Post Divider Register 2 (PDR2)

Figure 3-23 shows the register; Table 3-27 provides the register's field descriptions.

0x53F8_ 0064 (PDR2) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	MST2_PDF							MST1_PDF					
W																
Reset	0	0	0	0	0	0	1	0	1	0	0	0	0	1	0	1

Figure 3-23. Post Divider Register 2 (PDR2)

Table 3-27. PDR2 Field Descriptions

Field	Description
31–13	Reserved
12–7 MST2_PDF	These bits control the M-stick2 post divider. See Figure 3-24. 000001 Divide by 2 —— 111111 Divide by 64
6	Reserved
5–0 MST1_PDF	These bits control the M-stick1 post divider. See Figure 3-24. 000001 Divide by 2 —— 111111 Divide by 64

3.4.4 Functional Description

The general clock generation scheme is shown in Figure 3-24 which is shown without scan mux and clock gating. The sections that immediately in this chapter describe in more detail how the individual clocks are generated.

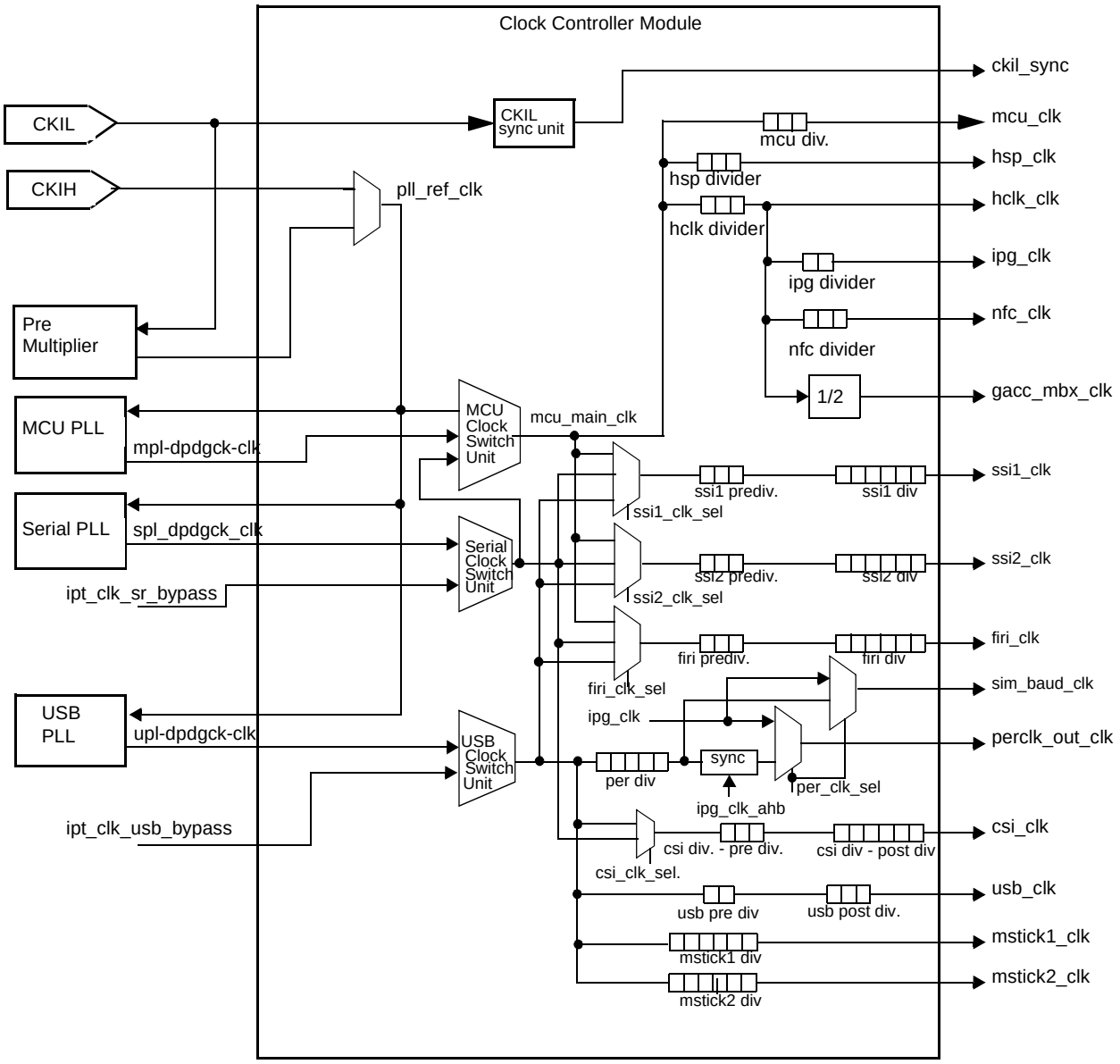


Figure 3-24. i.MX31 Clock Generation Scheme

3.4.4.1 Clock Sources

3.4.4.1.1 External Low Frequency Clock—CKIL

The i.MX31 and i.MX31L processors can use either a 32 kHz, 32.768 kHz or a 38.4 kHz crystal as the external low frequency source. Throughout this chapter, the low frequency crystal is referred to as the 32 kHz crystal, even though this can refer to the 32.768 or 38.4 kHz values.

The signal from the external 32 kHz crystal is the source of the CKIL signal that is sent to the real time clock (RTC). The output of the 32 kHz crystal is also input to the pre-multiplier PLL to produce the PLL reference clock by multiplying CKIL by a factor of 1024.

3.4.4.2 MCU Clock Domain Clocks

The CCM provides a large number of clock outputs used to supply clocks to the MCU and the peripherals. Each of the i.MX31 processors are partitioned into two asynchronous clock domains: *MCU* and *USB*, as there are different functionality and frequency requirements from the clocks (detailed in this section).

3.4.4.2.1 MCU Clock Domain Clock Source Switch Unit

The main clock of the MCU clock domain is mcu_main_clk and is generated by MCU clock switch unit.

The following signals are possible clock sources for the MCU clock switch unit: pll_ref_clk, mpl_dpdgck_clk (MCU PLL output), and spl_dpdgck_clk (SRPLL output). Whether the functional mode clock uses ref_pll_clk or the mpl_dpdgck_clk signal is determined in S/W via enabling/disabling the MCU PLL by writing MPE bit in the CCMR register, and by choosing the MCU clock domain source by writing the MDS bit in the CCMR register. The selection between a reference clock from the MCU PLL or the SRPLL can only be done when the DVFS is enabled. The ref_pll_clk signal will be selected automatically when the MCU enters the test mode to bypass the PLL.

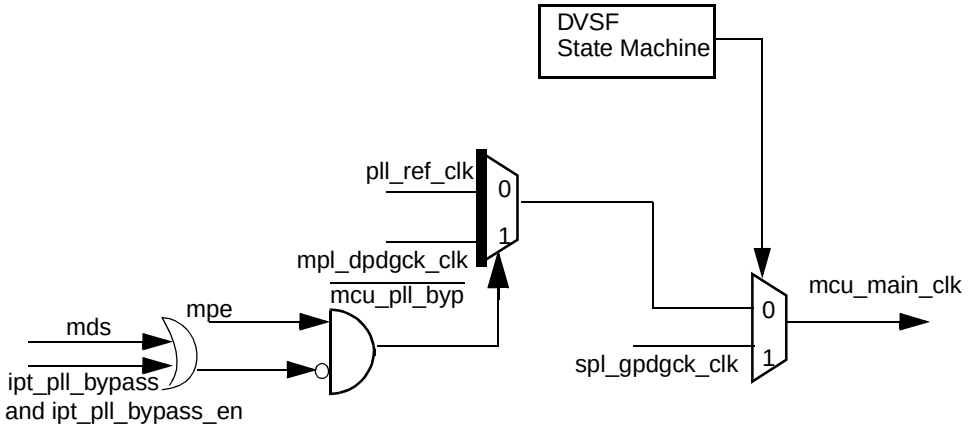


Figure 3-25. MCU Clock Switch Unit

3.4.4.2.2 MCU Clock Domain Clocks

The MCU clock domain is partitioned into four synchronous clocks and two sub-domains. These are described below. The main clock of this domain is called mcu_main_clk, and it is the output of the MCU clock switch unit.

- mcu_clk (ipg_clk_arm) is the clock of the ARM platform. The target frequency of this clock is 532 MHz. This clock is generated from MCU BRM with a division factor as defined by the BRMM bits in PDR0.
- max_clk sub-domain (ipg_clk_ahb) is the clock domain of the internal ARM platform peripherals like the cross bar switch and chip modules. Clocks in this domain are generated from the max

post-divider with a division factor as defined by the MAX_PDF bits in PDR0 register. These clocks should be an integer multiple (value of between 1 and 8) of the mcu_main_clk. Maximum target frequency of these clocks is 133 MHz.

- hsp_clk is the clock for the IPU. This clock is generated from the hsp post-divider with a division factor as defined by the HSP_PDF bits in PDR0 register. These clocks should be an integer multiple (value of between 1 and 8) of the mcu_main_clk. Maximum target frequency of this clock is 133 MHz for 1.2 V supply.
- ipg_clk sub-domain is the clock domain of certain parts of the IP peripherals. These clocks are generated from the ipg post-divider with a division factor defined by the IPG_PDF bits in the PDR0 register. These clocks should be an integer multiple (either 1 or 2) of the max_clk. Maximum target frequency of these clocks is 62.5 MHz.
- nfc_clk (ipg_clk_nfc_20m) is the clock for NAND Flash Controller. This clock is generated from the nfc post-divider with a division factor as defined by the NFC_PDF bits in the PDR0 register.
- ckil_mcu_sync_ipg is the clock for the peripheral modules. They require a 32-kHz clock
- ipg_clk_gacc_mbx_clk is the clock for the MBX module. It is 1/2 of the ipg_ahb_clk, which is 66 MHz.

NOTE

The MBX R-S graphics accelerator is not available in the i.MX31L.

The phases between mcu_clk, hsp_clk, ckil_mcu_sync_ipg, ipg_clk_gacc_mbx_clk and clocks from max_clk and per_clk subdomains are synchronized with the MCU domain master clock frequency (mcu_main_clk), but the frequencies can be different.

Each MCU clock sub-domain has a dedicated post-divider that can be programmed to generate the divided clock from the mcu_main_clk. The division factor is always an integer. Each clock port from the MCU domain is connected to a specific sub-domain.

All clocks in the MCU clock domain are balanced. Posedge (the edge at which point the voltage increases) of max_clk sub-domain clocks is always aligned with mcu_clk posedge. Posedge of ipg_clk sub-domain clocks is always aligned with posedge of max_clk sub-domain clocks.

3.4.4.2.3 Clock Generation—ipg_ckil_sync Clock

This clock is generated by synchronization of CKIL clock to ipg_clk, when the system is not in Deep Sleep mode or reset. When the system is in Deep Sleep mode and by default in reset, the synchronizer is bypassed.

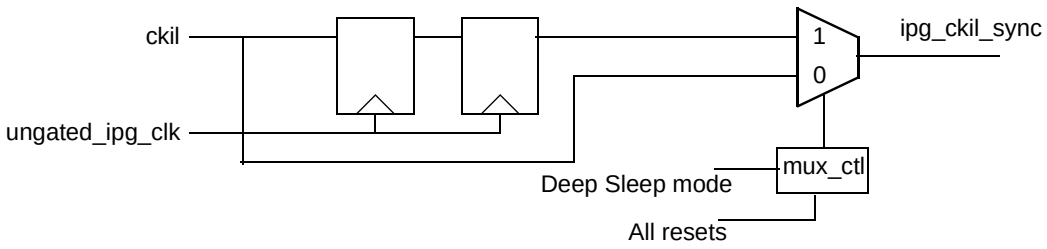


Figure 3-26. ckil_mcu_sync_ipg Clock Generation

3.4.4.3 USB Clock Domain

3.4.4.3.1 USB Clock Domain Switch Unit

The main clock of the USB clock domain is usb_main_clk. This signal is generated by the USB clock switch unit. The following signals are possible clock sources of the USB switch unit: USB DPLL output (upl_dpdgck_clk) and bypass clock (ipp_clk_usb_bypass - bypass clock).

Selection between functional mode clock upl_dpdgck_clk and test mode clock: ipt_clk_usb_bypass is done automatically, when the systems enters test mode.

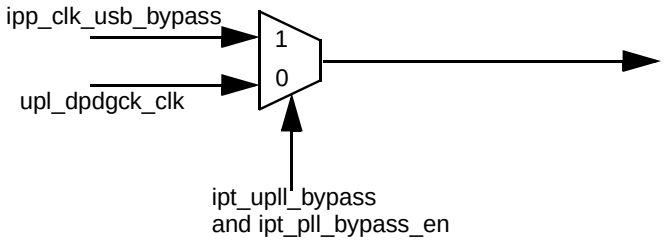


Figure 3-27. USB Clock Switch Unit

3.4.4.3.2 USB Clock Domain Clocks

The USB clock domain is partitioned into several asynchronous clocks.

- ipg_clk_usb_baud is the clock for the USBOTG module. This clock is generated from the usb post-divider with division factor, defined by the USB_PDF bits in the MPDR1 register.
- ipg_clk_firi_baud is the clock of the FIR module.
- ipg_clk_ssi1_baud is the clock for the SSI1 module.
- ipg_clk_ssi2_baud is the clocks for the SSI2 module.
- ipg_sim_baud is the clock for the SIM module.
- ipg_per_baud is the clock for the peripherals that will not be affected by the DVFS changes.
- ipg_clk_csi_baud is the clock for the IPU, used for the camera sensor chip. This clock is generated from the CSI post-divider with a division factor as defined by the CSI_PDF bits in the PDR0 register.
- ipg_clk_mstick1_baud is the clock for memstick1 module.
- ipg_clk_mstick2_baud is the clock for the memstick2 module.

USB domain clocks are not synchronized or balanced with each other.

3.4.4.3.3 Clock Generation—ipg_clk_firi_baud

The ipg_clk_firi_baud clock is generated as follows: First a source between MCU, USB or Serial PLL's outputs is chosen, according to the value of the FIRS bits in the CCMR register. Then, this source is passed through two post dividers, connected in series. The division factors of the post dividers are defined by defined by the FIRI_PRE_PDF and FIRI_PDF bits in the MPDR1 register.

3.4.4.3.4 Clock Generation—ipg_clk_ssi1_baud

The ipg_clk_ssi1_baud clock is generated as follows: First a source between MCU, USB or Serial PLL's outputs is chosen, according to the value of the SSI1S bits in the CCMR register. Then, this source is passed through two post dividers, connected in series. The division factors of the post dividers are defined by defined by the SSI1_PRE_PDF and SSI1_PDF bits in the MPDR1 register.

3.4.4.3.5 Clock Generation—ipg_clk_ssi2_baud

The ipg_clk_ssi2_baud clock is generated as follows: First a source between MCU, USB or Serial PLL's outputs is chosen, according to the value of the SSI2S bits in the CCMR register. Then, this source is passed through two post dividers, connected in series. The division factors of the post dividers are defined by defined by the SSI2_PRE_PDF and SSI2_PDF bits in the MPDR1 register.

3.4.4.3.6 Clock Generation—ipg_sim_baud

The ipg_sim_baud clock is generated as follows: The a source between USB PLL's output and ipg_clk is chosen according to the value of PERCS bit in the CCMR register. This source is passed though the peripheral divider that is defined in the PER_PODF bits in the PDR0 register.

3.4.4.3.7 Clock Generation—ipg_per_baud

The ipg_per_baud clock is generated as follows: A source between the synchronized USB PLL's output and ipg_clk is chosen according to the value of PERCS bit in CCMR register. The USB PLL's output is synchronized with ipg_clk_ahb after being divided by the peripheral post-divider using the PER_PODF bit in PDR0. There is a restriction on the value of the post divider bits, the created clock must be slower than the synchronizing clock ipg_clk_ahb by at least a half.

3.4.4.3.8 Clock Generation—ipg_clk_csi_baud

The ipg_clk_csi_baud clock is generated as follows: First a source between the MCU and USB PLL's output is chosen according to CSCS bit in CCMR register. Then, this source is passed throughout the CSI post-divider that is defined in CSI_PODF bits in the PDR0 register.

3.4.4.3.9 Clock Generation—ipg_clk_mstick1_baud

This clock is generated as follows: The USB PLL's output is passed though the memstick1 post-divider that is defined in the MST1_PDF bits in PDR2 register.

3.4.4.3.10 Clock Generation—ipg_clk_mstick2_baud

This clock is generated as follows: The USB PLL's output is passed though the memstick1 post-divider that is defined in the MST2_PDF bits in PDR2 register.

3.4.4.4 SR Clock Domain

3.4.4.4.1 SR Clock Switch Unit

The man clock of the SR clock domain in the sr_main_clk and is generated by the SR clock switch unit. The following signals are possible clock sources SRPLL output clock—spl_dpdgck_clk, bypass clock ipp_clk_sr_clk.

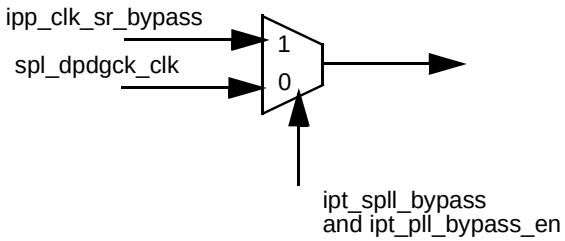


Figure 3-28. SR Clock Switch Unit

3.4.4.5 Clock Cleaner

3.4.4.6 The clock cleaner is connected to the ipp_ckih clock after the CAMP cell. Clock cleaner is actually a clock gating cell, that gates off PLL clocks until the output of CAMP is stable. ckih_osc_rdy signal controls the CKIH clock cleaner.**Low Power Clock Gating (LPCG)**

The LPCG block distributes clocks to all modules from the subdomain clocks and gates off clocks in low power mode. Clock gating for each module is carried out based on the specific low power mode and the relevant bits in the MCGR register.

3.4.4.7 SDRAM Controller Handshake Mechanism

The SDRAM controller requires high frequency clock (more than 1 MHz) to work properly with the external memories. If for some reason the clock controller will not provide such a clock to the SDRAM controller (software disable of SDRAM controller clock or entering the STOP mode), then before disabling the SDRAM controller clock, the handshake mechanism is activated, which causes the SDRAM controller to place the external memories in power down mode. CCM will then assert the ccm_sd_lpmd signal, which indicates that a request to place the SDRAM in self-refresh mode was made. After the EMI places the SDRAM in self refresh mode, the emi_sd_lpack signal will be asserted and the CCM will gate the clock to the EMI.

3.4.4.8 Power Fail

The power_fail IO port is connected to the CCM and generates a power fail interrupt for the MCU. The interrupt can be generated only if the MCU is not in State Retention or Deep Sleep mode. The goal of the interrupt routine is to enter Deep Sleep mode with minimum energy consumption.

3.5 Power Management

3.5.1 Power Domains

The i.MX31 and i.MX31L are partitioned into four power domains:

- MCU domain—includes the ARM11 Core, the MMU, and the Caches
- L2 cache domain—includes L2 cache data array
- Peripheral domain—includes all the peripherals except the MCU–PLL, USB-PLL, SR-PLL, and the FPM
- PLL domain—includes the MCU-PLL, USB-PLL, SR-PLL, and the FPM

3.5.2 Power Modes

The i.MX31 and i.MX31L processors support a versatile definition of power modes, including power and clock domains status and applied power techniques. The power modes described in the following sections were defined taking into account static and dynamic power consumption of blocks in different operational modes, the power consumption of clock sources, and the time required to exit low power modes.

3.5.2.1 Run Mode

This is the normal/functional operating mode of ARM11. The ARM clock frequency can vary between fmax and fmin, and the voltage between Vmax and Vmin.

3.5.2.2 Wait Mode

In this mode, the MCU clock is gated, but ARM11P MAX and all peripherals clocks are available. This mode is entered by the MCU executing a STANDBY FOR INTERRUPT command. The arm_clk_off output of ARM11P is asserted and the clock to the MCU arm_clk is gated by the CCM. The MCU exits the wait mode and enters the run mode by receiving of any interrupt (depend on mask bits) and negation of arm_clk_off signal. Clocks for specific modules can be gated off automatically in wait mode by programming MCGR registers.

3.5.2.3 Doze Mode

MCU and MAX clocks are gated. Clock source is available and peripherals that do not require MAX and MCU functionality can be active.

Doze mode is entered by programming the LPM bits in the MCR register. The next time the MCU executes the STANDBY FOR INTERRUPT command, a signal for MAX clock halt request will be asserted by the CCM and the clock to MAX will be gated off upon acknowledgement of the signal assertion. If WBEN bit is set to “1”, the ccm_wben signal will be asserted.

The MCU exits Doze mode to Run mode by receiving any enabled interrupt and negation of the arm_clk_off signal. If WBEN bit is set to “1”, PLL reference clock well-bias counter will be started and upon completion of its counting, the ccm_wben signal will be negated and clocks will then be provided to the MCU and peripherals. LPM bits in MCR register are cleared by exiting Doze mode.

Clocks for specific modules can be gated off automatically in Doze mode by programming the MCRG register.

3.5.2.4 State Retention Mode

This mode has the same functionality as Doze mode, except that the addition of clocks to peripherals are gated off, and MCU domain PLLs are disabled. SDRAM is put in Self-refresh mode.

State Retention mode is entered by programming the LPM bits in the MCR register the next time the STANDBY FOR INTERRUPT command is issued.

The sequence below is implemented by the CCM upon entering the State Retention mode:

1. Asserts ccm_sd_lpmd signal to EMI to put SDRAM in self refresh mode
2. Upon emi_sd_lpack assertion, stops clocks to the MCU and peripherals and then stops the PLLs, if any were enabled
3. If the SBYCS bit in the MCR register is set to “1”, disables selected clock source generator as follows:
   a) If an external oscillator is selected, disables CKIH receiver
   b) If FPM is selected, disables FPM
4. If the WBEN bit in the MCR register is set to “1”, assert the ccm_mcu_wb_en signal.
5. If the VSTBY bit was set to “1”, assert the ccm_vstby_pmic signal. This in turn will assert the pmic_stby output pin, which places the PMIC regulators in standby mode. If the L2PG bit is set to “1”, asserts the ccm_l2pg_pmic signal, which asserts the l2pg_pmic output pin.

The i.MX31 and i.MX31L exit State Retention mode by any internal or external interrupt.

The sequence below is implemented by the CCM to exit the State Retention mode:

1. If the VSTBY bit was set to “1”, negates the ccm_vstby_pmic signal, which negates the pmic_stby output pin. If the L2PG bit was set to “1”, negates the ccm_l2pg_pmic signal. A flash of the L2 cache should be performed before the MCU executes the WFI command.
2. If SBYCS bit is set to “1”, enables the clock source generator as follows:
   a) If external oscillator is selected, enables CKIH receiver
   b) If FPM is selected, enables FPM
   c) If the oscillator is selected as a clock source, starts the 32-KHz counter to count the time as defined by the OSCNT bits
   d) If WBEN bit is set to “1”, disables the well-bias
3. After reaching the value defined by the OSCNT bits in MCR register, and if the oscillator is selected as the clock source, the clock cleaner will be enabled. Indication on the FPM that the clock output is ready, if is selected, is the assertion of the lock-ready flag.

   The time required to disable the well-bias is significantly less than the settling time of the external oscillator. Indication that the voltage from the PMIC is valid is a PMIC interrupt, which is sent to the CCM.

4. After the selected clock source is ready and the voltage is valid, then if MPE, UPE or SPE bits are set to “1”, the relevant PLL’s will be started. If the WBEN bit is set to “1” and SBYCS to “0”, the well-bias will be disabled, and well-bias counter will be started, running on PLL reference clock.
5. With the MCU, USB, the Serial port (if enabled), and the PLL’s lock ready flag[1], negate the ccm_sd_lpmd signal to the EMI to remove the SDRAM from Self Refresh mode. Then start the ccm_emi_ahb_clk clock, and input this to the EMI.
6. Upon emi_sd_wack signal assertion, and when the well-bias disabled counter completes, if the SBYCS is set to “0”, the clocks to the ARM and peripherals will be restored. LPM bits in MCR register will be cleared by exiting Deep Sleep mode.

3.5.2.5 Deep Sleep Mode

This mode is similar to State retention mode, but supply of the ARM platform is shut down. If ARM State Retention is required, critical registers data should be saved before entering Deep Sleep mode (executing WFI instruction).

Entering the Deep Sleep mode follows the same steps as entering State Retention mode with the next addition to Step 5:

deassert ccm_pgen, ccm_reset_mcu, and ccm_mcupg_pmic signals, which will in turn assert the mcupg_pmic output pin.

Interrupts in this mode will be managed by a simplified interrupt controller unit in CCM and output of this unit will place the i.MX31 and i.MX31L processors in Run mode.

Exiting Deep Sleep mode is the same as exiting State Retention mode with the following additions:

Step 1: negate ccm_mcupg_pmic signal.

Step 2: negate mcu_power_gate_en and $\overline{\text{mcu\_reset\_out}}$.

1. If MPE bit is set to “0”, skip PLL lock ready flag requirement.

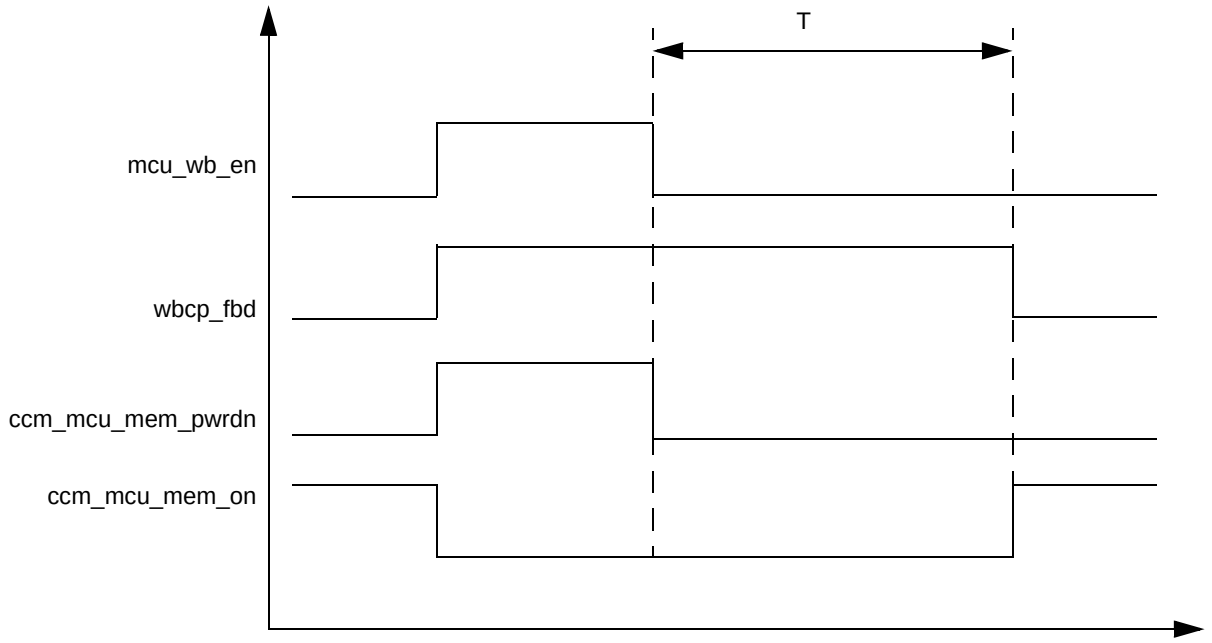


Figure 3-29. Well-Bias Activating and Deactivating and ARM Core Power Gating

T-In Doze mode, this is the WB counter.

In Deep Sleep Mode or State Retention Mode, and if SBYCS = 0, T is the OSCNT (if CKIH is the source) or FPM lock time (if FPM is the source). If SBYCS = 1 T is the WB counter.

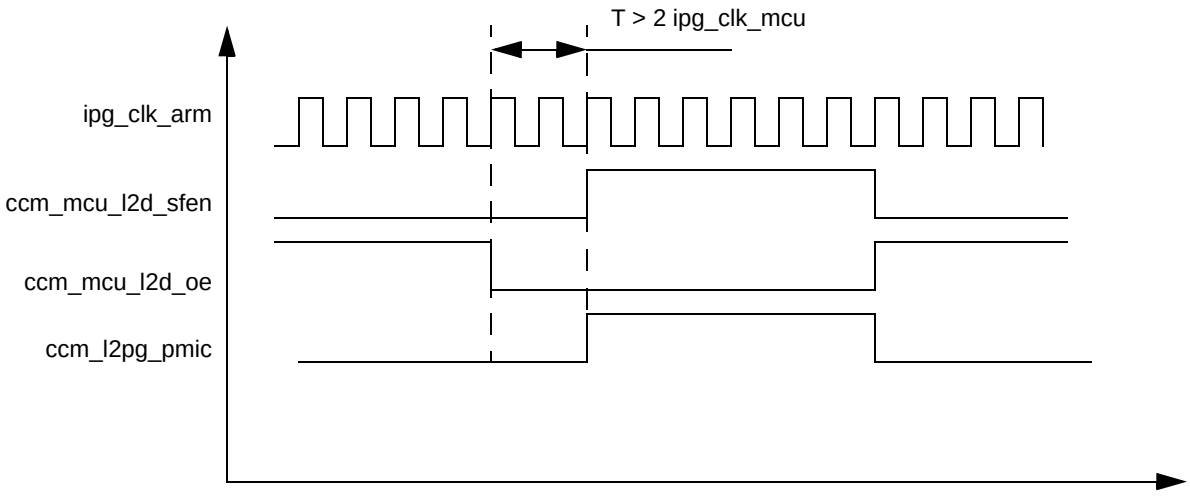


Figure 3-30. L2 Cache Power Gating

3.5.2.6 Hibernate Mode

The supply of the IC is shut down. The i.MX31 and i.MX31L processors can enter Run mode by external interrupt only with warm boot operation.

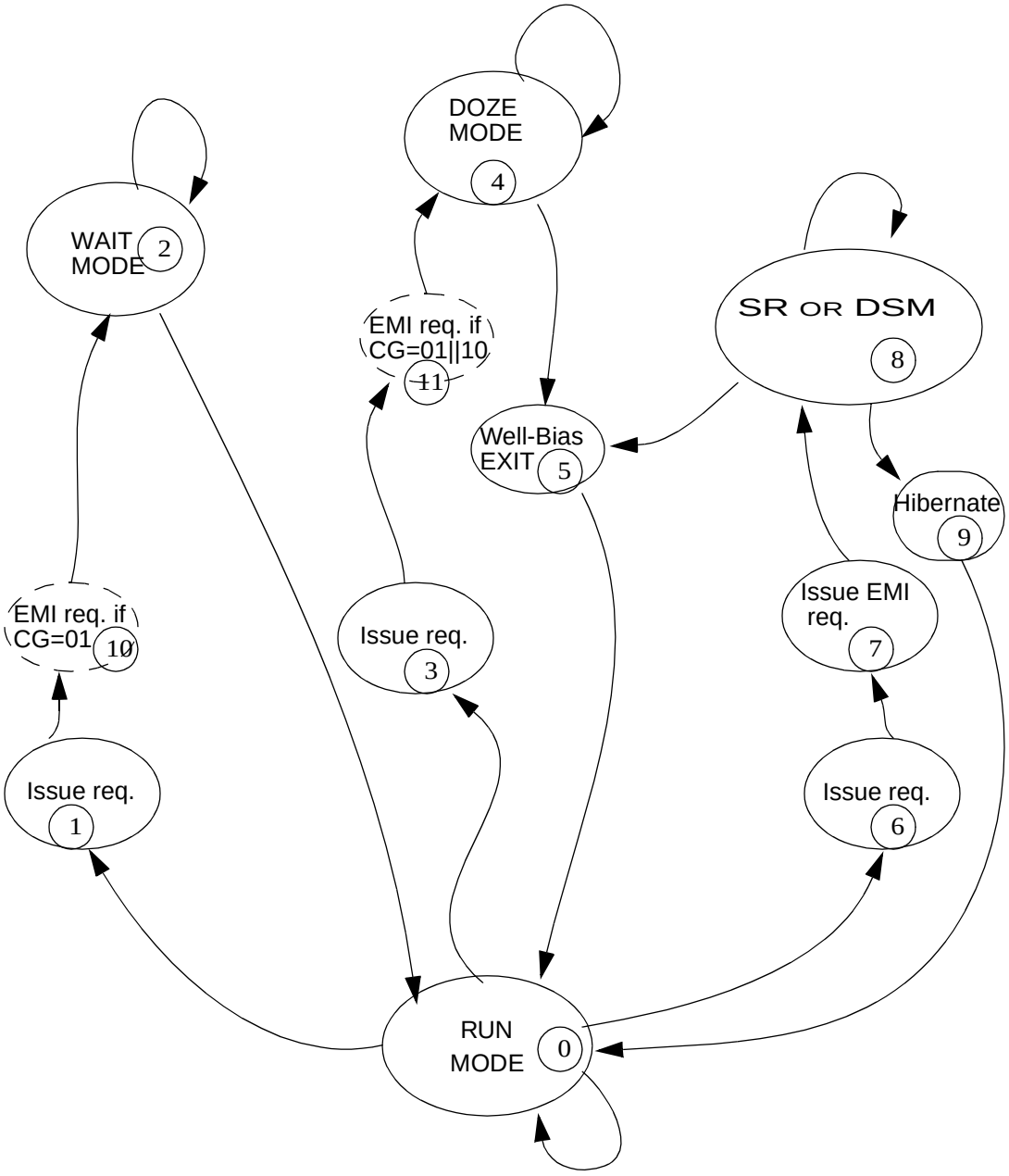


Figure 3-31. Low Power Modes State Machine

3.5.3 Power Management Techniques Overview

The CCM supports several power management techniques that reduce active and static power consumption:

Dynamic Voltage Frequency Scaling (DVFS)

Reduces active power consumption by scaling voltage and frequency accordingly to required MIPs.

Dynamic Process Temperature Compensation (DPTC)

Reduces active power consumption by adjusting supply voltage accordingly specific process cases, the manner in which the chip was fabricated, and the ambient temperature.

State Retention Voltage (SRV)

Reduces static power consumption by decreasing supply voltage to minimum State Retention level. Chip is not functional in this mode.

Active Well-Bias (AWB)

Reduces static power consumption by applying back bias on transistors. AWB can be applied on ARM11P. ARM11P is not functional when AWB is applied.

L2 Cache Power Gating

Reduces static power consumption by eliminating L2 Cache leakage.

ARM11P Power Gating

Reduces static power consumption by eliminating ARM11P leakage.

3.5.4 DVFS Support

The CCM enables simple S/W dynamic voltage frequency scaling. The frequency of the MCU clock domain and the voltage of the chip can be changed on the fly while all modules (including the MCU) continue their normal operation. The voltage of the chip can be changed by setting the DVS0 and DVS1 pins (connected to PMIC). The frequency of the MCU clock domain can be changed by switching to an alternate PLL clock (MCU or SR PLLs), already locked at a specific frequency, or by merely changing the post dividers division factors. Figure 3-32 shows the flow of frequency/voltage scaling.

The DVFS load tracking block enables hardware tracking on the MCU load and a generation of an interrupt when a frequency change is requested. Figure 3-33 shows the DVFS state machine.

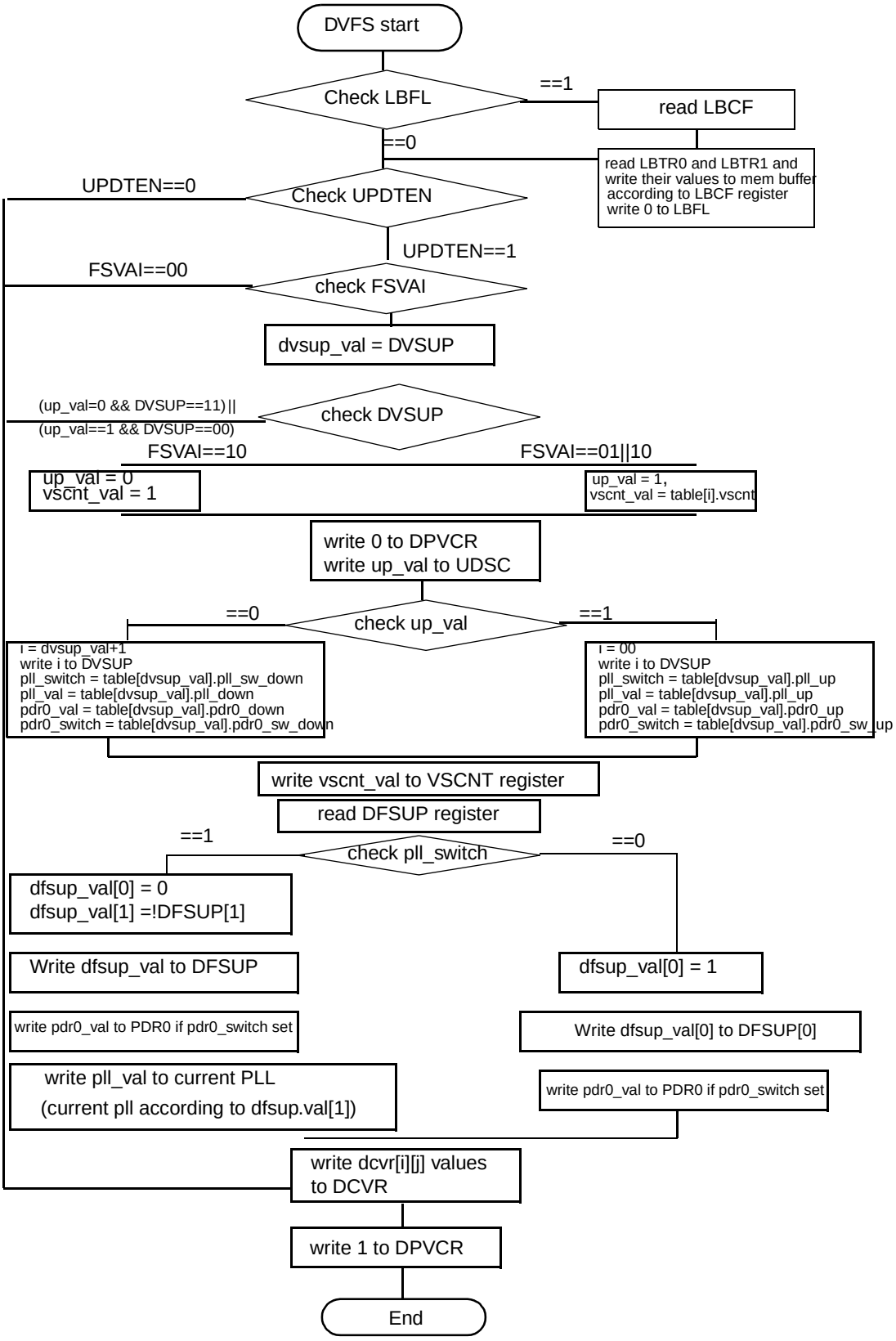


Figure 3-32. Frequency/Voltage Switching Procedure

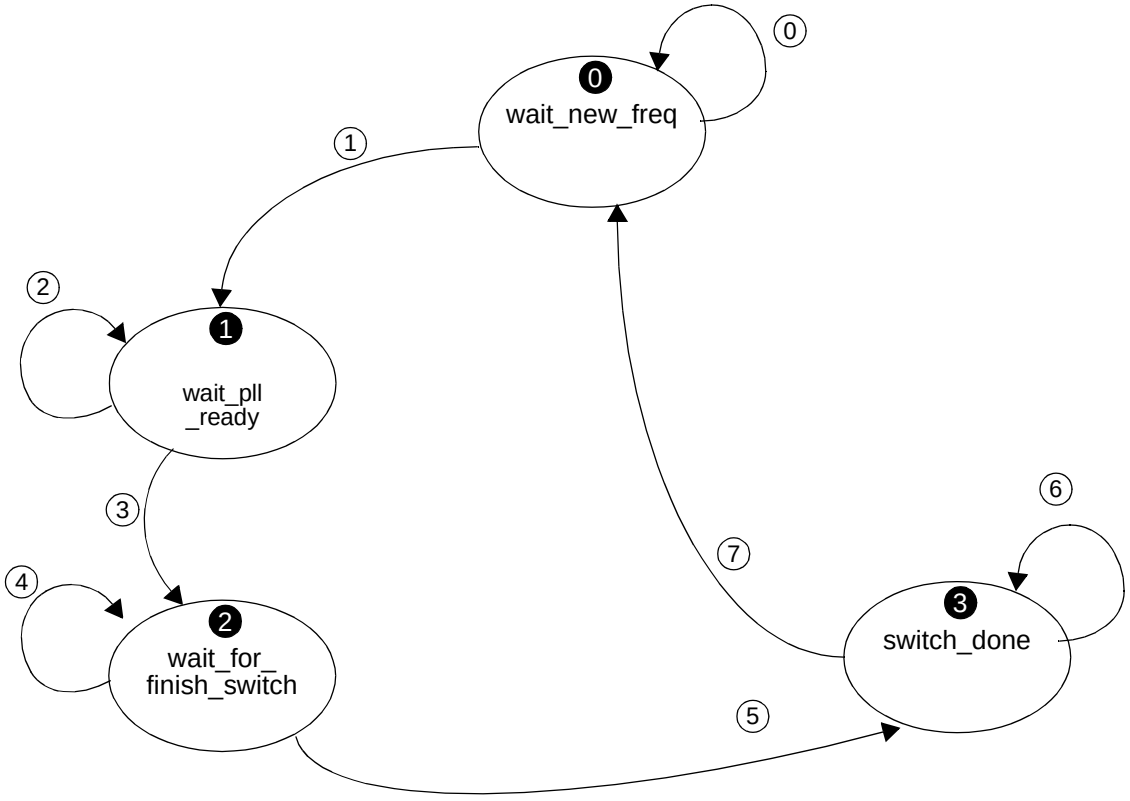


Figure 3-33. DVFS State Machine

Table 3-28. DVFS Transition Descriptions

Transition	Transition Condition	Output
0	not (1)	volt_count disabled dvfs_cnt_rst negated UPDTEN set to 0
1	new pll reg values written, pll restarted and pll change required or post-dividers only updated (if dfsup[0] ==1)	
2	not (3)	if freq_up => voltage_count enabled and DVSUP values written to PMIC pins
3	(pll lock finished | postdiv only update) and (voltage ready {voltage counter ended} in case of freq up)	
4	not (5)	volt_count disabled **Note:** plls are switching at all_counters_in_zero of post-divider
5	(switch to pll finished) | (postdiv only updated)	
6	1 clk delay in state	if freq_dw => DVSUP values are written to PMIC pins
7	always	

3.5.4.1 DVFS Load Tracking Block

The DVFS load tracking block includes the following features:

- Configurable include/exclude of input signals:
  - — ARM standby signal (idle/non-idle)
  - — 16 general purpose load tracking signals
- Configurable weight and edge/level detection of each input signal/set of signals.
- Configurable generated clocks and averaging time slicing (respond time).
- Configurable panic mode respond logic (for frequency up).
- Programmable buffer for last 4,8,12, or 16 load tracking samples. Based on value in LBCF in PMCR0. There is also a buffer full signal—LBFL.

The general purpose load tracking signals are (dvfs_w_sigs[15:0]) are connected at the device level to the signals in Table 3-29. Figure 3-34 shows the DVFS load tracking module block diagram.

Table 3-29. dvfs_w_sigs Connectivity

Bit #	Signal	Functionality
15:12	ccm_dvgp	Software controllable general purpose bits from the CCM
11	ipi_int_ipu_func	Interrupt line from IPU
10	$\overline{\text{ipi\_gpio1\_int0}}$	Interrupt line from GPIO
9	arm11p_fiq_b_rbt_gated	ARM fast interrupt
8	arm11p_irq_b_rbt_gated	ARM normal interrupt
7	m3if_hready_m7	Hready signal of M3IF's master #7 (IPU)
6	m3if_hready_m6	Hready signal of M3IF's master #6 (IPU)
5	m3if_hready_m5	Hready signal of M3IF's master #5 (mpeg4_vga_encoder)
4	m3if_hready_m4	Hready signal of M3IF's master #4 (SDMA)
3	m3if_hready_m3	Hready signal of M3IF's master #3 (MAX)
2	$\overline{\text{mbx\_mbxclkgate}}$	Hready signal of M3IF's master #2 (MBX)
1	m3if_hready_m1	Hready signal of M3IF's master #1 (L2 Cache)
0	m3if_hready_m0_buf	Hready signal of M3IF's master #0 (L2 Cache)

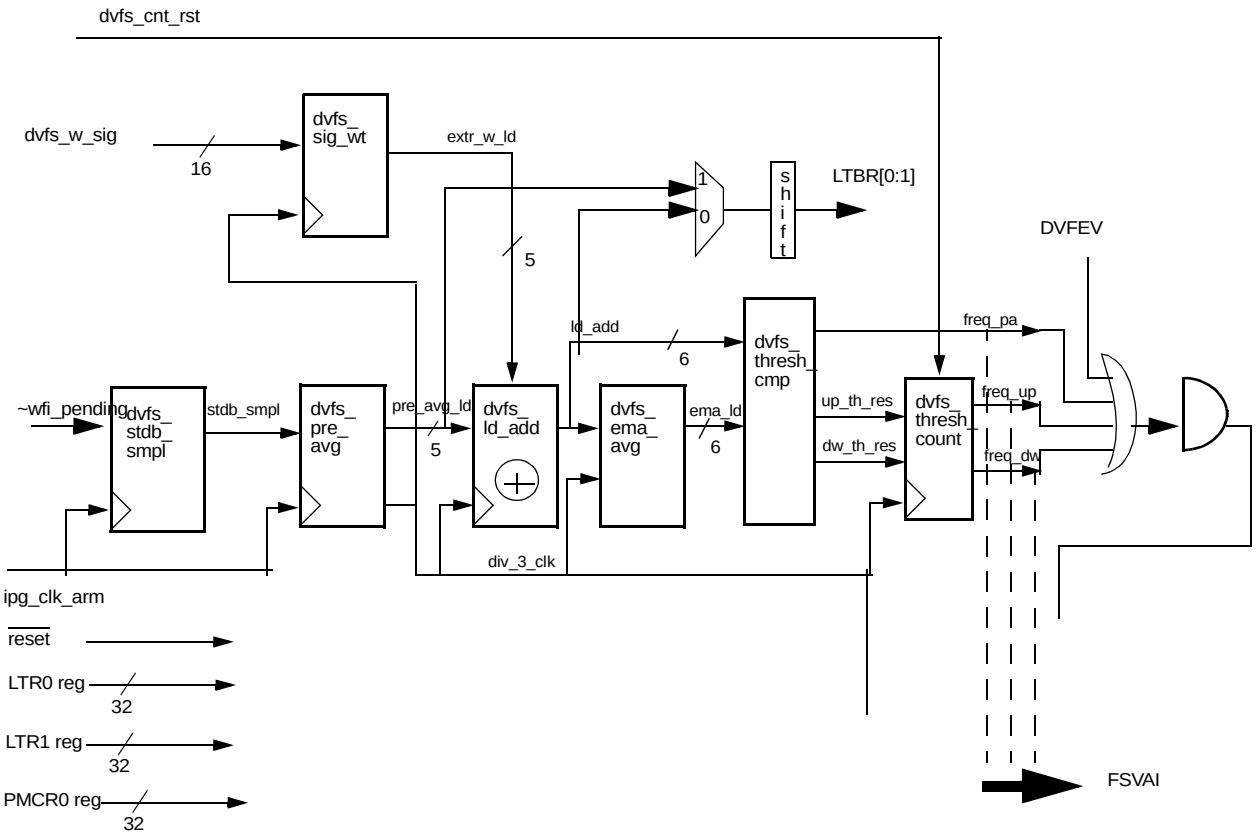


Figure 3-34. DVFS Load Tracking Module Block Diagram

3.5.4.1.1 Load Tracking Buffer Register

The purpose of the load tracking buffer register is to save last 16 samples of the tracked load (before EMA operation). Hence, the upper four bits of ld_add signal (ld_add[5:2]) are saved continuously, overwriting each time the last sample. Each save is carried upon detecting an edge of the div_3_clk signal.

3.5.5 DPTC support

The DPTC (Dynamic Process and Temperature Compensation) module is a power management module. The purpose of the DPTC module is to detect the minimum operation voltage for the IC, taking into account the extreme of processing activity and ambient temperature for a given frequency. It inputs predefined values for process speed performance measurement and generates an interrupt if the value of the supply voltage must be updated.

Figure 3-35 is a block diagram of the DPTC module.

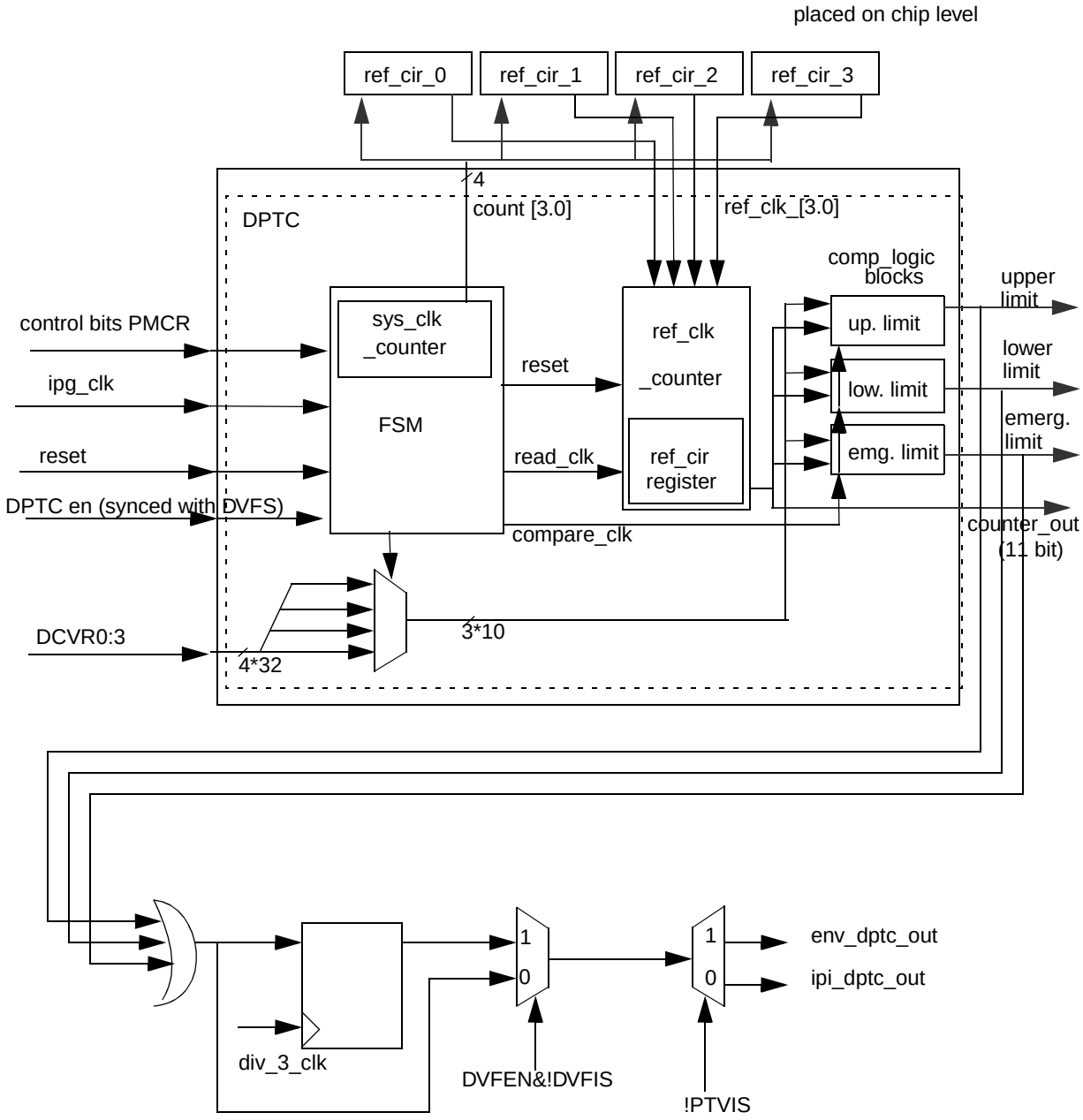


Figure 3-35. DPTC Block Diagram

3.5.5.1 Blocks Description

3.5.5.1.1 FSM Block

The FSM block is responsible for the internal control signals generation. For this purpose it includes a sys_clk_counter sub-block that creates a long-interval count enable signal, based on the system clock signal, and received from the sys_clk input pin. Table 3-30 provides the low-power FSM transition descriptions.

Table 3-30. Low Power FSM Transition Descriptions

Transition	Input	Output
0	ARM not in standby mode—arm_clk_off asserted||(WIMRandWAMO=11), reset, voltage valid interrupt (ccm_int_pmic) and valid clocks	* Enable clocks to ARM and peripherals. * Deassert stop, wait, and doze signals—ccm_ipg_stop, ccm_ipg_doze, ccm_ipg_wait. * Enable interrupts: deassert ccm_int_holdoff. * Clear LPM bits. * SDRAM (EMI) not in self refresh mode—deassert ccm_sd_lpmd * No request to stop MAX clock—deassert max_halt_req * No request for voltage change—deassert ccm_vstby_pmic * Well-Bias disabled—deassert ccm_wb_en, wbcp_fbd. * No ARM mem I/F power down—deassert ccm_mcu_mem_pwrdn * ARM mem. I/F enabled—ccm_mcu_mem_on * Enable ARM L2 cache memories—assert ccm_mcu_l2d_oe, deassert ccm_mcu_l2d_sfen. * No power down of L2 cache—deassert ccm_l2pg_pmic * Disable power gating of ARM—deassert mcu_power_gate_en * No reset to ARM—deassert $\overline{\text{mcu\_reset\_out}}$ * No power down to ARM domain—deassert ccm_mcupg_pmic * According to the clock source— If CKIH then enable clock amplifier and disable FPM: deassert ccm_ckih_camp_off, ccm_fpm_enable IF FPM then disable clock amplifier and enable FPM: assert ccm_ckih_camp_off, ccm_fpm_enable * Assert requests to IPU if the right glock gating is selected—assert ccm_ipu_stby_req CG(11)=00 in CGR1. Only after receiving ipu_stby_ack the IPU clock is stopped. * Assert request to SDMA if the right glock gating is selected—assert ccm_sdma_stby_req if CG(7)=00 in CGR0. Only after receiving sdma_stby_ack the SDMA clock is stopped * Assert request to EMI if the right clock gating is chosen—assert ccm_sd_lpmd if CG(4)=00 in CGR2. Only after receiving emi_sd_lpack with the EMI clocks be stopped. * No requests to rtic will be given since RTIC cannot be stopped at run mode—deassert ccm_rtic_stby_req.

Table 3-30. Low Power FSM Transition Descriptions (continued)

Transition	Input	Output
1	ARM in standby mode and LPM in WAIT—!arm_clk_off&LPM=WAIT	* Disable interrupts—assert ccm_int_holdoff. * Issue requests to IPU, SDMA, RTIC, and EMI if the clock gating is chosen for WAIT—assert ccm_ipu_stby_req if CG(11)=01 in CGR1, ccm_sdma_stby_req if CG(7)=01 in CGR0, ccm_rtic_stby_req if CG(5)=01 in CGR2. * Wait for acknowledges.
2	Received EMI acknowledges if needed (depending on clock gating)—emi_sd_lpack&(CG(4)=01 in CGR2)||~emi_sd_lpack&(CG(4)!=01)	* Assert the wait signal to peripherals—assert ccm_ipg_wait. * Wait 8 cycles. * Stop the clocks according to the CGR registers. * Enable interrupt—deassert ccm_int_holdoff.
3	ARM in standby mode and LPM in DOZE—!arm_clk_off&LPM=DOZE	* Disable interrupts—assert ccm_int_holdoff. * Issue requests to IPU, SDMA, RTIC, EMI and MAX if the clock gating is chosen for DOZE—assert ccm_ipu_stby_req if CG(11)=01||10 in CGR1, ccm_sdma_stby_req if CG(7)=01||10 in CGR0, ccm_rtic_stby_req if CG(5)=01||10 in CGR2, max_halt_request.
4	Received EMI acknowledges if needed (depending on clock gating)—emi_sd_lpack&(CG(4)=01||10 in CGR2)||~emi_sd_lpack&(CG(4)!=01||10)	* Assert doze signal to peripheral—assert ccm_ipg_doze. * Wait 8 cycles. * Stop the clock according to the CGR registers. * IF WBEN bit is enabled then well-bias will be enabled and ARM memory I/F will be powered down—assert mcu_wb_en, wbcp_fbd, ccm_mcu_mem_pwrdn, deassert ccm_mcu_mem_on. * Enable interrupts—deassert ccm_int_holdoff.

Table 3-30. Low Power FSM Transition Descriptions (continued)

Transition	Input	Output
5	ARM not in standby mode or wakeup interrupt—arm_clk_off|| WIMR&(WAMO=00||01||10)	** If LPM=DOZE then well-bias counter starts according to WBCN value. * Then well-bias will be disabled and ARM memory I/F will be enabled—deassert mcu_wb_en, ccm_mcu_mem_pwrdn, after WB counter (WBCN) deassert wbcp_fbd, assert ccm_mcu_mem_on. ** IF LPM=DSM||SR * If VSTBY bit set remove PMIC regulator from standby—deassert ccm_vstby_pmic. * If L2PG is set the power up L2 cache, a flash should be performed by MCU executing WFI command—deassert ccm_l2pg_pmic. ccm_mcu_l2d_sfen will be asserted and ccm_mcu_l2d_oe will be deasserted. * If LPM = DSM request to power up ARM—deassert ccm_mcupg_pmic. * Then if SBYCS is not set enable clock generator—IF CKIH deassert ccm_ckih_camp_off if FPM assert ccm_fpm_enable. * If WBEN is set disable well-bias and ARM mem I/F will be powered up—deassert mcu_wb_en, ccm_mcu_mem_pwrdn. * After OSCNT is over (if CKIH) or FPM lock (if FPM) MPE, UPE and SPE bits will be set, and the relevant PLLs will start. * If LPM = DSM release the ARM power gating in the chip, release ARM reset—deassert mcu_power_gate_en, assert $\overline{\text{mcu\_reset\_out}}$. * If WBEN is set and —If SBYCS is set then well-bias counter starts according to value in WBCN —If SBYCS is not set then continue. After either OSCNT is complete (if CKIH is chosen) or FPM is enabled (if FPM is chosen). * Then the final stages of releasing the well-bias and enabling the power of the ARM mem I/F occur—deassert wbcp_fbd, assert ccm_mcu_mem_on. * After PLLs are set and the pll_lock_ready flag is set (disregard if MPE is set to “0”) remove SDRAM (EMI) from self refresh mode and start EMI lock—deassert ccm_sd_lpmd.
6	ARM is standby and LPM in DSM||SR —arm_clk_off&LPM=DSM||SR	* Disable interrupts—assert ccm_int_holdoff. * Issue requests to IPU, SDMA, RTIC, and MAX—assert ccm_ipu_stby_req, ccm_sdma_stby_req, ccm_rtic_stby_req, max_halt_request.
7	Received acknowledges—ipu_stby_ack, sdma_stby_ack, rtic_stby_ack, max_halted	* Issue request to EMI for self refresh mode—assert ccm_sd_lpmd.

Table 3-30. Low Power FSM Transition Descriptions (continued)

Transition	Input	Output
8	Received EMI acknowledge—emi_sd_lpack	* Assert stop signal—assert ccm_ipg_stop. * Wait 8 cycles. * Stop clocks to MCU and peripherals. * Stop PLL's if any where enabled after there clock is held high—deassert ccm_mpl_cpen, ccm_spl_cpen, ccm_upl_cpen after mpl_dpdgck_clk, upl_dpdgck_clk, spl_dpdgck_clk are high respectively. * If SBYCS bit is set to "0" disable clock source generator—IF CKIH then assert ccm_ckih_camp_off, IF FPM deassert ccm_fpm_enable. * If WBEN bit is enabled then well-bias will be enabled—assert mcu_wb_en, wbcp_fbd. * If VSTBY bit is enabled then PMIC regulators are placed in standby—assert ccm_vstby_pmic. * If L2PG is set L2 cache memory will be shut down by deasserting ccm_mcu_l2d_oe and at least 2 cycles later ccm_mcu_l2d_sfen and then assert ccm_l2pg_pmic. * If LPM = DSM reset the ARM, turn on ARM power gating in the chip and power down the ARM mem I/F—deassert ccm_mcu_mem_pwrdn and then assert ccm_mcu_mem_on, assert mcu_power_gate_en, ccm_mcupg_pmic, deassert mcu_reset_out
9	Software request	No change from DSM||SR
10	Received acknowledges—ipu_stby_ack&(CG(11)=01 in CGR1), sdma_stby_ack&(CG(7)=01 in CGR0), rtic_stby_ack&(CG(5)=01 in CGR2)	* If EMI clock gating issue request—assert ccm_sd_lpmd if CG(4)=01 in CGR2. * If EMI not clock gated continue to next stage.
11	Received acknowledges—ipu_stby_ack&(CG(11) =01||10 in CGR1), sdma_stby_ack&(CG(7)=01||10 in CGR0), rtic_stby_ack&(CG(5)=01||10 in CGR2), max_halted	* If EMI clock gating issue request—assert ccm_sd_lpmd if CG(4)=01||10 in CGR2. * If EMI not clock gated continue to next stage.

Figure 3-36 shows the FSM loop.

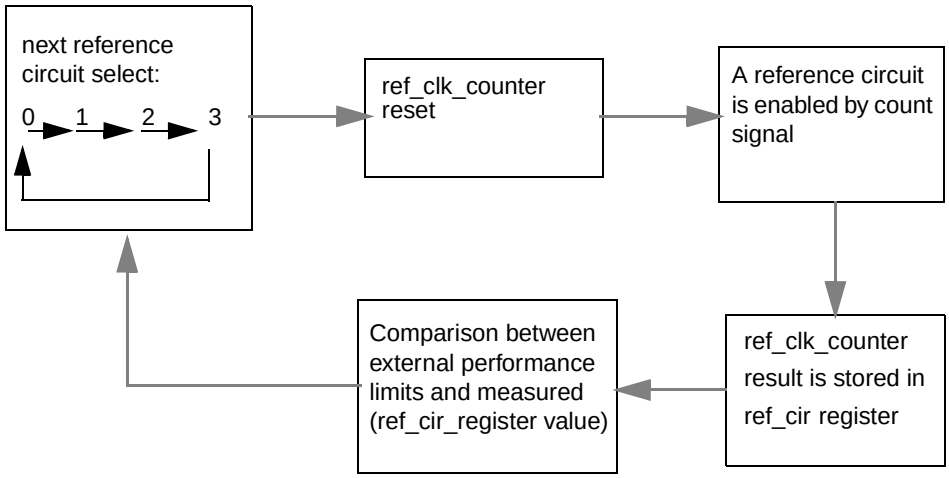


Figure 3-36. FSM Control Loop

Table 3-31 presents the FSM Control Loop stages.

Table 3-31. FSM Control Loop Stages

Operation	Explanation	Active Signals	Level	Misc.
Reference circuit select	Reference circuit 0–3 is selected	ref_circuit_select	2 bits value	
ref_clk_counter reset	Previous reference circuit counting result is deleted from ref_circuit_counter	$\overline{\text{ref\_count\_reset}}$	low	counter_out value is reset
Reference circuit performance evaluation	Amount of ref circuit clock cycles is counted by ref_clk_counter during defined number of sys clock	count	high	
Read reference circuit count result	Value of ref_circuit_counter is read to register	read_enable_clk	posedge	counter_out value is valid
Performance comparison: measured to limits	Reference circuit performance is compared to the limits from input register	comp_enable_clk	posedge	

Figure 3-37 shows the FSM control signals waveforms.

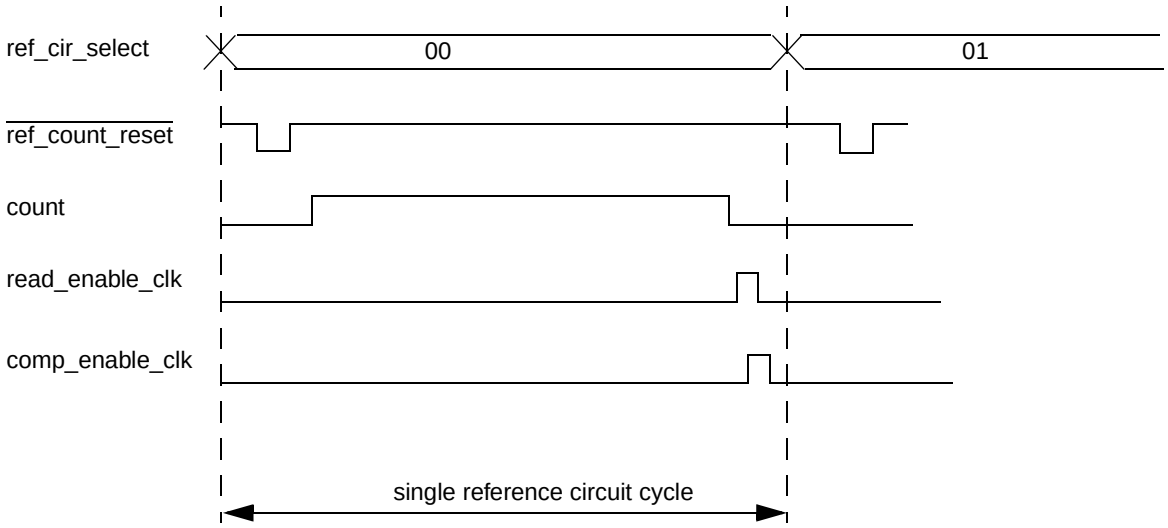


Figure 3-37. FSM Control Signals Waveforms

All signals are generated based on sys_clk. To create a long “count” signal, a sub-block sys_clk_counter is used. The sys_clk_counter counts the system clock and stops when the max value is reached. The max value for sys_clk_counter is set by the SCR bit in the status control register: ‘1’ is equivalent for 512 system clocks and ‘0’ is equivalent for 256 system clocks.

3.5.5.1.2 Ref_clk_counter Block

The ref_clk_counter block acts as a counter for the reference clocks. These clocks are generated by ref_cir blocks and includes the ref_cir_register (11 bit) for storing the last valid value of the counter.

Ref_cir_counter receives the following signals: $\overline{\text{ref\_clk\_reset}}$, read_enable_clk, ref_clk_0, ref_clk_1, ref_clk_2, ref_clk_3. The output (11 bits) is sent to the counter_out bus.

As long as the ref_clk_counter is not in the reset state (defined by $\overline{\text{ref\_clk\_reset}}$ active low signal), counting is enabled. Counter is an 11 bit counter, stored in the ref_cir_register.

3.5.5.1.3 Comp_logic Blocks

The comparison blocks are responsible for the comparison between performance values from performance limit registers and the actual performance as measured by the ref_clk_counter block. Each one of the compared signals are 11 bits (due to the fact that the input performance limits are extended by a “0” as the MSB).

The Comp_logic block inputs the following:

- Performance limit value (10 bits) (the desired performance limit value)
- Actual performance limit value (11 bits) (the measured performance limit value)
- ref_cir_select (2 bits)—index of currently active reference circuit
- RCSE (reference circuits selective enable) (4 bits) from status control register—to get information about enabled/disabled reference circuits.

- comp_enable_clk—clock of comparison enable.
- Limit type (1 bit)—hard config bit, connected to ground or supply.

The RCSE data is essential for taking/not taking in the account the result of comparison of the currently active reference circuit. For the disabled reference circuit, the comparison result should be considered as “right.” A critical situation can occur when all reference circuits are disabled (no real result—all artificial)—in this case all the outputs should be reset to “0” and the “error” bit in the control status register must be set.

Comp_logic includes FFs (one for single reference circuit), saving the comparison result for each reference circuit. The FFs data is used to generate the output signal of the comp_logic block. The data in the FF is continuously renewed so long as the appropriate reference circuit has completed its cycle and the data is valid. The FFs data is reset when the DPTC block is disabled or the external reset signal is activated.

Comp_logic blocks can be operated as “high_limit” type and “low_limit” type. For the “high_limit” type active output, the comparison of ALL reference circuits should exceed the high performance limit. For the “low_limit” type active output, *one* or more reference circuit comparison should exceed the low_limit /emergency_limit value.

3.5.5.1.4 Ref_cir Blocks

The ref_cir blocks are “reference circuit” units, constructed using a ring oscillator with an integrated clock gating cell. The inverting stages components and their values vary according to the type of ring oscillator. Ref_cir blocks are built to optimize speed and performance of the most critical paths in the chip for a given process/temperature.

3.5.5.1.5 Initialization Information

In the control status register, at least one of the RCSE bits should be set (to 1). Otherwise, the error bit will be activated and the output “limit” signals will be atrophied.

3.5.6 Synchronization Between DVFS and DPTC

To avoid a situation of DPTC trying to update events during voltage update done by DVFS a synchronization scheme has been created.

The DPTC machine can work in case the DPTC is enabled and the voltage is valid (DPVV = 1) and there is no voltage change request (DPVCR = 1).

Figure 3-38 shows the DVFS—DPTC synchronization.

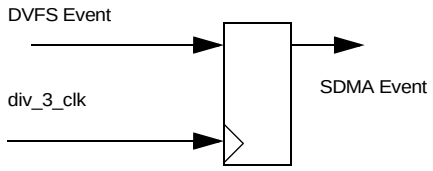


Figure 3-38. DVFS—DPTC Synchronization

3.5.7 Well-Bias Support

Well-bias is implemented with an on-chip charge pump. When well-bias is activated, the voltage of wells will be pumped to optimal values to enable minimum leakage. Parameters of the charge pump can be controlled in S/W by setting configuration bits in the PMCR register.

3.5.7.1 ARM Platform Well-Bias Activating

Well-bias of the ARM domain can be activated in Doze or State Retention modes by asserting the mcu_wb_en signal. The ARM Platform contains memory blocks with stdVt transistors in its interface sector. Well-Bias eliminates the leakage from these transistors by power gating of this interface—there is, therefore, no need for any data saving/restoring. Memories interface is power gated by asserting the ccm_mcu_mem_pwdn signal and negating the ccm_mcu_mem_on signal.

3.5.7.2 ARM Platform Well-Bias Deactivating

When ARM deasserts the arm_clk_off signal, the CCM deasserts mcu_wb_en signal. Clock will be provided to ARM11P after the time counted by the well-bias counter has elapsed (threshold is defined in the WBCN bits), or after any other event that lasts longer than 10µs. ARM11P memories interface power supply is restored by negation of the ccm_mcu_mem_pwdn signal together with mcu_wb_en negation and assertion of the ccm_mcu_mem_on signal upon clock restoration.

3.5.8 State Retention Voltage Support

In State Retention and Deep Sleep modes, the supply voltage of the i.MX31 and i.MX31L processors, if not power-gated, can be reduced to a State Retention value, which enables a reduction of chip leakage current while embedded memory state is dormant. Cores and modules using embedded memory are not functional in this mode. Voltage will be reduced by PMIC after assertion of the ccm_vstby_pmic signal, which in turn asserts the VSTBY I/O pin. Voltage will be restored to previous value upon negation of this signal. Valid voltage is indicated when the PMIC interrupt is asserted.

3.5.9 L2 Cache Power Gating Support

L2 cache power gating can be applied in State Retention and Deep Sleep modes. L2 cache power gating is implemented by negation of the ccm_l2d_oe signal and assertion of the ccm_l2d_sfen signal (after two cycles of the mcu_clk have passed). Supply of L2 cache is power gated by assertion of the ccm_l2pg_pmic signal, which in turn asserts the L2PG I/O pin. Upon exiting low power mode, the L2 cache supply is restored by negation of the ccm_l2pg_pmic signal and after voltage is valid and clocks are restored, negation of the ccm_l2d_sfen signal and assertion of the ccm_l2d_oe signal.

3.5.10 ARM Platform Power Gating Support

To provide interrupt monitoring during ARM11P power gating, the CCM contains a simplified interrupt controller—this consists of an “OR” between all interrupts with “enable” bit for each interrupt. Special power gasket cells are tied to all outputs of ARM11P to avoid contention during supply powering down

and restoring. ARM11P supply is power gated by assertion of the ccm_mcupg_pmic signal, which in turn asserts the MCUPG I/O pin. The reset_mcu signal is asserted during ARM11P power gating.

3.5.11 DFT Support

3.5.11.1 Overview

The CCM supports the following features to provide DFT support:

- Deterministic reset
- CCM Scan Div Mode
- CCM Long Chain Mode
- CCM SAF Scan Test Mode
- CCM Trans. System Mode
- CCM Trans. Last Shift Mode
- CCM Standalone Scan Mode
- Functional Mode

3.5.11.2 Deterministic Reset

The i.MX31 and i.MX31L processors support a deterministic reset test mode to enable synchronization between the tester clock and the EMI I/O port during test.

In this mode, the MCU PLL is applied in its phase lock mode and an equivalence path will be applied to get a minimum phase difference between the CKIH clock and the EMI BCLK IO pin. In this mode, system clocks will be frozen after the arm_active output of ARM is asserted, thus indicating that the first fetch has been executed by ARM. Clocks will be restored after the ipp_mcu_handshake signal assertion.

3.5.11.3 Clocks in Scan Divergence Mode

This mode is a failure analysis mode. In this mode all clocks are fetched from one source (mcu_pll). The CCM freezes all clocks after the scan divergence counter reaches the value defined by the ipt_scan_div_mcu_cnt[9:0] input signals. All the flops values then are shifted out.

3.5.11.4 Clocks in Long Chain Mode

In this mode, all the sequential elements except the BISTs are toggled in by the TCK clock. The BISTs are in Functional Mode and run on the functional clock frequency.

3.5.11.5 Clocks in SAF Scan Test Mode

In this mode, the CCM shunts test clocks that normally go to I/O pads directly (dividers and post-dividers in bypass state) to the operational clocks. See Figure 3-39 for its timing diagram.

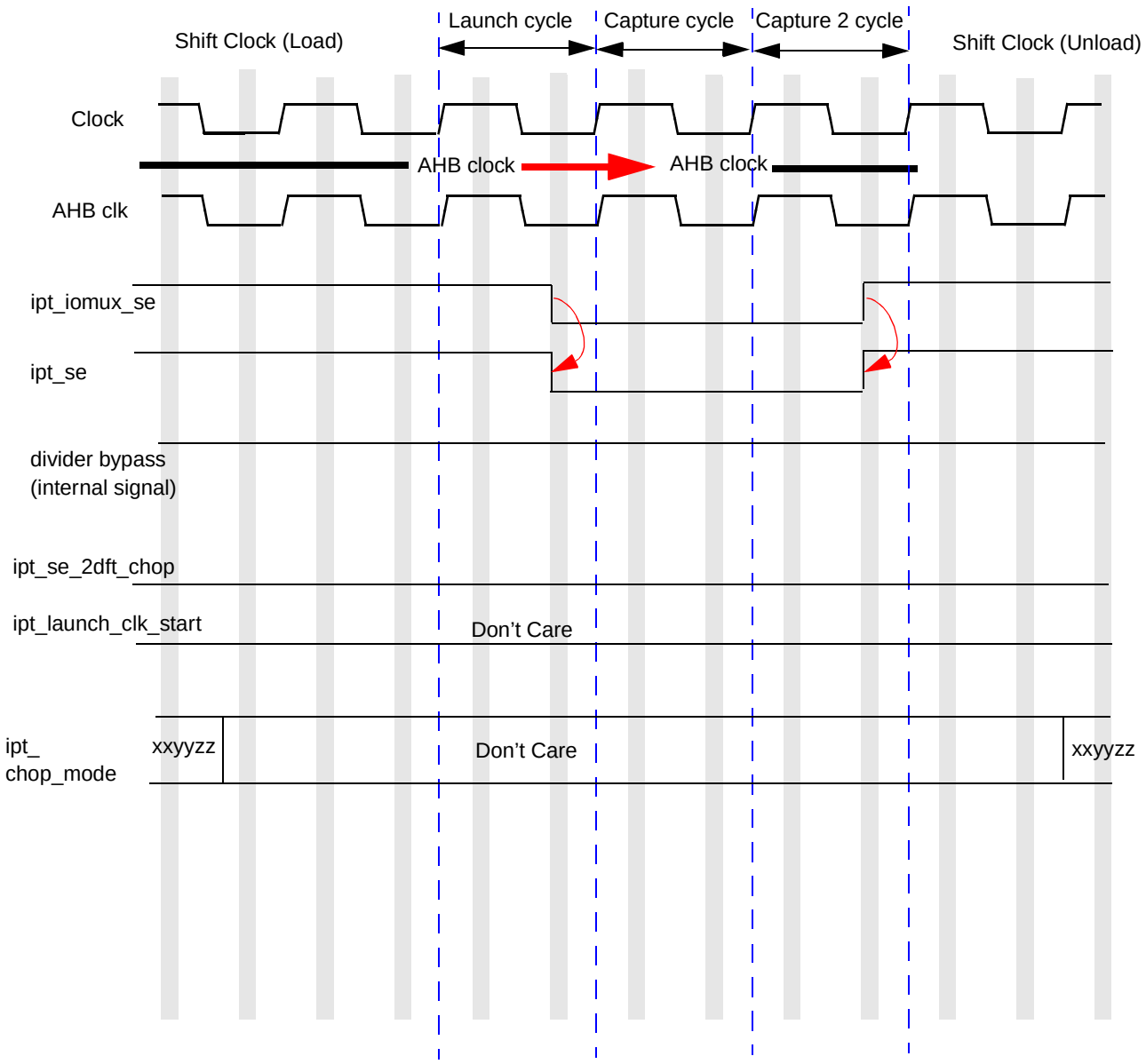


Figure 3-39. "Stuck-At Fault" (SAF) mode Launch and Capture Timing Diagram

3.5.11.6 Clocks in Transition Mode

In Transition mode, launch and capture clock cycles are generated at the functional frequency. See Figure 3-40 for its timing diagram.

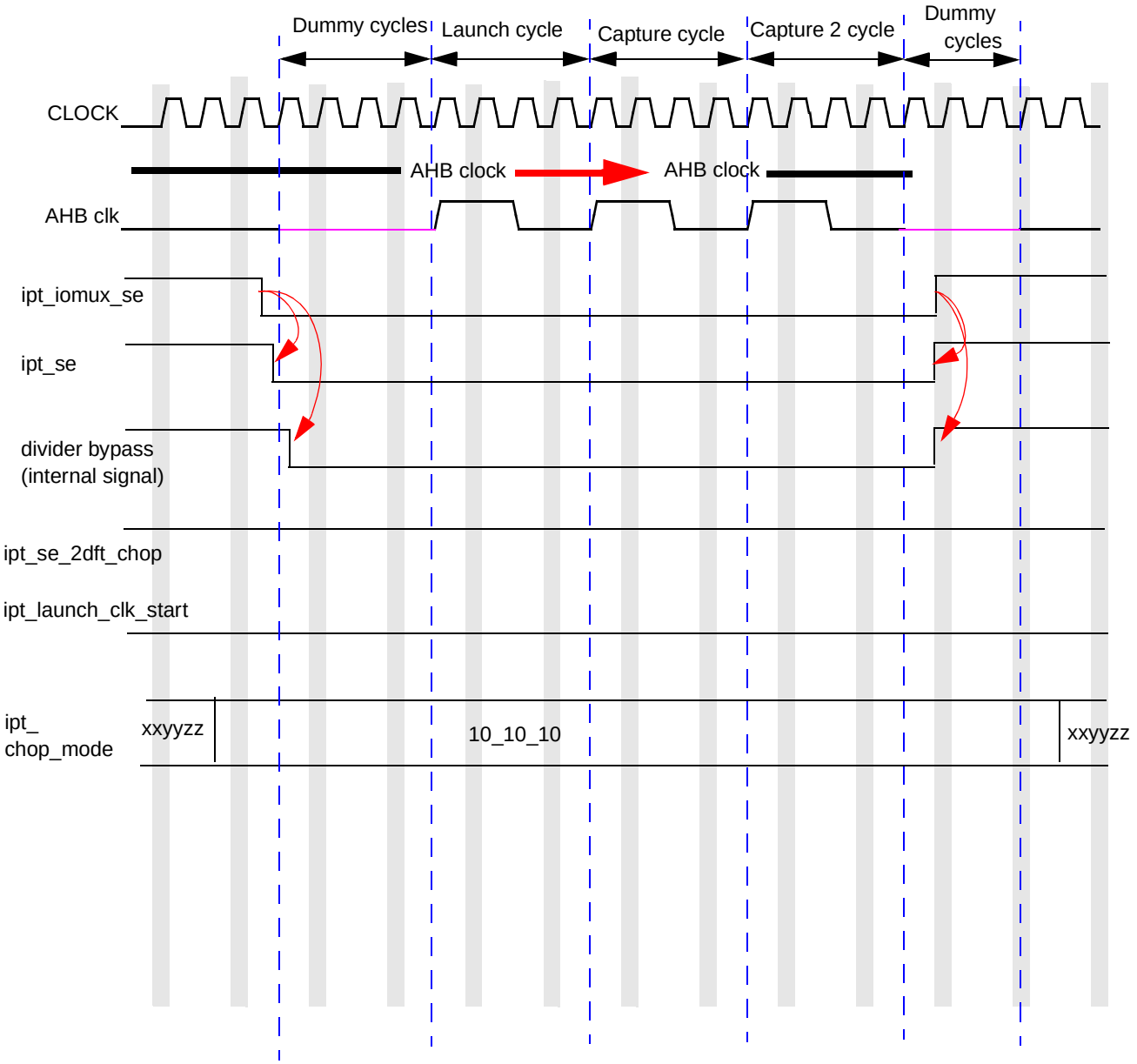


Figure 3-40. Transition Mode Launch and Capture Timing Diagram

3.5.11.7 Clocks in Transition Last Shift Mode

In the Transition Last Shift mode, launch and capture clock cycles are generated at the functional frequency (like Transition mode), but the scan enable signal is asserted after launching the clock cycle. See Figure 3-41 for its timing diagram.

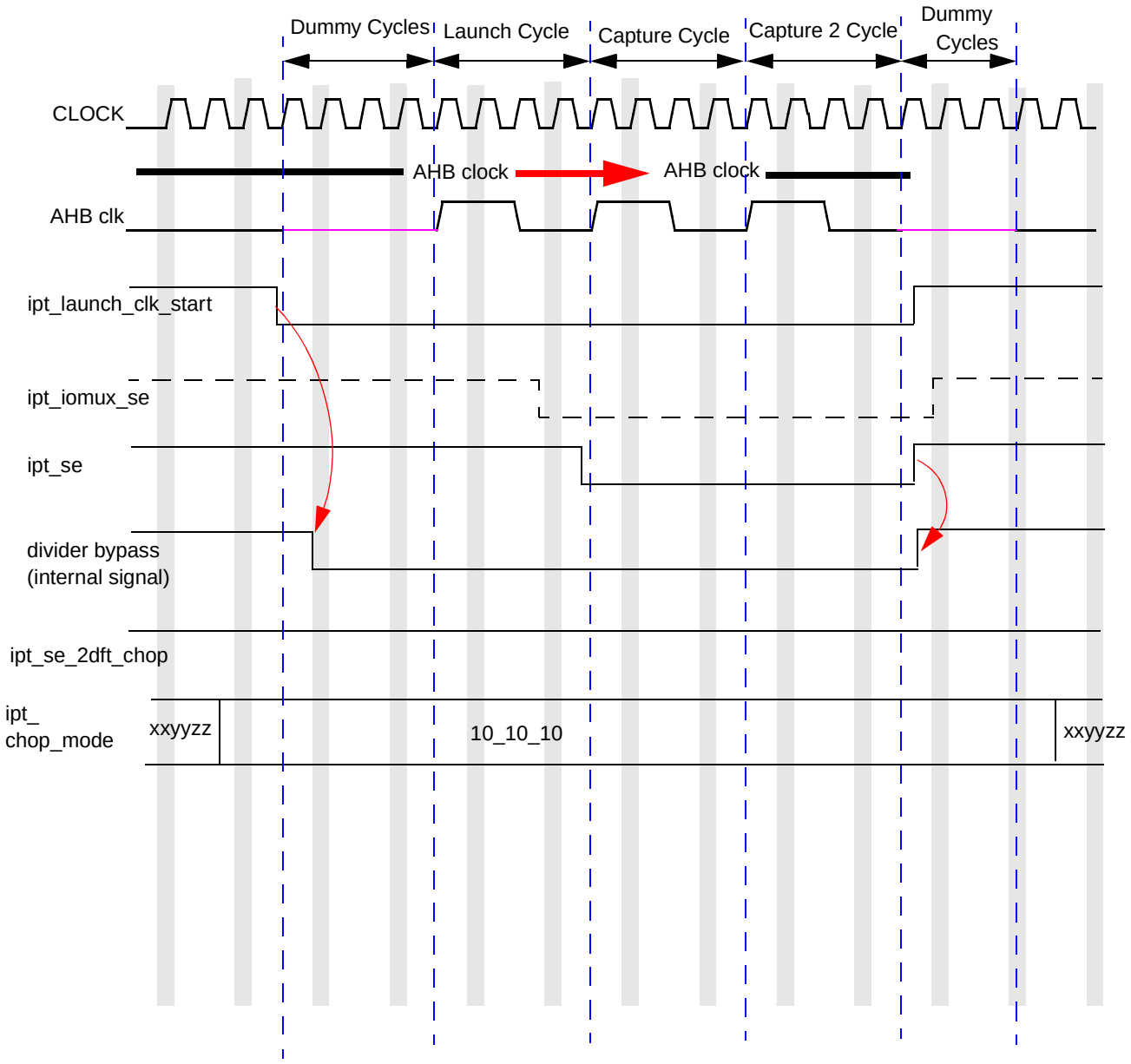


Figure 3-41. Transition Last Shift Mode Launch and Capture Timing Diagram

3.5.11.8 Clocks in Standalone Scan Mode

This is a unique test mode that was created to check the Clock Control Module (CCM). During this mode, no provided clocks are needed, and as such the clocks can be checked in various configurations regardless of the clock output form. A safe mode ability can be achieved by blocking all CCM output clocks to avoid or cancel any spike issues.

3.6 Reset Controller

3.6.1 Functional Description of the Reset Module

The reset module controls or distributes all of the system reset signals used by the i.MX31 and i.MX31L. The reset module generates eight distinct events.

- mcu_reset_out signal is connected to ARM11P, RTIC, and SCC.
- reset_fuse signal is connected to laser fuses in L2 cache data array through the FSH module.
- ccm_por_reset signal is connected to TCU and WDOG modules.
- periph_reset_out signal is connected to all peripherals except EMI.
- ccm_pll_reset1 signal is connected to three PLLs and FPM.
- ccm_pll_reset2 signal is connected to modules engaged in boot and security (IIM, RNGA, RTC).
- emi_reset_ signal is connected to EMI.
- ect_reset signal is connected to the ECT module.

3.6.2 Reset Negation Sequence

The reset exit sequence is as follows:

1. Fuse reset is generated (after por reset only) and then follows:
   a) FPM will be enabled
   b) Boot mode pins value will be sampled
   c) Reset to PLLs, peripherals including IIM and JTAG will be negated. The negation is synchronized to CKIL.
2. After FPM lock ready flag is asserted, and if ipp_clkss=0 or the oscillator counter has completed counting, and if ipp_clkss=1, all three PLLs will be restarted.
3. After the MCU PLL lock ready flag is asserted, a clock to the CCM and IIM is provided as follows:
   a) Generate NFC boot signals, if boot is done from NF
4. After the counter has completed counting, provide clocks to peripherals and MCU as follows:
   a) After 16 mcu_clk cycles, negate the mcu_reset_out. The negation is synchronized to CKIL.
   b) ARM11 processor begins fetching code from the internal bootstrap ROM, sync flash or CS0 space. The memory location of the fetch depends on the configuration of the BOOT pins and the value of the TEST pin on the rising edge of the reset.

3.6.3 Global Reset

Global reset will reset cores and all peripherals. Any one of the following events or conditions can cause a global reset:

- An external qualified low condition on the por pin
- An external qualified low condition on the reset_in pin

- A low condition on wdog_mcu_reset
- A JTAG command

3.6.4 MCU Reset

Any qualified global reset signal resets the ARM11 Platform and all related peripherals to their default state. After the internal reset is deasserted, the ARM11 processor begins fetching code from the internal bootstrap ROM, sync flash or CS0 space. The memory location of the fetch depends on the configuration of the BOOT pins and the value of the TEST pin on the rising edge of the hreset.

3.6.5 Watchdog Resets

There are two different watchdog reset events that can occur: a time-out event or a software reset. A watchdog module reset causes a reset of the chip, and the CCM thereby generates a reset pulse. All registers in the CCM—except those that can be reset by por_reset signal only—will be reset to their default values. When the MCU comes out of reset it can check the “reset source” bits in the CCM module and execute a different power mode transition routine accordingly.

3.6.5.1 The Reset Negation Sequence on a Watchdog Event

The following defines the reset negation sequence on a watchdog event (see Figure 3-42 and Figure 3-43):

1. A reset that is received from the watchdog is synchronized by CKIL.
2. The exit of the reset is also synchronized by CKIL.
3. After the two samples of the reset, the reset is given to the PLLs and peripheral, causing the watchdog to release the reset.
4. After 1 CKIL, the reset given to the peripherals and PLLs will be released.

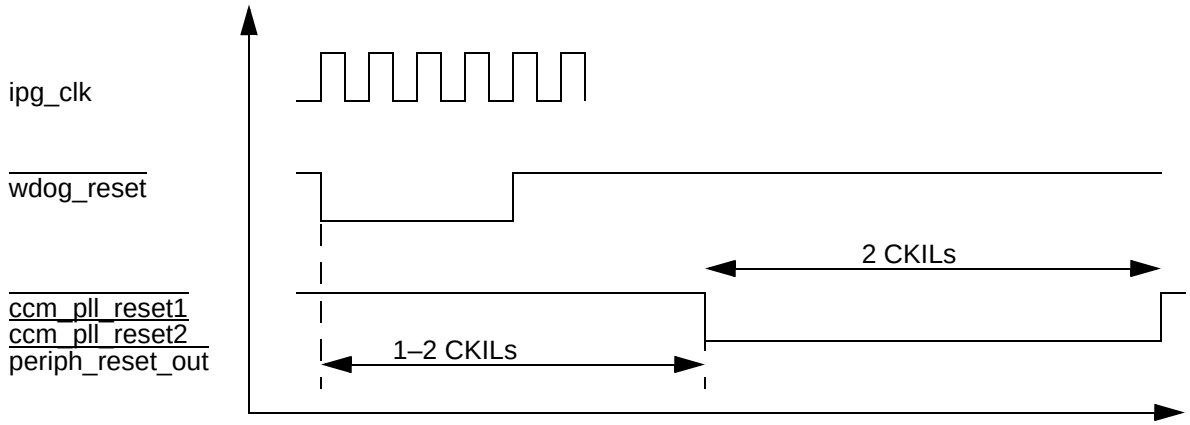


Figure 3-42. Watchdog Software Reset Diagram

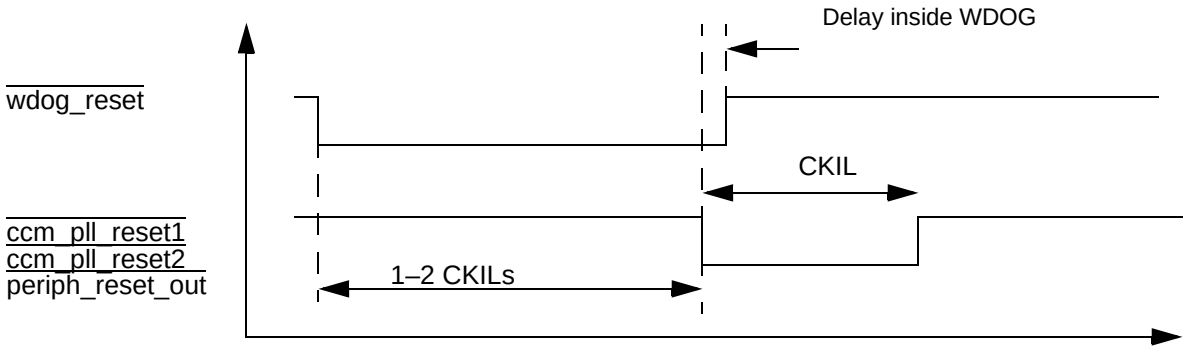


Figure 3-43. Watchdog Timeout Event Reset Diagram

3.6.6 S/W Peripheral Reset

MCU peripherals (except CCM and PLLs) can be reset by writing the PERES bit in the PCSR register.

3.6.7 JTAG S/W Reset

The i.MX31 and i.MX31L processors can be reset via S/W from a JTAG module, either by a signal or by a general purpose bit.

3.7 Power On Reset (Boot)

For more information, see Chapter 7, “i.MX31 and i.MX31L Boot.”

Chapter 4
Signal Multiplexing

This chapter identifies and describes the I/O module, which configures and controls the multiplexing of signals going to and from the i.MX31 or i.MX31L IC. This chapter also provides information about the I/O needed to configure and operate the I/O module.

NOTE

In previous versions of this manual, this chapter was named "*Signal Descriptions and Pin Assignments*."

4.1 Overview

This section provides an overview of the configuration and operation of the I/O MUX Controller (IOMUXC) module. The IOMUXC module, shown in Figure 4-1, is composed of three hardware blocks:

- **MUX**—Routes signals to and from the I/O
- **Buffers**—Logic level converters and drivers for interfacing the signals to the external contacts of the IC. I/O characteristics of each line are determined by these buffers.
- **Control Registers**
  - Software MUX Control (SW_MUX_CTL)—These register control the configuration of the signal lines connecting the On-chip peripherals to the contacts. See SW_MUX_CTL registers.
  - I/O Line Characteristics (SW_PAD_CTL)—These registers configure features such as pull-ups, drive strength, and hysteresis. See SW_PAD_CTL registers.

While there is interaction between the GPIO and the IOMUXC, the operation of the GPIO is described in Chapter 5, "General Purpose Input/Output (GPIO)."

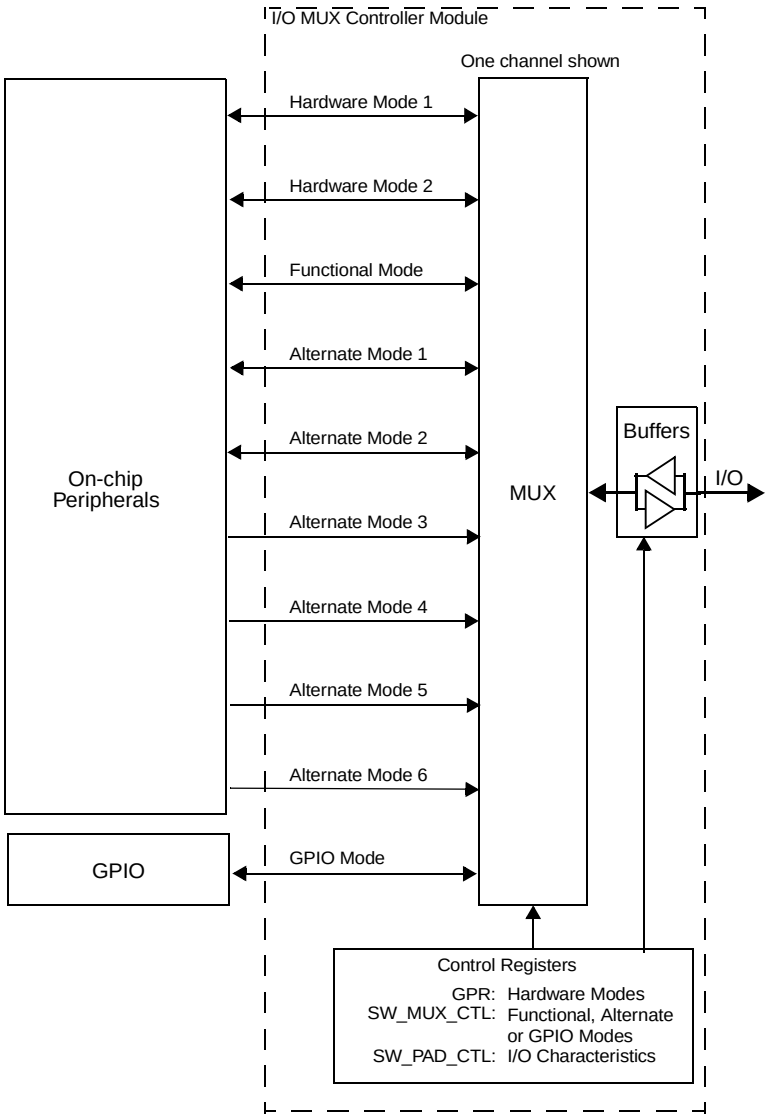


Figure 4-1. I/O Signal Multiplexing Block Diagram

4.2 IOMUX Controller (IOMUXC)

The IOMUXC registers control features in the IOMUXC. These registers perform the following tasks:

- Controls the IOMUX input and output paths.
- Controls I/O line properties such as pull-up, pull-down, and hysteresis.
- Monitors off-chip interrupts.

Figure 4-2 shows the registers that control the IOMUXC.

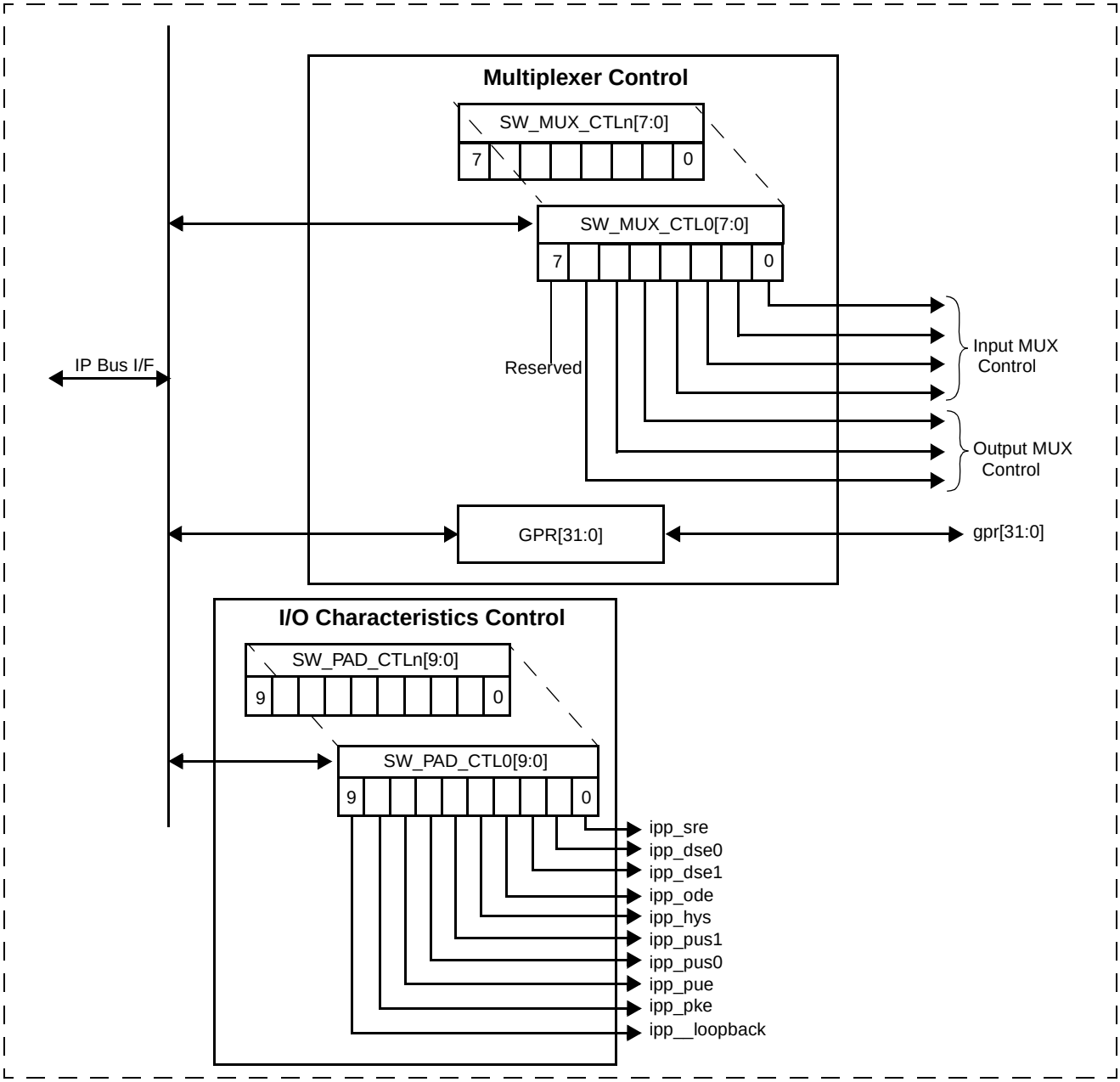


Figure 4-2. IOMUXC Registers

4.2.1 Software Multiplexor Control (SW_MUX_CTL)

The SW_MUX_CTL block consists of registers that control each IOMUX on an individual I/O line basis. The SW_MUX_CTL registers are partitioned into 4×8 bit fields. Each field is mapped to a specific I/O line and is partitioned as follows: four bits to control the input path, three bits to control the output path, and one reserved bit.

4.2.2 Software Pad Control (SW_PAD_CTL)

The SW_PAD_CTL block consists of registers that control the characteristics of each I/O line. The SW_PAD_CTL registers are partitioned into 3×9 bit fields, with each field mapped to a specific I/O line. Each field in the register controls several parameters of the I/O characteristics (for example, pull-up/pull-down, keeper, max drive, hysteresis, and open-drain).

4.3 Memory Map and Register Definition

Table 4-1 shows the IOMUXC memory map.

Table 4-1. IOMUX Memory Map

Address	Register	Access	Reset Value	Section/Page
0x43FA_C008	General Purpose Register (GPR)	R/W	0x0000_0000	4.3.2/4-5
0x43FA_C00C to 0x43FA_C150	Software MUX Control Register (SW_MUX_CTL)	R/W	See register descriptions.	4.3.3/4-9
0x43FA_C154 to 0x43FA_C308	Software Pad Control Register (SW_PAD_CTL)	R/W	See register descriptions.	4.3.7/4-78

4.3.1 Register Summary

Figure 4-3 shows the key to the register fields and Table 4-2 shows the register figure conventions.

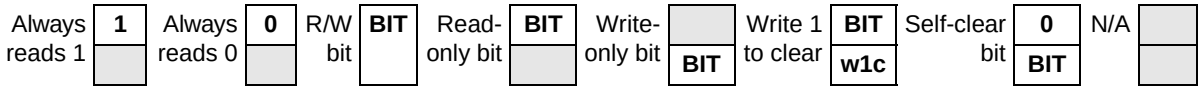


Figure 4-3. Key to Register Fields

Table 4-2. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.

Table 4-2. Register Figure Conventions (continued)

Convention	Description
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 4-3. IOMUXC Register Summary

Field		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x43FA_C008	R	GPR[31:0]															
	W																
	R																
	W																
0x43FA_C00C to 0x43FA_C150	R	SW_MUX_CTLx[31:0]															
	W																
	R																
	W																
0x43FA_C154 to 0x43FA_C308	R	SW_PAD_CTLx[31:0]															
	W																
	R																
	W																

4.3.2 General Purpose Register (GPR)

The General Purpose Register (GPR) is used to configure the IOMUXC. Bits in the GPR control combinations of predefined I/O type signals in the IOMUXC which are prioritized as Hardware modes 1 and 2 (HW1 and HW2). The priority of these hardware modes is shown in Table 4-7. Figure 4-4 shows the GPR register; Table 4-4 provides its field descriptions, and the bit definitions. Each bit controls a combination of I/O lines, functions or modes. The operation of each bit is defined in Table 4-5.

0x43FA_C008 Access: User Read/Write

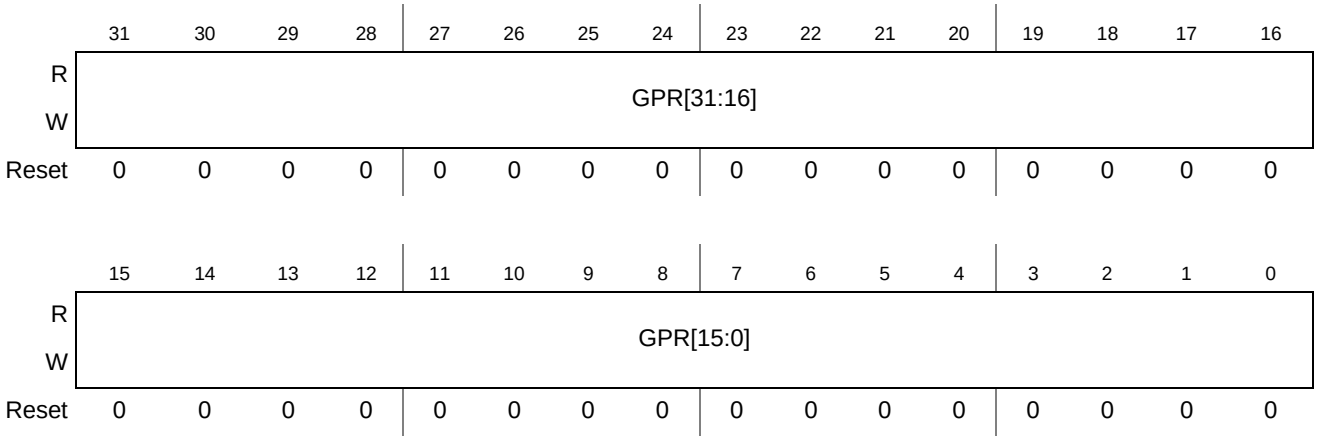


Figure 4-4. General Purpose Register (GPR)

Table 4-4. GPR Register Field Description

Name	Description
31–0 GPR[31:0]	Each bit in this register controls a combination of I/O Type signals or selection/modes. The signals are routed through the IOMUXC, providing 32 combinations of settings.

Table 4-5 shows which signals or hardware modes are controlled by each GPR bit. The multiplexing data about the signals being controlled is listed in Table 4-8 on page 41.

Table 4-5. Hardware Mode Definitions by GPR Bit Position

Bit	Definition	HW Mode	GPR bit = 0	GPR bit = 1
GPR[0]	Selects FIR or UART2 SDMA events	—	UART2 DMA requests are selected for DMA events 16 and 17.	FIR DMA requests are selected for DMA events 16 and 17.
GPR[1]	Forces all DDR type contacts to DDR Drive Strength setting.	—	Inactive (recommended)	Forces DDR type I/O contacts to DDR mode[1]
GPR[2]	Overrides the Full UART group default signals on page 4-63 with CSPI1	HW1	Inactive	Replaces Full UART Group with CSPI1 signals.
GPR[3]	Overrides the PWMO default signal with the ATA IORDY signal	HW1	Inactive	Enable ATA IORDY signal on PWMO contact.
GPR[4]	Overrides the USBH2 default signals with the following ATA signals • DA[2:0] • DMARQ • BUFFER_EN • INTRQ	HW1	Inactive	Enable ATA signals on USBH2 contacts
GPR[5]	Overrides the EMI Group NANDF default signals with ATA Data[13:7]	HW1	Inactive	Enable ATA DATA7-13 on NANDF contacts.

Table 4-5. Hardware Mode Definitions by GPR Bit Position (continued)

Bit	Definition	HW Mode	GPR bit = 0	GPR bit = 1
GPR[6]	Overrides the default EMI Group signals with the following ATA signals: • DA[2:0] • DMARQ • BUFFER_EN • INTRQ	HW2	Inactive	Enable ATA signals on NANDF contacts
GPR[7]	Override the default IPU (CSI) /I2C Group signals with the ATA Data.	HW1	Inactive	Enable DATA0-13 signals of ATA on IPU (CSI) and DATA14-15 on I2C
GPR[8]	Overrides the default AudioPort 3 and AudioPort6 Group signals with the ATA Data signals.	HW1	Inactive	Enable DATA7-10 signals of ATA on AudioPort3 and DATA11-13 on AudioPort6
GPR[9]	Overrides the default Timer/CSPI1 Group signals with ATA Data signals.	HW1	Inactive	Enable ATA DATA14-15 on Timer Group contacts and DATA0-6 on CSPI1 Group contacts.
GPR[10]	Overrides the default CSPI1 signals with the following ATA signals • DA[2:0] • DMARQ • BUFFER_EN • INTRQ	HW2	Inactive	Enable ATA signals on CSPI1 Group contacts
GPR[11]	Overrides the default AudioPort 3 and AudioPort6 Group signals with USBH2 signals.	HW2	Inactive	Enable USBH2 signals on AudioPort 3 and AudioPort6
GPR[12]	Selects either CSD0 or WEIM on EMI CS2 contact.	—	CSD0 is selected for EMI CS2	WEIM is selected for EMI CS2
GPR[13]	Selects either CSD1 or WEIM on EMI CS3 contact.	—	CSD1 is selected for EMI CS3	WEIM is selected for EMI CS3
GPR[14]	Selects either CSPI1 or UART3 DMA requests.	—	CSPI1 DMA request is selected for events 8 and 9.	UART3 DMA request is selected for events 8 and 9.
GPR[15]	Selects either External or MBX DMA requests.	—	External DMA Request2 is selected for event 14	MBX DMA request is selected for event 14.
GPR[16]	Enables Tamper Detect Logic.	—	Inactive	Tamper detect logic is enabled.
GPR[17]	Overrides default DSR_DCE1 signal with the USBOTG_DATA4 signal.	HW2	Inactive	Enable USBOTG_DATA4 on DSR_DCE1 contact.
GPR[18]	Overrides the default Full UART Group signals DCD_DCE1, DSR_DCE1 and RI_DCE1 with USBOTG_DATA[5:3].	HW2	Inactive	Enable USBOTG_DATA[5:3] on Full UART Group contacts

Table 4-5. Hardware Mode Definitions by GPR Bit Position (continued)

Bit	Definition	HW Mode	GPR bit = 0	GPR bit = 1
GPR[19]	Selects either SDHC1 or MSHC1 DMA requests.	—	SDHC1 DMA Request Is Selected for event 20	MSHC1 DMA request is selected for event 20.
GPR[20]	Selects either SDHC2 or MSHC2 DMA requests.	—	SDHC2 DMA Request Is Selected for event 21	MSHC2 DMA request is selected for event 21.
GPR[21]	Selects GPIO3_0 or SPLL_BYPASS_CLK. **Note:** SPLL_BYPASS_CLK is intended for testing purposes.	HW1	Inactive	Enable SPLL clock bypass through GPIO3_0 contact.
GPR[22]	Selects GPIO3_1 or UPLL_BYPASS_CLK. **Note:** UPLL_BYPASS_CLK is intended for manufacturing testing.	HW1	Inactive	Enable UPLL clock bypass through GPIO3_1 contact.
GPR[23]	When MSHC2 clock is selected using Alternate Mode 2, this bit controls the drive strength on PC_CD1_B as either a standard/high or maximum drive strength.	—	Standard or high drive strength	Maximum drive strength
GPR[24]	When MSHC2 clock is selected using Alternate Mode 2, this bit controls the output on PC_CD1_B as either a slow or fast slew rate signal	—	Slow slew rate	Fast slew rate
GPR[25]	Selects either CSPI3 or UART5 DMA requests.	—	CSPI3 DMA requests are selected for events 10 and 11.	UART5 DMA requests are selected for events 10 and 11.
GPR[26]	Overrides the default Keypad Group signals with the following ATA signals • DA[2:0] • DMARQ • BUFFER_EN • INTRQ	HW1	Inactive	Enable ATA signals on Keypad Group contacts
GPR[27]	Overrides the default signal SFS6 with USBH1_SUSPEND.	HW1	Inactive	Enable USBH1_SUSPEND signal on SFS6 contact.
GPR[28]	Enables USBOTG loopback **Note:** This is intended for manufacturing testing.	—	Inactive	Turn on sw_input_on (loopback) on some USBOTG contacts
GPR[29]	Enables USBH1 loopback **Note:** This is intended for manufacturing testing.	—	Inactive	Turn on sw_input_on (loopback) on some USBH1 contacts
GPR[30]	Enables USBH2 loopback **Note:** This is intended for manufacturing testing.	—	Inactive	Turn on sw_input_on (loopback) on some USBH2 contacts
GPR[31]	Enables DDR Drive Strength setting on the CLKO contact.	—	Inactive	Enable DDR mode on CLKO contact

[1] Setting this bit is not recommended as it may produce excessive overshoot.

4.3.3 Software Multiplexor Control Register (SW_MUX_CTL)

The SW_MUX_CTL register controls the IOMUX. Figure 4-5 describes an example generic SW_MUX_CTL register. Table 4-6 provides the register's field descriptions; Table 4-7 lists its priorities.

0x43FA_C00C
to
0x43FA_C150

Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		SW_MUX_CTL_SIGNAL 4								SW_MUX_CTL_SIGNAL 3						
W		SW_MUX_OUT_EN			MUX2_IN	MUX1_IN	FUNC_IN	GPIO_PSR/ISR		SW_MUX_OUT_EN			MUX2_IN	MUX1_IN	FUNC_IN	GPIO_PSR/ISR
Field Bits	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		SW_MUX_CTL_SIGNAL 2								SW_MUX_CTL_SIGNAL 1						
W		SW_MUX_OUT_EN			MUX2_IN	MUX1_IN	FUNC_IN	GPIO_PSR/ISR		SW_MUX_OUT_EN			MUX2_IN	MUX1_IN	FUNC_IN	GPIO_PSR/ISR
Field Bits	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

Figure 4-5. SW_MUX_CTL Register

Table 4-6. SW_MUX_CTL Register Field Descriptions

Register Bit	Bit Name	Setting
SW_MUX_CTL[31] SW_MUX_CTL[23] SW_MUX_CTL[15] SW_MUX_CTL[7]	—	Reserved

Table 4-6. SW_MUX_CTL Register Field Descriptions (continued)

Register Bit	Bit Name	Setting
SW_MUX_CTL[30:28] SW_MUX_CTL[22:20] SW_MUX_CTL[14:12] SW_MUX_CTL[6:4]	sw_mux_out_en	**MUX Output Selection** 000 GPIO DR (data register) output 001 Functional output 010 Alternate mode 1 output 011 Alternate mode 2 output 100 Alternate mode 3 output 101 Alternate mode 4 output 110 Alternate mode 5 output 111 Alternate mode 6 output
SW_MUX_CTL[27:24] SW_MUX_CTL[19:16] SW_MUX_CTL[11:8] SW_MUX_CTL[3:0]	mux2_in, mux1_in, func_in, gpio_psr/isr	**MUX Input Selection** 0000 No inputs selected 0001 GPIO PSR/ISR input 0010 Functional input 0100 Alternate Mode 1 input 1000 Alternate Mode 2 input 0011 Not recommended 0101 Not recommended 0110 Not recommended 0111 Not recommended 1001 Not recommended 1010 Not recommended 1011 Not recommended 1100 Not recommended 1101 Not recommended 1110 Not recommended 1111 Not recommended

4.3.4 Register Descriptions for SW MUX Control (SW_MUX_CTL)

Figure 4-6 through Figure 4-87 show the sw_mux_ctl registers. The functional multiplexing information shown in Table 4-8 enables the user to select the function of each I/O line by configuring the GPR or appropriate SW_MUX_CTL registers. The data in the table applies to any I/O that is multiplexed to provide different functions.

Additional information about EMI Multiplexing is shown in Table 4-13.

Absolute: 0x43FA_C00C Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_cspi3_miso								sw_mux_ctl_cspi3_sclk						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_cspi3_spi_rdy								sw_mux_ctl_ttm_pad						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0

Figure 4-6. Register Description sw_mux_ctl_cspi3_miso_cspi3_sclk_cspi3_spi_rdy_ttm_pad

Absolute: 0x43FA_C010 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_ata_reset_b								sw_mux_ctl_ce_control						
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_clkss								sw_mux_ctl_cspi3_mosi						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0

Figure 4-7. Register Description sw_mux_ctl_ata_reset_b_ce_control_clkss_cspi3_mosi

Absolute: 0x43FA_C014 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_ata_cs1								sw_mux_ctl_ata_dior						
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_ata_diow								sw_mux_ctl_ata_dmack						
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

Figure 4-8. Register Description sw_mux_ctl_ata_cs1_ata_dior_ata_diow_ata_dmack

Absolute: 0x43FA_C018 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_sd1_data1								sw_mux_ctl_sd1_data2						
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_sd1_data3								sw_mux_ctl_ata_cs0						
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

Figure 4-9. Register Description sw_mux_ctl_sd1_data1_sd1_data2_sd1_data3_ata_cs0

Absolute: 0x43FA_C01C Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_d3_spl								sw_mux_ctl_sd1_cmd						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_sd1_clk								sw_mux_ctl_sd1_data0						
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

Figure 4-10. Register Description sw_mux_ctl_d3_spl_sd1_cmd_sd1_clk_sd1_data0

Absolute: 0x43FA_C020 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_vsync3								sw_mux_ctl_contrast						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_d3_rev								sw_mux_ctl_d3_cls						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-11. Register Description sw_mux_ctl_vsync3_contrast_d3_rev_d3_cls

Absolute: 0x43FA_C024 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_ser_rs								sw_mux_ctl_par_rs						
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_write								sw_mux_ctl_read						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-12. Register Description sw_mux_ctl_ser_rs_par_rs_write_read

Absolute: 0x43FA_C028 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_sd_d_io								sw_mux_ctl_sd_d_clk						
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_lcs0								sw_mux_ctl_lcs1						
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

Figure 4-13. Register Description sw_mux_ctl_sd_d_io_sd_d_clk_lcs0_lcs1

Absolute: 0x43FA_C02C Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_hsync								sw_mux_ctl_fpshift						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_drdy0								sw_mux_ctl_sd_d_i						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 4-14. Register Description sw_mux_ctl_hsync_fpshift_drdy0_sd_d_i

Absolute: 0x43FA_C030 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_ld15								sw_mux_ctl_ld16						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_ld17								sw_mux_ctl_vsync0						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-15. Register Description sw_mux_ctl_ld15_ld16_ld17_vsync0

Absolute: 0x43FA_C034 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_ld11								sw_mux_ctl_ld12						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_ld13								sw_mux_ctl_ld14						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-16. Register Description sw_mux_ctl_ld11_ld12_ld13_ld14

Absolute: 0x43FA_C038 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_ld7								sw_mux_ctl_ld8						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_ld9								sw_mux_ctl_ld10						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-17. Register Description sw_mux_ctl_ld7_ld8_ld9_ld10

Absolute: 0x43FA_C03C Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_ld3								sw_mux_ctl_ld4						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_ld5								sw_mux_ctl_ld6						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-18. Register Description sw_mux_ctl_ld3_ld4_ld5_ld6

Absolute: 0x43FA_C040 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_usbh2_data1								sw_mux_ctl_ld0						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_ld1								sw_mux_ctl_ld2						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-19. Register Description sw_mux_ctl_usbh2_data1_ld0_ld1_ld2

Absolute: 0x43FA_C044 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_usbh2_dir								sw_mux_ctl_usbh2_stp						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_usbh2_nxt								sw_mux_ctl_usbh2_data0						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-20. Register Description sw_mux_ctl_usbh2_dir_usbh2_stp_usbh2_nxt_usbh2_data0

Absolute: 0x43FA_C048 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_usbotg_data5								sw_mux_ctl_usbotg_data6						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_usbotg_data7								sw_mux_ctl_usbh2_clk						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-21. Register Description sw_mux_ctl_usbotg_data5_usbotg_data6_usbotg_data7_usbh2_clk

Absolute: 0x43FA_C04C Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_usbotg_data1								sw_mux_ctl_usbotg_data2						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_usbotg_data3								sw_mux_ctl_usbotg_data4						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-22. Register Description sw_mux_ctl_usbotg_data1_usbotg_data2_usbotg_data3_usbotg_data4

Absolute: 0x43FA_C050 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_usbotg_dir								sw_mux_ctl_usbotg_stp						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_usbotg_nxt								sw_mux_ctl_usbotg_data0						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-23. Register Description sw_mux_ctl_usbotg_dir_usbotg_stp_usbotg_nxt_usbotg_data0

Absolute: 0x43FA_C054 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_usb_pwr								sw_mux_ctl_usb_oc						
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_usb_byp								sw_mux_ctl_usbotg_clk						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0

Figure 4-24. Register Description sw_mux_ctl_usb_pwr_usb_oc_usb_byp_usbotg_clk

Absolute: 0x43FA_C058 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_tdo								sw_mux_ctl_trstb						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R				$\overline{\text{sw\_mux\_ctl\_de}}$						sw_mux_ctl_sjc_mod						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

Figure 4-25. Register Description sw_mux_ctl_tdo_trstb_de_b_sjc_mod

Absolute: 0x43FA_C05C Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_rtck								sw_mux_ctl_tck						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_tms								sw_mux_ctl_tdi						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

Figure 4-26. Register Description sw_mux_ctl_rtck_tck_tms_tdi

Absolute: 0x43FA_C060 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_key_col4								sw_mux_ctl_key_col5						
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_key_col6								sw_mux_ctl_key_col7						
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

Figure 4-27. Register Description sw_mux_ctl_key_col4_key_col5_key_col6_key_col7

Absolute: 0x43FA_C064 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_key_col0								sw_mux_ctl_key_col1						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_key_col2								sw_mux_ctl_key_col3						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-28. Register Description sw_mux_ctl_key_col0_key_col1_key_col2_key_col3

Absolute: 0x43FA_C068 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_key_row4								sw_mux_ctl_key_row5						
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_key_row6								sw_mux_ctl_key_row7						
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

Figure 4-29. Register Description sw_mux_ctl_key_row4_key_row5_key_row6_key_row7

Absolute: 0x43FA_C06C Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_key_row0								sw_mux_ctl_key_row1						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_key_row2								sw_mux_ctl_key_row3						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-30. Register Description sw_mux_ctl_key_row0_key_row1_key_row2_key_row3

Absolute: 0x43FA_C070 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_txd2								sw_mux_ctl_rts2						
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_cts2								sw_mux_ctl_batt_line						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 4-31. Register Description sw_mux_ctl_txd2_rts2_cts2_batt_line

Absolute: 0x43FA_C074 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_ri_dte1								sw_mux_ctl_dcd_dte1						
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_dtr_dce2								sw_mux_ctl_rxd2						
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

Figure 4-32. Register Description sw_mux_ctl_ri_dte1_dcd_dte1_dtr_dce2_rxd2

Absolute: 0x43FA_C078 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_ri_dce1								sw_mux_ctl_dcd_dce1						
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_dtr_dte1								sw_mux_ctl_dsr_dte1						
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

Figure 4-33. Register Description sw_mux_ctl_ri_dce1_dcd_dce1_dtr_dte1_dsr_dte1

Absolute: 0x43FA_C07C Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_rts1								sw_mux_ctl_cts1						
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_dtr_dce1								sw_mux_ctl_dsr_dce1						
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

Figure 4-34. Register Description sw_mux_ctl_rts1_cts1_dtr_dce1_dsr_dce1

Absolute: 0x43FA_C080 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_cspi2_sclk								sw_mux_ctl_cspi2_spi_rdy						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_rxd1								sw_mux_ctl_txd1						
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

Figure 4-35. Register Description sw_mux_ctl_cspi2_sclk_cspi2_spi_rdy_rxd1_txd1

Absolute: 0x43FA_C084 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_cspi2_miso								sw_mux_ctl_cspi2_ss0						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_cspi2_ss1								sw_mux_ctl_cspi2_ss2						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-36. Register Description sw_mux_ctl_cspi2_miso_cspi2_ss0_cspi2_ss1_cspi2_ss2

Absolute: 0x43FA_C088 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_cspi1_ss2								sw_mux_ctl_cspi1_sclk						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_cspi1_spi_rdy								sw_mux_ctl_cspi2_mosi						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-37. Register Description sw_mux_ctl_cspi1_ss2_cspi1_sclk_cspi1_spi_rdy_cspi2_mosi

Absolute: 0x43FA_C08C Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_cspi1_mosi								sw_mux_ctl_cspi1_miso						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_cspi1_ss0								sw_mux_ctl_cspi1_ss1						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-38. Register Description sw_mux_ctl_cspi1_mosi_cspi1_miso_cspi1_ss0_cspi1_ss1

Absolute: 0x43FA_C090 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_stxd6								sw_mux_ctl_srxd6						
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_sck6								sw_mux_ctl_sfs6						
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

Figure 4-39. Register Description sw_mux_ctl_stxd6_srxd6_sck6_sfs6

Absolute: 0x43FA_C094 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_stxd5								sw_mux_ctl_srxd5						
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_sck5								sw_mux_ctl_sfs5						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-40. Register Description sw_mux_ctl_stxd5_srxd5_sck5_sfs5

Absolute: 0x43FA_C098 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_stxd4								sw_mux_ctl_srxd4						
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_sck4								sw_mux_ctl_sfs4						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-41. Register Description sw_mux_ctl_stxd4_srxd4_sck4_sfs4

Absolute: 0x43FA_C09C Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_stxd3								sw_mux_ctl_srxd3						
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_sck3								sw_mux_ctl_sfs3						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-42. Register Description sw_mux_ctl_stxd3_srxd3_sck3_sfs3

Absolute: 0x43FA_C0A0 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_csi_hsync								sw_mux_ctl_csi_pixclk						
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_i2c_clk								sw_mux_ctl_i2c_dat						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-43. Register Description sw_mux_ctl_csi_hsync_csi_pixclk_i2c_clk_i2c_dat

Absolute: 0x43FA_C0A4 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_csi_d14								sw_mux_ctl_csi_d15						
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_csi_mclk								sw_mux_ctl_csi_vsync						
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

Figure 4-44. Register Description sw_mux_ctl_csi_d14_csi_d15_csi_mclk_csi_vsync

Absolute: 0x43FA_C0A8 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_csi_d10								sw_mux_ctl_csi_d11						
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_csi_d12								sw_mux_ctl_csi_d13						
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

Figure 4-45. Register Description sw_mux_ctl_csi_d10_csi_d11_csi_d12_csi_d13

Absolute: 0x43FA_C0AC Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_csi_d6								sw_mux_ctl_csi_d7						
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_csi_d8								sw_mux_ctl_csi_d9						
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

Figure 4-46. Register Description sw_mux_ctl_csi_d6_csi_d7_csi_d8_csi_d9

Absolute: 0x43FA_C0B0 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_m_request								sw_mux_ctl_m_grant						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_csi_d4								sw_mux_ctl_csi_d5						
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

Figure 4-47. Register Description sw_mux_ctl_m_request_m_grant_csi_d4_csi_d5

Absolute: 0x43FA_C0B4 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_pc_rst								sw_mux_ctl_iois16						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_pc_rw_b								sw_mux_ctl_pc_poe						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

Figure 4-48. Register Description sw_mux_ctl_pc_rst_iois16_pc_rw_b_pc_poe

Absolute: 0x43FA_C0B8 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_pc_vs1								sw_mux_ctl_pc_vs2						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_pc_bvd1								sw_mux_ctl_pc_bvd2						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-49. Register Description sw_mux_ctl_pc_vs1_pc_vs2_pc_bvd1_pc_bvd2

Absolute: 0x43FA_C0BC Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_pc_cd2_b								sw_mux_ctl_pc_wait_b						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_pc_ready								sw_mux_ctl_pc_pwron						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-50. Register Description sw_mux_ctl_pc_cd2_b_pc_wait_b_pc_ready_pc_pwron

Absolute: 0x43FA_C0C0 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_d2								sw_mux_ctl_d1						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_d0								sw_mux_ctl_pc_cd1_b						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

Figure 4-51. Register Description sw_mux_ctl_d2_d1_d0_pc_cd1_b

Absolute: 0x43FA_C0C4 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_d6								sw_mux_ctl_d5						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_d4								sw_mux_ctl_d3						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

Figure 4-52. Register Description sw_mux_ctl_d6_d5_d4_d3

Absolute: 0x43FA_C0C8 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_d10								sw_mux_ctl_d9						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_d8								sw_mux_ctl_d7						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

Figure 4-53. Register Description sw_mux_ctl_d10_d9_d8_d7

Absolute: 0x43FA_C0CC Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_d14								sw_mux_ctl_d13						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_d12								sw_mux_ctl_d11						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

Figure 4-54. Register Description sw_mux_ctl_d14_d13_d12_d11

Absolute: 0x43FA_C0D0 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_nfwp_b								sw_mux_ctl_nfce_b						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_nfrb								sw_mux_ctl_d15						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

Figure 4-55. Register Description sw_mux_ctl_nfwp_b_nfce_b_nfrb_d15

Absolute: 0x43FA_C0D4 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_nfwe_b								sw_mux_ctl_nfre_b						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_nfale								sw_mux_ctl_nfcle						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

Figure 4-56. Register Description sw_mux_ctl_nfwe_b_nfre_b_nfale_nfcle

Absolute: 0x43FA_C0D8 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_sdqs0								sw_mux_ctl_sdqs1						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_sdqs2								sw_mux_ctl_sdqs3						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

Figure 4-57. Register Description sw_mux_ctl_sdqs0_sdqs1_sdqs2_sdqs3

Absolute: 0x43FA_C0DC Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_sdcke0								sw_mux_ctl_sdcke1						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_sdclk*								Reserved						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

*Bits 8–14 control the differential output pair SDCLK and $\overline{\text{SDCLK}}$.

Figure 4-58. Register Description sw_mux_ctl_sdcke0_sdcke1_sdclk_sdclk_b

Absolute: 0x43FA_C0E0 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_rw								sw_mux_ctl_ras						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_cas								sw_mux_ctl_sdwe						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

Figure 4-59. Register Description sw_mux_ctl_rw_ras_cas_sdwe

Absolute: 0x43FA_C0E4 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_cs5								sw_mux_ctl_ecb						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_lba								sw_mux_ctl_bclk						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

Figure 4-60. Register Description sw_mux_ctl_cs5_ecb_lba_bclk

Absolute: 0x43FA_C0E8 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_cs1								sw_mux_ctl_cs2						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_cs3								sw_mux_ctl_cs4						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

Figure 4-61. Register Description sw_mux_ctl_cs1_cs2_cs3_cs4

Absolute: 0x43FA_C0EC Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_eb0								sw_mux_ctl_eb1						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_oe								sw_mux_ctl_cs0						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

Figure 4-62. Register Description sw_mux_ctl_eb0_eb1_oe_cs0

Absolute: 0x43FA_C0F0 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_dqm0								sw_mux_ctl_dqm1						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_dqm2								sw_mux_ctl_dqm3						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

Figure 4-63. Register Description sw_mux_ctl_dqm0_dqm1_dqm2_dqm3

Absolute: 0x43FA_C0F4 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_sd28								sw_mux_ctl_sd29						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_sd30								sw_mux_ctl_sd31						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

Figure 4-64. Register Description sw_mux_ctl_sd28_sd29_sd30_sd31

Absolute: 0x43FA_C0F8 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_sd24								sw_mux_ctl_sd25						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_sd26								sw_mux_ctl_sd27						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

Figure 4-65. Register Description sw_mux_ctl_sd24_sd25_sd26_sd27

Absolute: 0x43FA_C0FC Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_sd20								sw_mux_ctl_sd21						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_sd22								sw_mux_ctl_sd23						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

Figure 4-66. Register Description sw_mux_ctl_sd20_sd21_sd22_sd23

Absolute: 0x43FA_C100 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_sd16								sw_mux_ctl_sd17						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_sd18								sw_mux_ctl_sd19						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

Figure 4-67. Register Description sw_mux_ctl_sd16_sd17_sd18_sd19

Absolute: 0x43FA_C104 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_sd12								sw_mux_ctl_sd13						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_sd14								sw_mux_ctl_sd15						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

Figure 4-68. Register Description sw_mux_ctl_sd12_sd13_sd14_sd15

Absolute: 0x43FA_C108 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_sd8								sw_mux_ctl_sd9						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_sd10								sw_mux_ctl_sd11						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

Figure 4-69. Register Description sw_mux_ctl_sd8_sd9_sd10_sd11

Absolute: 0x43FA_C10C Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_sd4								sw_mux_ctl_sd5						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_sd6								sw_mux_ctl_sd7						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

Figure 4-70. Register Description sw_mux_ctl_sd4_sd5_sd6_sd7

Absolute: 0x43FA_C110 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_sd0								sw_mux_ctl_sd1						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_sd2								sw_mux_ctl_sd3						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

Figure 4-71. Register Description sw_mux_ctl_sd0_sd1_sd2_sd3

Absolute: 0x43FA_C114 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_a24								sw_mux_ctl_a25						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_sdba1								sw_mux_ctl_sdba0						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

Figure 4-72. Register Description sw_mux_ctl_a24_a25_sdba1_sdba0

Absolute: 0x43FA_C118 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_a20								sw_mux_ctl_a21						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_a22								sw_mux_ctl_a23						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

Figure 4-73. Register Description sw_mux_ctl_a20_a21_a22_a23

Absolute: 0x43FA_C11C Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_a16								sw_mux_ctl_a17						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_a18								sw_mux_ctl_a19						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

Figure 4-74. Register Description sw_mux_ctl_a16_a17_a18_a19

Absolute: 0x43FA_C120 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_a12								sw_mux_ctl_a13						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_a14								sw_mux_ctl_a15						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

Figure 4-75. Register Description sw_mux_ctl_a12_a13_a14_a15

Absolute: 0x43FA_C124 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_a9								sw_mux_ctl_a10						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_ma10								sw_mux_ctl_a11						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

Figure 4-76. Register Description sw_mux_ctl_a9_a10_ma10_a11

Absolute: 0x43FA_C128 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_a5								sw_mux_ctl_a6						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_a7								sw_mux_ctl_a8						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

Figure 4-77. Register Description sw_mux_ctl_a5_a6_a7_a8

Absolute: 0x43FA_C12C Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_a1								sw_mux_ctl_a2						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_a3								sw_mux_ctl_a4						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

Figure 4-78. Register Description sw_mux_ctl_a1_a2_a3_a4

Absolute: 0x43FA_C130 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_dvfs1								sw_mux_ctl_vpg0						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_vpg1								sw_mux_ctl_a0						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

Figure 4-79. Register Description sw_mux_ctl_dvfs1_vpg0_vpg1_a0

Absolute: 0x43FA_C134 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_ckil								sw_mux_ctl_power_fail						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_vstby								sw_mux_ctl_dvfs0						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

Figure 4-80. Register Description sw_mux_ctl_ckil_power_fail_vstby_dvfs0

Absolute: 0x43FA_C138 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_boot_mode1								sw_mux_ctl_boot_mode2						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_boot_mode3								sw_mux_ctl_boot_mode4						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

Figure 4-81. Register Description sw_mux_ctl_boot_mode1_boot_mode2_boot_mode3_boot_mode4

Absolute: 0x43FA_C13C Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_reset_in_b								sw_mux_ctl_por_b						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_clko								sw_mux_ctl_boot_mode0						
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

Figure 4-82. Register Description sw_mux_ctl_reset_in_b_por_b_clko_boot_mode0

Absolute: 0x43FA_C140 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_stx0								sw_mux_ctl_srx0						
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_simpd0								sw_mux_ctl_ckih						
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1	0

Figure 4-83. Register Description sw_mux_ctl_stx0_srx0_simpd0_ckih

Absolute: 0x43FA_C144 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_gpio3_1								sw_mux_ctl_sclk0						
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_srst0								sw_mux_ctl_sven0						
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

Figure 4-84. Register Description sw_mux_ctl_gpio3_1_sclk0_srst0_sven0

Absolute: 0x43FA_C148 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		sw_mux_ctl_gpio1_4								sw_mux_ctl_gpio1_5						
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		sw_mux_ctl_gpio1_6								sw_mux_ctl_gpio3_0						
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

Figure 4-85. Register Description sw_mux_ctl_gpio1_4_gpio1_5_gpio1_6_gpio3_0

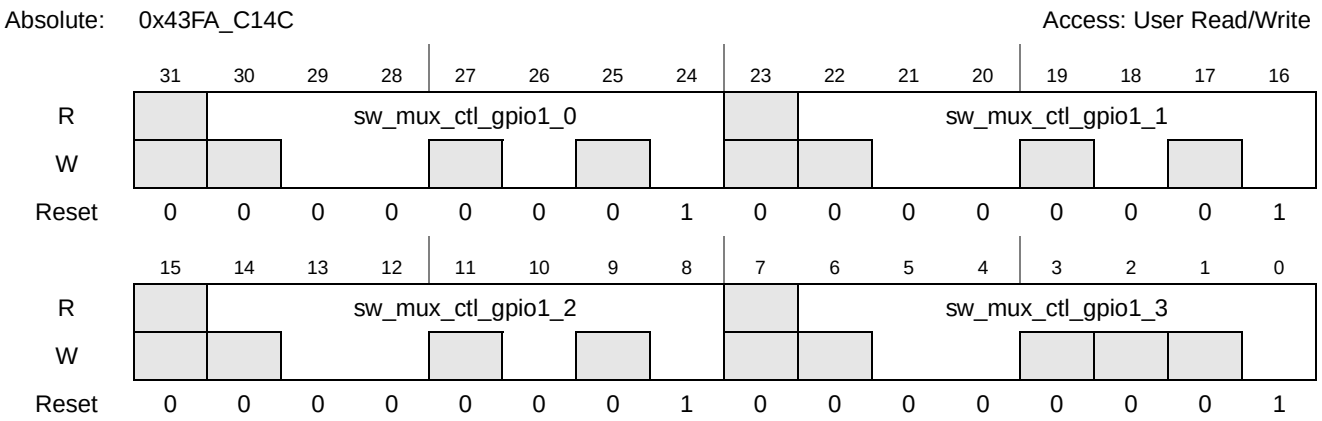


Figure 4-86. Register Description sw_mux_ctl_gpio1_0_gpio1_1_gpio1_2_gpio1_3

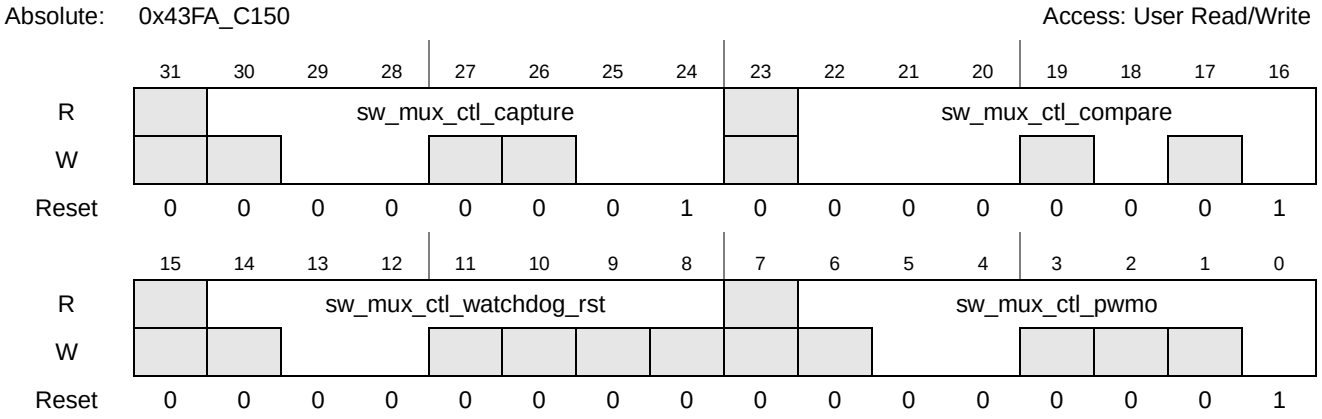


Figure 4-87. Register Description sw_mux_ctl_capture_compare_watchdog_rst_pwmo

4.3.5 Functional Multiplexing Modes

The IOMUX has four configurable modes: Hardware, Functional, Alternate, and GPIO. Each mode has a priority. Setting an I/O with a higher priority overrides the previous mode setting. Likewise, if a priority setting being applied is a lower priority than the priority of the current mode, the new mode setting is ignored. Table 4-7 shows the priority for each of the modes.

Table 4-7. Multiplexing Priorities

Priority	Mode
1	Hardware mode 2
2	Hardware mode 1
3	Alternate mode 2
4	Alternate mode 1
5	Functional mode

The four multiplexing modes are defined in the following sections.

4.3.5.1 Hardware Mode

This mode is used to set multiple sets of I/O signals by setting or clearing a single bit in the General Purpose Register (GPR).

4.3.5.2 Functional Mode

This is the primary mode of the I/O line. Like a default setting, the primary mode routes the signals for which it is named. For example, the functional mode of the RXD1 I/O line routes the RXD signal of UART1 to the external contact of the IC.

4.3.5.3 Alternate Modes

Each I/O has software-programmable bits that can select between the functional mode and other I/O muxing options. Each I/O signal has the potential of six alternate modes which are individually defined by software. For example, using an alternate mode allows the RXD signal to be routed to RI_DCE1 I/O. The six alternate mode options are as follows:

- Alternate Mode 1–2 (I/O)
- Alternate Modes 3–6 (output only)

4.3.5.4 GPIO Mode

In GPIO mode, configuration of the I/O is controlled by the GPIO module. For example, in the GPIO column of Table 4-8, MCU indicates MCU1_7, which is associated with GPIO1 bit 7.

Table 4-8. Functional Multiplexing Configuration

Contact Name	Group	Description	Configured by GPR		SW_MUX_CTL Bits	Configured by SW_MUX_CTL							
			Hardware Mode 1	Hardware Mode 2		Functional Mode	Alternate Mode 1	Alternate Mode 2	Alternate Mode 3	Alternate Mode 4	Alternate Mode 5	Alternate Mode 6	GPIO Mode
CAPTURE	Timer	Timer1 input capture, GPIO suggested for memstick1 card detect	ATA_DATA 14	—	sw_mux_ctl_capture[6:0]	Capture1	—	Compare2	—	—	—	—	MCU1_7
COMPARE	Timer	Timer1 output compare, GPIO suggested for memstick2 card detect	ATA_DATA 15	—	sw_mux_ctl_compare[6:0]	Compare1	Capture2	Compare3	—	EPIT1	EPIT2	—	MCU1_8
WATCHDOG_RST	WTDG	Watchdog reset output	—	—	sw_mux_ctl_watchdog_rst[6:0]	WDOG Reset	—	CSI Flash strobe	—	—	—	—	—
PWMO	PWM	PWM output	ATA_IORDY	—	sw_mux_ctl_pwmo[6:0]	PWM Out	PC spkr out	—	—	—	—	—	MCU1_9
GPIO1_0	GPIO	GPIO1, bit 0	—	—	sw_mux_ctl_gpio1_0[6:0]	—	ext DMA 0	—	—	—	—	—	MCU1_0
GPIO1_1	GPIO	GPIO1, bit 1	—	—	sw_mux_ctl_gpio1_1[6:0]	—	ext DMA 1	—	—	—	—	—	MCU1_1
GPIO1_2	GPIO	GPIO1, bit 2	—	—	sw_mux_ctl_gpio1_2[6:0]	—	ext DMA 2	—	—	—	—	—	MCU1_2
GPIO1_3	GPIO	GPIO1, bit 3	—	—	sw_mux_ctl_gpio1_3[6:0]	—	—	—	—	—	—	—	MCU1_3

Table 4-8. Functional Multiplexing Configuration (continued)

Contact Name	Group	Description	Configured by GPR		SW_MUX_CTL Bits	Configured by SW_MUX_CTL							
			Hardware Mode 1	Hardware Mode 2		Functional Mode	Alternate Mode 1	Alternate Mode 2	Alternate Mode 3	Alternate Mode 4	Alternate Mode 5	Alternate Mode 6	GPIO Mode
GPIO1_4	GPIO	GPIO1, bit 4	—	—	sw_mux_ctl_gpio1_4 [6:0]	—	USB Host1 SUSPEND	—	—	—	—	—	MCU1_4
GPIO1_5	GPIO	Reserved input for connect to power ready output of a power management IC	—	—	sw_mux_ctl_gpio1_5[6:0]	Power Ready Input	—	—	—	—	—	—	MCU1_5
GPIO1_6	GPIO	GPIO suggested for Tamper detect input	TAMPER_DETECT	—	sw_mux_ctl_gpio1_6[6:0]	—	—	—	—	—	—	—	MCU1_6
GPIO3_0	GPIO	GPIO suggested as IPU CSI chip select3	SPLL_BYPASS_CLK	—	sw_mux_ctl_gpio3_0[6:0]	—	—	—	—	—	—	—	MCU3_0
GPIO3_1	GPIO	GPIO suggested as IPU CSI chip select4	UPLL_BYPASS_CLK	—	sw_mux_ctl_gpio3_1[6:0]	—	—	—	—	—	—	—	MCU3_1
SCLK0	SIM	SIM Port 0	—	—	sw_mux_ctl_sclk0[6:0]	SCLK0	CTI_TRIG_IN_1_4	DISPB_D2_CS	—	—	—	—	MCU3_2
SRST0	SIM	SIM Port 0	—	—	sw_mux_ctl_srst0[6:0]	RST0	—	DISPB_D12_VSYNC	—	—	—	—	MCU3_3
SVEN0	SIM	SIM Port 0	—	—	sw_mux_ctl_sven0[6:0]	SVEN0	CTI_TRIG_IN_1_6	—	—	—	—	—	MCU2_0

Table 4-8. Functional Multiplexing Configuration (continued)

Contact Name	Group	Description	Configured by GPR		SW_MUX_CTL Bits	Configured by SW_MUX_CTL							
			Hardware Mode 1	Hardware Mode 2		Functional Mode	Alternate Mode 1	Alternate Mode 2	Alternate Mode 3	Alternate Mode 4	Alternate Mode 5	Alternate Mode 6	GPIO Mode
STX0	SIM	SIM Port 0	—	—	sw_mux_ctl_stx0[6:0]	DATA0_TX_OUT	CTI_TRIG_IN_1_5	—	—	—	—	—	MCU2_1
SRX0	SIM	SIM Port 0	—	—	sw_mux_ctl_srx0[6:0]	RCVD0_IN	—	—	—	—	—	—	MCU2_2
SIMPD0	SIM	SIM Port 0	—	—	sw_mux_ctl_simpd0[6:0]	SIMPD0	—	—	—	—	—	—	MCU2_3
CKIH	Clock and Reset and PM	High frequency clock input	—	—	—	CKIH	—	—	—	—	—	—	—
$\overline{\text{RESET\_IN}}$	Clock and Reset and PM	Master Reset Input	$\overline{\text{RESET\_IN}}$	—	sw_mux_ctl_reset_in_b[6:0]	—	—	—	—	—	—	—	—
$\overline{\text{POR}}$	Clock and Reset and PM	Power On Reset input	—	—	—	POR_B	—	—	—	—	—	—	—
CLKO	Clock and Reset and PM	Clock out signal	—	—	—	CCM_CLKO	—	—	—	—	—	—	—

Table 4-8. Functional Multiplexing Configuration (continued)

Contact Name	Group	Description	Configured by GPR		SW_MUX_CTL Bits	Configured by SW_MUX_CTL							
			Hardware Mode 1	Hardware Mode 2		Functional Mode	Alternate Mode 1	Alternate Mode 2	Alternate Mode 3	Alternate Mode 4	Alternate Mode 5	Alternate Mode 6	GPIO Mode
BOOT_MODE 0	Clock and Reset and PM	Boot Mode 0	—	—	—	BOOT0	—	—	—	—	—	—	—
BOOT_MODE 1	Clock and Reset and PM	Boot Mode 1	—	—	—	BOOT1	—	—	—	—	—	—	—
BOOT_MODE 2	Clock and Reset and PM	Boot Mode 2	—	—	—	BOOT2	—	—	—	—	—	—	—
BOOT_MODE 3	Clock and Reset and PM	Boot Mode 3	—	—	—	BOOT3	—	—	—	—	—	—	—
BOOT_MODE 4	Clock and Reset and PM	Boot Mode 4	—	—	—	BOOT4	—	—	—	—	—	—	—
CKIL	Clock and Reset and PM	Low frequency clock input	—	—	—	CKIL	—	—	—	—	—	—	—

Table 4-8. Functional Multiplexing Configuration (continued)

Contact Name	Group	Description	Configured by GPR		SW_MUX_CTL Bits	Configured by SW_MUX_CTL							
			Hardware Mode 1	Hardware Mode 2		Functional Mode	Alternate Mode 1	Alternate Mode 2	Alternate Mode 3	Alternate Mode 4	Alternate Mode 5	Alternate Mode 6	GPIO Mode
POWER_FAIL	Clock and Reset and PM	power shut-off input	—	—	—	POWER_FAIL	—	—	—	—	—	—	—
VSTBY	Clock and Reset and PM	Power management State retention output	—	—	sw_mux_ctl_vstby[6:0]	VSTBY	—	—	—	—	—	—	—
DVFS0	Clock and Reset and PM	Power management voltage change output	—	—	sw_mux_ctl_dvfs0[6:0]	DVFS0	—	—	—	—	—	—	—
DVFS1	Clock and Reset and PM	Power management voltage change output	—	—	sw_mux_ctl_dvfs1[6:0]	DVFS1	—	—	—	—	—	—	—
VPG0	Clock and Reset and PM	Power management power gating output for ARM	—	—	sw_mux_ctl_vpg0[6:0]	VPG0	—	—	—	—	—	—	—
VPG1	Clock and Reset and PM	Power management power gating output for the L2 Cache	—	—	sw_mux_ctl_vpg1[6:0]	VPG1	—	—	—	—	—	—	—
A0	EMI	EIM address 0	—	—	sw_mux_ctl_a0[6:0]	EMI_ADDR[0]	—	—	—	—	—	—	—

Table 4-8. Functional Multiplexing Configuration (continued)

Contact Name	Group	Description	Configured by GPR		SW_MUX_CTL Bits	Configured by SW_MUX_CTL							
			Hardware Mode 1	Hardware Mode 2		Functional Mode	Alternate Mode 1	Alternate Mode 2	Alternate Mode 3	Alternate Mode 4	Alternate Mode 5	Alternate Mode 6	GPIO Mode
A1	EMI	EIM address 1	—	—	sw_mux_ctl_a1[6:0]	EMI_ADDR[1]	—	—	—	—	—	—	—
A2	EMI	EIM address 2	—	—	sw_mux_ctl_a2[6:0]	EMI_ADDR[2]	—	—	—	—	—	—	—
A3	EMI	EIM address 3	—	—	sw_mux_ctl_a3[6:0]	EMI_ADDR[3]	—	—	—	—	—	—	—
A4	EMI	EIM address 4	—	—	sw_mux_ctl_a4[6:0]	EMI_ADDR[4]	—	—	—	—	—	—	—
A5	EMI	EIM address 5	—	—	sw_mux_ctl_a5[6:0]	EMI_ADDR[5]	—	—	—	—	—	—	—
A6	EMI	EIM address 6	—	—	sw_mux_ctl_a6[6:0]	EMI_ADDR[6]	—	—	—	—	—	—	—
A7	EMI	EIM address 7	—	—	sw_mux_ctl_a7[6:0]	EMI_ADDR[7]	—	—	—	—	—	—	—
A8	EMI	EIM address 8	—	—	sw_mux_ctl_a8[6:0]	EMI_ADDR[8]	—	—	—	—	—	—	—
A9	EMI	EIM address 9	—	—	sw_mux_ctl_a9[6:0]	EMI_ADDR[9]	—	—	—	—	—	—	—
A10	EMI	EIM address 10	—	—	sw_mux_ctl_a10[6:0]	EMI_ADDR[10]	—	—	—	—	—	—	—
MA10	EMI	SDRAM controller address 10	—	—	sw_mux_ctl_ma10[6:0]	M3IF_MA10	—	—	—	—	—	—	—
A11	EMI	EIM address 11	—	—	sw_mux_ctl_a11[6:0]	EMI_ADDR[11]	—	—	—	—	—	—	—
A12	EMI	EIM address 12	—	—	sw_mux_ctl_a12[6:0]	EMI_ADDR[12]	—	—	—	—	—	—	—

Table 4-8. Functional Multiplexing Configuration (continued)

Contact Name	Group	Description	Configured by GPR		SW_MUX_CTL Bits	Configured by SW_MUX_CTL							
			Hardware Mode 1	Hardware Mode 2		Functional Mode	Alternate Mode 1	Alternate Mode 2	Alternate Mode 3	Alternate Mode 4	Alternate Mode 5	Alternate Mode 6	GPIO Mode
A13	EMI	EIM address 13	—	—	sw_mux_ctl_a13[6:0]	EMI_ADDR[13]	—	—	—	—	—	—	—
A14	EMI	EIM address 14	—	—	sw_mux_ctl_a14[6:0]	EMI_ADDR[14]	—	—	—	—	—	—	—
A15	EMI	EIM address 15	—	—	sw_mux_ctl_a15[6:0]	EMI_ADDR[15]	—	—	—	—	—	—	—
A16	EMI	EIM address 16	—	—	sw_mux_ctl_a16[6:0]	EMI_ADDR[16]	—	—	—	—	—	—	—
A17	EMI	EIM address 17	—	—	sw_mux_ctl_a17[6:0]	EMI_ADDR[17]	—	—	—	—	—	—	—
A18	EMI	EIM address 18	—	—	sw_mux_ctl_a18[6:0]	EMI_ADDR[18]	—	—	—	—	—	—	—
A19	EMI	EIM address 19	—	—	sw_mux_ctl_a19[6:0]	EMI_ADDR[19]	—	—	—	—	—	—	—
A20	EMI	EIM address 20	—	—	sw_mux_ctl_a20[6:0]	EMI_ADDR[20]	—	—	—	—	—	—	—
A21	EMI	EIM address 21	—	—	sw_mux_ctl_a21[6:0]	EMI_ADDR[21]	—	—	—	—	—	—	—
A22	EMI	EIM address 22	—	—	sw_mux_ctl_a22[6:0]	EMI_ADDR[22]	—	—	—	—	—	—	—
A23	EMI	EIM address 23	—	—	sw_mux_ctl_a23[6:0]	EMI_ADDR[23]	—	—	—	—	—	—	—
A24	EMI	EIM address 24	—	—	sw_mux_ctl_a24[6:0]	EMI_ADDR[24]	—	—	—	—	—	—	—
A25	EMI	EIM address 25	—	—	sw_mux_ctl_a25[6:0]	EMI_ADDR[25]	—	—	—	—	—	—	—

Table 4-8. Functional Multiplexing Configuration (continued)

Contact Name	Group	Description	Configured by GPR		SW_MUX_CTL Bits	Configured by SW_MUX_CTL							
			Hardware Mode 1	Hardware Mode 2		Functional Mode	Alternate Mode 1	Alternate Mode 2	Alternate Mode 3	Alternate Mode 4	Alternate Mode 5	Alternate Mode 6	GPIO Mode
SDBA1	EMI	EIM Bank Address	—	—	sw_mux_ctl_sdba1[6:0]	SDBA1	—	—	—	—	—	—	—
SDBA0	EMI	EIM Bank Address	—	—	sw_mux_ctl_sdba0[6:0]	SDBA0	—	—	—	—	—	—	—
SD0	EMI	DDR/SDRAM Data 0	—	—	—	EMI_DATA[0]	—	—	—	—	—	—	—
SD1	EMI	DDR/SDRAM Data 1	—	—	—	EMI_DATA[1]	—	—	—	—	—	—	—
SD2	EMI	DDR/SDRAM Data 2	—	—	—	EMI_DATA[2]	—	—	—	—	—	—	—
SD3	EMI	DDR/SDRAM Data 3	—	—	—	EMI_DATA[3]	—	—	—	—	—	—	—
SD4	EMI	DDR/SDRAM Data 4	—	—	—	EMI_DATA[4]	—	—	—	—	—	—	—
SD5	EMI	DDR/SDRAM Data 5	—	—	—	EMI_DATA[5]	—	—	—	—	—	—	—
SD6	EMI	DDR/SDRAM Data 6	—	—	—	EMI_DATA[6]	—	—	—	—	—	—	—
SD7	EMI	DDR/SDRAM Data 7	—	—	—	EMI_DATA[7]	—	—	—	—	—	—	—
SD8	EMI	DDR/SDRAM Data 8	—	—	—	EMI_DATA[8]	—	—	—	—	—	—	—
SD9	EMI	DDR/SDRAM Data 9	—	—	—	EMI_DATA[9]	—	—	—	—	—	—	—
SD10	EMI	DDR/SDRAM Data 10	—	—	—	EMI_DATA[10]	—	—	—	—	—	—	—

Table 4-8. Functional Multiplexing Configuration (continued)

Contact Name	Group	Description	Configured by GPR		SW_MUX_CTL Bits	Configured by SW_MUX_CTL							
			Hardware Mode 1	Hardware Mode 2		Functional Mode	Alternate Mode 1	Alternate Mode 2	Alternate Mode 3	Alternate Mode 4	Alternate Mode 5	Alternate Mode 6	GPIO Mode
SD11	EMI	DDR/SDRAM Data 11	—	—	—	EMI_DATA[11]	—	—	—	—	—	—	—
SD12	EMI	DDR/SDRAM Data 12	—	—	—	EMI_DATA[12]	—	—	—	—	—	—	—
SD13	EMI	DDR/SDRAM Data 13	—	—	—	EMI_DATA[13]	—	—	—	—	—	—	—
SD14	EMI	DDR/SDRAM Data 14	—	—	—	EMI_DATA[14]	—	—	—	—	—	—	—
SD15	EMI	DDR/SDRAM Data 15	—	—	—	EMI_DATA[15]	—	—	—	—	—	—	—
SD16	EMI	DDR/SDRAM Data 16	—	—	—	EMI_DATA[16]	—	—	—	—	—	—	—
SD17	EMI	DDR/SDRAM Data 17	—	—	—	EMI_DATA[17]	—	—	—	—	—	—	—
SD18	EMI	DDR/SDRAM Data 18	—	—	—	EMI_DATA[18]	—	—	—	—	—	—	—
SD19	EMI	DDR/SDRAM Data 19	—	—	—	EMI_DATA[19]	—	—	—	—	—	—	—
SD20	EMI	DDR/SDRAM Data 20	—	—	—	EMI_DATA[20]	—	—	—	—	—	—	—
SD21	EMI	DDR/SDRAM Data 21	—	—	—	EMI_DATA[21]	—	—	—	—	—	—	—
SD22	EMI	DDR/SDRAM Data 22	—	—	—	EMI_DATA[22]	—	—	—	—	—	—	—
SD23	EMI	DDR/SDRAM Data 23	—	—	—	EMI_DATA[23]	—	—	—	—	—	—	—

Table 4-8. Functional Multiplexing Configuration (continued)

Contact Name	Group	Description	Configured by GPR		SW_MUX_CTL Bits	Configured by SW_MUX_CTL							
			Hardware Mode 1	Hardware Mode 2		Functional Mode	Alternate Mode 1	Alternate Mode 2	Alternate Mode 3	Alternate Mode 4	Alternate Mode 5	Alternate Mode 6	GPIO Mode
SD24	EMI	DDR/SDRAM Data 24	—	—	—	EMI_DATA[24]	—	—	—	—	—	—	—
SD25	EMI	DDR/SDRAM Data 25	—	—	—	EMI_DATA[25]	—	—	—	—	—	—	—
SD26	EMI	DDR/SDRAM Data 26	—	—	—	EMI_DATA[26]	—	—	—	—	—	—	—
SD27	EMI	DDR/SDRAM Data 27	—	—	—	EMI_DATA[27]	—	—	—	—	—	—	—
SD28	EMI	DDR/SDRAM Data 28	—	—	—	EMI_DATA[28]	—	—	—	—	—	—	—
SD29	EMI	DDR/SDRAM Data 29	—	—	—	EMI_DATA[29]	—	—	—	—	—	—	—
SD30	EMI	DDR/SDRAM Data 30	—	—	—	EMI_DATA[30]	—	—	—	—	—	—	—
SD31	EMI	DDR/SDRAM Data 31	—	—	—	EMI_DATA[31]	—	—	—	—	—	—	—
DQM0	EMI	Byte strobe DDR data enable	—	—	sw_mux_ctl_dqm0[6:0]	DQM[0]	—	—	—	—	—	—	—
DQM1	EMI	Byte strobe DDR data enable	—	—	sw_mux_ctl_dqm1[6:0]	DQM[1]	—	—	—	—	—	—	—
DQM2	EMI	Byte strobe DDR data enable	—	—	sw_mux_ctl_dqm2[6:0]	DQM[2]	—	—	—	—	—	—	—
DQM3	EMI	Byte strobe DDR data enable	—	—	sw_mux_ctl_dqm3[6:0]	DQM[3]	—	—	—	—	—	—	—
EB0	EMI	LSB Byte strobe WEIM data enable; Controls D[7:0]	—	—	sw_mux_ctl_eb0[6:0]	EB_B[0]	—	—	—	—	—	—	—

Table 4-8. Functional Multiplexing Configuration (continued)

Contact Name	Group	Description	Configured by GPR		SW_MUX_CTL Bits	Configured by SW_MUX_CTL							
			Hardware Mode 1	Hardware Mode 2		Functional Mode	Alternate Mode 1	Alternate Mode 2	Alternate Mode 3	Alternate Mode 4	Alternate Mode 5	Alternate Mode 6	GPIO Mode
EB1	EMI	LSB Byte strobe WEIM data enable Controls D[15:8]	—	—	sw_mux_ctl_eb1[6:0]	EB_B[1]	—	—	—	—	—	—	—
OE	EMI	Memory Output enable	—	—	sw_mux_ctl_oe[6:0]	EMI_OE_B	—	—	—	—	—	—	—
CS0	EMI	Chip select 0	—	—	sw_mux_ctl_cs0[6:0]	WEIM_CS_B0	—	—	—	—	—	—	—
CS1	EMI	Chip select 1	—	—	sw_mux_ctl_cs1[6:0]	WEIM_CS_B1	—	—	—	—	—	—	—
CS2	EMI	Chip select 2/SDRAM Sync Flash chip select	—	—	sw_mux_ctl_cs2[6:0]	WEIM_CS_B2	—	—	—	—	—	—	—
CS3	EMI	Chip select 3/SDRAM Sync Flash chip select	—	—	sw_mux_ctl_cs3[6:0]	WEIM_CS_B3	—	—	—	—	—	—	—
CS4	EMI	Chip select 4	—	—	sw_mux_ctl_cs4[6:0]	WEIM_CS_B4	$\overline{\text{DTACK}}$	—	—	—	—	—	—
CS5	EMI	Chip select 5	—	—	sw_mux_ctl_cs5[6:0]	WEIM_CS_B5	—	—	—	—	—	—	—
ECB	EMI	End Current Burst	—	—	—	WEIM_ECB_B	—	—	—	—	—	—	—
LBA	EMI	Load Base Address	—	—	sw_mux_ctl_lba[6:0]	WEIM_LBA_B	—	—	—	—	—	—	—
BCLK	EMI	used by Flash for burst mode	—	—	sw_mux_ctl_bclk[6:0]	WEIM_BCLK	—	—	—	—	—	—	—
RW	EMI	read/write signal or WE for external dram	—	—	sw_mux_ctl_rw[6:0]	WEIM_RW_B	—	—	—	—	—	—	—
RAS	EMI	SDRAM row address select	—	—	sw_mux_ctl_ras[6:0]	M3IF_RAS_B	—	—	—	—	—	—	—

Table 4-8. Functional Multiplexing Configuration (continued)

Contact Name	Group	Description	Configured by GPR		SW_MUX_CTL Bits	Configured by SW_MUX_CTL							
			Hardware Mode 1	Hardware Mode 2		Functional Mode	Alternate Mode 1	Alternate Mode 2	Alternate Mode 3	Alternate Mode 4	Alternate Mode 5	Alternate Mode 6	GPIO Mode
CAS	EMI	SDRAM column address select	—	—	sw_mux_ctl_cas[6:0]	M3IF_CAS_B	—	—	—	—	—	—	—
SDWE	EMI	SDRAM write enable	—	—	sw_mux_ctl_sdwe[6:0]	SDRC_SDWE	—	—	—	—	—	—	—
SDCKE0	EMI	SDRAM clock enable0	—	—	sw_mux_ctl_sdcke0[6:0]	SDRC_SDCKE[0]	—	—	—	—	—	—	—
SDCKE1	EMI	SDRAM clock enable1	—	—	sw_mux_ctl_sdcke1[6:0]	SDRC_SDCKE[1]	—	—	—	—	—	—	—
SDCLK	EMI	SDRAM clock DDR clock pad	—	—	sw_mux_ctl_sdclk[6:0]	SDRC_SDCLK	—	—	—	—	—	—	—
SDCLK_B	DDR	False pad DDR_CLK	—	—	—	SDRC_SDCLK_B	—	—	—	—	—	—	—
SDQS0	EMI	DDR sample strobe	—	—	—	DQS[0]	—	—	—	—	—	—	—
SDQS1	EMI	DDR sample strobe	—	—	—	DQS[1]	—	—	—	—	—	—	—
SDQS2	EMI	DDR sample strobe	—	—	—	DQS[2]	—	—	—	—	—	—	—
SDQS3	EMI	DDR sample strobe	—	—	—	DQS[3]	—	—	—	—	—	—	—
$\overline{\text{NFWE}}$	EMI	NANDF write enable	ATA_DATA7	ATA_INTRQ	sw_mux_ctl_nfwe_b[6:0]	NFC_WE	—	USBH2 DATA2	—	TRACEDATA_0	—	—	MCU1_10
$\overline{\text{NFRE}}$	EMI	NANDF read enable	ATA_DATA8	ATA_BUFFER_EN	sw_mux_ctl_nfre_b[6:0]	NFC_RE	—	USBH2 DATA3	—	TRACEDATA_1	—	—	MCU1_11
NFALE	EMI	NANDF address latch enable	ATA_DATA9	ATA_DMARQ	sw_mux_ctl_nfale[6:0]	NFC_ALE	—	USBH2_DATA4	—	TRACEDATA_2	—	—	MCU1_12

Table 4-8. Functional Multiplexing Configuration (continued)

Contact Name	Group	Description	Configured by GPR		SW_MUX_CTL Bits	Configured by SW_MUX_CTL							
			Hardware Mode 1	Hardware Mode 2		Functional Mode	Alternate Mode 1	Alternate Mode 2	Alternate Mode 3	Alternate Mode 4	Alternate Mode 5	Alternate Mode 6	GPIO Mode
NFCLE	EMI	NANDF command latch enable	ATA_DATA 10	ATA_DA0	sw_mux_ctl_nfcle[6:0]	NFC_CLE	—	USBH2_DATA5	—	TRACEDATA_3	—	—	MCU1_13
$\overline{\text{NFWP}}$	EMI	NANDF write protect	ATA_DATA 11	ATA_DA1	sw_mux_ctl_nfwp_b[6:0]	NFC_WP	NFWP_B	USBH2_DATA6	—	TRACEDATA_4	—	—	MCU1_14
$\overline{\text{NFCE}}$	EMI	NANDF chip enable	ATA_DATA 12	ATA_DA2	sw_mux_ctl_nfce_b[6:0]	NFC_CE	—	USBH2_DATA7	—	TRACEDATA_5	—	—	MCU1_15
NFRB	EMI	NANDF ready/busy	ATA_DATA 13	—	sw_mux_ctl_nfrb[6:0]	NFC_RB	—	—	—	TRACEDATA_6	—	—	MCU1_16
D15	EMI	PCMCIA/WEIM/NANDF Data 15	—	—	—	NFC_DATA 15	—	—	—	—	—	—	—
D14	EMI	PCMCIA/WEIM/NANDF Data 14	—	—	—	NFC_DATA 14	—	—	—	—	—	—	—
D13	EMI	PCMCIA/WEIM/NANDF Data 13	—	—	—	NFC_DATA 13	—	—	—	—	—	—	—
D12	EMI	PCMCIA/WEIM/NANDF Data 12	—	—	—	NFC_DATA 12	—	—	—	—	—	—	—
D11	EMI	PCMCIA/WEIM/NANDF Data 11	—	—	—	NFC_DATA 11	—	—	—	—	—	—	—
D10	EMI	PCMCIA/WEIM/NANDF Data 10	—	—	—	NFC_DATA 10	—	—	—	—	—	—	—
D9	EMI	PCMCIA/WEIM/NANDF Data 9	—	—	—	NFC_DATA 9	—	—	—	—	—	—	—
D8	EMI	PCMCIA/WEIM/NANDF Data 8	—	—	—	NFC_DATA 8	—	—	—	—	—	—	—

Table 4-8. Functional Multiplexing Configuration (continued)

Contact Name	Group	Description	Configured by GPR		SW_MUX_CTL Bits	Configured by SW_MUX_CTL							
			Hardware Mode 1	Hardware Mode 2		Functional Mode	Alternate Mode 1	Alternate Mode 2	Alternate Mode 3	Alternate Mode 4	Alternate Mode 5	Alternate Mode 6	GPIO Mode
D7	EMI	PCMCIA/WEIM/NANDF Data 7	—	—	—	NFC_DATA 7	—	—	—	—	—	—	—
D6	EMI	PCMCIA/WEIM/NANDF Data 6	—	—	—	NFC_DATA 6	—	—	—	—	—	—	—
D5	EMI	PCMCIA/WEIM/NANDF Data 5	—	—	—	NFC_DATA 5	—	—	—	—	—	—	—
D4	EMI	PCMCIA/WEIM/NANDF Data 4	—	—	—	NFC_DATA 4	—	—	—	—	—	—	—
D3	EMI	PCMCIA/WEIM/NANDF Data 3	—	—	—	NFC_DATA 3	—	—	—	—	—	—	—
D2	EMI	PCMCIA/WEIM/NANDF Data 2	—	—	—	NFC_DATA 2	—	—	—	—	—	—	—
D1	EMI	PCMCIA/WEIM/NANDF Data 1	—	—	—	NFC_DATA 1	—	—	—	—	—	—	—
D0	EMI	PCMCIA/WEIM/NANDF Data 0	—	—	—	NFC_DATA 0	—	—	—	—	—	—	—
PC_CD1_B	EMI	PCMCIA card detect 1 input	—	—	sw_mux_ctl_pc_cd1_b[6:0]	CD1_B	SD2_CMD	MSHC2_SCLK	—	—	—	—	—
PC_CD2_B	EMI	PCMCIA card detect 2 input	—	—	sw_mux_ctl_pc_cd2_b[6:0]	CD2_B	SD2_CLK	MSHC2_BS	—	—	—	—	—
$\overline{\text{PC\_WAIT}}$	EMI	PCMCIA pending cycle delay input	—	—	sw_mux_ctl_pc_wait_b[6:0]	WAIT_B	SD2_DATA0	MSHC2_SDIO_DATA0	—	—	—	—	—
PC_READY	EMI	PCMCIA ready/busy input	—	—	sw_mux_ctl_pc_ready[6:0]	RDY_IRQ_B	SD2_DATA1	MSHC2_DATA1	—	—	—	—	—

Table 4-8. Functional Multiplexing Configuration (continued)

Contact Name	Group	Description	Configured by GPR		SW_MUX_CTL Bits	Configured by SW_MUX_CTL							
			Hardware Mode 1	Hardware Mode 2		Functional Mode	Alternate Mode 1	Alternate Mode 2	Alternate Mode 3	Alternate Mode 4	Alternate Mode 5	Alternate Mode 6	GPIO Mode
PC_PWRON	EMI	PCMCIA supply ready input	—	—	sw_mux_ctl_pc_pwron[6:0]	PWR_ON	SD2_DATA3	MSHC2_DATA2	—	—	—	—	—
PC_VS1	EMI	PCMCIA voltage sense 1 input	—	—	sw_mux_ctl_pc_vs1[6:0]	VS1	SD2_DATA2	MSHC2_DATA3	—	—	—	—	—
PC_VS2	EMI	PCMCIA voltage sense 2 input	—	—	sw_mux_ctl_pc_vs2[6:0]	VS2	USBH2_DATA2	UART5_RTS	—	—	—	—	—
PC_BVD1	EMI	PCMCIA battery voltage detect 1 input	—	—	sw_mux_ctl_pc_bvd1[6:0]	BVD1	USBH2_DATA3	UART5_RXD	—	—	—	—	—
PC_BVD2	EMI	PCMCIA battery voltage detect 2 input	—	—	sw_mux_ctl_pc_bvd2[6:0]	BVD2	USBH2_DATA4	UART5_TXD	—	—	—	—	—
PC_RST	EMI	PCMCIA reset output	—	—	sw_mux_ctl_pc_rst[6:0]	CARD_RESET	USBH2_DATA5	UART5_CTS	—	—	—	—	—
IOIS16	EMI	PCMCIA bus width input	—	—	sw_mux_ctl_iois16[6:0]	IND_WP	USBH2_DATA6	—	—	—	—	—	—
PC_R$\overline{W}$	EMI	PCMCIA read/$\overline{write}$ - external transceiver direction control output.	—	—	sw_mux_ctl_pc_rw_b[6:0]	CARD_RW_B	USBH2_DATA7	—	—	—	—	—	—
PC_POE	EMI	PCMCIA buffers output enable output	—	—	sw_mux_ctl_pc_poe[6:0]	CARD_POE_O_B	—	—	—	—	—	—	—
M_REQUEST	EMI	reserved	—	—	—	M_REQUEST	—	—	—	—	—	—	—

Table 4-8. Functional Multiplexing Configuration (continued)

Contact Name	Group	Description	Configured by GPR		SW_MUX_CTL Bits	Configured by SW_MUX_CTL							
			Hardware Mode 1	Hardware Mode 2		Functional Mode	Alternate Mode 1	Alternate Mode 2	Alternate Mode 3	Alternate Mode 4	Alternate Mode 5	Alternate Mode 6	GPIO Mode
M_GRANT	EMI	reserved	—	—	—	M_GRANT	—	—	—	—	—	—	—
CSI_D4	IPU (CSI)	When using a 10-bit sensor this line is not needed for the sensor data and the suggested use is as the GPIO for IPU CSI chip select1	—	—	sw_mux_ctl_csi_d4[6:0]	SENSB_DATA[4]	—	—	—	—	—	CTI_TRIG_OUT_1_2	MCU3_4
CSI_D5	IPU (CSI)	When using a 10-bit sensor this line is not needed for the sensor data and the suggested use is as the GPIO for IPU CSI chip select2	—	—	sw_mux_ctl_csi_d5[6:0]	SENSB_DATA[5]	—	—	—	—	—	CTI_TRIG_OUT_1_3	MCU3_5
CSI_D6	IPU (CSI)	When used with a 10-bit sensor this signal is Sensor Port Data bit 0. If a 16-bit sensor is used it is Sensor Port Data bit 6.	ATA_DATA0	—	sw_mux_ctl_csi_d6[6:0]	SENSB_DATA[6]	—	—	—	—	—	CTI_TRIG_OUT_1_4	MCU3_6
CSI_D7	IPU (CSI)	When used with a 10-bit sensor this signal is Sensor Port Data bit 1. If a 16-bit sensor is used it is Sensor Port Data bit 7.	ATA_DATA1	—	sw_mux_ctl_csi_d7[6:0]	SENSB_DATA[7]	—	—	—	—	—	CTI_TRIG_OUT_1_5	MCU3_7
CSI_D8	IPU (CSI)	When used with a 10-bit sensor this signal is Sensor Port Data bit 2. If a 16-bit sensor is used it is Sensor Port Data bit 8.	ATA_DATA2	—	sw_mux_ctl_csi_d8[6:0]	SENSB_DATA[8]	—	—	—	—	—	—	MCU3_8

Table 4-8. Functional Multiplexing Configuration (continued)

Contact Name	Group	Description	Configured by GPR		SW_MUX_CTL Bits	Configured by SW_MUX_CTL							
			Hardware Mode 1	Hardware Mode 2		Functional Mode	Alternate Mode 1	Alternate Mode 2	Alternate Mode 3	Alternate Mode 4	Alternate Mode 5	Alternate Mode 6	GPIO Mode
CSI_D9	IPU (CSI)	When used with a 10-bit sensor this signal is Sensor Port Data bit 3. If a 16-bit sensor is used it is Sensor Port Data bit 9.	ATA_DATA 3	—	sw_mux_ctl _csi_d9[6:0]	SENSB_D ATA[9]	—	—	—	—	—	—	MCU3_9
CSI_D10	IPU (CSI)	When used with a 10-bit sensor this signal is Sensor Port Data bit 4. If a 16-bit sensor is used it is Sensor Port Data bit 10.	ATA_DATA 4	—	sw_mux_ctl _csi_d10[6: 0]	SENSB_D ATA[10]	—	—	—	—	—	—	MCU3_1 0
CSI_D11	IPU (CSI)	When used with a 10-bit sensor this signal is Sensor Port Data bit 5. If a 16-bit sensor is used it is Sensor Port Data bit 11.	ATA_DATA 5	—	sw_mux_ctl _csi_d11[6: 0]	SENSB_D ATA[11]	—	—	—	—	—	—	MCU3_1 1
CSI_D12	IPU (CSI)	When used with a 10-bit sensor this signal is Sensor Port Data bit 6. If a 16-bit sensor is used it is Sensor Port Data bit 12.	ATA_DATA 6	—	sw_mux_ctl _csi_d12[6: 0]	SENSB_D ATA[12]	—	—	—	—	—	—	MCU3_1 2
CSI_D13	IPU (CSI)	When used with a 10-bit sensor this signal is Sensor Port Data bit 7. If a 16-bit sensor is used it is Sensor Port Data bit 13.	ATA_DATA 7	—	sw_mux_ctl _csi_d13[6: 0]	SENSB_D ATA[13]	—	—	—	—	—	—	MCU3_1 3

Table 4-8. Functional Multiplexing Configuration (continued)

Contact Name	Group	Description	Configured by GPR		SW_MUX_CTL Bits	Configured by SW_MUX_CTL							
			Hardware Mode 1	Hardware Mode 2		Functional Mode	Alternate Mode 1	Alternate Mode 2	Alternate Mode 3	Alternate Mode 4	Alternate Mode 5	Alternate Mode 6	GPIO Mode
CSI_D14	IPU (CSI)	When used with a 10-bit sensor this signal is Sensor Port Data bit 8. If a 16-bit sensor is used it is Sensor Port Data bit 14.	ATA_DATA 8	—	sw_mux_ctl_csi_d14[6:0]	SENSB_DATA[14]	—	—	—	—	—	—	MCU3_14
CSI_D15	IPU (CSI)	When used with a 10-bit sensor this signal is Sensor Port Data bit 9. If a 16-bit sensor is used it is Sensor Port Data bit 15.	ATA_DATA 9	—	sw_mux_ctl_csi_d15[6:0]	SENSB_DATA[15]	—	—	—	—	—	—	MCU3_15
CSI_MCLK	IPU (CSI)	Sensor Port master Clock	ATA_DATA 10	—	sw_mux_ctl_csi_mclk[6:0]	SENSB_MCLK	—	—	—	—	—	—	MCU3_16
CSI_VSYNC	IPU (CSI)	Sensor port vertical sync	ATA_DATA 11	—	sw_mux_ctl_csi_vsync[6:0]	SENSB_VSYNC	—	—	—	—	—	—	MCU3_17
CSI_HSYNC	IPU (CSI)	Sensor port horizontal Sync	ATA_DATA 12	—	sw_mux_ctl_csi_hsync[6:0]	SENSB_HSYNC	—	—	—	—	—	—	MCU3_18
CSI_PIXCLK	IPU (CSI)	Sensor port data latch clock	ATA_DATA 13	—	sw_mux_ctl_csi_pixclk[6:0]	SENSB_PIX_CLK	—	—	—	—	—	—	MCU3_19
I2C_CLK	I2C	I2C clock	ATA_DATA 14	—	sw_mux_ctl_i2c_clk[6:0]	I2C1_SCL	—	—	IPU_DIAGB[0]	—	—	—	—
I2C_DAT	I2C	I2C data	ATA_DATA 15	—	sw_mux_ctl_i2c_dat[6:0]	I2C1_SDA	—	—	IPU_DIAGB[1]	—	—	—	—

Table 4-8. Functional Multiplexing Configuration (continued)

Contact Name	Group	Description	Configured by GPR		SW_MUX_CTL Bits	Configured by SW_MUX_CTL							
			Hardware Mode 1	Hardware Mode 2		Functional Mode	Alternate Mode 1	Alternate Mode 2	Alternate Mode 3	Alternate Mode 4	Alternate Mode 5	Alternate Mode 6	GPIO Mode
STXD3	AudioPort3-BB (HP3)	TxD	ATA_DATA7	USBH2_DATA2	sw_mux_ctl_stxd3[6:0]	HP3_TXDAT	—	—	IPU_DIAGB[2]	TRACEDATA_7	EVNTBUS_0	EMI_DEBUG0	MCU1_17
SRXD3	AudioPort3-BB (HP3)	RxD	ATA_DATA8	USBH2_DATA3	sw_mux_ctl_srxd3[6:0]	HP3_RXDAT	—	—	IPU_DIAGB[3]	TRACEDATA_8	EVNTBUS_1	EMI_DEBUG1	MCU1_18
SCK3	AudioPort3-BB (HP3)	Tx Serial Clock	ATA_DATA9	USBH2_DATA4	sw_mux_ctl_sck3[6:0]	HP3_TXCLK	—	—	IPU_DIAGB[4]	TRACEDATA_9	EVNTBUS_2	EMI_DEBUG2	—
SFS3	AudioPort3-BB (HP3)	Tx Frame Sync	ATA_DATA10	USBH2_DATA5	sw_mux_ctl_sfs3[6:0]	HP3_TXFS	—	—	IPU_DIAGB[5]	TRACEDATA_10	EVNTBUS_3	EMI_DEBUG3	—
STXD4	AudioPort4-PM_NB (PP1)	TxD	—	—	sw_mux_ctl_stxd4[6:0]	PP1_TXDAT	RXFS3	—	IPU_DIAGB[6]	—	EVNTBUS_4	EMI_DEBUG4	MCU1_19
SRXD4	AudioPort4-PM_NB (PP1)	RxD	—	—	sw_mux_ctl_srxd4[6:0]	PP1_RXDAT	RXCLK3	—	IPU_DIAGB[7]	—	EVNTBUS_5	ARM_COREASID0	MCU1_20
SCK4	AudioPort4-PM_NB (PP1)	Tx Serial Clock	—	—	sw_mux_ctl_sck4[6:0]	PP1_TXCLK	RXFS5	—	IPU_DIAGB[8]	—	EVNTBUS_6	ARM_COREASID1	—
SFS4	AudioPort4-PM_NB (PP1)	Tx Frame Sync	—	—	sw_mux_ctl_sfs4[6:0]	PP1_TXFS	RXCLK5	—	IPU_DIAGB[9]	—	EVNTBUS_7	ARM_COREASID2	—

Table 4-8. Functional Multiplexing Configuration (continued)

Contact Name	Group	Description	Configured by GPR		SW_MUX_CTL Bits	Configured by SW_MUX_CTL							
			Hardware Mode 1	Hardware Mode 2		Functional Mode	Alternate Mode 1	Alternate Mode 2	Alternate Mode 3	Alternate Mode 4	Alternate Mode 5	Alternate Mode 6	GPIO Mode
STXD5	AudioPort5-PM_WB (PP2)	TxD	—	—	sw_mux_ctl_stxd5[6:0]	PP2_TXDAT	—	—	IPU_DIAGB[10]	—	EVNTBUS_8	ARM_COREASID3	MCU1_21
SRXD5	AudioPort5-PM_WB (PP2)	RxD	—	—	sw_mux_ctl_srxd5[6:0]	PP2_RXDAT	—	—	IPU_DIAGB[11]	—	EVNTBUS_9	ARM_COREASID4	MCU1_22
SCK5	AudioPort5-PM_WB (PP2)	Tx Serial Clock	—	—	sw_mux_ctl_sck5[6:0]	PP2_TXCLK	—	—	IPU_DIAGB[12]	—	EVNTBUS_10	ARM_COREASID5	—
SFS5	AudioPort5-PM_WB (PP2)	Tx Frame Sync	—	—	sw_mux_ctl_sfs5[6:0]	PP2_TXFS	—	—	IPU_DIAGB[13]	—	EVNTBUS_11	ARM_COREASID6	—
STXD6	AudioPort6-BT (PP3)	TxD	ATA_DATA11	USBH2_DATA6	sw_mux_ctl_stxd6[6:0]	PP3_TXDAT	—	—	IPU_DIAGB[14]	TRACEDATA_11	EVNTBUS_12	ARM_COREASID7	MCU1_23
SRXD6	AudioPort6-BT (PP3)	RxD	ATA_DATA12	USBH2_DATA7	sw_mux_ctl_srxd6[6:0]	PP3_RXDAT	—	—	IPU_DIAGB[15]	TRACEDATA_12	EVNTBUS_13	M3IF_CHOSEN_MASTER_0	MCU1_24
SCK6	AudioPort6-BT (PP3)	Tx Serial Clock	ATA_DATA13	—	sw_mux_ctl_sck6[6:0]	PP3_TXCLK	—	—	IPU_DIAGB[16]	TRACEDATA_13	EVNTBUS_14	M3IF_CHOSEN_MASTER_1	MCU1_25

Table 4-8. Functional Multiplexing Configuration (continued)

Contact Name	Group	Description	Configured by GPR		SW_MUX_CTL Bits	Configured by SW_MUX_CTL							
			Hardware Mode 1	Hardware Mode 2		Functional Mode	Alternate Mode 1	Alternate Mode 2	Alternate Mode 3	Alternate Mode 4	Alternate Mode 5	Alternate Mode 6	GPIO Mode
SFS6	AudioPort6-BT (PP3)	Tx Frame Sync	USBH1_SUSPEND	—	sw_mux_ctl_sfs6[6:0]	PP3_TXFS	—	—	IPU_DIAGB[17]	TRACEDATA_14	EVNTBUS_15	M3IF_CHOSEN_MASTER_2	MCU1_26
CSPI1_MOSI	CSPI1_BB	Master Out/Slave In.	ATA_DATA0	ATA_INTRQ	sw_mux_ctl_cspi1_mosi[6:0]	CSPI1_MOSI	USBH1_RXDM	RXD3	IPU_DIAGB[18]	TRACEDATA_15	—	—	—
CSPI1_MISO	CSPI1_BB	Slave In/Master Out.	ATA_DATA1	ATA_BUFFER_EN	sw_mux_ctl_cspi1_miso[6:0]	CSPI1_MISO	USBH1_RXDP	TXD3	IPU_DIAGB[19]	TRACEDATA_16	—	—	—
CSPI1_SS0	CSPI1_BB	Slave Select (Selectable polarity).	ATA_DATA2	ATA_DMARQ	sw_mux_ctl_cspi1_ss0[6:0]	CSPI1_SS0_B	USBH1_TXDM	CSPI3_SS2	IPU_DIAGB[20]	TRACEDATA_17	—	—	—
CSPI1_SS1	CSPI1_BB	Slave Select (Selectable polarity).	ATA_DATA3	ATA_DA0	sw_mux_ctl_cspi1_ss1[6:0]	CSPI1_SS1_B	USBH1_TXDP	CSPI2_SS3	IPU_DIAGB[21]	TRACEDATA_18	—	—	—
CSPI1_SS2	CSPI1_BB	Slave Select (Selectable polarity).	ATA_DATA4	ATA_DA1	sw_mux_ctl_cspi1_ss2[6:0]	CSPI1_SS2_B	USBH1_RCV	CSPI3_SS3	IPU_DIAGB[22]	TRACEDATA_19	—	—	—
CSPI1_SCLK	CSPI1_BB	Serial Clock.	ATA_DATA5	ATA_DA2	sw_mux_ctl_cspi1_sclk[6:0]	CSPI1_CLK	USBH1_OEB	RTS3	IPU_DIAGB[23]	—	—	—	—
CSPI1_SPI_RDY	CSPI1_BB	Serial Data Ready.	ATA_DATA6	—	sw_mux_ctl_cspi1_spi_rdy[6:0]	CSPI1_IND_DATAREADY_B	USBH1_FS	CTS3	IPU_DIAGB[24]	—	—	—	—
CSPI2_MOSI	CSPI2_PM	Master Out/Slave In.	—	—	sw_mux_ctl_cspi2_mosi[6:0]	CSPI2_MOSI	I2C2_SCL	—	—	—	—	—	—

Table 4-8. Functional Multiplexing Configuration (continued)

Contact Name	Group	Description	Configured by GPR		SW_MUX_CTL Bits	Configured by SW_MUX_CTL							
			Hardware Mode 1	Hardware Mode 2		Functional Mode	Alternate Mode 1	Alternate Mode 2	Alternate Mode 3	Alternate Mode 4	Alternate Mode 5	Alternate Mode 6	GPIO Mode
CSPI2_MISO	CSPI2_PM	Slave In/Master Out.	—	—	sw_mux_ctl_cspi2_miso[6:0]	CSPI2_MISO	I2C2_SDA	—	—	—	—	—	—
CSPI2_SS0	CSPI2_PM	Slave Select (Selectable polarity).	—	—	sw_mux_ctl_cspi2_ss0[6:0]	CSPI2_SS0_B	CSPI3_SS0	—	—	—	—	—	—
CSPI2_SS1	CSPI2_PM	Slave Select (Selectable polarity).	—	—	sw_mux_ctl_cspi2_ss1[6:0]	CSPI2_SS1_B	CSPI3_SS1	CSPI1_SS3	—	—	—	—	—
CSPI2_SS2	CSPI2_PM	Slave Select (Selectable polarity).	—	—	sw_mux_ctl_cspi2_ss2[6:0]	CSPI2_SS2_B	I2C3_SDA	CSI_FLASH_STROBE	—	—	—	—	—
CSPI2_SCLK	CSPI2_PM	Serial Clock.	—	—	sw_mux_ctl_cspi2_sclk[6:0]	CSPI2_CLK	I2C3_SCL	—	—	—	—	—	—
CSPI2_SPI_RDY	CSPI2_PM	—	—	—	sw_mux_ctl_cspi2_spi_rdy[6:0]	CSPI2_DATAREADY_B	—	—	—	—	—	—	—
RXD1	UART1_GPS	Rx Data. (+CE Bus 12)	TRSTB	—	sw_mux_ctl_rxd1[6:0]	UART1_RXD_MUX	USBOTG_DATA4	PP4_TXDAT/STDA	—	DSR_DCE1	—	—	MCU2_4
TXD1	UART1_GPS	Tx Data. + (CE Bus 10)	TCK	—	sw_mux_ctl_txd1[6:0]	UART1_TXD_MUX	USBOTG_DATA1	PP4_TXCLK/SCK	—	RI_DCE1	—	—	MCU2_5
RTS1	UART1_GPS	Request to send. + (CE Bus 9)	—	—	sw_mux_ctl_rts1[6:0]	UART1_RTS_B	—	PP4_TXFS/FS	—	DCD_DCE1	—	—	MCU2_6
CTS1	UART1_GPS	Clear to send. + CE Bus 8)	DE_B	—	sw_mux_ctl_cts1[6:0]	UART1_CTS_B	—	—	—	—	—	—	MCU2_7

Table 4-8. Functional Multiplexing Configuration (continued)

Contact Name	Group	Description	Configured by GPR		SW_MUX_CTL Bits	Configured by SW_MUX_CTL							
			Hardware Mode 1	Hardware Mode 2		Functional Mode	Alternate Mode 1	Alternate Mode 2	Alternate Mode 3	Alternate Mode 4	Alternate Mode 5	Alternate Mode 6	GPIO Mode
DTR_DCE1	UART1_GPS	Full UART IF + CE Bus 11	TMS	—	sw_mux_ctl_dtr_dce1[6:0]	UART1_DTR_DCE_I_B	—	PP4_RXDAT/SRDA	—	—	—	—	MCU2_8
DSR_DCE1	Full UART IF	Full UART IF + CE Bus 4	TDO	USB OTG_DATA4	sw_mux_ctl_dsr_dce1[6:0]	UART1_DSR_DCE_O_B	CSPI1_SCLK	TXD1	DSR_DCE2	—	—	—	MCU2_9
RI_DCE1	Full UART IF	Full UART IF + CE Bus 5	TDI	USB OTG_DATA3	sw_mux_ctl_ri_dce1[6:0]	UART1_RI_DCE_O_B	CSPI1_SPI_RDY	RXD1	RI_DCE2	—	—	—	MCU2_10
DCD_DCE1	Full UART IF	Full UART IF + CE Bus 6	RESET_IN	USB OTG_DATA5	sw_mux_ctl_dcd_dce1[6:0]	UART1_DCD_DCE_O_B	CSPI1_SS3	RTS1	DCD_DCE2	USB_PWR	—	—	MCU2_11
DTR_DTE1	Full UART IF	—	CSPI1_MOSI	—	sw_mux_ctl_dtr_dte1[6:0]	UART1_DTR_DTE_O_B	—	—	DTR_DTE2	—	EVNTBUS_16	—	MCU2_12
DSR_DTE1	Full UART IF	—	CSPI1_MISO	—	sw_mux_ctl_dsr_dte1[6:0]	UART1_DSR_DTE_I_B	DSR_DTE2	—	—	—	EVNTBUS_17	—	MCU2_13
RI_DTE1	Full UART IF	—	CSPI1_SS0	—	sw_mux_ctl_ri_dte1[6:0]	UART1_RI_DTE_I_B	RI_DTE2	I2C2_SCL	IPU_DIAGB[25]	—	EVNTBUS_18	—	MCU2_14
DCD_DTE1	Full UART IF	—	CSPI1_SS1	—	sw_mux_ctl_dcd_dte1[6:0]	UART1_DCD_DTE_I_B	DCD_DTE2	I2C2_SDA	IPU_DIAGB[26]	—	EVNTBUS_19	—	MCU2_15
DTR_DCE2	Full UART IF	—	CSPI1_SS2	—	sw_mux_ctl_dtr_dce2[6:0]	UART2_DTR_DCE_I_B	—	—	IPU_DIAGB[27]	—	—	—	MCU2_16

Table 4-8. Functional Multiplexing Configuration (continued)

Contact Name	Group	Description	Configured by GPR		SW_MUX_CTL Bits	Configured by SW_MUX_CTL							
			Hardware Mode 1	Hardware Mode 2		Functional Mode	Alternate Mode 1	Alternate Mode 2	Alternate Mode 3	Alternate Mode 4	Alternate Mode 5	Alternate Mode 6	GPIO Mode
RXD2	UART2_IR	RX Data	FIRI RXD	—	sw_mux_ctl_rxd2[6:0]	UART2_RXD_MUX	—	—	IPU_DIAGB[28]	—	—	—	MCU1_27
TXD2	UART2_IR	TX Data	FIRI TXD	—	sw_mux_ctl_txd2[6:0]	UART2_TXD_MUX	—	—	IPU_DIAGB[29]	—	—	—	MCU1_28
RTS2	UART2_IR	Request to send	FIRI RXD	—	sw_mux_ctl_rts2[6:0]	UART2_RTS_B	—	—	IPU_DIAGB[30]	—	—	—	—
CTS2	UART2_IR	Clear to send	FIRI TXD	—	sw_mux_ctl_cts2[6:0]	UART2_CTS_B	—	—	IPU_DIAGB[31]	—	—	—	—
BATT_LINE	1-Wire	1-Wire data, external battery monitor	—	—	sw_mux_ctl_batt_line[6:0]	OWIRE_BATTERY_LINE	—	—	—	—	—	—	MCU2_17
KEY_ROW0	Keypad	keypad row sense 0	—	—	sw_mux_ctl_key_row0[6:0]	ROW[0]	—	—	—	—	—	—	—
KEY_ROW1	Keypad	keypad row sense 1	—	—	sw_mux_ctl_key_row1[6:0]	ROW[1]	—	—	—	—	—	—	—
KEY_ROW2	Keypad	keypad row sense 2	—	—	sw_mux_ctl_key_row2[6:0]	ROW[2]	—	—	—	—	—	—	—
KEY_ROW3	Keypad	keypad row sense 3	—	—	sw_mux_ctl_key_row3[6:0]	ROW[3]	—	—	—	TRACECTL	—	—	—
KEY_ROW4	Keypad	keypad row sense 4	—	—	sw_mux_ctl_key_row4[6:0]	ROW[4]	—	—	—	TRACECLK	—	—	MCU2_18

Table 4-8. Functional Multiplexing Configuration (continued)

Contact Name	Group	Description	Configured by GPR		SW_MUX_CTL Bits	Configured by SW_MUX_CTL							
			Hardware Mode 1	Hardware Mode 2		Functional Mode	Alternate Mode 1	Alternate Mode 2	Alternate Mode 3	Alternate Mode 4	Alternate Mode 5	Alternate Mode 6	GPIO Mode
KEY_ROW5	Keypad	keypad row sense 5	—	—	sw_mux_ctl_key_row5[6:0]	ROW[5]	—	—	—	TRACEDATA_0	—	—	MCU2_19
KEY_ROW6	Keypad	keypad row sense 6	ATA_INTRQ	—	sw_mux_ctl_key_row6[6:0]	ROW[6]	—	—	—	TRACEDATA_1	—	—	MCU2_20
KEY_ROW7	Keypad	keypad row sense 7	ATA_BUFFER_EN	—	sw_mux_ctl_key_row7[6:0]	ROW[7]	—	—	—	TRACEDATA_2	—	—	MCU2_21
KEY_COL0	Keypad	keypad column driver 0	—	—	sw_mux_ctl_key_col0[6:0]	COL[0]	—	—	—	—	—	—	—
KEY_COL1	Keypad	keypad column driver 1	—	—	sw_mux_ctl_key_col1[6:0]	COL[1]	—	—	—	—	—	—	—
KEY_COL2	Keypad	keypad column driver 2	—	—	sw_mux_ctl_key_col2[6:0]	COL[2]	—	—	—	—	—	—	—
KEY_COL3	Keypad	keypad column driver 3	—	—	sw_mux_ctl_key_col3[6:0]	COL[3]	—	—	—	TRACEDATA_3	—	—	—
KEY_COL4	Keypad	keypad column driver 4	ATA_DMARQ	—	sw_mux_ctl_key_col4[6:0]	COL[4]	—	—	—	TRACEDATA_4	—	—	MCU2_22
KEY_COL5	Keypad	keypad column driver 5	ATA_DA0	—	sw_mux_ctl_key_col5[6:0]	COL[5]	—	—	—	TRACEDATA_5	—	—	MCU2_23
KEY_COL6	Keypad	keypad column driver 6	ATA_DA1	—	sw_mux_ctl_key_col6[6:0]	COL[6]	—	—	—	TRACEDATA_6	—	—	MCU2_24

Table 4-8. Functional Multiplexing Configuration (continued)

Contact Name	Group	Description	Configured by GPR		SW_MUX_CTL Bits	Configured by SW_MUX_CTL							
			Hardware Mode 1	Hardware Mode 2		Functional Mode	Alternate Mode 1	Alternate Mode 2	Alternate Mode 3	Alternate Mode 4	Alternate Mode 5	Alternate Mode 6	GPIO Mode
KEY_COL7	Keypad	keypad column driver 7	ATA_DA2	—	sw_mux_ctl_key_col7[6:0]	COL[7]	—	—	—	TRACEDATA_7	—	—	MCU2_25
RTCK	JTAG	ARM debug test clock	—	—	—	DBGRTCK	—	—	—	—	—	—	—
TCK	JTAG	JTAG Tap clock	—	—	sw_mux_ctl_tck[6:0]	TCK	—	—	—	—	—	—	—
TMS	JTAG	JTAG Tap mode select	—	—	sw_mux_ctl_tms[6:0]	TMS	—	—	—	—	—	—	—
TDI	JTAG	JTAG Tap Data In	—	—	sw_mux_ctl_tdi[6:0]	TDI	—	—	—	—	—	—	—
TDO	JTAG	JTAG Tap data out	—	—	—	TDO	—	—	—	—	—	—	—
TRSTB	JTAG	JTAG Tap reset	—	—	sw_mux_ctl_trstb[6:0]	TRST_B	—	—	—	—	—	—	—
DE_B	JTAG	JTAG debug enable	—	—	sw_mux_ctl_de_b[6:0]	DEBUG_IN_B	—	—	—	—	—	—	—
SJC_MOD	JTAG	JTAG Mode	—	—	—	SJC_MOD	—	—	—	—	—	—	—
USB_PWR	USB GEN	USB Generic	—	—	sw_mux_ctl_usb_pwr[6:0]	USB_PWR	—	—	—	—	—	MAX1_HMASTER_0	MCU1_29
USB_OC	USB GEN	USB Generic	—	—	sw_mux_ctl_usb_oc[6:0]	USB_OC	—	—	—	—	—	MAX1_HMASTER_1	MCU1_30
USB_BYP	USB GEN	USB Generic	—	—	sw_mux_ctl_usb_byp[6:0]	USB_BYPASS_B	—	—	—	—	—	MAX1_HMASTER_2	MCU1_31

Table 4-8. Functional Multiplexing Configuration (continued)

Contact Name	Group	Description	Configured by GPR		SW_MUX_CTL Bits	Configured by SW_MUX_CTL							
			Hardware Mode 1	Hardware Mode 2		Functional Mode	Alternate Mode 1	Alternate Mode 2	Alternate Mode 3	Alternate Mode 4	Alternate Mode 5	Alternate Mode 6	GPIO Mode
USBOTG_CLK	USBOTG	USB OTG FS/ULPI Port	—	—	sw_mux_ctl_usbotg_clk[6:0]	OTG_CLK	—	—	—	—	—	MAX1_HMASTER_3	—
USBOTG_DIR	USBOTG	USB OTG FS/ULPI Port	—	—	sw_mux_ctl_usbotg_dir[6:0]	OTG_DIR	—	—	—	—	—	MAX0_HMASTER_0	—
USBOTG_STP	USBOTG	USB OTG FS/ULPI Port	—	—	sw_mux_ctl_usbotg_stp[6:0]	OTG_STP	—	—	—	—	—	MAX0_HMASTER_1	—
USBOTG_NXT	USBOTG	USB OTG FS/ULPI Port	—	—	sw_mux_ctl_usbotg_nxt[6:0]	OTG_NXT	—	—	—	—	—	MAX0_HMASTER_2	—
USBOTG_DATA0	USBOTG	USB OTG FS/ULPI Port + CE Bus	—	—	sw_mux_ctl_usbotg_data0[6:0]	OTG_DATA0	—	UART4_CTS	—	—	—	MAX0_HMASTER_3	—
USBOTG_DATA1	USBOTG	USB OTG FS/ULPI Port	—	—	sw_mux_ctl_usbotg_data1[6:0]	OTG_DATA1	—	—	—	—	—	—	—
USBOTG_DATA2	USBOTG	USB OTG FS/ULPI Port + CE Bus	—	—	sw_mux_ctl_usbotg_data2[6:0]	OTG_DATA2	—	—	—	—	—	—	—
USBOTG_DATA3	USBOTG	USB OTG FS/ULPI Port	—	—	sw_mux_ctl_usbotg_data3[6:0]	OTG_DATA3	—	UART4_RXD	—	—	—	—	—

Table 4-8. Functional Multiplexing Configuration (continued)

Contact Name	Group	Description	Configured by GPR		SW_MUX_CTL Bits	Configured by SW_MUX_CTL							
			Hardware Mode 1	Hardware Mode 2		Functional Mode	Alternate Mode 1	Alternate Mode 2	Alternate Mode 3	Alternate Mode 4	Alternate Mode 5	Alternate Mode 6	GPIO Mode
USBOTG_DATA4	USBOTG	USB OTG FS/ULPI Port	—	—	sw_mux_ctl_usbotg_data4[6:0]	OTG_DATA4	—	UART4_TXD	—	—	—	—	—
USBOTG_DATA5	USBOTG	USB OTG FS/ULPI Port	—	—	sw_mux_ctl_usbotg_data5[6:0]	OTG_DATA5	—	UART4_RTS	—	—	—	—	—
USBOTG_DATA6	USBOTG	USB OTG FS/ULPI Port	—	—	sw_mux_ctl_usbotg_data6[6:0]	OTG_DATA6	—	—	—	—	—	—	—
USBOTG_DATA7	USBOTG	USB OTG FS/ULPI Port	—	—	sw_mux_ctl_usbotg_data7[6:0]	OTG_DATA7	—	—	—	—	—	—	—
USBH2_CLK	USBH2	USB Host2 FS/ULPI	ATA_INTRQ	—	sw_mux_ctl_usbh2_clk[6:0]	UH2_CLK	UART5_RTS	—	—	TRACEDATA_20	—	—	—
USBH2_DIR	USBH2	USB Host2 FS/ULPI	ATA_BUFFER_EN	—	sw_mux_ctl_usbh2_dir[6:0]	UH2_DIR	UART5_RXD	—	—	TRACEDATA_21	—	—	—
USBH2_STP	USBH2	USB Host2 FS/ULPI	ATA_DMARQ	—	sw_mux_ctl_usbh2_stp[6:0]	UH2_STP	UART5_TXD	—	—	TRACEDATA_22	—	—	—
USBH2_NXT	USBH2	USB Host2 FS/ULPI	ATA_DA0	—	sw_mux_ctl_usbh2_nxt[6:0]	UH2_NXT	UART5_CTS	—	—	TRACEDATA_23	—	—	—
USBH2_DATA0	USBH2	USB Host2 FS/ULPI	ATA_DA1	—	sw_mux_ctl_usbh2_data0[6:0]	UH2_DATA0	—	—	—	TRACECTL	—	—	—
USBH2_DATA1	USBH2	USB Host2 FS/ULPI	ATA_DA2	—	sw_mux_ctl_usbh2_data1[6:0]	UH2_DATA1	—	—	—	TRACECLK	—	—	—

Table 4-8. Functional Multiplexing Configuration (continued)

Contact Name	Group	Description	Configured by GPR		SW_MUX_CTL Bits	Configured by SW_MUX_CTL							
			Hardware Mode 1	Hardware Mode 2		Functional Mode	Alternate Mode 1	Alternate Mode 2	Alternate Mode 3	Alternate Mode 4	Alternate Mode 5	Alternate Mode 6	GPIO Mode
LD0	IPU (LCD)	—	—	—	sw_mux_ctl_ld0[6:0]	DISPB_DATA[0]	—	—	—	—	—	SDMA_DEBUG_PC_0	—
LD1	IPU (LCD)	—	—	—	sw_mux_ctl_ld1[6:0]	DISPB_DATA[1]	—	—	—	—	—	SDMA_DEBUG_PC_1	—
LD2	IPU (LCD)	—	—	—	sw_mux_ctl_ld2[6:0]	DISPB_DATA[2]	—	—	—	—	—	SDMA_DEBUG_PC_2	—
LD3	IPU (LCD)	—	—	—	sw_mux_ctl_ld3[6:0]	DISPB_DATA[3]	—	—	—	—	—	SDMA_DEBUG_PC_3	—
LD4	IPU (LCD)	—	—	—	sw_mux_ctl_ld4[6:0]	DISPB_DATA[4]	—	—	—	—	—	SDMA_DEBUG_PC_4	—
LD5	IPU (LCD)	—	—	—	sw_mux_ctl_ld5[6:0]	DISPB_DATA[5]	—	—	—	—	—	SDMA_DEBUG_PC_5	—
LD6	IPU (LCD)	—	—	—	sw_mux_ctl_ld6[6:0]	DISPB_DATA[6]	—	—	—	—	—	SDMA_DEBUG_PC_6	—

Table 4-8. Functional Multiplexing Configuration (continued)

Contact Name	Group	Description	Configured by GPR		SW_MUX_CTL Bits	Configured by SW_MUX_CTL							
			Hardware Mode 1	Hardware Mode 2		Functional Mode	Alternate Mode 1	Alternate Mode 2	Alternate Mode 3	Alternate Mode 4	Alternate Mode 5	Alternate Mode 6	GPIO Mode
LD7	IPU (LCD)	—	—	—	sw_mux_ctl_ld7[6:0]	DISPB_DATA[7]	—	—	—	—	—	SDMA_DEBUG_PC_7	—
LD8	IPU (LCD)	—	—	—	sw_mux_ctl_ld8[6:0]	DISPB_DATA[8]	—	—	—	—	—	SDMA_DEBUG_PC_8	—
LD9	IPU (LCD)	—	—	—	sw_mux_ctl_ld9[6:0]	DISPB_DATA[9]	—	—	—	—	—	SDMA_DEBUG_PC_9	—
LD10	IPU (LCD)	—	—	—	sw_mux_ctl_ld10[6:0]	DISPB_DATA[10]	—	—	—	—	—	SDMA_DEBUG_PC_10	—
LD11	IPU (LCD)	—	—	—	sw_mux_ctl_ld11[6:0]	DISPB_DATA[11]	—	—	—	—	—	SDMA_DEBUG_PC_11	—
LD12	IPU (LCD)	—	—	—	sw_mux_ctl_ld12[6:0]	DISPB_DATA[12]	—	—	—	—	—	SDMA_DEBUG_PC_12	—
LD13	IPU (LCD)	—	—	—	sw_mux_ctl_ld13[6:0]	DISPB_DATA[13]	—	—	—	—	—	SDMA_DEBUG_PC_13	—

Table 4-8. Functional Multiplexing Configuration (continued)

Contact Name	Group	Description	Configured by GPR		SW_MUX_CTL Bits	Configured by SW_MUX_CTL							
			Hardware Mode 1	Hardware Mode 2		Functional Mode	Alternate Mode 1	Alternate Mode 2	Alternate Mode 3	Alternate Mode 4	Alternate Mode 5	Alternate Mode 6	GPIO Mode
LD14	IPU (LCD)	—	—	—	sw_mux_ctl_ld14[6:0]	DISPB_DATA[14]	—	—	—	—	—	SDMA_DEBUG_EVENT_CHANNEL_0	—
LD15	IPU (LCD)	—	—	—	sw_mux_ctl_ld15[6:0]	DISPB_DATA[15]	—	—	—	—	—	SDMA_DEBUG_EVENT_CHANNEL_1	—
LD16	IPU (LCD)	—	—	—	sw_mux_ctl_ld16[6:0]	DISPB_DATA[16]	—	—	—	—	—	SDMA_DEBUG_EVENT_CHANNEL_2	—
LD17	IPU (LCD)	—	—	—	sw_mux_ctl_ld17[6:0]	DISPB_DATA[17]	—	—	—	—	—	SDMA_DEBUG_EVENT_CHANNEL_3	—

Table 4-8. Functional Multiplexing Configuration (continued)

Contact Name	Group	Description	Configured by GPR		SW_MUX_CTL Bits	Configured by SW_MUX_CTL							
			Hardware Mode 1	Hardware Mode 2		Functional Mode	Alternate Mode 1	Alternate Mode 2	Alternate Mode 3	Alternate Mode 4	Alternate Mode 5	Alternate Mode 6	GPIO Mode
VSYNC0	IPU (LCD)	frame sync	—	—	sw_mux_ctl_vsync0[6:0]	DISPB_D0_VSYNC	—	—	—	—	—	SDMA_DEBUG_EVENT_CHANNEL_4	—
HSYNC	IPU (LCD)	line sync	—	—	sw_mux_ctl_hsync[6:0]	DISPB_D3_HSYNC	—	—	—	—	—	SDMA_DEBUG_EVENT_CHANNEL_5	—
FPSHIFT	IPU (LCD)	shift	—	—	sw_mux_ctl_fpshift[6:0]	DISPB_D3_CLK	DISPB_BCLK	—	—	—	—	SDMA_DEBUG_CORE_STATUS_0	—
DRDY0	IPU (LCD)	DRDY/VLD	—	—	sw_mux_ctl_drdy0[6:0]	DISPB_D3_DRDY	—	—	—	—	—	SDMA_DEBUG_CORE_STATUS_1	—

Table 4-8. Functional Multiplexing Configuration (continued)

Contact Name	Group	Description	Configured by GPR		SW_MUX_CTL Bits	Configured by SW_MUX_CTL							
			Hardware Mode 1	Hardware Mode 2		Functional Mode	Alternate Mode 1	Alternate Mode 2	Alternate Mode 3	Alternate Mode 4	Alternate Mode 5	Alternate Mode 6	GPIO Mode
SD_D_I	IPU (LCD)	Data in for Serial Display	—	—	sw_mux_ctl_sd_d_i[6:0]	—	—	SD_D_I	—	—	—	SDMA_DEBUG_CORE_STATUS_2	MCU3_20
SD_D_IO	IPU (LCD)	Data in/out for Serial Display	—	—	sw_mux_ctl_sd_d_io[6:0]	DISPB_SD_D	—	—	—	—	—	SDMA_DEBUG_CORE_STATUS_3	MCU3_21
SD_D_CLK	IPU (LCD)	Serial Display clock	—	—	sw_mux_ctl_sd_d_clk[6:0]	DISPB_SD_D_CLK	—	—	—	—	—	—	MCU3_22
LCS0	IPU (LCD)	Asynch. Port chip select	—	—	sw_mux_ctl_lcs0[6:0]	DISPB_D0_CS	DISPB_BCLK	—	—	—	—	—	MCU3_23
LCS1	IPU (LCD)	Asynch. Port chip select	—	—	sw_mux_ctl_lcs1[6:0]	DISPB_D1_C	—	—	—	—	—	—	MCU3_24
SER_RS	IPU (LCD)	Asynch. Serial Port data/comm	—	—	sw_mux_ctl_ser_rs[6:0]	DISPB_SER_RS	—	—	—	—	—	—	MCU3_25
PAR_RS	IPU (LCD)	Asynch.Parallel Port data/comm	—	—	sw_mux_ctl_par_rs[6:0]	DISPB_PAR_RS	—	—	—	—	—	—	—
WRITE	IPU (LCD)	Asynch. Port write	—	—	sw_mux_ctl_write[6:0]	DISPB_WR	—	—	—	—	—	—	—

Table 4-8. Functional Multiplexing Configuration (continued)

Contact Name	Group	Description	Configured by GPR		SW_MUX_CTL Bits	Configured by SW_MUX_CTL							
			Hardware Mode 1	Hardware Mode 2		Functional Mode	Alternate Mode 1	Alternate Mode 2	Alternate Mode 3	Alternate Mode 4	Alternate Mode 5	Alternate Mode 6	GPIO Mode
READ	IPU (LCD)	Asynch. Port read	—	—	sw_mux_ctl_read[6:0]	DISPB_RD	—	—	—	—	—	—	—
VSYNC3	IPU (LCD)	vsync	—	—	sw_mux_ctl_vsync3[6:0]	DISPB_D3_VSYNC	—	—	—	—	—	—	—
CONTRAST	IPU (LCD)	—	—	—	sw_mux_ctl_contrast[6:0]	DISPB_CONTRAST	—	—	—	—	—	—	—
D3_REV	IPU (LCD)	—	—	—	sw_mux_ctl_d3_rev[6:0]	DISPB_D3_REV	—	—	—	—	—	—	—
D3_CLS	IPU (LCD)	—	—	—	sw_mux_ctl_d3_cls[6:0]	DISPB_D3_CLS	—	—	—	—	—	—	—
D3_SPL	IPU (LCD)	—	—	—	sw_mux_ctl_d3_spl[6:0]	DISPB_D3_SPL	—	—	—	—	—	—	—
SD1_CMD	SD/MMC1	—	—	—	sw_mux_ctl_sd1_cmd[6:0]	SDHC1_CMD	MSHC1_SCLK	—	—	TRACEDATA_0	—	—	MCU2_26
SD1_CLK	SD/MMC1	—	—	—	sw_mux_ctl_sd1_clk[6:0]	SDHC1_MMC_CLK	MSHC1_BS	—	—	TRACEDATA_1	—	—	MCU2_27
SD1_DATA0	SD/MMC1	—	—	—	sw_mux_ctl_sd1_data0[6:0]	SDHC1_DATA0	MSHC1_SDIO_DATA0	—	—	TRACEDATA_2	—	—	MCU2_28

Table 4-8. Functional Multiplexing Configuration (continued)

Contact Name	Group	Description	Configured by GPR		SW_MUX_CTL Bits	Configured by SW_MUX_CTL							
			Hardware Mode 1	Hardware Mode 2		Functional Mode	Alternate Mode 1	Alternate Mode 2	Alternate Mode 3	Alternate Mode 4	Alternate Mode 5	Alternate Mode 6	GPIO Mode
SD1_DATA1	SD/MMC 1	—	—	—	sw_mux_ctl_sd1_data 1[6:0]	SDHC1_DATA1	MSHC1_DATA1	—	—	TRACEDATA_3	—	—	MCU2_29
SD1_DATA2	SD/MMC 1	—	—	—	sw_mux_ctl_sd1_data 2[6:0]	SDHC1_DATA2	MSHC1_DATA2	—	—	TRACEDATA_4	—	—	MCU2_30
SD1_DATA3	SD/MMC 1	—	—	—	sw_mux_ctl_sd1_data 3[6:0]	SDHC1_DATA3	MSHC1_DATA3	CTI_TRIG_IN_1_7	—	TRACEDATA_5	—	—	MCU2_31
ATA_CS0	ATA	—	—	—	sw_mux_ctl_ata_cs0[6:0]	ATA_CS0	UART4_RXD	CSI_D0	SD_D_CLK	TRACEDATA_6	—	—	MCU3_26
ATA_CS1	ATA	—	—	—	sw_mux_ctl_ata_cs1[6:0]	ATA_CS1	UART4_RTS	CSI_D1	LCS1	TRACEDATA_7	—	—	MCU3_27
ATA_DIOR	ATA	—	—	—	sw_mux_ctl_ata_dior[6:0]	ATA_DIOR	UART4_TXD	CSI_D2	SER_RS	TRACECTL	—	—	MCU3_28
ATA_DIOW	ATA	—	—	—	sw_mux_ctl_ata_diow[6:0]	ATA_DIOW	UART4_CTS	CSI_D3	—	TRACECLK	—	—	MCU3_29
ATA_DMACK	ATA	—	—	—	sw_mux_ctl_ata_dmack[6:0]	ATA_DMACK	SD_D_O	—	—	—	—	—	MCU3_30
ATA_RESET_B	ATA	—	—	—	sw_mux_ctl_ata_reset_b[6:0]	ATA_RESET_B	SD_D	—	—	—	—	—	MCU3_31
CE_CONTROL	CE CONTROL	—	—	—	—	CE_CONTROL	—	—	—	—	—	—	—

Table 4-8. Functional Multiplexing Configuration (continued)

Contact Name	Group	Description	Configured by GPR		SW_MUX_CTL Bits	Configured by SW_MUX_CTL							
			Hardware Mode 1	Hardware Mode 2		Functional Mode	Alternate Mode 1	Alternate Mode 2	Alternate Mode 3	Alternate Mode 4	Alternate Mode 5	Alternate Mode 6	GPIO Mode
CLKSS	Clock and Reset and PM	Clock Source Select at reset	—	—	—	CLKSS	—	—	—	—	—	—	—
CSPI3_MOSI	CSPI3_MM	Master Out/Slave In	—	—	sw_mux_ctl_cspi3_mosi[6:0]	CSPI3_MOSI	RXD3	—	—	—	—	—	—
CSPI3_MISO	CSPI3_MM	Slave In/Master Out	—	—	sw_mux_ctl_cspi3_miso[6:0]	CSPI3_MISO	TXD3	—	—	—	—	—	—
CSPI3_SCLK	CSPI3_MM	Serial Clock	—	—	sw_mux_ctl_cspi3_sclk[6:0]	CSPI3_CLK	RTS3	—	—	—	—	—	—
CSPI3_SPI_RDY	CSPI3_MM	Serial Data Ready	—	—	sw_mux_ctl_cspi3_spi_rdy[6:0]	CSPI3_IND_DATAREADY_B	CTS3	—	—	—	—	—	—
TTM_PAD	TTM_PAD	Special TTM pad, factory use only.	—	—	—	—	—	—	—	—	—	—	—

4.3.6 ATA Routing Options

Table 4-9 lists seven signal routing options for the ATA signals. The primary purpose of this table is to provide a summary of the seven most commonly used routing scenarios of the ATA signals that are controlled by the hardware modes.

Table 4-9. ATA Signal Routing Options using Hardware Modes

Group	Scenario A MUX Mode; GPR Bit; ATA Signals	Scenario B MUX Mode; GPR Bit; ATA Signals	Scenario C MUX Mode; GPR Bit; ATA Signals	Scenario D MUX Mode; GPR Bit; ATA Signals	Scenario E MUX Mode; GPR Bit; ATA Signals	Scenario F MUX Mode; GPR Bit; ATA Signals	Scenario G MUX Mode; GPR Bit; ATA Signals
PWM	HW1; GPR[3]; IORDY	HW1; GPR[3]; IORDY	HW1; GPR[3]; IORDY	HW1; GPR[3]; IORDY	HW1; GPR[3]; IORDY	HW1; GPR[3]; IORDY	HW1; GPR[3]; IORDY
Timer	HW1; GPR[9]; DATA[14,15]	HW1; GPR[9]; DATA[14,15]	–	–	–	HW1; GPR[9]; DATA[14,15]	HW1; GPR[9]; DATA[14,15]
EMI (NANDFC)	HW1; GPR[5]; DATA[7:13]	–	–	–	HW2; GPR[6]; DA0-2, DMARQ, INTRQ, BUFFER_EN	HW2; GPR[6]; DA0-2, DMARQ, INTRQ, BUFFER_EN	–
IPU (CSI)	–	–	HW1; GPR[7]; DATA[0:13]	HW1; GPR[7]; DATA[0:13]	HW1; GPR[7]; DATA[0:13]	–	–
AudioPort3	–	HW1; GPR[8]; DATA[7:10]	–	–	–	HW1; GPR[8]; DATA[7:10]	HW1; GPR[8]; DATA[7:10]
AudioPort6	–	HW1; GPR[8]; DATA[11:13]	–	–	–	HW1; GPR[8]; DATA[11:13]	HW1; GPR[8]; DATA[11:13]
Keypad	–	HW1; GPR[26]; DA0-2, DMARQ, INTRQ, BUFFER_EN	–	–	–	–	–
CSPI1	HW1; GPR[9]; DATA[0:6]	HW1; GPR[9]; DATA[0:6]	–	HW2; GPR[10]; DA0-2, DMARQ, INTRQ, BUFFER_EN	–	HW1; GPR[9]; DATA[0:6]	HW1; GPR[9]; DATA[0:6]
USBH2	HW1; GPR[4]; DA0-2, DMARQ, INTRQ, BUFFER_EN	–	HW1; GPR[4]; DA0-2, DMARQ, INTRQ, BUFFER_EN	–	–	–	HW1; GPR[4]; DA0-2, DMARQ, INTRQ, BUFFER_EN

Table 4-9. ATA Signal Routing Options using Hardware Modes (continued)

Group	Scenario A MUX Mode; GPR Bit; ATA Signals	Scenario B MUX Mode; GPR Bit; ATA Signals	Scenario C MUX Mode; GPR Bit; ATA Signals	Scenario D MUX Mode; GPR Bit; ATA Signals	Scenario E MUX Mode; GPR Bit; ATA Signals	Scenario F MUX Mode; GPR Bit; ATA Signals	Scenario G MUX Mode; GPR Bit; ATA Signals
I2C	–	–	HW1; GPR[7]; DATA[14,15]	HW1; GPR[7]; DATA[14,15]	HW1; GPR[7]; DATA[14,15]	–	–
ATA	Functional; (no GPR bit); DIOR, DIOW, CS0, CS1, RESET_B, DMACK	Functional; (no GPR bit); DIOR, DIOW, CS0, CS1, RESET_B, DMACK	Functional; (no GPR bit); DIOR, DIOW, CS0, CS1, RESET_B, DMACK	Functional; (no GPR bit); DIOR, DIOW, CS0, CS1, RESET_B, DMACK	Functional; (no GPR bit); DIOR, DIOW, CS0, CS1, RESET_B, DMACK	Functional; (no GPR bit); DIOR, DIOW, CS0, CS1, RESET_B, DMACK	Functional; (no GPR bit); DIOR, DIOW, CS0, CS1, RESET_B, DMACK

See Table 4-8 for (Contact) Group, (Functional) MUX Modes, and associated Contact Signal list. See Table 4-5 for GPR bit and MUX Mode information.
A dash indicates that a Contact Group is not used for a particular Scenario. Therefore, the Group may be used for other signal multiplexing.

4.3.7 Software Pad Control Register (SW_PAD_CTL)

The SW_PAD_CTL registers control the characteristics of the I/O lines. Figure 4-88 provides the register's field descriptions; Table 4-10 lists the control by bit.

0x43FA_C154
to
0x43FA_C308

Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			SW_PAD_CTL_IO3										SW_PAD_CTL_IO2			
W			loopback	ipp_pke	ipp_pue	ipp_pus0	ipp_pus1	ipp_hys	ipp_ode	ipp_dse1	ipp_dse0	ipp_sre	loopback	ipp_pke	ipp_pue	ipp_pus0
Field bits	—	—	9	8	7	6	5	4	3	2	1	0	9	8	7	6

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SW_PAD_CTL_IO2						SW_PAD_CTL_IO1									
W	ipp_pus1	ipp_hys	ipp_ode	ipp_dse1	ipp_dse0	ipp_sre	loopback	ipp_pke	ipp_pue	ipp_pus0	ipp_pus1	ipp_hys	ipp_ode	ipp_dse1	ipp_dse0	ipp_sre
Field bits	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0

Figure 4-88. SW_PAD_CTL

Table 4-10. SW_PAD_CTL Bit Descriptions

Register Bit	Bit Name	Description
SW_PAD_CTL[31:30]	—	Unused
SW_PAD_CTL[29] SW_PAD_CTL[19] SW_PAD_CTL[9]	Loopback	Used to loop back the logic level at the BGA ball to a module input. Used when programming an output value to verify the value is achieved at I/O 0 Disable 1 Enable
SW_PAD_CTL[28:27] SW_PAD_CTL[18:17] SW_PAD_CTL[8:7]	ipp_pke ipp_pue	Pull-up, pull-down, and keeper 00 Disable pull-up/down, and keeper 01 Disable pull-up/down, and keeper 10 Enable keeper 11 Enable pull-up or pull-down
SW_PAD_CTL[26:25] SW_PAD_CTL[16:15] SW_PAD_CTL[6:5]	ipp_pus0 ipp_pus1	Size of pull resistor and up/down control 00 100 kΩ pull-down 01 100 kΩ pull-up 10 47 kΩ pull-up (Not used in i.MX31 and i.MX31L.) 11 22 kΩ pull-up (Not used in i.MX31 and i.MX31L.)
SW_PAD_CTL[24] SW_PAD_CTL[14] SW_PAD_CTL[4]	ipp_hys	Hysteresis control 0 Standard input 1 Input with hysteresis, Schmitt trigger engaged
SW_PAD_CTL[23] SW_PAD_CTL[13] SW_PAD_CTL[3]	ipp_ode	Open-drain control 0 Standard CMOS output, push-pull 1 Output is open-drain (requires pull-up)
SW_PAD_CTL[22:21] SW_PAD_CTL[12:11] SW_PAD_CTL[2:1]	ipp_dse1 ipp_dse0	Output drive strength 00 Standard (std) 01 High 10 Max 11 Max
SW_PAD_CTL[20] SW_PAD_CTL[10] SW_PAD_CTL[0]	ipp_sre	Slew rate control 0 Slow 1 Fast

4.3.8 Register Descriptions for SW Pad Control (SW_PAD_CTL)

Figure 4-89 through Figure 4-198 show the sw_pad_ctl registers. The I/O settings shown in Table 4-12 enables the user to select the characteristics of each I/O line by configuring the appropriate SW_PAD_CTL registers.

Absolute: 0x43FA_C154 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_ttm_pad										Reserved			
W																
Reset	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved						Reserved									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-89. Register Description sw_pad_ctl_ttm_pad__x__x

Absolute: 0x43FA_C158 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_cspi3_miso										sw_pad_ctl_cspi3_sclk			
W																
Reset	0	0	0	1	1	0	1	0	0	0	0	0	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_cspi3_sclk						sw_pad_ctl_cspi3_spi_rdy									
W																
Reset	1	1	0	0	0	0	0	1	1	0	1	0	0	0	0	0

Figure 4-90. Register Description sw_pad_ctl_cspi3_miso_cspi3_sclk_cspi3_spi_rdy

Absolute: 0x43FA_C15C Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_ce_control										sw_pad_ctl_clkss			
W																
Reset	0	0	0	0	1	0	0	0	0	0	0	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_clkss						sw_pad_ctl_cspi3_mosi									
W																
Reset	1	0	0	0	0	1	0	1	1	0	1	0	0	0	0	0

Figure 4-91. Register Description sw_pad_ctl_ce_control_clkss_cspi3_mosi

Absolute: 0x43FA_C160 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_ata_diow										sw_pad_ctl_ata_dmack			
W																
Reset	0	0	0	1	1	0	1	0	0	0	0	0	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_ata_dmack						sw_pad_ctl_ata_reset_b									
W																
Reset	1	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0

Figure 4-92. Register Description sw_pad_ctl_ata_diow_ata_dmack_ata_reset_b

Absolute: 0x43FA_C164 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_ata_cs0										sw_pad_ctl_ata_cs1			
W																
Reset	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_ata_cs1						sw_pad_ctl_ata_dior									
W																
Reset	1	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0

Figure 4-93. Register Description sw_pad_ctl_ata_cs0_ata_cs1_ata_dior

Absolute: 0x43FA_C168 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_sd1_data1										sw_pad_ctl_sd1_data2			
W																
Reset	0	0	0	0	1	0	1	0	0	0	0	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_sd1_data2						sw_pad_ctl_sd1_data3									
W																
Reset	1	0	0	0	0	1	0	0	1	0	0	0	0	0	0	1

Figure 4-94. Register Description sw_pad_ctl_sd1_data1_sd1_data2_sd1_data3

Absolute: 0x43FA_C16C Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_sd1_cmd										sw_pad_ctl_sd1_clk			
W																
Reset	0	0	0	0	1	0	1	0	0	0	0	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_sd1_clk						sw_pad_ctl_sd1_data0									
W																
Reset	1	0	0	0	0	1	0	0	1	0	1	0	0	0	0	1

Figure 4-95. Register Description sw_pad_ctl_sd1_cmd_sd1_clk_sd1_data0

Absolute: 0x43FA_C170 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_d3_rev										sw_pad_ctl_d3_cls			
W																
Reset	0	0	0	0	1	0	1	0	0	0	1	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_d3_cls						sw_pad_ctl_d3_spl									
W																
Reset	1	0	0	0	1	1	0	0	1	0	1	0	0	0	1	1

Figure 4-96. Register Description sw_pad_ctl_d3_rev_d3_cls_d3_spl

Absolute: 0x43FA_C174 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_read										sw_pad_ctl_vsync3			
W																
Reset	0	0	0	0	1	0	1	0	0	0	1	1	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_vsync3						sw_pad_ctl_contrast									
W																
Reset	1	0	0	0	1	1	0	0	1	0	1	0	0	0	1	1

Figure 4-97. Register Description sw_pad_ctl_read_vsync3_contrast

Absolute: 0x43FA_C178 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_ser_rs										sw_pad_ctl_par_rs			
W																
Reset	0	0	0	0	1	0	1	0	0	0	1	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_par_rs						sw_pad_ctl_write									
W																
Reset	1	0	0	0	1	1	0	0	1	0	1	0	0	0	1	1

Figure 4-98. Register Description sw_pad_ctl_ser_rs_par_rs_write

Absolute: 0x43FA_C17C Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_sd_d_clk										sw_pad_ctl_lcs0			
W																
Reset	0	0	0	0	1	0	1	0	0	0	1	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_lcs0						sw_pad_ctl_lcs1									
W																
Reset	1	0	0	0	1	1	0	0	1	0	1	0	0	0	1	1

Figure 4-99. Register Description sw_pad_ctl_sd_d_clk_lcs0_lcs1

Absolute: 0x43FA_C180 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_drdy0										sw_pad_ctl_sd_d_i			
W																
Reset	0	0	0	0	1	0	1	0	0	0	1	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_sd_d_i						sw_pad_ctl_sd_d_io									
W																
Reset	1	0	0	0	1	1	0	0	1	0	1	0	0	0	1	1

Figure 4-100. Register Description sw_pad_ctl_drdy0_sd_d_i_sd_d_io

Absolute: 0x43FA_C184 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_vsync0										sw_pad_ctl_hsync			
W																
Reset	0	0	0	1	1	0	1	0	0	0	1	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_hsync						sw_pad_ctl_fpshift									
W																
Reset	1	0	0	0	1	1	0	0	1	0	1	0	0	0	1	1

Figure 4-101. Register Description sw_pad_ctl_vsync0_hsync_fpshift

Absolute: 0x43FA_C188 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_ld15										sw_pad_ctl_ld16			
W																
Reset	0	0	0	1	1	0	1	0	0	0	1	1	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_ld16						sw_pad_ctl_ld17									
W																
Reset	1	0	0	0	1	1	0	1	1	0	1	0	0	0	1	1

Figure 4-102. Register Description sw_pad_ctl_ld15_ld16_ld17

Absolute: 0x43FA_C18C Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_ld12										sw_pad_ctl_ld13			
W																
Reset	0	0	0	1	1	0	1	0	0	0	1	1	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_ld13						sw_pad_ctl_ld14									
W																
Reset	1	0	0	0	1	1	0	1	1	0	1	0	0	0	1	1

Figure 4-103. Register Description sw_pad_ctl_ld12_ld13_ld14

Absolute: 0x43FA_C190 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_ld9										sw_pad_ctl_ld10			
W																
Reset	0	0	0	1	1	0	1	0	0	0	1	1	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_ld10						sw_pad_ctl_ld11									
W																
Reset	1	0	0	0	1	1	0	1	1	0	1	0	0	0	1	1

Figure 4-104. Register Description sw_pad_ctl_ld9_ld10_ld11

Absolute: 0x43FA_C194 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_ld6										sw_pad_ctl_ld7			
W																
Reset	0	0	0	1	1	0	1	0	0	0	1	1	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_ld7						sw_pad_ctl_ld8									
W																
Reset	1	0	0	0	1	1	0	1	1	0	1	0	0	0	1	1

Figure 4-105. Register Description sw_pad_ctl_ld6_ld7_ld8

Absolute: 0x43FA_C198 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_ld3										sw_pad_ctl_ld4			
W																
Reset	0	0	0	1	1	0	1	0	0	0	1	1	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_ld4						sw_pad_ctl_ld5									
W																
Reset	1	0	0	0	1	1	0	1	1	0	1	0	0	0	1	1

Figure 4-106. Register Description sw_pad_ctl_ld3_ld4_ld5

Absolute: 0x43FA_C19C Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_ld0										sw_pad_ctl_ld1			
W																
Reset	0	0	0	1	1	0	1	0	0	0	1	1	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_ld1						sw_pad_ctl_ld2									
W																
Reset	1	0	0	0	1	1	0	1	1	0	1	0	0	0	1	1

Figure 4-107. Register Description sw_pad_ctl_ld0_ld1_ld2

Absolute: 0x43FA_C1A0 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_usbh2_nxt										sw_pad_ctl_usbh2_data0			
W																
Reset	0	0	0	1	1	0	1	0	0	0	0	0	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_usbh2_data0						sw_pad_ctl_usbh2_data1									
W																
Reset	1	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0

Figure 4-108. Register Description sw_pad_ctl_usbh2_nxt_usbh2_data0_usbh2_data1

Absolute: 0x43FA_C1A4 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_usbh2_clk										sw_pad_ctl_usbh2_dir			
W																
Reset	0	0	0	1	1	0	1	1	0	0	0	0	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_usbh2_dir						sw_pad_ctl_usbh2_stp									
W																
Reset	1	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0

Figure 4-109. Register Description sw_pad_ctl_usbh2_clk_usbh2_dir_usbh2_stp

Absolute: 0x43FA_C1A8 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_usbotg_data5										sw_pad_ctl_usbotg_data6			
W																
Reset	0	0	0	1	1	0	1	0	0	0	0	0	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_usbotg_data6						sw_pad_ctl_usbotg_data7									
W																
Reset	1	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0

Figure 4-110. Register Description sw_pad_ctl_usbotg_data5_usbotg_data6_usbotg_data7

Absolute: 0x43FA_C1AC Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_usbotg_data2										sw_pad_ctl_usbotg_data3			
W																
Reset	0	0	0	1	1	0	1	0	0	0	0	0	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_usbotg_data3						sw_pad_ctl_usbotg_data4									
W																
Reset	1	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0

Figure 4-111. Register Description sw_pad_ctl_usbotg_data2_usbotg_data3_usbotg_data4

Absolute: 0x43FA_C1B0 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_usbotg_nxt										sw_pad_ctl_usbotg_data0			
W																
Reset	0	0	0	1	1	0	1	0	0	0	0	0	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_usbotg_data0						sw_pad_ctl_usbotg_data1									
W																
Reset	1	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0

Figure 4-112. Register Description sw_pad_ctl_usbotg_nxt_usbotg_data0_usbotg_data1

Absolute: 0x43FA_C1B4 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_usbotg_clk										sw_pad_ctl_usbotg_dir			
W																
Reset	0	0	0	1	1	0	1	1	0	0	0	0	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_usbotg_dir						sw_pad_ctl_usbotg_stp									
W																
Reset	1	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0

Figure 4-113. Register Description sw_pad_ctl_usbotg_clk_usbotg_dir_usbotg_stp

Absolute: 0x43FA_C1B8 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_usb_pwr										sw_pad_ctl_usb_oc			
W																
Reset	0	0	0	1	1	0	1	0	0	0	0	0	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_usb_oc						sw_pad_ctl_usb_byp									
W																
Reset	1	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0

Figure 4-114. Register Description sw_pad_ctl_usb_pwr_usb_oc_usb_byp

Absolute: 0x43FA_C1BC Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_trstb										sw_pad_ctl_de_b			
W																
Reset	0	0	0	1	1	0	1	0	0	0	0	0	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_de_b						sw_pad_ctl_sjc_mod									
W																
Reset	1	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0

Figure 4-115. Register Description sw_pad_ctl_trstb_de_b_sjc_mod

Absolute: 0x43FA_C1C0 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_tms										sw_pad_ctl_tdi			
W																
Reset	0	0	0	1	1	0	1	1	0	0	0	0	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_tdi						sw_pad_ctl_tdo									
W																
Reset	1	1	0	0	0	0	0	0	1	0	1	0	0	0	1	1

Figure 4-116. Register Description sw_pad_ctl_tms_tdi_tdo

Absolute: 0x43FA_C1C4 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_key_col7										sw_pad_ctl_rtck			
W																
Reset	0	0	0	1	1	0	1	0	0	0	0	0	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_rtck						sw_pad_ctl_tck									
W																
Reset	0	0	0	0	1	1	0	1	1	0	0	1	0	0	0	0

Figure 4-117. Register Description sw_pad_ctl_key_col7_rtck_tck

Absolute: 0x43FA_C1C8 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_key_col4										sw_pad_ctl_key_col5			
W																
Reset	0	0	0	1	1	0	1	0	0	0	0	0	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_key_col5						sw_pad_ctl_key_col6									
W																
Reset	1	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0

Figure 4-118. Register Description sw_pad_ctl_key_col4_key_col5_key_col6

Absolute: 0x43FA_C1CC Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	sw_pad_ctl_key_col1												sw_pad_ctl_key_col2			
W																
Reset	0	0	0	1	1	0	1	0	0	0	0	0	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_key_col2						sw_pad_ctl_key_col3									
W																
Reset	1	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0

Figure 4-119. Register Description sw_pad_ctl_key_col1_key_col2_key_col3

Absolute: 0x43FA_C1D0 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	sw_pad_ctl_key_row6												sw_pad_ctl_key_row7			
W																
Reset	0	0	0	1	1	0	1	0	0	0	0	0	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_key_row7						sw_pad_ctl_key_col0									
W																
Reset	1	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0

Figure 4-120. Register Description sw_pad_ctl_key_row6_key_row7_key_col0

Absolute: 0x43FA_C1D4 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	sw_pad_ctl_key_row3												sw_pad_ctl_key_row4			
W																
Reset	0	0	0	1	1	0	1	0	0	0	0	0	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_key_row4						sw_pad_ctl_key_row5									
W																
Reset	1	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0

Figure 4-121. Register Description sw_pad_ctl_key_row3_key_row4_key_row5

Absolute: 0x43FA_C1D8 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_key_row0										sw_pad_ctl_key_row1			
W																
Reset	0	0	0	1	1	0	1	0	0	0	0	0	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_key_row1						sw_pad_ctl_key_row2									
W																
Reset	1	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0

Figure 4-122. Register Description sw_pad_ctl_key_row0_key_row1_key_row2

Absolute: 0x43FA_C1DC Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_rts2										sw_pad_ctl_cts2			
W																
Reset	0	0	0	1	1	0	1	0	0	0	0	0	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_cts2						sw_pad_ctl_batt_line									
W																
Reset	1	0	0	0	0	0	1	1	1	0	1	0	1	0	0	0

Figure 4-123. Register Description sw_pad_ctl_rts2_cts2_batt_line

Absolute: 0x43FA_C1E0 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_dtr_dce2										sw_pad_ctl_rxd2			
W																
Reset	0	0	0	1	1	0	1	0	0	0	0	0	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_rxd2						sw_pad_ctl_txd2									
W																
Reset	1	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0

Figure 4-124. Register Description sw_pad_ctl_dtr_dce2_rxd2_txd2

Absolute: 0x43FA_C1E4 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_dsr_dte1										sw_pad_ctl_ri_dte1			
W																
Reset	0	0	0	1	1	0	1	0	0	0	0	0	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_ri_dte1						sw_pad_ctl_dcd_dte1									
W																
Reset	1	1	0	0	0	0	0	1	1	0	1	0	0	0	0	0

Figure 4-125. Register Description sw_pad_ctl_dsr_dte1_ri_dte1_dcd_dte1

Absolute: 0x43FA_C1E8 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_ri_dce1										sw_pad_ctl_dcd_dce1			
W																
Reset	0	0	0	1	1	0	1	0	0	0	0	0	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_dcd_dce1						sw_pad_ctl_dtr_dte1									
W																
Reset	1	0	0	0	0	0	0	1	1	0	1	1	0	0	0	0

Figure 4-126. Register Description sw_pad_ctl_ri_dce1_dcd_dce1_dtr_dte1

Absolute: 0x43FA_C1EC Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_cts1										sw_pad_ctl_dtr_dce1			
W																
Reset	0	0	0	1	1	0	1	0	0	0	0	0	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_dtr_dce1						sw_pad_ctl_dsr_dce1									
W																
Reset	1	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0

Figure 4-127. Register Description sw_pad_ctl_cts1_dtr_dce1_dsr_dce1

Absolute: 0x43FA_C1F0 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_rxd1										sw_pad_ctl_txd1			
W																
Reset	0	0	0	1	1	0	1	0	0	0	0	0	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_txd1						sw_pad_ctl_rts1									
W																
Reset	1	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0

Figure 4-128. Register Description sw_pad_ctl_rxd1_txd1_rts1

Absolute: 0x43FA_C1F4 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_cspi2_ss2										sw_pad_ctl_cspi2_sclk			
W																
Reset	0	0	0	1	1	0	1	1	0	0	0	0	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_cspi2_sclk						sw_pad_ctl_cspi2_spi_rdy									
W																
Reset	1	1	0	0	0	0	0	1	1	0	1	0	0	0	0	0

Figure 4-129. Register Description sw_pad_ctl_cspi2_ss2_cspi2_sclk_cspi2_spi_rdy

Absolute: 0x43FA_C1F8 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_cspi2_miso										sw_pad_ctl_cspi2_ss0			
W																
Reset	0	0	0	1	1	0	1	1	0	0	0	0	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_cspi2_ss0						sw_pad_ctl_cspi2_ss1									
W																
Reset	1	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0

Figure 4-130. Register Description sw_pad_ctl_cspi2_miso_cspi2_ss0_cspi2_ss1

Absolute: 0x43FA_C1FC Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_cspi1_sclk										sw_pad_ctl_cspi1_spi_rdy			
W																
Reset	0	0	0	1	1	0	1	1	0	0	0	0	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_cspi1_spi_rdy						sw_pad_ctl_cspi2_mosi									
W																
Reset	1	0	0	0	0	0	0	1	1	0	1	1	0	0	0	0

Figure 4-131. Register Description sw_pad_ctl_cspi1_sclk_cspi1_spi_rdy_cspi2_mosi

Absolute: 0x43FA_C200 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_cspi1_ss0										sw_pad_ctl_cspi1_ss1			
W																
Reset	0	0	0	1	1	0	1	0	0	0	0	0	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_cspi1_ss1						sw_pad_ctl_cspi1_ss2									
W																
Reset	1	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0

Figure 4-132. Register Description sw_pad_ctl_cspi1_ss0_cspi1_ss1_cspi1_ss2

Absolute: 0x43FA_C204 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_sfs6										sw_pad_ctl_cspi1_mosi			
W																
Reset	0	0	0	1	1	0	1	0	0	0	0	0	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_cspi1_mosi						sw_pad_ctl_cspi1_miso									
W																
Reset	1	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0

Figure 4-133. Register Description sw_pad_ctl_sfs6_cspi1_mosi_cspi1_miso

Absolute: 0x43FA_C208 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_stxd6										sw_pad_ctl_srxd6			
W																
Reset	0	0	0	1	1	0	1	0	0	0	0	0	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_srxd6						sw_pad_ctl_sck6									
W																
Reset	1	0	0	0	0	0	0	1	1	0	1	1	0	0	0	0

Figure 4-134. Register Description sw_pad_ctl_stxd6_srxd6_sck6

Absolute: 0x43FA_C20C Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_srxd5										sw_pad_ctl_sck5			
W																
Reset	0	0	0	1	1	0	1	0	0	0	0	0	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_sck5						sw_pad_ctl_sfs5									
W																
Reset	1	1	0	0	0	0	0	1	1	0	1	0	0	0	0	0

Figure 4-135. Register Description sw_pad_ctl_srxd5_sck5_sfs5

Absolute: 0x43FA_C210 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_sck4										sw_pad_ctl_sfs4			
W																
Reset	0	0	0	1	1	0	1	1	0	0	0	0	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_sfs4						sw_pad_ctl_stxd5									
W																
Reset	1	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0

Figure 4-136. Register Description sw_pad_ctl_sck4_sfs4_stxd5

Absolute: 0x43FA_C214 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_sfs3										sw_pad_ctl_stxd4			
W																
Reset	0	0	0	1	1	0	1	0	0	0	0	0	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_stxd4						sw_pad_ctl_srxd4									
W																
Reset	1	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0

Figure 4-137. Register Description sw_pad_ctl_sfs3_stxd4_srxd4

Absolute: 0x43FA_C218 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_stxd3										sw_pad_ctl_srxd3			
W																
Reset	0	0	0	1	1	0	1	0	0	0	0	0	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_srxd3						sw_pad_ctl_sck3									
W																
Reset	1	0	0	0	0	0	0	1	1	0	1	1	0	0	0	0

Figure 4-138. Register Description sw_pad_ctl_stxd3_srxd3_sck3

Absolute: 0x43FA_C21C Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_csi_pixclk										sw_pad_ctl_i2c_clk			
W																
Reset	0	0	0	1	0	0	1	1	0	0	1	1	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_i2c_clk						sw_pad_ctl_i2c_dat									
W																
Reset	1	1	0	0	0	0	0	1	1	0	1	1	0	0	0	0

Figure 4-139. Register Description sw_pad_ctl_csi_pixclk_i2c_clk_i2c_dat

Absolute: 0x43FA_C220 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_csi_mclk										sw_pad_ctl_csi_vsync			
W																
Reset	0	0	0	1	0	0	1	0	0	0	1	1	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_csi_vsync						sw_pad_ctl_csi_hsync									
W																
Reset	1	0	0	0	1	1	0	1	0	0	1	0	0	0	1	1

Figure 4-140. Register Description sw_pad_ctl_csi_mclk_csi_vsync_csi_hsync

Absolute: 0x43FA_C224 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_csi_d13										sw_pad_ctl_csi_d14			
W																
Reset	0	0	0	1	0	0	1	0	0	0	1	1	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_csi_d14						sw_pad_ctl_csi_d15									
W																
Reset	1	0	0	0	1	1	0	1	0	0	1	0	0	0	1	1

Figure 4-141. Register Description sw_pad_ctl_csi_d13_csi_d14_csi_d15

Absolute: 0x43FA_C228 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_csi_d10										sw_pad_ctl_csi_d11			
W																
Reset	0	0	0	1	0	0	1	0	0	0	1	1	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_csi_d11						sw_pad_ctl_csi_d12									
W																
Reset	1	0	0	0	1	1	0	1	0	0	1	0	0	0	1	1

Figure 4-142. Register Description sw_pad_ctl_csi_d10_csi_d11_csi_d12

Absolute: 0x43FA_C22C Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_csi_d7										sw_pad_ctl_csi_d8			
W																
Reset	0	0	0	1	0	0	1	0	0	0	1	1	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_csi_d8						sw_pad_ctl_csi_d9									
W																
Reset	1	0	0	0	1	1	0	1	0	0	1	0	0	0	1	1

Figure 4-143. Register Description sw_pad_ctl_csi_d7_csi_d8_csi_d9

Absolute: 0x43FA_C230 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_csi_d4										sw_pad_ctl_csi_d5			
W																
Reset	0	0	0	1	0	0	1	0	0	0	1	1	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_csi_d5						sw_pad_ctl_csi_d6									
W																
Reset	1	0	0	0	1	1	0	1	0	0	1	0	0	0	1	1

Figure 4-144. Register Description sw_pad_ctl_csi_d4_csi_d5_csi_d6

Absolute: 0x43FA_C234 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_pc_poe										sw_pad_ctl_m_request			
W																
Reset	0	0	0	0	1	0	1	0	0	0	1	0	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_m_request						sw_pad_ctl_m_grant									
W																
Reset	1	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0

Figure 4-145. Register Description sw_pad_ctl_pc_poe_m_request_m_grant

Absolute: 0x43FA_C238 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_pc_rst										sw_pad_ctl_iois16			
W																
Reset	0	0	0	1	1	0	1	0	0	0	1	0	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_iois16						sw_pad_ctl_pc_rw_b									
W																
Reset	1	0	0	0	1	0	0	1	1	0	1	0	0	0	1	0

Figure 4-146. Register Description sw_pad_ctl_pc_rst_iois16_pc_rw_b

Absolute: 0x43FA_C23C Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_pc_vs2										sw_pad_ctl_pc_bvd1			
W																
Reset	0	0	0	1	1	0	1	0	0	0	1	0	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_pc_bvd1						sw_pad_ctl_pc_bvd2									
W																
Reset	1	0	0	0	1	0	0	1	1	0	1	0	0	0	1	0

Figure 4-147. Register Description sw_pad_ctl_pc_vs2_pc_bvd1_pc_bvd2

Absolute: 0x43FA_C240 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_pc_ready										sw_pad_ctl_pc_pwron			
W																
Reset	0	0	0	1	1	0	1	0	0	0	1	0	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_pc_pwron						sw_pad_ctl_pc_vs1									
W																
Reset	0	0	0	0	1	0	0	1	1	0	1	0	0	0	1	0

Figure 4-148. Register Description sw_pad_ctl_pc_ready_pc_pwron_pc_vs1

Absolute: 0x43FA_C244 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_pc_cd1_b										sw_pad_ctl_pc_cd2_b			
W																
Reset	0	0	0	1	1	0	1	0	0	0	1	0	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_pc_cd2_b						sw_pad_ctl_pc_wait_b									
W																
Reset	1	0	0	0	1	0	0	1	1	0	1	0	0	0	1	0

Figure 4-149. Register Description sw_pad_ctl_pc_cd1_b_pc_cd2_b_pc_wait_b

Absolute: 0x43FA_C248 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_d2										sw_pad_ctl_d1			
W																
Reset	0	0	0	1	0	0	1	0	0	1	1	1	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_d1						sw_pad_ctl_d0									
W																
Reset	1	0	0	1	1	1	0	1	0	0	1	0	0	1	1	1

Figure 4-150. Register Description sw_pad_ctl_d2_d1_d0

Absolute: 0x43FA_C24C Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_d5										sw_pad_ctl_d4			
W																
Reset	0	0	0	1	0	0	1	0	0	1	1	1	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_d4						sw_pad_ctl_d3									
W																
Reset	1	0	0	1	1	1	0	1	0	0	1	0	0	1	1	1

Figure 4-151. Register Description sw_pad_ctl_d5_d4_d3

Absolute: 0x43FA_C250 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_d8										sw_pad_ctl_d7			
W																
Reset	0	0	0	1	0	0	1	0	0	1	1	1	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_d7						sw_pad_ctl_d6									
W																
Reset	1	0	0	1	1	1	0	1	0	0	1	0	0	1	1	1

Figure 4-152. Register Description sw_pad_ctl_d8_d7_d6

Absolute: 0x43FA_C254 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_d11										sw_pad_ctl_d10			
W																
Reset	0	0	0	1	0	0	1	0	0	1	1	1	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_d10						sw_pad_ctl_d9									
W																
Reset	1	0	0	1	1	1	0	1	0	0	1	0	0	1	1	1

Figure 4-153. Register Description sw_pad_ctl_d11_d10_d9

Absolute: 0x43FA_C258 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_d14										sw_pad_ctl_d13			
W																
Reset	0	0	0	1	0	0	1	0	0	1	1	1	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_d13						sw_pad_ctl_d12									
W																
Reset	1	0	0	1	1	1	0	1	0	0	1	0	0	1	1	1

Figure 4-154. Register Description sw_pad_ctl_d14_d13_d12

Absolute: 0x43FA_C25C Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_nfce_b										sw_pad_ctl_nfrb			
W																
Reset	0	0	0	1	1	0	1	0	0	0	1	1	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_nfrb						sw_pad_ctl_d15									
W																
Reset	1	0	0	0	1	1	0	1	0	0	1	0	0	1	1	1

Figure 4-155. Register Description sw_pad_ctl_nfce_b_nfrb_d15

Absolute: 0x43FA_C260 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_nfale										sw_pad_ctl_nfcle			
W																
Reset	0	0	0	1	1	0	1	0	0	0	1	1	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_nfcle						sw_pad_ctl_nfwp_b									
W																
Reset	1	0	0	0	1	1	0	1	1	0	1	0	0	0	1	1

Figure 4-156. Register Description sw_pad_ctl_nfale_nfcle_nfwp_b

Absolute: 0x43FA_C264 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_sdqs3										sw_pad_ctl_nfwe_b			
W																
Reset	0	0	0	1	1	0	0	0	0	0	0	1	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_nfwe_b						sw_pad_ctl_nfre_b									
W																
Reset	0	0	0	0	1	1	0	1	1	0	1	0	0	0	1	1

Figure 4-157. Register Description sw_pad_ctl_sdqs3_nfwe_b_nfre_b

Absolute: 0x43FA_C268 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_sdqs0										sw_pad_ctl_sdqs1			
W																
Reset	0	0	0	1	1	0	0	0	0	0	0	1	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_sdqs1						sw_pad_ctl_sdqs2									
W																
Reset	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	1

Figure 4-158. Register Description sw_pad_ctl_sdqs0_sdqs1_sdqs2

Absolute: 0x43FA_C26C Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_sdcke1										sw_pad_ctl_sdclk*			
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	1	1	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_sdclk*						Reserved									
W																
Reset	1	0	0	1	1	1	0	1	1	0	1	0	0	0	0	0

*Bits 10–19 control the differential output pair SDCLK and $\overline{\text{SDCLK}}$.

Figure 4-159. Register Description sw_pad_ctl_sdcke1_sdclk_sdclk_b

Absolute: 0x43FA_C270 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_cas										sw_pad_ctl_sdwe			
W																
Reset	0	0	0	1	0	0	1	0	0	1	1	1	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_sdwe						sw_pad_ctl_sdcke0									
W																
Reset	1	0	0	0	0	1	0	1	0	0	1	0	0	0	0	1

Figure 4-160. Register Description sw_pad_ctl_cas_sdwe_sdcke0

Absolute: 0x43FA_C274 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_bclk										sw_pad_ctl_rw			
W																
Reset	0	0	1	0	1	0	1	0	0	0	1	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_rw						sw_pad_ctl_ras									
W																
Reset	1	0	0	0	1	1	0	1	0	0	1	0	0	1	1	1

Figure 4-161. Register Description sw_pad_ctl_bclk_rw_ras

Absolute: 0x43FA_C278 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_cs5										sw_pad_ctl_ecb			
W																
Reset	0	0	0	0	1	0	1	0	0	0	1	1	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_ecb						sw_pad_ctl_lba									
W																
Reset	1	0	0	0	1	1	0	0	1	0	1	0	0	0	1	1

Figure 4-162. Register Description sw_pad_ctl_cs5_ecb_lba

Absolute: 0x43FA_C27C Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_cs2										sw_pad_ctl_cs3			
W																
Reset	0	0	0	1	0	0	1	0	0	1	1	1	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_cs3						sw_pad_ctl_cs4									
W																
Reset	1	0	0	1	1	1	0	0	1	0	1	0	0	0	1	1

Figure 4-163. Register Description sw_pad_ctl_cs2_cs3_cs4

Absolute: 0x43FA_C280 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_oe										sw_pad_ctl_cs0			
W																
Reset	0	0	0	0	1	0	1	0	0	0	1	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_cs0						sw_pad_ctl_cs1									
W																
Reset	1	0	0	0	1	1	0	0	1	0	1	0	0	0	1	1

Figure 4-164. Register Description sw_pad_ctl_oe_cs0_cs1

Absolute: 0x43FA_C284 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_dqm3										sw_pad_ctl_eb0			
W																
Reset	0	0	0	1	0	0	1	0	0	1	1	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_eb0						sw_pad_ctl_eb1									
W																
Reset	1	0	0	0	1	1	0	0	1	0	1	0	0	0	1	1

Figure 4-165. Register Description sw_pad_ctl_dqm3_eb0_eb1

Absolute: 0x43FA_C288 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_dqm0										sw_pad_ctl_dqm1			
W																
Reset	0	0	0	1	0	0	1	0	0	1	1	1	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_dqm1						sw_pad_ctl_dqm2									
W																
Reset	1	0	0	1	1	1	0	1	0	0	1	0	0	1	1	1

Figure 4-166. Register description sw_pad_ctl_dqm0_dqm1_dqm2

Absolute: 0x43FA_C28C Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_sd29										sw_pad_ctl_sd30			
W																
Reset	0	0	0	1	0	0	1	0	0	1	1	1	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_sd30						sw_pad_ctl_sd31									
W																
Reset	1	0	0	1	1	1	0	1	0	0	1	0	0	1	1	1

Figure 4-167. Register Description sw_pad_ctl_sd29_sd30_sd31

Absolute: 0x43FA_C290 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_sd26										sw_pad_ctl_sd27			
W																
Reset	0	0	0	1	0	0	1	0	0	1	1	1	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_sd27						sw_pad_ctl_sd28									
W																
Reset	1	0	0	1	1	1	0	1	0	0	1	0	0	1	1	1

Figure 4-168. Register Description sw_pad_ctl_sd26_sd27_sd28

Absolute: 0x43FA_C294 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_sd23										sw_pad_ctl_sd24			
W																
Reset	0	0	0	1	0	0	1	0	0	1	1	1	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_sd24						sw_pad_ctl_sd25									
W																
Reset	1	0	0	1	1	1	0	1	0	0	1	0	0	1	1	1

Figure 4-169. Register Description sw_pad_ctl_sd23_sd24_sd25

Absolute: 0x43FA_C298 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_sd20										sw_pad_ctl_sd21			
W																
Reset	0	0	0	1	0	0	1	0	0	1	1	1	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_sd21						sw_pad_ctl_sd22									
W																
Reset	1	0	0	1	1	1	0	1	0	0	1	0	0	1	1	1

Figure 4-170. Register Description sw_pad_ctl_sd20_sd21_sd22

Absolute: 0x43FA_C29C Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_sd17										sw_pad_ctl_sd18			
W																
Reset	0	0	0	1	0	0	1	0	0	1	1	1	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_sd18						sw_pad_ctl_sd19									
W																
Reset	1	0	0	1	1	1	0	1	0	0	1	0	0	1	1	1

Figure 4-171. Register Description sw_pad_ctl_sd17_sd18_sd19

Absolute: 0x43FA_C2A0 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_sd14										sw_pad_ctl_sd15			
W																
Reset	0	0	0	1	0	0	1	0	0	1	1	1	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_sd15						sw_pad_ctl_sd16									
W																
Reset	1	0	0	1	1	1	0	1	0	0	1	0	0	1	1	1

Figure 4-172. Register Description sw_pad_ctl_sd14_sd15_sd16

Absolute: 0x43FA_C2A4 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_sd11										sw_pad_ctl_sd12			
W																
Reset	0	0	0	1	0	0	1	0	0	1	1	1	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_sd12						sw_pad_ctl_sd13									
W																
Reset	1	0	0	1	1	1	0	1	0	0	1	0	0	1	1	1

Figure 4-173. Register Description sw_pad_ctl_sd11_sd12_sd13

Absolute: 0x43FA_C2A8 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_sd8										sw_pad_ctl_sd9			
W																
Reset	0	0	0	1	0	0	1	0	0	1	1	1	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_sd9						sw_pad_ctl_sd10									
W																
Reset	1	0	0	1	1	1	0	1	0	0	1	0	0	1	1	1

Figure 4-174. Register Description sw_pad_ctl_sd8_sd9_sd10

Absolute: 0x43FA_C2AC Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_sd5										sw_pad_ctl_sd6			
W																
Reset	0	0	0	1	0	0	1	0	0	1	1	1	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_sd6						sw_pad_ctl_sd7									
W																
Reset	1	0	0	1	1	1	0	1	0	0	1	0	0	1	1	1

Figure 4-175. Register Description sw_pad_ctl_sd5_sd6_sd7

Absolute: 0x43FA_C2B0 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_sd2										sw_pad_ctl_sd3			
W																
Reset	0	0	0	1	0	0	1	0	0	1	1	1	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_sd3						sw_pad_ctl_sd4									
W																
Reset	1	0	0	1	1	1	0	1	0	0	1	0	0	1	1	1

Figure 4-176. Register Description sw_pad_ctl_sd2_sd3_sd4

Absolute: 0x43FA_C2B4 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_sdba0										sw_pad_ctl_sd0			
W																
Reset	0	0	0	0	1	0	1	0	0	0	0	1	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_sd0						sw_pad_ctl_sd1									
W																
Reset	1	0	0	1	1	1	0	1	0	0	1	0	0	1	1	1

Figure 4-177. Register Description sw_pad_ctl_sdba0_sd0_sd1

Absolute: 0x43FA_C2B8 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_a24										sw_pad_ctl_a25			
W																
Reset	0	0	0	0	1	0	1	0	0	1	1	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_a25						sw_pad_ctl_sdba1									
W																
Reset	1	0	0	1	1	1	0	0	1	0	1	0	0	0	0	1

Figure 4-178. Register Description sw_pad_ctl_a24_a25_sdba1

Absolute: 0x43FA_C2BC Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_a21										sw_pad_ctl_a22			
W																
Reset	0	0	0	0	1	0	1	0	0	1	1	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_a22						sw_pad_ctl_a23									
W																
Reset	1	0	0	1	1	1	0	0	1	0	1	0	0	1	1	1

Figure 4-179. Register Description sw_pad_ctl_a21_a22_a23

Absolute: 0x43FA_C2C0 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_a18										sw_pad_ctl_a19			
W																
Reset	0	0	0	0	1	0	1	0	0	1	1	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_a19						sw_pad_ctl_a20									
W																
Reset	1	0	0	1	1	1	0	0	1	0	1	0	0	1	1	1

Figure 4-180. Register Description sw_pad_ctl_a18_a19_a20

Absolute: 0x43FA_C2C4 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_a15										sw_pad_ctl_a16			
W																
Reset	0	0	0	0	1	0	1	0	0	1	1	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_a16						sw_pad_ctl_a17									
W																
Reset	1	0	0	1	1	1	0	0	1	0	1	0	0	1	1	1

Figure 4-181. Register Description sw_pad_ctl_a15_a16_a17

Absolute: 0x43FA_C2C8 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_a12										sw_pad_ctl_a13			
W																
Reset	0	0	0	0	1	0	1	0	0	1	1	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_a13						sw_pad_ctl_a14									
W																
Reset	1	0	0	1	1	1	0	0	1	0	1	0	0	1	1	1

Figure 4-182. Register Description sw_pad_ctl_a12_a13_a14

Absolute: 0x43FA_C2CC Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_a10										sw_pad_ctl_ma10			
W																
Reset	0	0	0	0	1	0	1	0	0	1	1	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_ma10						sw_pad_ctl_a11									
W																
Reset	1	0	0	1	1	1	0	0	1	0	1	0	0	1	1	1

Figure 4-183. Register Description sw_pad_ctl_a10_ma10_a11

Absolute: 0x43FA_C2D0 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_a7										sw_pad_ctl_a8			
W																
Reset	0	0	0	0	1	0	1	0	0	1	1	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_a8						sw_pad_ctl_a9									
W																
Reset	1	0	0	1	1	1	0	0	1	0	1	0	0	1	1	1

Figure 4-184. Register Description sw_pad_ctl_a7_a8_a9

Absolute: 0x43FA_C2D4 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_a4										sw_pad_ctl_a5			
W																
Reset	0	0	0	0	1	0	1	0	0	1	1	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_a5						sw_pad_ctl_a6									
W																
Reset	1	0	0	1	1	1	0	0	1	0	1	0	0	1	1	1

Figure 4-185. Register Description sw_pad_ctl_a4_a5_a6

Absolute: 0x43FA_C2D8 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_a1										sw_pad_ctl_a2			
W																
Reset	0	0	0	0	1	0	1	0	0	1	1	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_a2						sw_pad_ctl_a3									
W																
Reset	1	0	0	1	1	1	0	0	1	0	1	0	0	1	1	1

Figure 4-186. Register Description sw_pad_ctl_a1_a2_a3

Absolute: 0x43FA_C2DC Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_vpg0										sw_pad_ctl_vpg1			
W																
Reset	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_vpg1						sw_pad_ctl_a0									
W																
Reset	1	0	0	0	0	0	0	0	1	0	1	0	0	1	1	1

Figure 4-187. Register Description sw_pad_ctl_vpg0_vpg1_a0

Absolute: 0x43FA_C2E0 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_vstby										sw_pad_ctl_dvfs0			
W																
Reset	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_dvfs0						sw_pad_ctl_dvfs1									
W																
Reset	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Figure 4-188. Register Description sw_pad_ctl_vstby_dvfs0_dvfs1

Absolute: 0x43FA_C2E4 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_boot_mode4										sw_pad_ctl_ckil			
W																
Reset	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_ckil						sw_pad_ctl_power_fail									
W																
Reset	1	1	0	0	0	0	0	1	1	0	0	0	0	0	0	0

Figure 4-189. Register Description sw_pad_ctl_boot_mode4_ckil_power_fail

Absolute: 0x43FA_C2E8 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_boot_mode1										sw_pad_ctl_boot_mode2			
W																
Reset	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_boot_mode2						sw_pad_ctl_boot_mode3									
W																
Reset	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Figure 4-190. Register Description sw_pad_ctl_boot_mode1_boot_mode2_boot_mode3

Absolute: 0x43FA_C2EC Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_por_b										sw_pad_ctl_clko			
W																
Reset	0	0	0	1	1	0	1	1	0	0	0	0	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_clko						sw_pad_ctl_boot_mode0									
W																
Reset	1	0	0	1	1	1	0	0	1	0	1	0	0	0	0	0

Figure 4-191. Register Description sw_pad_ctl_por_b_clko_boot_mode0

Absolute: 0x43FA_C2F0 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_simpd0										sw_pad_ctl_ckih			
W																
Reset	0	0	0	1	1	0	1	0	0	0	0	0	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_ckih						sw_pad_ctl_reset_in_b									
W																
Reset	1	1	0	0	0	0	0	1	1	0	1	1	0	0	0	0

Figure 4-192. Register Description sw_pad_ctl_simpd0_ckih_reset_in_b

Absolute: 0x43FA_C2F4 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_sven0										sw_pad_ctl_stx0			
W																
Reset	0	0	0	1	1	0	1	0	0	0	0	0	1	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_stx0						sw_pad_ctl_srx0									
W																
Reset	1	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0

Figure 4-193. Register Description sw_pad_ctl_sven0_stx0_srx0

Absolute: 0x43FA_C2F8 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_gpio3_1										sw_pad_ctl_sclk0			
W																
Reset	0	0	0	1	1	0	1	0	0	0	0	0	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_sclk0						sw_pad_ctl_srst0									
W																
Reset	1	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0

Figure 4-194. Register Description sw_pad_ctl_gpio3_1_sclk0_srst0

Absolute: 0x43FA_C2FC Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_gpio1_5										sw_pad_ctl_gpio1_6			
W																
Reset	0	0	0	1	1	0	1	0	0	0	0	0	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_gpio1_6						sw_pad_ctl_gpio3_0									
W																
Reset	1	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0

Figure 4-195. Register Description sw_pad_ctl_gpio1_5_gpio1_6_gpio3_0

Absolute: 0x43FA_C300 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_gpio1_2										sw_pad_ctl_gpio1_3			
W																
Reset	0	0	0	1	1	0	1	0	0	0	0	0	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_gpio1_3						sw_pad_ctl_gpio1_4									
W																
Reset	1	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0

Figure 4-196. Register Description sw_pad_ctl_gpio1_2_gpio1_3_gpio1_4

Absolute: 0x43FA_C304 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_pwmo										sw_pad_ctl_gpio1_0			
W																
Reset	0	0	0	0	1	0	1	1	0	0	0	0	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_gpio1_0						sw_pad_ctl_gpio1_1									
W																
Reset	1	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0

Figure 4-197. Register Description sw_pad_ctl_pwmo_gpio1_0_gpio1_1

Absolute: 0x43FA_C308 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			sw_pad_ctl_capture										sw_pad_ctl_compare			
W																
Reset	0	0	0	1	1	0	1	0	0	0	0	0	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	sw_pad_ctl_compare						sw_pad_ctl_watchdog_rst									
W																
Reset	1	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0

Figure 4-198. Register Description sw_pad_ctl_capture_compare_watchdog_rst

4.4 I/O Settings and Signal Multiplexing Scheme

This section contains detailed information about the settings for each I/O line and the functional multiplexing scheme. Table 4-7 through Table 4-13 describe the pin settings and the functional multiplexing between I/O and signals.

4.4.1 I/O Settings

The settings for each I/O are described in Table 4-12. The table includes bit descriptions for all settings and whether they are software configurable.

4.4.1.1 Special I/O and Exceptions for SW_PAD_CTL

Table 4-11 lists the exceptions to the I/O settings shown in Table 4-12. These exception apply only to ICs that are Revision 1.2. For each of these signals, the I/O characteristics override the value shown in the corresponding SW_PAD_CTL register; thus, the SW_PAD_CTL register has no impact on these signals. In addition, the signal characteristics shown in Table 4-11 are hardwired and are not configurable.

NOTE

For ICs previous to version 1.2, the information in Table 4-11 should be ignored.

Table 4-11. I/O Setting Exceptions and Special Pad Descriptions (IC Rev. 1.2)

Signal Name	Register	Bit	Description	Hardwired Setting
SDQS3	sw_pad_ctl_sdqs3_nfwe_b_nfre	21	Register bit “dse0” shows value of 0, but is overridden per Hardwired Setting column.	Drive strength set to high drive; cannot be configured for nominal or max drive.
SDQS2	sw_pad_ctl_sdqs0_sdqs1_sdqs2	1	Register bit “dse0” shows value of 0, but is overridden per Hardwired Setting column.	Drive strength set to high drive; cannot be configured for nominal or max drive.
SDQS1	sw_pad_ctl_sdqs0_sdqs1_sdqs2	11	Register bit “dse0” shows value of 0, but is overridden per Hardwired Setting column.	Drive strength set to high drive; cannot be configured for nominal or max drive.
SDQS0	sw_pad_ctl_sdqs0_sdqs1_sdqs2	21	Register bit “dse0” shows value of 0, but is overridden per Hardwired Setting column.	Drive strength set to high drive; cannot be configured for nominal or max drive.
SDBA1	sw_pad_ctl_a24_a25_sdba1	1	Register bit “dse0” shows value of 0, but is overridden per Hardwired Setting column.	Drive strength set to high drive; cannot be configured for nominal or max drive.
SDBA0	sw_pad_ctl_sdba0_sd0_sd1	21	Register bit “dse0” shows value of 0, but is overridden per Hardwired Setting column.	Drive strength set to high drive; cannot be configured for nominal or max drive.
SDCKE1	sw_pad_ctl_sdcke1_sdclk_sdclk	21	Register bit “dse0” shows value of 0, but is overridden per Hardwired Setting column.	Drive strength set to high drive; cannot be configured for nominal or max drive.
SDCKE0	sw_pad_ctl_cas_sdwe_sdcke0	1	Register bit “dse0” shows value of 0, but is overridden per Hardwired Setting column.	Drive strength set to high drive; cannot be configured for nominal or max drive.
SDWE	sw_pad_ctl_cas_sdwe_sdcke0	11	Register bit “dse0” shows value of 0, but is overridden per Hardwired Setting column.	Drive strength set to high drive; cannot be configured for nominal or max drive.
PC_CD1_B	sw_pad_ctl_pc_cd1_b_pc_cd2_b_pc_wait	20	Register bit “sre” shows value of 0, but is overridden per Hardwired Setting column.	Slew rate set to fast, cannot be configured for slow slew rate.

Table 4-11. I/O Setting Exceptions and Special Pad Descriptions (IC Rev. 1.2) (continued)

Signal Name	Register	Bit	Description	Hardwired Setting
PC_CD2_B	sw_pad_ctl_pc_cd1_b_pc_cd2_b_pc_wait	10	Register bit “sre” shows value of 0, but is overridden per Hardwired Setting column.	Slew rate set to fast, cannot be configured for slow slew rate.
PC_WAIT_B	sw_pad_ctl_pc_cd1_b_pc_cd2_b_pc_wait	0	Register bit “sre” shows value of 0, but is overridden per Hardwired Setting column.	Slew rate set to fast, cannot be configured for slow slew rate.
PC_READY	sw_pad_ctl_pc_ready_pc_pwron_pc_vs1	20	Register bit “sre” shows value of 0, but is overridden per Hardwired Setting column.	Slew rate set to fast, cannot be configured for slow slew rate.
PC_PWRON	sw_pad_ctl_pc_ready_pc_pwron_pc_vs1	10	Register bit “sre” shows value of 0, but is overridden per Hardwired Setting column.	Slew rate set to fast, cannot be configured for slow slew rate.
PC_VS1	sw_pad_ctl_pc_ready_pc_pwron_pc_vs1	0	Register bit “sre” shows value of 0, but is overridden per Hardwired Setting column.	Slew rate set to fast, cannot be configured for slow slew rate.
PC_RST	sw_pad_ctl_pc_rst_iois16_pc_rw	20	Register bit “sre” shows value of 0, but is overridden per Hardwired Setting column.	Slew rate set to fast, cannot be configured for slow slew rate.
PC_RW_B	sw_pad_ctl_pc_rst_iois16_pc_rw	0	Register bit “sre” shows value of 0, but is overridden per Hardwired Setting column.	Slew rate set to fast, cannot be configured for slow slew rate.
PC_POE	sw_pad_ctl_pc_poe_m_request_m_grant	20	Register bit “sre” shows value of 0, but is overridden per Hardwired Setting column.	Slew rate set to fast, cannot be configured for slow slew rate.
COMPARE	sw_pad_ctl_capture_compare_watchdog_rst	11	Register bit “dse0” shows value of 0, but is overridden per Hardwired Setting column.	Drive strength set to high drive; cannot be configured for nominal or max drive.
CAPTURE	sw_pad_ctl_capture_compare_watchdog_rst	21	Register bit “dse0” shows value of 0, but is overridden per Hardwired Setting column.	Drive strength set to high drive; cannot be configured for nominal or max drive.

4.4.1.2 Table Headings

The Table 4-12 headings are defined as follows:

Package Contact Name

Lists the name assigned to the BGA ball. The ball map and signal listing are in the data sheet.

Supply — Specifies the power supply associated with the I/O signal.

Value After Reset — After the reset sequence is completed, specifies the direction of the signal and logic state.

I/O Type — Indicates the configuration of the output driver which determines the DC and AC characteristics. Data sheet DC Electrical Parameters table for GPIO is applicable when “regular” is listed. Data sheet DC Electrical Parameters table for DDR is applicable when DDR is listed. AC parameters are also in the data sheet.

Slew Rate — Indicates the output transition time capability of the output. If the signal is an input, ignore this column. The terms slow and fast are found in the data sheet DC Electrical Parameters and AC Electrical Characteristics tables.

Loopback — A connection that sends an output signal to the module’s input. The loopback feature allows the input buffer and output buffer to be enabled simultaneously. Note that most signal lines indicate No, which means there is no loopback capability.

Drive Strength — Output driver’s current source and sink capability. The terms “std”, “high”, and “max” are found in the data sheet DC Electrical Parameters table.*Pull Value and Direction*

The on-chip nominal resistor value that is potentially available at each I/O. For i.MX31 and i.MX31L, the only value available is 100 kohms. Direction is either pull-up or pull-down. Note that a “pull-up” or “pull-dn” may be listed, but may be overridden by the Pull/Keeper Control column indicating “disable” which means the pull resistor is not enabled.

Pull/Keeper Control — The termination capability selected. “Pull” means pull-up/down is selected; refer to “Pull Value and Direction” column for details. “Keeper” indicates that a keeper circuit is enabled which holds a logic value without the extra current drain required by a simple pull-up/down resistor.
“Disable” means there is no pull-up, pull-down, or keeper. In this case, the “Pull Value and Direction” column must be ignored.

Open Drain/Push-Pull — Specifies the output structure type. “PP” indicates push-pull which is a standard CMOS output with both active pull-up and pull-down MOSFETs. “OD” indicates open-drain which is an output with an active pull-down MOSFET but no active pull-up MOSFET.

Schmitt Trigger — Specifies whether an input has hysteresis or not. “No” indicates that an input is a standard CMOS input with no hysteresis. “Yes” indicates that a Schmitt-trigger circuit is engaged.

4.4.1.3 Table Subheadings

Table 4-12 subheadings are defined as follows:

Control Bit(s) — Lists the associated sw_pad_ctl register bit(s). A dash in this column indicates that the default value is not controllable by software and the default cannot be overridden.

Default — Indicates the default setting after the power-up sequence is completed.

Table 4-12. i.MX31 and i.MX31L I/O Settings

Package Contact Name	Value After Reset		Supply	I/O Type	Slew Rate		Loop back?		Drive Strength		Pull Value and Direction		Pull/Keeper Control		Open-Drain/ Push-Pull		Schmitt Trigger?	
	Direction	State			Control Bit	Default	Control Bit	Default	Control Bits	Default	Control Bits	Default	Control Bit	Default	Control Bit	Default	Control Bit	Default
Note: The following notes apply to all of the contacts described in this table: • This table represents IC revision 1.1 and 1.15. Exceptions for revision 1.2 are listed in Table 4-11. • IC Revision can be read on the SREV register. See IIM chapter for details. • Each unused input not specifically detailed in this table must be either (a) set up as a no connect with the associated on-chip pull-up or pull-down device enabled by software or (b) externally terminated to GND or associated positive supply directly or through a 1 kΩ resistor. • All unused outputs should be floated.																		
CAPTURE	input	pulled up	NVCC1	regular	sw_pad_ctl_capture[0]	slow	—	no	—	std	—	100k pull-up	sw_pad_ctl_capture[8]	pull	—	PP	—	no
COMPARE	input	pulled up	NVCC1	regular	sw_pad_ctl_compare[0]	slow	—	no	—	std	—	100k pull-up	—	pull	—	PP	—	no
WATCHDOG_RST	input	pulled up	NVCC1	regular	—	slow	—	no	—	std	—	100k pull-up	—	pull	—	PP	—	no
PWMO	input	floating	NVCC3	regular	sw_pad_ctl_pwmo[0]	slow	—	no	—	std	—	100k pull-up	sw_pad_ctl_pwmo[8]	disable	—	PP	sw_pad_ctl_pwmo[4]	yes
GPIO1_0	input	pulled up	NVCC1	regular	sw_pad_ctl_gpio1_0[0]	slow	—	no	sw_pad_ctl_gpio1_0[2:1]	std	—	100k pull-up	sw_pad_ctl_gpio1_0[8:7]	pull	sw_pad_ctl_gpio1_0[3]	PP	sw_pad_ctl_gpio1_0[4]	no
GPIO1_1	input	pulled up	NVCC1	regular	sw_pad_ctl_gpio1_1[0]	slow	—	no	sw_pad_ctl_gpio1_1[2:1]	std	—	100k pull-up	sw_pad_ctl_gpio1_1[8:7]	pull	sw_pad_ctl_gpio1_1[3]	PP	sw_pad_ctl_gpio1_1[4]	no

Table 4-12. i.MX31 and i.MX31L I/O Settings (continued)

Package Contact Name	Value After Reset		Supply	I/O Type	Slew Rate		Loop back?		Drive Strength		Pull Value and Direction		Pull/Keeper Control		Open-Drain/ Push-Pull		Schmitt Trigger?	
	Direction	State			Control Bit	Default	Control Bit	Default	Control Bits	Default	Control Bits	Default	Control Bit	Default	Control Bit	Default	Control Bit	Default
GPIO1_2	input	pulled up	NVCC1	regular	sw_pad_ctl_gpio1_2[0]	slow	—	no	sw_pad_ctl_gpio1_2[2:1]	std	—	100k pull-up	sw_pad_ctl_gpio1_2[8:7]	pull	sw_pad_ctl_gpio1_2[3]	PP	sw_pad_ctl_gpio1_2[4]	no
GPIO1_3	input	pulled up	NVCC1	regular	sw_pad_ctl_gpio1_3[0]	slow	—	no	sw_pad_ctl_gpio1_3[2:1]	std	—	100k pull-up	sw_pad_ctl_gpio1_3[8:7]	pull	sw_pad_ctl_gpio1_3[3]	PP	sw_pad_ctl_gpio1_3[4]	no
GPIO1_4	input	pulled up	NVCC1	regular	sw_pad_ctl_gpio1_4[0]	slow	—	no	sw_pad_ctl_gpio1_4[2:1]	std	—	100k pull-up	sw_pad_ctl_gpio1_4[8:7]	pull	sw_pad_ctl_gpio1_4[3]	PP	sw_pad_ctl_gpio1_4[4]	no
GPIO1_5 **note 1**	input	pulled up	NVCC1	regular	sw_pad_ctl_gpio1_5[0]	slow	—	no	sw_pad_ctl_gpio1_5[2:1]	std	—	100k pull-up	sw_pad_ctl_gpio1_5[8:7]	pull	sw_pad_ctl_gpio1_5[3]	PP	sw_pad_ctl_gpio1_5[4]	no
Note 1: GPIO1_5 should be connected to an external power management IC power ready output signal. If not used, GPIO1_5 must either be (a) externally pulled-up to NVCC1 or (b) a no connect, internally pulled-up by enabling the on-chip pull-up resistor. GPIO1_5 is a dedicated input and cannot be used as a general-purpose input/output.																		
GPIO1_6	input	pulled up	NVCC1	regular	sw_pad_ctl_gpio1_6[0]	slow	—	no	sw_pad_ctl_gpio1_6[2:1]	std	—	100k pull-up	sw_pad_ctl_gpio1_6[8:7]	pull	sw_pad_ctl_gpio1_6[3]	PP	sw_pad_ctl_gpio1_6[4]	no
GPIO3_0	input	pulled up	NVCC4	regular	sw_pad_ctl_gpio3_0[0]	slow	—	no	sw_pad_ctl_gpio3_0[2:1]	std	—	100k pull-up	sw_pad_ctl_gpio3_0[8:7]	pull	sw_pad_ctl_gpio3_0[3]	PP	sw_pad_ctl_gpio3_0[4]	no
GPIO3_1	input	pulled up	NVCC4	regular	sw_pad_ctl_gpio3_1[0]	slow	—	no	sw_pad_ctl_gpio3_1[2:1]	std	—	100k pull-up	sw_pad_ctl_gpio3_1[8:7]	pull	sw_pad_ctl_gpio3_1[3]	PP	sw_pad_ctl_gpio3_1[4]	no
SCLK0	input	pulled up	NVCC9	regular	sw_pad_ctl_sclk0[0]	slow	sw_pad_ctl_sclk0[9]	no	sw_pad_ctl_sclk0[2:1]	std	—	100k pull-up	sw_pad_ctl_sclk0[8]	pull	—	PP	—	no

Table 4-12. i.MX31 and i.MX31L I/O Settings (continued)

Package Contact Name	Value After Reset		Supply	I/O Type	Slew Rate		Loop back?		Drive Strength		Pull Value and Direction		Pull/Keeper Control		Open-Drain/ Push-Pull		Schmitt Trigger?	
	Direction	State			Control Bit	Default	Control Bit	Default	Control Bits	Default	Control Bits	Default	Control Bit	Default	Control Bit	Default	Control Bit	Default
SRST0	input	pulled up	NVCC9	regular	sw_pad_ctl_srst0[0]	slow	—	no	sw_pad_ctl_srst0 [2:1]	std	—	100k pull-up	sw_pad_ctl_srst0[8]	pull	—	PP	—	no
SVEN0	input	pulled up	NVCC9	regular	sw_pad_ctl_sven0[0]	slow	—	no	sw_pad_ctl_sven0 [2:1]	std	—	100k pull-up	sw_pad_ctl_sven0[8]	pull	—	PP	—	no
STX0	input	pulled up	NVCC9	regular	sw_pad_ctl_stx0[0]	slow	sw_pad_ctl_stx 0[9]	yes	sw_pad_ctl_stx0 [2]	std	—	100k pull-up	sw_pad_ctl_stx0[8]	pull	—	PP	—	no
SRX0	input	pulled up	NVCC9	regular	sw_pad_ctl_srx0[0]	slow	—	no	sw_pad_ctl_srx0[2]	std	—	100k pull-up	sw_pad_ctl_srx0[8]	pull	—	PP	—	no
SIMPD0	input	pulled up	NVCC9	regular	sw_pad_ctl_simpd0[0]	slow	—	no	sw_pad_ctl_simpd 0[2:1]	std	—	100k pull-up	sw_pad_ctl_simpd0[8]	pull	—	PP	—	no
CKIH	input	floating	NVCC1	regular	—	slow	—	no	—	std	—	100k pull-up	—	disable	—	PP	sw_pad_ctl_ckih[4]	yes
$\overline{\text{RESET\_IN}}$	input	pulled up	NVCC1	regular	—	slow	—	no	—	std	—	100k pull-up	—	pull	—	PP	—	yes
$\overline{\text{POR}}$	input	pulled up	NVCC1	regular	—	slow	—	no	—	std	—	100k pull-up	—	pull	—	PP	—	yes
CLKO	output	toggling	NVCC1	regular	—	fast	—	no	—	max	—	100k pull-up	—	disable	—	PP	—	no
BOOT_MODE0	input	floating	NVCC1	regular	—	slow	—	no	—	std	—	100k pull-up	—	disable	—	PP	—	no
BOOT_MODE1	input	floating	NVCC1	regular	—	slow	—	no	—	std	—	100k pull-up	—	disable	—	PP	—	no
BOOT_MODE2	input	floating	NVCC1	regular	—	slow	—	no	—	std	—	100k pull-up	—	disable	—	PP	—	no

Table 4-12. i.MX31 and i.MX31L I/O Settings (continued)

Package Contact Name	Value After Reset		Supply	I/O Type	Slew Rate		Loop back?		Drive Strength		Pull Value and Direction		Pull/Keeper Control		Open-Drain/ Push-Pull		Schmitt Trigger?	
	Direction	State			Control Bit	Default	Control Bit	Default	Control Bits	Default	Control Bits	Default	Control Bit	Default	Control Bit	Default	Control Bit	Default
BOOT_MODE3	input	floating	NVCC1	regular	—	slow	—	no	—	std	—	100k pull-up	—	disable	—	PP	—	no
BOOT_MODE4	input	floating	NVCC1	regular	—	slow	—	no	—	std	—	100k pull-up	—	disable	—	PP	—	no
CKIL	input	floating	NVCC1	regular	—	slow	—	no	—	std	—	100k pull-up	—	disable	—	PP	—	yes
POWER_FAIL	input	pulled down	NVCC1	regular	—	slow	—	no	—	std	—	100k pull-dn	—	pull	—	PP	—	no
VSTBY	output	low	NVCC1	regular	—	slow	—	no	—	std	—	100k pull-up	—	disable	—	PP	—	no
DVFS0	output	low	NVCC1	regular	—	slow	—	no	—	std	—	100k pull-up	—	disable	—	PP	—	no
DVFS1	output	low	NVCC1	regular	—	slow	—	no	—	std	—	100k pull-up	—	disable	—	PP	—	no
VPG0	output	low	NVCC1	regular	—	slow	—	no	—	std	—	100k pull-up	—	disable	—	PP	—	no
VPG1	output	low	NVCC1	regular	—	slow	—	no	—	std	—	100k pull-up	—	disable	—	PP	—	no
A0	output	low	NVCC22	regular	—	fast	—	no	sw_pad_ctl_a0[2]	max	—	100k pull-up	—	disable	—	PP	—	no
A1	output	low	NVCC22	regular	—	fast	—	no	sw_pad_ctl_a1[2]	max	—	100k pull-up	—	disable	—	PP	—	no
A2	output	low	NVCC22	regular	—	fast	—	no	sw_pad_ctl_a2[2]	max	—	100k pull-up	—	disable	—	PP	—	no
A3	output	low	NVCC22	regular	—	fast	—	no	sw_pad_ctl_a3[2]	max	—	100k pull-up	—	disable	—	PP	—	no
A4	output	low	NVCC22	regular	—	fast	—	no	sw_pad_ctl_a4[2]	max	—	100k pull-up	—	disable	—	PP	—	no

Table 4-12. i.MX31 and i.MX31L I/O Settings (continued)

Package Contact Name	Value After Reset		Supply	I/O Type	Slew Rate		Loop back?		Drive Strength		Pull Value and Direction		Pull/Keeper Control		Open-Drain/ Push-Pull		Schmitt Trigger?	
	Direction	State			Control Bit	Default	Control Bit	Default	Control Bits	Default	Control Bits	Default	Control Bit	Default	Control Bit	Default	Control Bit	Default
A5	output	low	NVCC22	regular	—	fast	—	no	sw_pad_ctl_a5[2]	max	—	100k pull-up	—	disable	—	PP	—	no
A6	output	low	NVCC22	regular	—	fast	—	no	sw_pad_ctl_a6[2]	max	—	100k pull-up	—	disable	—	PP	—	no
A7	output	low	NVCC22	regular	—	fast	—	no	sw_pad_ctl_a7[2]	max	—	100k pull-up	—	disable	—	PP	—	no
A8	output	low	NVCC22	regular	—	fast	—	no	sw_pad_ctl_a8[2]	max	—	100k pull-up	—	disable	—	PP	—	no
A9	output	low	NVCC22	regular	—	fast	—	no	sw_pad_ctl_a9[2]	max	—	100k pull-up	—	disable	—	PP	—	no
A10	output	low	NVCC2	regular	—	fast	—	no	sw_pad_ctl_a10[2]	max	—	100k pull-up	—	disable	—	PP	—	no
MA10	output	low	NVCC22	regular	—	fast	—	no	sw_pad_ctl_ma10[2]	max	—	100k pull-up	—	disable	—	PP	—	no
A11	output	low	NVCC22	regular	—	fast	—	no	sw_pad_ctl_a11[2]	max	—	100k pull-up	—	disable	—	PP	—	no
A12	output	low	NVCC22	regular	—	fast	—	no	sw_pad_ctl_a12[2]	max	—	100k pull-up	—	disable	—	PP	—	no
A13	output	low	NVCC22	regular	—	fast	—	no	sw_pad_ctl_a13[2]	max	—	100k pull-up	—	disable	—	PP	—	no
A14	output	low	NVCC21	regular	—	fast	—	no	sw_pad_ctl_a14[2]	max	—	100k pull-up	—	disable	—	PP	—	no
A15	output	low	NVCC21	regular	—	fast	—	no	sw_pad_ctl_a15[2]	max	—	100k pull-up	—	disable	—	PP	—	no

Table 4-12. i.MX31 and i.MX31L I/O Settings (continued)

Package Contact Name	Value After Reset		Supply	I/O Type	Slew Rate		Loop back?		Drive Strength		Pull Value and Direction		Pull/Keeper Control		Open-Drain/ Push-Pull		Schmitt Trigger?	
	Direction	State			Control Bit	Default	Control Bit	Default	Control Bits	Default	Control Bits	Default	Control Bit	Default	Control Bit	Default	Control Bit	Default
A16	output	low	NVCC21	regular	—	fast	—	no	sw_pad_ctl_a16[2]	max	—	100k pull-up	—	disable	—	PP	—	no
A17	output	low	NVCC22	regular	—	fast	—	no	sw_pad_ctl_a17[2]	max	—	100k pull-up	—	disable	—	PP	—	no
A18	output	low	NVCC22	regular	—	fast	—	no	sw_pad_ctl_a18[2]	max	—	100k pull-up	—	disable	—	PP	—	no
A19	output	low	NVCC22	regular	—	fast	—	no	sw_pad_ctl_a19[2]	max	—	100k pull-up	—	disable	—	PP	—	no
A20	output	low	NVCC22	regular	—	fast	—	no	sw_pad_ctl_a20[2]	max	—	100k pull-up	—	disable	—	PP	—	no
A21	output	low	NVCC22	regular	—	fast	—	no	sw_pad_ctl_a21[2]	max	—	100k pull-up	—	disable	—	PP	—	no
A22	output	low	NVCC22	regular	—	fast	—	no	sw_pad_ctl_a22[2]	max	—	100k pull-up	—	disable	—	PP	—	no
A23	output	low	NVCC22	regular	—	fast	—	no	sw_pad_ctl_a23[2]	max	—	100k pull-up	—	disable	—	PP	—	no
A24	output	low	NVCC22	regular	—	fast	—	no	sw_pad_ctl_a24[2]	max	—	100k pull-up	—	disable	—	PP	—	no
A25	output	low	NVCC22	regular	—	fast	—	no	sw_pad_ctl_a25[2]	max	—	100k pull-up	—	disable	—	PP	—	no
SDBA1	output	low	NVCC22	regular	—	fast	—	no	—	std	—	100k pull-up	—	disable	—	PP	—	no

Table 4-12. i.MX31 and i.MX31L I/O Settings (continued)

Package Contact Name	Value After Reset		Supply	I/O Type	Slew Rate		Loop back?		Drive Strength		Pull Value and Direction		Pull/Keeper Control		Open-Drain/ Push-Pull		Schmitt Trigger?	
	Direction	State			Control Bit	Default	Control Bit	Default	Control Bits	Default	Control Bits	Default	Control Bit	Default	Control Bit	Default	Control Bit	Default
SDBA0	output	low	NVCC22	regular	—	fast	—	no	—	std	—	100k pull-up	—	disable	—	PP	—	no
SD0	input	low	NVCC21	DDR	—	fast	—	no	sw_pad_ctl_sd0[2]	max	—	100k pull-up	—	keep	—	PP	—	no
SD1	input	low	NVCC21	DDR	—	fast	—	no	sw_pad_ctl_sd1[2]	max	—	100k pull-up	—	keep	—	PP	—	no
SD2	input	low	NVCC21	DDR	—	fast	—	no	sw_pad_ctl_sd2[2]	max	—	100k pull-up	—	keep	—	PP	—	no
SD3	input	low	NVCC21	DDR	—	fast	—	no	sw_pad_ctl_sd3[2]	max	—	100k pull-up	—	keep	—	PP	—	no
SD4	input	low	NVCC21	DDR	—	fast	—	no	sw_pad_ctl_sd4[2]	max	—	100k pull-up	—	keep	—	PP	—	no
SD5	input	low	NVCC21	DDR	—	fast	—	no	sw_pad_ctl_sd5[2]	max	—	100k pull-up	—	keep	—	PP	—	no
SD6	input	low	NVCC21	DDR	—	fast	—	no	sw_pad_ctl_sd6[2]	max	—	100k pull-up	—	keep	—	PP	—	no
SD7	input	low	NVCC21	DDR	—	fast	—	no	sw_pad_ctl_sd7[2]	max	—	100k pull-up	—	keep	—	PP	—	no
SD8	input	low	NVCC21	DDR	—	fast	—	no	sw_pad_ctl_sd8[2]	max	—	100k pull-up	—	keep	—	PP	—	no
SD9	input	low	NVCC21	DDR	—	fast	—	no	sw_pad_ctl_sd9[2]	max	—	100k pull-up	—	keep	—	PP	—	no

Table 4-12. i.MX31 and i.MX31L I/O Settings (continued)

Package Contact Name	Value After Reset		Supply	I/O Type	Slew Rate		Loop back?		Drive Strength		Pull Value and Direction		Pull/Keeper Control		Open-Drain/ Push-Pull		Schmitt Trigger?	
	Direction	State			Control Bit	Default	Control Bit	Default	Control Bits	Default	Control Bits	Default	Control Bit	Default	Control Bit	Default	Control Bit	Default
SD10	input	low	NVCC22	DDR	—	fast	—	no	sw_pad_ctl_sd10[2]	max	—	100k pull-up	—	keep	—	PP	—	no
SD11	input	low	NVCC22	DDR	—	fast	—	no	sw_pad_ctl_sd11[2]	max	—	100k pull-up	—	keep	—	PP	—	no
SD12	input	low	NVCC22	DDR	—	fast	—	no	sw_pad_ctl_sd12[2]	max	—	100k pull-up	—	keep	—	PP	—	no
SD13	input	low	NVCC22	DDR	—	fast	—	no	sw_pad_ctl_sd13[2]	max	—	100k pull-up	—	keep	—	PP	—	no
SD14	input	low	NVCC22	DDR	—	fast	—	no	sw_pad_ctl_sd14[2]	max	—	100k pull-up	—	keep	—	PP	—	no
SD15	input	low	NVCC22	DDR	—	fast	—	no	sw_pad_ctl_sd15[2]	max	—	100k pull-up	—	keep	—	PP	—	no
SD16	input	low	NVCC22	DDR	—	fast	—	no	sw_pad_ctl_sd16[2]	max	—	100k pull-up	—	keep	—	PP	—	no
SD17	input	low	NVCC22	DDR	—	fast	—	no	sw_pad_ctl_sd17[2]	max	—	100k pull-up	—	keep	—	PP	—	no
SD18	input	low	NVCC22	DDR	—	fast	—	no	sw_pad_ctl_sd18[2]	max	—	100k pull-up	—	keep	—	PP	—	no
SD19	input	low	NVCC22	DDR	—	fast	—	no	sw_pad_ctl_sd19[2]	max	—	100k pull-up	—	keep	—	PP	—	no

Table 4-12. i.MX31 and i.MX31L I/O Settings (continued)

Package Contact Name	Value After Reset		Supply	I/O Type	Slew Rate		Loop back?		Drive Strength		Pull Value and Direction		Pull/Keeper Control		Open-Drain/ Push-Pull		Schmitt Trigger?	
	Direction	State			Control Bit	Default	Control Bit	Default	Control Bits	Default	Control Bits	Default	Control Bit	Default	Control Bit	Default	Control Bit	Default
SD20	input	low	NVCC22	DDR	—	fast	—	no	sw_pad_ctl_sd20[2]	max	—	100k pull-up	—	keep	—	PP	—	no
SD21	input	low	NVCC22	DDR	—	fast	—	no	sw_pad_ctl_sd21[2]	max	—	100k pull-up	—	keep	—	PP	—	no
SD22	input	low	NVCC22	DDR	—	fast	—	no	sw_pad_ctl_sd22[2]	max	—	100k pull-up	—	keep	—	PP	—	no
SD23	input	low	NVCC22	DDR	—	fast	—	no	sw_pad_ctl_sd23[2]	max	—	100k pull-up	—	keep	—	PP	—	no
SD24	input	low	NVCC22	DDR	—	fast	—	no	sw_pad_ctl_sd24[2]	max	—	100k pull-up	—	keep	—	PP	—	no
SD25	input	low	NVCC22	DDR	—	fast	—	no	sw_pad_ctl_sd25[2]	max	—	100k pull-up	—	keep	—	PP	—	no
SD26	input	low	NVCC22	DDR	—	fast	—	no	sw_pad_ctl_sd26[2]	max	—	100k pull-up	—	keep	—	PP	—	no
SD27	input	low	NVCC22	DDR	—	fast	—	no	sw_pad_ctl_sd27[2]	max	—	100k pull-up	—	keep	—	PP	—	no
SD28	input	low	NVCC22	DDR	—	fast	—	no	sw_pad_ctl_sd28[2]	max	—	100k pull-up	—	keep	—	PP	—	no
SD29	input	low	NVCC22	DDR	—	fast	—	no	sw_pad_ctl_sd29[2]	max	—	100k pull-up	—	keep	—	PP	—	no

Table 4-12. i.MX31 and i.MX31L I/O Settings (continued)

Package Contact Name	Value After Reset		Supply	I/O Type	Slew Rate		Loop back?		Drive Strength		Pull Value and Direction		Pull/Keeper Control		Open-Drain/ Push-Pull		Schmitt Trigger?	
	Direction	State			Control Bit	Default	Control Bit	Default	Control Bits	Default	Control Bits	Default	Control Bit	Default	Control Bit	Default	Control Bit	Default
SD30	input	low	NVCC22	DDR	—	fast	—	no	sw_pad_ctl_sd30[2]	max	—	100k pull-up	—	keep	—	PP	—	no
SD31	input	low	NVCC22	DDR	—	fast	—	no	sw_pad_ctl_sd31[2]	max	—	100k pull-up	—	keep	—	PP	—	no
DQM0	output	low	NVCC2	DDR	—	fast	—	no	sw_pad_ctl_dqm0[2]	max	—	100k pull-up	—	keep	—	PP	—	no
DQM1	output	low	NVCC2	DDR	—	fast	—	no	sw_pad_ctl_dqm1[2]	max	—	100k pull-up	—	keep	—	PP	—	no
DQM2	output	low	NVCC2	DDR	—	fast	—	no	sw_pad_ctl_dqm2[2]	max	—	100k pull-up	—	keep	—	PP	—	no
DQM3	output	low	NVCC21	DDR	—	fast	—	no	sw_pad_ctl_dqm3[2]	max	—	100k pull-up	—	keep	—	PP	—	no
EB0	output	high	NVCC2	regular	—	fast	—	no	sw_pad_ctl_eb0[2]	high	—	100k pull-up	—	disable	—	PP	—	no
EB1	output	high	NVCC2	regular	—	fast	—	no	sw_pad_ctl_eb1[2]	high	—	100k pull-up	—	disable	—	PP	—	no
OE	output	high	NVCC2	regular	—	fast	—	no	sw_pad_ctl_oe[2]	high	—	100k pull-up	—	disable	—	PP	—	no
CS0	output	high	NVCC2	regular	—	fast	—	no	sw_pad_ctl_cs0[2]	high	—	100k pull-up	—	disable	—	PP	—	no
CS1	output	high	NVCC2	regular	—	fast	—	no	sw_pad_ctl_cs1[2]	high	—	100k pull-up	—	disable	—	PP	—	no
CS2	output	high	NVCC2	regular	—	fast	—	no	sw_pad_ctl_cs2[2]	max	—	100k pull-up	—	keep	—	PP	—	no

Table 4-12. i.MX31 and i.MX31L I/O Settings (continued)

Package Contact Name	Value After Reset		Supply	I/O Type	Slew Rate		Loop back?		Drive Strength		Pull Value and Direction		Pull/Keeper Control		Open-Drain/ Push-Pull		Schmitt Trigger?	
	Direction	State			Control Bit	Default	Control Bit	Default	Control Bits	Default	Control Bits	Default	Control Bit	Default	Control Bit	Default	Control Bit	Default
CS3	output	high	NVCC2	regular	—	fast	—	no	sw_pad_ctl_cs3[2]	max	—	100k pull-up	—	keep	—	PP	—	no
CS4	output	high	NVCC2	regular	—	fast	—	no	sw_pad_ctl_cs4[2]	high	—	100k pull-up	—	disable	—	PP	—	no
CS5	output	high	NVCC2	regular	—	fast	—	no	sw_pad_ctl_cs5[2]	high	—	100k pull-up	—	disable	—	PP	—	no
ECB	input	pulled up	NVCC2	regular	—	fast	—	no	sw_pad_ctl_ecb[2]	high	—	100k pull-up	—	pull	—	PP	—	no
LBA	output	high	NVCC2	regular	—	fast	—	no	sw_pad_ctl_lba[2]	high	—	100k pull-up	—	disable	—	PP	—	no
BCLK	output	low	NVCC2	regular	—	fast	—	yes	sw_pad_ctl_bclk[2]	high	—	100k pull-up	—	disable	—	PP	—	no
RW	output	high	NVCC2	regular	—	fast	—	no	sw_pad_ctl_rw[2]	high	—	100k pull-up	—	disable	—	PP	—	no
RAS	output	high	NVCC2	regular	—	fast	—	no	sw_pad_ctl_ras[2]	max	—	100k pull-up	—	keep	—	PP	—	no
CAS	output	high	NVCC2	regular	—	fast	—	no	sw_pad_ctl_cas[2]	max	—	100k pull-up	—	keep	—	PP	—	no
SDWE	output	high	NVCC2	regular	—	fast	—	no	—	std	—	100k pull-up	—	keep	—	PP	—	no
SDCKE0	output	low	NVCC2	regular	—	fast	—	no	—	std	—	100k pull-up	—	keep	—	PP	—	no
SDCKE1	output	low	NVCC2	regular	—	fast	—	no	—	std	—	100k pull-up	—	keep	—	PP	—	no
SDCLK **note 2**	output	toggling	NVCC2	regular	—	fast	—	yes	sw_pad_ctl_sdclk[2]	max	—	100k pull-up	—	disable	—	PP	—	no

Note 2: **SDCLK** and **$\overline{\text{SDCLK}}$** is a differential output pair that is controlled by the same Drive Strength bit.

Table 4-12. i.MX31 and i.MX31L I/O Settings (continued)

Package Contact Name	Value After Reset		Supply	I/O Type	Slew Rate		Loop back?		Drive Strength		Pull Value and Direction		Pull/Keeper Control		Open-Drain/ Push-Pull		Schmitt Trigger?	
	Direction	State			Control Bit	Default	Control Bit	Default	Control Bits	Default	Control Bits	Default	Control Bit	Default	Control Bit	Default	Control Bit	Default
$\overline{\text{SDCLK}}$ **note 2**	output	toggling	NVCC2	regular	—	fast	—	yes	sw_pad_ctl_sdclk [2]	max	—	100k pull-up	—	disable	—	PP	—	no
SDQS0	output	low	NVCC21	DDR	—	fast	—	no	—	std	—	100k pull-dn	—	pull	—	PP	—	no
SDQS1	output	low	NVCC22	DDR	—	fast	—	no	—	std	—	100k pull-dn	—	pull	—	PP	—	no
SDQS2	output	low	NVCC22	DDR	—	fast	—	no	—	std	—	100k pull-dn	—	pull	—	PP	—	no
SDQS3	output	low	NVCC22	DDR	—	fast	—	no	—	std	—	100k pull-dn	—	pull	—	PP	—	no
$\overline{\text{NFWE}}$	output	high	NVCC10	regular	$\overline{\text{sw_pad_ctl_nfwe}}$[0]	fast	—	no	$\overline{\text{sw_pad_ctl_nfwe}}$ [2:1]	high	—	100k pull-dn	$\overline{\text{sw_pad_ctl_nfwe}}$[8]	pull	—	PP	—	no
$\overline{\text{NFRE}}$	output	high	NVCC10	regular	$\overline{\text{sw_pad_ctl_nfre}}$[0]	fast	—	no	$\overline{\text{sw_pad_ctl_nfre}}$ [2:1]	high	—	100k pull-up	$\overline{\text{sw_pad_ctl_nfre}}$[8]	pull	—	PP	—	no
NFALE	output	low	NVCC10	regular	sw_pad_ctl_nfale[0]	fast	—	no	sw_pad_ctl_nfale [2:1]	high	—	100k pull-up	sw_pad_ctl_nfale[8]	pull	—	PP	—	no
NFCLE	output	low	NVCC10	regular	sw_pad_ctl_nfcle[0]	fast	—	no	sw_pad_ctl_nfcle [2:1]	high	—	100k pull-up	sw_pad_ctl_nfcle[8]	pull	—	PP	—	no
$\overline{\text{NFWP}}$	output	low during reset/ high after reset	NVCC10	regular	$\overline{\text{sw_pad_ctl_nfwp}}$[0]	fast	—	no	$\overline{\text{sw_pad_ctl_nfwp}}$ [2:1]	high	—	100k pull-up	$\overline{\text{sw_pad_ctl_nfwp}}$[8]	pull	—	PP	—	no
$\overline{\text{NFCE}}$	output	high	NVCC10	regular	$\overline{\text{sw_pad_ctl_nfce}}$[0]	fast	—	no	$\overline{\text{sw_pad_ctl_nfce}}$ [2:1]	high	—	100k pull-up	$\overline{\text{sw_pad_ctl_nfce}}$[8]	pull	—	PP	—	no

Table 4-12. i.MX31 and i.MX31L I/O Settings (continued)

Package Contact Name	Value After Reset		Supply	I/O Type	Slew Rate		Loop back?		Drive Strength		Pull Value and Direction		Pull/Keeper Control		Open-Drain/ Push-Pull		Schmitt Trigger?	
	Direction	State			Control Bit	Default	Control Bit	Default	Control Bits	Default	Control Bits	Default	Control Bit	Default	Control Bit	Default	Control Bit	Default
NFRB	input	pulled up	NVCC10	regular	sw_pad_ctl_nfrb[0]	fast	—	no	sw_pad_ctl_nfrb[2:1]	high	—	100k pull-up	sw_pad_ctl_nfrb[8]	pull	—	PP	—	no
D15	input	low	NVCC10	regular	—	fast	—	no	sw_pad_ctl_d15[2]	max	—	100k pull-up	—	keep	—	PP	—	no
D14	input	low	NVCC10	regular	—	fast	—	no	sw_pad_ctl_d14[2]	max	—	100k pull-up	—	keep	—	PP	—	no
D13	input	low	NVCC10	regular	—	fast	—	no	sw_pad_ctl_d13[2]	max	—	100k pull-up	—	keep	—	PP	—	no
D12	input	low	NVCC10	regular	—	fast	—	no	sw_pad_ctl_d12[2]	max	—	100k pull-up	—	keep	—	PP	—	no
D11	input	low	NVCC10	regular	—	fast	—	no	sw_pad_ctl_d11[2]	max	—	100k pull-up	—	keep	—	PP	—	no
D10	input	low	NVCC10	regular	—	fast	—	no	sw_pad_ctl_d10[2]	max	—	100k pull-up	—	keep	—	PP	—	no
D9	input	low	NVCC10	regular	—	fast	—	no	sw_pad_ctl_d9[2]	max	—	100k pull-up	—	keep	—	PP	—	no
D8	input	low	NVCC10	regular	—	fast	—	no	sw_pad_ctl_d8[2]	max	—	100k pull-up	—	keep	—	PP	—	no
D7	input	low	NVCC10	regular	—	fast	—	no	sw_pad_ctl_d7[2]	max	—	100k pull-up	—	keep	—	PP	—	no
D6	input	low	NVCC10	regular	—	fast	—	no	sw_pad_ctl_d6[2]	max	—	100k pull-up	—	keep	—	PP	—	no
D5	input	low	NVCC10	regular	—	fast	—	no	sw_pad_ctl_d5[2]	max	—	100k pull-up	—	keep	—	PP	—	no

Table 4-12. i.MX31 and i.MX31L I/O Settings (continued)

Package Contact Name	Value After Reset		Supply	I/O Type	Slew Rate		Loop back?		Drive Strength		Pull Value and Direction		Pull/Keeper Control		Open-Drain/ Push-Pull		Schmitt Trigger?	
	Direction	State			Control Bit	Default	Control Bit	Default	Control Bits	Default	Control Bits	Default	Control Bit	Default	Control Bit	Default	Control Bit	Default
D4	input	low	NVCC10	regular	—	fast	—	no	sw_pad_ctl_d4[2]	max	—	100k pull-up	—	keep	—	PP	—	no
D3	input	low	NVCC10	regular	—	fast	—	no	sw_pad_ctl_d3[2]	max	—	100k pull-up	—	keep	—	PP	—	no
D2	input	low	NVCC10	regular	—	fast	—	no	sw_pad_ctl_d2[2]	max	—	100k pull-up	—	keep	—	PP	—	no
D1	input	low	NVCC10	regular	—	fast	—	no	sw_pad_ctl_d1[2]	max	—	100k pull-up	—	keep	—	PP	—	no
D0	input	low	NVCC10	regular	—	fast	—	no	sw_pad_ctl_d0[2]	max	—	100k pull-up	—	keep	—	PP	—	no
$\overline{\text{PC\_CD1}}$ **note 3**	input	pulled up	NVCC3	regular	—	slow	$\overline{\text{sw\_pad\_ctl\_pc\_cd1}}$[9]	no	—	high	—	100k pull-up	$\overline{\text{sw\_pad\_ctl\_pc\_cd1}}$[8]	pull	—	PP	$\overline{\text{sw\_pad\_ctl\_pc\_cd1}}$[4]	no
Note 3: $\overline{\text{PC\_CD1}}$ and $\overline{\text{PC\_CD2}}$ are controlled by same Pull/Keeper control bit.																		
$\overline{\text{PC\_CD2}}$ **note 3, 4**	input	pulled up	NVCC3	regular	—	slow	—	no	—	high	sw_pad_ctl_pc_vs1[8]	100k pull-up	$\overline{\text{sw\_pad\_ctl\_pc\_cd1}}$[8]	pull	—	PP	—	no
Note 4: The setting of $\overline{\text{sw\_pad\_ctl\_pc\_cd2}}$[8] does not effect bit **ipp_pke** of **PC_CD2**. This bit is used as the pull up/down (0=pull dn/1=pull up) select (ipp_pus1) for PC_PWRON.																		
$\overline{\text{PC\_WAIT}}$	input	pulled up	NVCC3	regular	—	slow	$\overline{\text{sw\_pad\_ctl\_pc\_wait}}$[9]	no	—	high	sw_pad_ctl_pc_vs1[8] **note 5**	100k pull-up	$\overline{\text{sw\_pad\_ctl\_pc\_wait}}$[8] **note 6**	pull	—	PP	—	no

Table 4-12. i.MX31 and i.MX31L I/O Settings (continued)

Package Contact Name	Value After Reset		Supply	I/O Type	Slew Rate		Loop back?		Drive Strength		Pull Value and Direction		Pull/Keeper Control		Open-Drain/ Push-Pull		Schmitt Trigger?	
	Direction	State			Control Bit	Default	Control Bit	Default	Control Bits	Default	Control Bits	Default	Control Bit	Default	Control Bit	Default	Control Bit	Default
PC_READY	input	pulled up	NVCC3	regular	—	slow	sw_pad_ctl_pc_ready [9]	no	—	high	sw_pad_ctl_pc_vs1[8] **note 5**	100k pull-up	sw_pad_ctl_pc_ready [8]	pull	—	PP	—	no
PC_PWRON	input	pulled down	NVCC3	regular	—	slow	sw_pad_ctl_pc_pwron [9]	no	—	high	sw_pad_ctl_pc_cd2[8] **note 4**	100k pull-dn	sw_pad_ctl_pc_pwron [8]	pull	—	PP	—	no
PC_VS1	input	pulled up	NVCC3	regular	—	slow	sw_pad_ctl_pc_vs1 [9]	no	—	high	sw_pad_ctl_pc_VS1[8] **note 5**	100k pull-up	$\overline{\text{sw\_pad\_ctl\_pc\_wait}}$ [8] **note 6**	pull	—	PP	—	no
Note 5: The setting of bit 8 does not effect bit **ipp_pke** of **PC_VS1**. This bit is used as the pull up/down (0=pull dn/1=pull up) select (ipp_pus1) for $\overline{\text{PC\_WAIT}}$, PC_READY, PC_VS1																		
Note 6: PC_WAIT and PC_VS1 are controlled by the same Pull/Keeper control bit.																		
PC_VS2	input	pulled up	NVCC3	regular	—	slow	—	no	—	high	—	100k pull-up	sw_pad_ctl_pc_vs2[8]	pull	—	PP	—	no
PC_BVD1	input	pulled up	NVCC3	regular	—	slow	—	no	—	high	—	100k pull-up	sw_pad_ctl_pc_bvd1 [8]	pull	—	PP	—	no
PC_BVD2	input	pulled up	NVCC3	regular	—	slow	—	no	—	high	—	100k pull-up	sw_pad_ctl_pc_bvd2 [8]	pull	—	PP	—	no

Table 4-12. i.MX31 and i.MX31L I/O Settings (continued)

Package Contact Name	Value After Reset		Supply	I/O Type	Slew Rate		Loop back?		Drive Strength		Pull Value and Direction		Pull/Keeper Control		Open-Drain/ Push-Pull		Schmitt Trigger?	
	Direction	State			Control Bit	Default	Control Bit	Default	Control Bits	Default	Control Bits	Default	Control Bit	Default	Control Bit	Default	Control Bit	Default
PC_RST	output	low	NVCC3	regular	—	slow	—	no	—	high	—	100k pull-up	sw_pad_ctl_pc_rst[8]	pull	—	PP	—	no
IOIS16	input	pulled up	NVCC3	regular	—	slow	—	no	—	high	—	100k pull-up	sw_pad_ctl_pc_iois16 [8]	pull	—	PP	—	no
$\overline{\text{PC\_RW}}$	output	high	NVCC3	regular	—	slow	—	no	—	high	—	100k pull-up	$\overline{\text{sw\_pad\_ctl\_pc\_rw}}$[8]	pull	—	PP	—	no
PC_POE	output	high	NVCC3	regular	—	slow	—	no	—	high	—	100k pull-up	—	disable	—	PP	—	no
M_REQUEST **note 7**	output	low	NVCC2	regular	—	slow	—	no	—	std	—	100k pull-up	—	disable	—	PP	—	no
Note 7: M_REQUEST and M_GRANT are not utilized in MX31/31L. No connections should be made to these signals.																		
M_GRANT **note 7**	input	pulled up	NVCC2	regular	—	slow	—	no	—	std	—	100k pull-up	—	pull	—	PP	—	no
CSI_D4	input	not floating	NVCC4	regular	sw_pad_ctl_csi_d4[0]	fast	—	no	sw_pad_ctl_csi_d4[2:1]	high	—	100k pull-up	sw_pad_ctl_csi_d4 [7:8]	keep	—	PP	—	no
CSI_D5	input	not floating	NVCC4	regular	sw_pad_ctl_csi_d5[0]	fast	—	no	sw_pad_ctl_csi_d5[2:1]	high	—	100k pull-up	sw_pad_ctl_csi_d5 [7:8]	keep	—	PP	—	no
CSI_D6	input	not floating	NVCC4	regular	sw_pad_ctl_csi_d6[0]	fast	—	no	sw_pad_ctl_csi_d6[2:1]	high	—	100k pull-up	sw_pad_ctl_csi_d6 [7:8]	keep	—	PP	—	no
CSI_D7	input	not floating	NVCC4	regular	sw_pad_ctl_csi_d7[0]	fast	—	no	sw_pad_ctl_csi_d7[2:1]	high	—	100k pull-up	sw_pad_ctl_csi_d7 [7:8]	keep	—	PP	—	no

Table 4-12. i.MX31 and i.MX31L I/O Settings (continued)

Package Contact Name	Value After Reset		Supply	I/O Type	Slew Rate		Loop back?		Drive Strength		Pull Value and Direction		Pull/Keeper Control		Open-Drain/ Push-Pull		Schmitt Trigger?	
	Direction	State			Control Bit	Default	Control Bit	Default	Control Bits	Default	Control Bits	Default	Control Bit	Default	Control Bit	Default	Control Bit	Default
CSI_D8	input	not floating	NVCC4	regular	sw_pad_ctl_csi_d8[0]	fast	—	no	sw_pad_ctl_csi_d8[2:1]	high	—	100k pull-up	sw_pad_ctl_csi_d8 [7:8]	keep	—	PP	—	no
CSI_D9	input	not floating	NVCC4	regular	sw_pad_ctl_csi_d9[0]	fast	—	no	sw_pad_ctl_csi_d9[2:1]	high	—	100k pull-up	sw_pad_ctl_csi_d9 [7:8]	keep	—	PP	—	no
CSI_D10	input	not floating	NVCC4	regular	sw_pad_ctl_csi_d10[0]	fast	—	no	sw_pad_ctl_csi_d10[2:1]	high	—	100k pull-up	sw_pad_ctl_csi_d10 [7:8]	keep	—	PP	—	no
CSI_D11	input	not floating	NVCC4	regular	sw_pad_ctl_csi_d11[0]	fast	—	no	sw_pad_ctl_csi_d11[2:1]	high	—	100k pull-up	sw_pad_ctl_csi_d11 [7:8]	keep	—	PP	—	no
CSI_D12	input	not floating	NVCC4	regular	sw_pad_ctl_csi_d12[0]	fast	—	no	sw_pad_ctl_csi_d12[2:1]	high	—	100k pull-up	sw_pad_ctl_csi_d12 [7:8]	keep	—	PP	—	no
CSI_D13	input	not floating	NVCC4	regular	sw_pad_ctl_csi_d13[0]	fast	—	no	sw_pad_ctl_csi_d13[2:1]	high	—	100k pull-up	sw_pad_ctl_csi_d13 [7:8]	keep	—	PP	—	no
CSI_D14	input	not floating	NVCC4	regular	sw_pad_ctl_csi_d14[0]	fast	—	no	sw_pad_ctl_csi_d14[2:1]	high	—	100k pull-up	sw_pad_ctl_csi_d14 [7:8]	keep	—	PP	—	no
CSI_D15	input	not floating	NVCC4	regular	sw_pad_ctl_csi_d15[0]	fast	—	no	sw_pad_ctl_csi_d15[2:1]	high	—	100k pull-up	sw_pad_ctl_csi_d15 [7:8]	keep	—	PP	—	no
CSI_MCLK	input	not floating	NVCC4	regular	sw_pad_ctl_csi_mclk[0]	fast	—	no	sw_pad_ctl_csi_mclk[2:1]	high	—	100k pull-up	sw_pad_ctl_csi_mclk [7:8]	keep	—	PP	—	no
CSI_VSYNC	input	not floating	NVCC4	regular	sw_pad_ctl_csi_vsync[0]	fast	—	no	sw_pad_ctl_csi_vsync[2:1]	high	—	100k pull-up	sw_pad_ctl_csi_vsync [7:8]	keep	—	PP	—	no

Table 4-12. i.MX31 and i.MX31L I/O Settings (continued)

Package Contact Name	Value After Reset		Supply	I/O Type	Slew Rate		Loop back?		Drive Strength		Pull Value and Direction		Pull/Keeper Control		Open-Drain/ Push-Pull		Schmitt Trigger?	
	Direction	State			Control Bit	Default	Control Bit	Default	Control Bits	Default	Control Bits	Default	Control Bit	Default	Control Bit	Default	Control Bit	Default
CSI_HSYNC	input	not floating	NVCC4	regular	sw_pad_ctl_csi_hsync[0]	fast	—	no	sw_pad_ctl_csi_hsync[2:1]	high	—	100k pull-up	sw_pad_ctl_csi_hsync[7:8]	keep	—	PP	—	no
CSI_PIXCLK	input	not floating	NVCC4	regular	sw_pad_ctl_csi_pixclk[0]	fast	—	no	sw_pad_ctl_csi_pixclk[2:1]	high	—	100k pull-up	sw_pad_ctl_csi_pixclk [7:8]	keep	—	PP	sw_pad_ctl_csi_pixclk[4]	yes
I2C_CLK	input	pulled up	NVCC4	regular	sw_pad_ctl_i2c_clk[0]	slow	—	no	sw_pad_ctl_i2c_clk[2:1]	std	—	100k pull-up	sw_pad_ctl_i2c_clk [8]	pull	sw_pad_ctl_i2c_clk [3]	PP	sw_pad_ctl_i2c_clk [4]	yes
I2C_DAT	input	pulled up	NVCC4	regular	sw_pad_ctl_i2c_dat[0]	slow	—	no	sw_pad_ctl_i2c_dat[2:1]	std	—	100k pull-up	sw_pad_ctl_i2c_dat[8]	pull	sw_pad_ctl_i2c_dat[3]	PP	—	yes
STXD3	input	pulled up	NVCC10	regular	sw_pad_ctl_stxd3[0]	slow	—	no	sw_pad_ctl_stxd3[2:1]	std	—	100k pull-up	sw_pad_ctl_stxd3[8]	pull	—	PP	—	no
SRXD3	input	pulled up	NVCC10	regular	sw_pad_ctl_srxd3[0]	slow	—	no	sw_pad_ctl_srxd3 [2:1]	std	—	100k pull-up	sw_pad_ctl_srxd3[8]	pull	—	PP	—	no
SCK3	input	pulled up	NVCC10	regular	sw_pad_ctl_sck3[0]	slow	—	no	sw_pad_ctl_sck3 [2:1]	std	—	100k pull-up	sw_pad_ctl_sck3[8]	pull	—	PP	sw_pad_ctl_sck3[4]	yes
SFS3	input	pulled up	NVCC10	regular	sw_pad_ctl_sfs3[0]	slow	—	no	sw_pad_ctl_sfs3 [2:1]	std	—	100k pull-up	sw_pad_ctl_sfs3[8]	pull	—	PP	—	no
STXD4	input	pulled up	NVCC5	regular	sw_pad_ctl_stxd4[0]	slow	—	no	sw_pad_ctl_stxd4 [2:1]	std	—	100k pull-up	sw_pad_ctl_stxd4[8]	pull	—	PP	—	no
SRXD4	input	pulled up	NVCC5	regular	sw_pad_ctl_srxd4[0]	slow	—	no	sw_pad_ctl_srxd4 [2:1]	std	—	100k pull-up	sw_pad_ctl_srxd4[8]	pull	—	PP	sw_pad_ctl_srxd4[4]	no

Table 4-12. i.MX31 and i.MX31L I/O Settings (continued)

Package Contact Name	Value After Reset		Supply	I/O Type	Slew Rate		Loop back?		Drive Strength		Pull Value and Direction		Pull/Keeper Control		Open-Drain/ Push-Pull		Schmitt Trigger?	
	Direction	State			Control Bit	Default	Control Bit	Default	Control Bits	Default	Control Bits	Default	Control Bit	Default	Control Bit	Default	Control Bit	Default
SCK4	input	pulled up	NVCC5	regular	sw_pad_ctl_sck4[0]	slow	—	no	sw_pad_ctl_sck4 [2:1]	std	—	100k pull-up	sw_pad_ctl_sck4[8]	pull	—	PP	sw_pad_ctl_sck4[4]	yes
SFS4	input	pulled up	NVCC5	regular	sw_pad_ctl_sfs4[0]	slow	—	no	sw_pad_ctl_sfs4 [2:1]	std	—	100k pull-up	sw_pad_ctl_sfs4[8]	pull	—	PP	sw_pad_ctl_sfs4[4]	no
STXD5	input	pulled up	NVCC5	regular	sw_pad_ctl_stxd5[0]	slow	—	no	sw_pad_ctl_stxd5 [2:1]	std	—	100k pull-up	sw_pad_ctl_stxd5[8]	pull	—	PP	—	no
SRXD5	input	pulled up	NVCC5	regular	sw_pad_ctl_srxd5[0]	slow	—	no	sw_pad_ctl_srxd5 [2:1]	std	—	100k pull-up	sw_pad_ctl_srxd5[8]	pull	—	PP	—	no
SCK5	input	pulled up	NVCC5	regular	sw_pad_ctl_sck5[0]	slow	—	no	sw_pad_ctl_sck5 [2:1]	std	—	100k pull-up	sw_pad_ctl_sck5[8]	pull	—	PP	sw_pad_ctl_sck5[4]	yes
SFS5	input	pulled up	NVCC5	regular	sw_pad_ctl_sfs5[0]	slow	—	no	sw_pad_ctl_sfs5 [2:1]	std	—	100k pull-up	sw_pad_ctl_sfs5[8]	pull	—	PP	—	no
STXD6	input	pulled up	NVCC10	regular	sw_pad_ctl_stxd6[0]	slow	—	no	sw_pad_ctl_stxd6 [2:1]	std	—	100k pull-up	sw_pad_ctl_stxd6[8]	pull	—	PP	—	no
SRXD6	input	pulled up	NVCC10	regular	sw_pad_ctl_srxd6[0]	slow	—	no	sw_pad_ctl_srxd6 [2:1]	std	—	100k pull-up	sw_pad_ctl_srxd6[8]	pull	—	PP	—	no
SCK6	input	pulled up	NVCC10	regular	sw_pad_ctl_sck6[0]	slow	—	no	sw_pad_ctl_sck6 [2:1]	std	—	100k pull-up	sw_pad_ctl_sck6[8]	pull	—	PP	sw_pad_ctl_sck6[4]	yes
SFS6	input	pulled up	NVCC10	regular	sw_pad_ctl_sfs6[0]	slow	—	no	sw_pad_ctl_sfs6 [2:1]	std	—	100k pull-up	sw_pad_ctl_sfs6[8]	pull	—	PP	—	no

Table 4-12. i.MX31 and i.MX31L I/O Settings (continued)

Package Contact Name	Value After Reset		Supply	I/O Type	Slew Rate		Loop back?		Drive Strength		Pull Value and Direction		Pull/Keeper Control		Open-Drain/ Push-Pull		Schmitt Trigger?	
	Direction	State			Control Bit	Default	Control Bit	Default	Control Bits	Default	Control Bits	Default	Control Bit	Default	Control Bit	Default	Control Bit	Default
CSPI1_MOSI	input	pulled up	NVCC10	regular	sw_pad_ctl_cspi1_mosi[0]	slow	—	no	sw_pad_ctl_cspi1_mosi [2:1]	std	—	100k pull-up	sw_pad_ctl_cspi1_mosi[8]	pull	—	PP	—	no
CSPI1_MISO	input	pulled up	NVCC10	regular	sw_pad_ctl_cspi1_miso[0]	slow	—	no	sw_pad_ctl_cspi1_miso [2:1]	std	—	100k pull-up	sw_pad_ctl_cspi1_miso[8]	pull	—	PP	—	no
CSPI1_SS0	input	pulled up	NVCC10	regular	sw_pad_ctl_cspi1_ss0[0]	slow	—	no	sw_pad_ctl_cspi1_ss0[2:1]	std	—	100k pull-up	sw_pad_ctl_cspi1_ss0[8]	pull	—	PP	—	no
CSPI1_SS1	input	pulled up	NVCC10	regular	sw_pad_ctl_cspi1_ss1[0]	slow	—	no	sw_pad_ctl_cspi1_ss1[2:1]	std	—	100k pull-up	sw_pad_ctl_cspi1_ss1[8]	pull	—	PP	—	no
CSPI1_SS2	input	pulled up	NVCC10	regular	sw_pad_ctl_cspi1_ss2[0]	slow	—	no	sw_pad_ctl_cspi1_ss2[2:1]	std	—	100k pull-up	sw_pad_ctl_cspi1_ss2[8]	pull	—	PP	—	no
CSPI1_SCLK	input	pulled up	NVCC10	regular	sw_pad_ctl_cspi1_sclk[0]	slow	—	no	sw_pad_ctl_cspi1_sclk[2:1]	std	—	100k pull-up	sw_pad_ctl_cspi1_sclk[8]	pull	—	PP	sw_pad_ctl_cspi1_sclk[4]	yes
CSPI1_SPI_RDY	input	pulled up	NVCC10	regular	sw_pad_ctl_cspi1_spi_rdy[0]	slow	—	no	sw_pad_ctl_cspi1_spi_rdy[2:1]	std	—	100k pull-up	sw_pad_ctl_cspi1_spi_rdy[8]	pull	—	PP	—	no
CSPI2_MOSI	input	pulled up	NVCC5	regular	sw_pad_ctl_cspi2_mosi[0]	slow	—	no	sw_pad_ctl_cspi2_mosi[2:1]	std	—	100k pull-up	sw_pad_ctl_cspi2_mosi[8]	pull	sw_pad_ctl_cspi2_mosi[3]	PP	—	yes
CSPI2_MISO	input	pulled up	NVCC5	regular	sw_pad_ctl_cspi2_miso[0]	slow	—	no	sw_pad_ctl_cspi2_miso[2:1]	std	—	100k pull-up	sw_pad_ctl_cspi2_miso[8]	pull	sw_pad_ctl_cspi2_miso[3]	PP	—	yes

Table 4-12. i.MX31 and i.MX31L I/O Settings (continued)

Package Contact Name	Value After Reset		Supply	I/O Type	Slew Rate		Loop back?		Drive Strength		Pull Value and Direction		Pull/Keeper Control		Open-Drain/ Push-Pull		Schmitt Trigger?	
	Direction	State			Control Bit	Default	Control Bit	Default	Control Bits	Default	Control Bits	Default	Control Bit	Default	Control Bit	Default	Control Bit	Default
CSPI2_SS0	input	pulled up	NVCC5	regular	sw_pad_ctl_cspi2_ss0[0]	slow	—	no	sw_pad_ctl_cspi2_ss0[2:1]	std	—	100k pull-up	sw_pad_ctl_cspi2_ss0[8]	pull	—	PP	—	no
CSPI2_SS1	input	pulled up	NVCC5	regular	sw_pad_ctl_cspi2_ss1[0]	slow	—	no	sw_pad_ctl_cspi2_ss1[2:1]	std	—	100k pull-up	sw_pad_ctl_cspi2_ss1[8]	pull	—	PP	—	no
CSPI2_SS2	input	pulled up	NVCC5	regular	sw_pad_ctl_cspi2_ss2[0]	slow	—	no	sw_pad_ctl_cspi2_ss2[2:1]	std	—	100k pull-up	sw_pad_ctl_cspi2_ss2[8]	pull	sw_pad_ctl_cspi2_ss2[3]	PP	—	yes
CSPI2_SCLK	input	pulled up	NVCC5	regular	sw_pad_ctl_cspi2_sclk[0]	slow	—	no	sw_pad_ctl_cspi2_sclk[2:1]	std	—	100k pull-up	sw_pad_ctl_cspi2_sclk[8]	pull	sw_pad_ctl_cspi2_sclk[3]	PP	sw_pad_ctl_cspi2_sclk[4]	yes
CSPI2_SPI_RDY	input	pulled up	NVCC5	regular	sw_pad_ctl_cspi2_spi_rdy[0]	slow	—	no	sw_pad_ctl_cspi2_spi_rdy[2:1]	std	—	100k pull-up	sw_pad_ctl_cspi2_spi_rdy[8]	pull	—	PP	—	no
RXD1	input	pulled up	NVCC8	regular	sw_pad_ctl_rxd1[0]	slow	—	no	sw_pad_ctl_rxd1[2:1]	std	—	100k pull-up	sw_pad_ctl_rxd1[8]	pull	—	PP	—	no
TXD1	input	pulled up	NVCC8	regular	sw_pad_ctl_txd1[0]	slow	—	no	sw_pad_ctl_txd1[2:1]	std	—	100k pull-up	sw_pad_ctl_txd1[8]	pull	—	PP	sw_pad_ctl_txd1[4]	no
RTS1	input	pulled up	NVCC8	regular	sw_pad_ctl_rts1[0]	slow	—	no	sw_pad_ctl_rts1[2:1]	std	—	100k pull-up	sw_pad_ctl_rts1[8]	pull	—	PP	—	no
CTS1	input	pulled up	NVCC8	regular	sw_pad_ctl_cts1[0]	slow	—	no	sw_pad_ctl_cts1[2:1]	std	—	100k pull-up	sw_pad_ctl_cts1[8]	pull	—	PP	—	no

Table 4-12. i.MX31 and i.MX31L I/O Settings (continued)

Package Contact Name	Value After Reset		Supply	I/O Type	Slew Rate		Loop back?		Drive Strength		Pull Value and Direction		Pull/Keeper Control		Open-Drain/ Push-Pull		Schmitt Trigger?	
	Direction	State			Control Bit	Default	Control Bit	Default	Control Bits	Default	Control Bits	Default	Control Bit	Default	Control Bit	Default	Control Bit	Default
DTR_DCE1	input	pulled up	NVCC8	regular	sw_pad_ctl_dtr_dce1[0]	slow	—	no	sw_pad_ctl_dtr_dce1[2:1]	std	—	100k pull-up	sw_pad_ctl_dtr_dce1 [8]	pull	—	PP	—	no
DSR_DCE1	input	pulled up	NVCC8	regular	sw_pad_ctl_dsr_dce1[0]	slow	—	no	sw_pad_ctl_dsr_dce1[2:1]	std	—	100k pull-up	sw_pad_ctl_dsr_dce1 [8]	pull	—	PP	sw_pad_ctl_dsr_dce1[4]	no
RI_DCE1	input	pulled up	NVCC8	regular	sw_pad_ctl_ri_dce1[0]	slow	—	no	sw_pad_ctl_ri_dce1[2:1]	std	—	100k pull-up	sw_pad_ctl_ri_dce1[8]	pull	—	PP	—	no
DCD_DCE1	input	pulled up	NVCC8	regular	sw_pad_ctl_dcd_dce1[0]	slow	—	no	sw_pad_ctl_dcd_dce1[2:1]	std	—	100k pull-up	sw_pad_ctl_dcd_dce1[8]	pull	—	PP	—	no
DTR_DTE1	input	pulled up	NVCC8	regular	sw_pad_ctl_dtr_dte1[0]	slow	—	no	sw_pad_ctl_dtr_dte1[2:1]	std	—	100k pull-up	sw_pad_ctl_dtr_dte1[8]	pull	—	PP	—	yes
DSR_DTE1	input	pulled up	NVCC8	regular	sw_pad_ctl_dsr_dte1[0]	slow	—	no	sw_pad_ctl_dsr_dte1[2:1]	std	—	100k pull-up	sw_pad_ctl_dsr_dte1[8]	pull	—	PP	—	no
RI_DTE1	input	pulled up	NVCC8	regular	sw_pad_ctl_ri_dte1[0]	slow	—	no	sw_pad_ctl_ri_dte1[2:1]	std	—	100k pull-up	sw_pad_ctl_ri_dte1[8]	pull	sw_pad_ctl_ri_dte1[3]	PP	—	yes
DCD_DTE1	input	pulled up	NVCC8	regular	sw_pad_ctl_dcd_dte1[0]	slow	—	no	sw_pad_ctl_dcd_dte1[2:1]	std	—	100k pull-up	sw_pad_ctl_dcd_dte1[8]	pull	sw_pad_ctl_dcd_dte1[3]	PP	—	no
DTR_DCE2	input	pulled up	NVCC8	regular	sw_pad_ctl_dtr_dce2[0]	slow	—	no	sw_pad_ctl_dtr_dce2[2:1]	std	—	100k pull-up	sw_pad_ctl_dtr_dce2[8]	pull	—	PP	—	no

Table 4-12. i.MX31 and i.MX31L I/O Settings (continued)

Package Contact Name	Value After Reset		Supply	I/O Type	Slew Rate		Loop back?		Drive Strength		Pull Value and Direction		Pull/Keeper Control		Open-Drain/ Push-Pull		Schmitt Trigger?	
	Direction	State			Control Bit	Default	Control Bit	Default	Control Bits	Default	Control Bits	Default	Control Bit	Default	Control Bit	Default	Control Bit	Default
RXD2	input	pulled up	NVCC8	regular	sw_pad_ctl_rxd2[0]	slow	—	no	sw_pad_ctl_rxd2[2:1]	std	—	100k pull-up	sw_pad_ctl_rxd2[8]	pull	—	PP	—	no
TXD2	input	pulled up	NVCC8	regular	sw_pad_ctl_txd2[0]	slow	—	no	sw_pad_ctl_txd2[2:1]	std	—	100k pull-up	sw_pad_ctl_txd2[8]	pull	—	PP	—	no
RTS2	input	pulled up	NVCC8	regular	sw_pad_ctl_rts2[0]	slow	—	no	sw_pad_ctl_rts2[2:1]	std	—	100k pull-up	sw_pad_ctl_rts2[8]	pull	—	PP	—	no
CTS2	input	pulled up	NVCC8	regular	sw_pad_ctl_cts2[0]	slow	—	no	sw_pad_ctl_cts2[2:1]	std	—	100k pull-up	sw_pad_ctl_cts2[8]	pull	—	PP	—	no
BATT_LINE	input	pulled up	NVCC5	regular	—	slow	—	yes	—	std	—	100k pull-up	sw_pad_ctl_batt_line[8]	pull	—	OD	—	no
KEY_ROW0	input	pulled up	NVCC6	regular	sw_pad_ctl_key_row0[0]	slow	—	no	—	std	—	100k pull-up	sw_pad_ctl_key_row0[8]	pull	—	PP	—	no
KEY_ROW1	input	pulled up	NVCC6	regular	sw_pad_ctl_key_row1[0]	slow	—	no	sw_pad_ctl_key_row1[2:1]	std	—	100k pull-up	sw_pad_ctl_key_row1[8]	pull	—	PP	—	no
KEY_ROW2	input	pulled up	NVCC6	regular	sw_pad_ctl_key_row2[0]	slow	—	no	sw_pad_ctl_key_row2[2:1]	std	—	100k pull-up	sw_pad_ctl_key_row2[8]	pull	—	PP	—	no
KEY_ROW3	input	pulled up	NVCC6	regular	sw_pad_ctl_key_row3[0]	slow	—	no	sw_pad_ctl_key_row3[2:1]	std	—	100k pull-up	sw_pad_ctl_key_row3[8]	pull	—	PP	—	no
KEY_ROW4	input	pulled up	NVCC6	regular	sw_pad_ctl_key_row4[0]	slow	—	no	sw_pad_ctl_key_row4[2:1]	std	—	100k pull-up	sw_pad_ctl_key_row4[8]	pull	—	PP	—	no

Table 4-12. i.MX31 and i.MX31L I/O Settings (continued)

Package Contact Name	Value After Reset		Supply	I/O Type	Slew Rate		Loop back?		Drive Strength		Pull Value and Direction		Pull/Keeper Control		Open-Drain/ Push-Pull		Schmitt Trigger?	
	Direction	State			Control Bit	Default	Control Bit	Default	Control Bits	Default	Control Bits	Default	Control Bit	Default	Control Bit	Default	Control Bit	Default
KEY_ROW5	input	pulled up	NVCC6	regular	sw_pad_ctl_key_row5[0]	slow	—	no	sw_pad_ctl_key_row5[2:1]	std	—	100k pull-up	sw_pad_ctl_key_row5[8]	pull	—	PP	—	no
KEY_ROW6	input	pulled up	NVCC6	regular	sw_pad_ctl_key_row6[0]	slow	—	no	sw_pad_ctl_key_row6[2:1]	std	—	100k pull-up	sw_pad_ctl_key_row6[8]	pull	—	PP	—	no
KEY_ROW7	input	pulled up	NVCC6	regular	sw_pad_ctl_key_row7[0]	slow	—	no	sw_pad_ctl_key_row7[2:1]	std	—	100k pull-up	sw_pad_ctl_key_row7[8]	pull	—	PP	—	no
KEY_COL0	input	pulled up	NVCC6	regular	sw_pad_ctl_key_col0[0]	slow	—	no	sw_pad_ctl_key_col0[2:1]	std	—	100k pull-up	sw_pad_ctl_key_col0[8]	pull	sw_pad_ctl_key_col0[3]	PP	—	no
KEY_COL1	input	pulled up	NVCC6	regular	sw_pad_ctl_key_col1[0]	slow	—	no	sw_pad_ctl_key_col1[2:1]	std	—	100k pull-up	sw_pad_ctl_key_col1[8]	pull	sw_pad_ctl_key_col1[3]	PP	—	no
KEY_COL2	input	pulled up	NVCC6	regular	sw_pad_ctl_key_col2[0]	slow	—	no	sw_pad_ctl_key_col2[2:1]	std	—	100k pull-up	sw_pad_ctl_key_col2[8]	pull	sw_pad_ctl_key_col2[3]	PP	—	no
KEY_COL3	input	pulled up	NVCC6	regular	sw_pad_ctl_key_col3[0]	slow	—	no	sw_pad_ctl_key_col3[2:1]	std	—	100k pull-up	sw_pad_ctl_key_col3[8]	pull	sw_pad_ctl_key_col3[3]	PP	—	no
KEY_COL4	input	pulled up	NVCC6	regular	sw_pad_ctl_key_col4[0]	slow	—	no	sw_pad_ctl_key_col4[2:1]	std	—	100k pull-up	sw_pad_ctl_key_col4[8]	pull	sw_pad_ctl_key_col4[3]	PP	—	no
KEY_COL5	input	pulled up	NVCC6	regular	sw_pad_ctl_key_col5[0]	slow	—	no	sw_pad_ctl_key_col5[2:1]	std	—	100k pull-up	sw_pad_ctl_key_col5[8]	pull	sw_pad_ctl_key_col5[3]	PP	—	no

Table 4-12. i.MX31 and i.MX31L I/O Settings (continued)

Package Contact Name	Value After Reset		Supply	I/O Type	Slew Rate		Loop back?		Drive Strength		Pull Value and Direction		Pull/Keeper Control		Open-Drain/ Push-Pull		Schmitt Trigger?	
	Direction	State			Control Bit	Default	Control Bit	Default	Control Bits	Default	Control Bits	Default	Control Bit	Default	Control Bit	Default	Control Bit	Default
KEY_COL6	input	pulled up	NVCC6	regular	sw_pad_ctl_key_col6[0]	slow	—	no	sw_pad_ctl_key_col6[2:1]	std	—	100k pull-up	sw_pad_ctl_key_col6[8]	pull	sw_pad_ctl_key_col6[3]	PP	—	no
KEY_COL7	input	pulled up	NVCC6	regular	sw_pad_ctl_key_col7[0]	slow	—	no	sw_pad_ctl_key_col7[2:1]	std	—	100k pull-up	sw_pad_ctl_key_col7[8]	pull	sw_pad_ctl_key_col7[3]	PP	—	no
RTCK	output	—	NVCC6	regular	—	fast	—	no	—	high	—	100k pull-dn	—	disable	—	PP	—	no
TCK	input	pulled down	NVCC6	regular	—	slow	—	no	—	std	—	100k pull-dn	—	pull	—	PP	—	yes
TMS	input	pulled up	NVCC6	regular	—	slow	—	no	—	std	—	100k pull-up	—	pull	—	PP	—	yes
TDI	input	pulled up	NVCC6	regular	—	slow	—	no	—	std	—	100k pull-up	—	pull	—	PP	—	yes
TDO	—	—	NVCC6	regular	—	fast	—	no	—	high	—	100k pull-up	—	disable	—	PP	—	no
TRSTB	input	pulled up	NVCC6	regular	—	slow	—	no	—	std	—	100k pull-up	—	pull	—	PP	—	no
DE_B	—	—	NVCC6	regular	—	slow	—	no	—	std	—	100k pull-up	—	pull	—	PP	—	no
SJC_MOD **note 8**	input	pulled up	NVCC6	regular	sw_pad_ctl_sjc_mod[0]	slow	—	no	sw_pad_ctl_sjc_mod[2:1]	std	—	100k pull-up	—	pull	—	PP	—	no
Note 8: SJC_MOD must be externally connected to GND for normal operation. Termination to GND through an external pull-down resistor (such as 1 kΩ) is allowed, but the value should be much smaller than the on-chip pull-up.																		
USB_PWR	input	pulled up	NVCC5	regular	sw_pad_ctl_usb_pwr[0]	slow	—	no	sw_pad_ctl_usb_pwr[2:1]	std	—	100k pull-up	sw_pad_ctl_usb_pwr[8]	pull	—	PP	—	no

Table 4-12. i.MX31 and i.MX31L I/O Settings (continued)

Package Contact Name	Value After Reset		Supply	I/O Type	Slew Rate		Loop back?		Drive Strength		Pull Value and Direction		Pull/Keeper Control		Open-Drain/ Push-Pull		Schmitt Trigger?	
	Direction	State			Control Bit	Default	Control Bit	Default	Control Bits	Default	Control Bits	Default	Control Bit	Default	Control Bit	Default	Control Bit	Default
USB_OC	input	pulled up	NVCC5	regular	sw_pad_ctl_usb_oc[0]	slow	—	no	sw_pad_ctl_usb_oc[2:1]	std	—	100k pull-up	sw_pad_ctl_usb_oc[8]	pull	—	PP	—	no
USB_BYP	input	pulled up	NVCC5	regular	sw_pad_ctl_usb_byp[0]	slow	—	no	sw_pad_ctl_usb_byp[2:1]	std	—	100k pull-up	sw_pad_ctl_usb_byp [8]	pull	—	PP	—	no
USBOTG_CLK	input	pulled up	NVCC5	regular	sw_pad_ctl_usbotg_clk[0]	slow	—	no	sw_pad_ctl_usbotg_clk[2:1]	std	—	100k pull-up	sw_pad_ctl_usbotg_clk[8]	pull	—	PP	sw_pad_ctl_usbotg_clk[4]	yes
USBOTG_DIR	input	pulled up	NVCC5	regular	sw_pad_ctl_usbotg_dir[0]	slow	—	no	sw_pad_ctl_usbotg_dir[2:1]	std	—	100k pull-up	sw_pad_ctl_usbotg_dir[8]	pull	—	PP	—	no
USBOTG_STP	input	pulled up	NVCC5	regular	sw_pad_ctl_usbotg_stp[0]	slow	—	no	sw_pad_ctl_usbotg_stp[2:1]	std	—	100k pull-up	sw_pad_ctl_usbotg_stp[8]	pull	—	PP	—	no
USBOTG_NXT	input	pulled up	NVCC5	regular	sw_pad_ctl_usbotg_nxt[0]	slow	—	no	sw_pad_ctl_usbotg_nxt[2:1]	std	—	100k pull-up	sw_pad_ctl_usbotg_nxt[8]	pull	—	PP	—	no
USBOTG_DATA0	input	pulled up	NVCC5	regular	sw_pad_ctl_usbotg_data0[0]	slow	—	no	sw_pad_ctl_usbotg_data0[2:1]	std	—	100k pull-up	sw_pad_ctl_usbotg_data0[8]	pull	—	PP	—	no
USBOTG_DATA1	input	pulled up	NVCC5	regular	sw_pad_ctl_usbotg_data1[0]	slow	—	no	sw_pad_ctl_usbotg_data1[2:1]	std	—	100k pull-up	sw_pad_ctl_usbotg_data1[8]	pull	—	PP	—	no
USBOTG_DATA2	input	pulled up	NVCC5	regular	sw_pad_ctl_usbotg_data2[0]	slow	—	no	sw_pad_ctl_usbotg_data2[2:1]	std	—	100k pull-up	sw_pad_ctl_usbotg_data2[8]	pull	—	PP	—	no

Table 4-12. i.MX31 and i.MX31L I/O Settings (continued)

Package Contact Name	Value After Reset		Supply	I/O Type	Slew Rate		Loop back?		Drive Strength		Pull Value and Direction		Pull/Keeper Control		Open-Drain/ Push-Pull		Schmitt Trigger?	
	Direction	State			Control Bit	Default	Control Bit	Default	Control Bits	Default	Control Bits	Default	Control Bit	Default	Control Bit	Default	Control Bit	Default
USBOTG_DATA3	input	pulled up	NVCC5	regular	sw_pad_ctl_usbotg_data3[0]	slow	—	no	sw_pad_ctl_usbotg_data3[2:1]	std	—	100k pull-up	sw_pad_ctl_usbotg_data3[8]	pull	—	PP	—	no
USBOTG_DATA4	input	pulled up	NVCC5	regular	sw_pad_ctl_usbotg_data4[0]	slow	—	no	sw_pad_ctl_usbotg_data4[2:1]	std	—	100k pull-up	sw_pad_ctl_usbotg_data4[8]	pull	—	PP	—	no
USBOTG_DATA5	input	pulled up	NVCC5	regular	sw_pad_ctl_usbotg_data5[0]	slow	—	no	sw_pad_ctl_usbotg_data5[2:1]	std	—	100k pull-up	sw_pad_ctl_usbotg_data5[8]	pull	—	PP	—	no
USBOTG_DATA6	input	pulled up	NVCC5	regular	sw_pad_ctl_usbotg_data6[0]	slow	—	no	sw_pad_ctl_usbotg_data6[2:1]	std	—	100k pull-up	sw_pad_ctl_usbotg_data6[8]	pull	—	PP	—	no
USBOTG_DATA7	input	pulled up	NVCC5	regular	sw_pad_ctl_usbotg_data7[0]	slow	—	no	sw_pad_ctl_usbotg_data7[2:1]	std	—	100k pull-up	sw_pad_ctl_usbotg_data7[8]	pull	—	PP	—	no
USBH2_CLK	input	pulled up	NVCC10	regular	sw_pad_ctl_usbh2_clk[0]	slow	—	no	sw_pad_ctl_usbh2_clk[2:1]	std	—	100k pull-up	sw_pad_ctl_usbh2_clk[8]	pull	—	PP	sw_pad_ctl_usbh2_clk[4]	yes
USBH2_DIR	input	pulled up	NVCC10	regular	sw_pad_ctl_usbh2_dir[0]	slow	—	no	sw_pad_ctl_usbh2_dir[2:1]	std	—	100k pull-up	sw_pad_ctl_usbh2_dir[8]	pull	—	PP	—	no
USBH2_STP	input	pulled up	NVCC10	regular	sw_pad_ctl_usbh2_stp[0]	slow	—	no	sw_pad_ctl_usbh2_stp[2:1]	std	—	100k pull-up	sw_pad_ctl_usbh2_stp[8]	pull	—	PP	—	no

Table 4-12. i.MX31 and i.MX31L I/O Settings (continued)

Package Contact Name	Value After Reset		Supply	I/O Type	Slew Rate		Loop back?		Drive Strength		Pull Value and Direction		Pull/Keeper Control		Open-Drain/ Push-Pull		Schmitt Trigger?	
	Direction	State			Control Bit	Default	Control Bit	Default	Control Bits	Default	Control Bits	Default	Control Bit	Default	Control Bit	Default	Control Bit	Default
USBH2_NXT	input	pulled up	NVCC10	regular	sw_pad_ctl_usbh2_nxt[0]	slow	—	no	sw_pad_ctl_usbh2_nxt[2:1]	std	—	100k pull-up	sw_pad_ctl_usbh2_nxt[8]	pull	—	PP	—	no
USBH2_DATA0	input	pulled up	NVCC10	regular	sw_pad_ctl_usbh2_data0[0]	slow	—	no	sw_pad_ctl_usbh2_data0[2:1]	std	—	100k pull-up	sw_pad_ctl_usbh2_data0[8]	pull	—	PP	—	no
USBH2_DATA1	input	pulled up	NVCC10	regular	sw_pad_ctl_usbh2_data1[0]	slow	—	no	sw_pad_ctl_usbh2_data1[2:1]	std	—	100k pull-up	sw_pad_ctl_usbh2_data1[8]	pull	—	PP	—	no
LD0	input	pulled up	NVCC7	regular	—	fast	—	no	sw_pad_ctl_ld0[2]	high	—	100k pull-up	sw_pad_ctl_ld0[7]	pull	—	PP	—	no
LD1	input	pulled up	NVCC7	regular	—	fast	—	no	sw_pad_ctl_ld1[2]	high	—	100k pull-up	sw_pad_ctl_ld1[7]	pull	—	PP	—	no
LD2	input	pulled up	NVCC7	regular	—	fast	—	no	sw_pad_ctl_ld2[2]	high	—	100k pull-up	sw_pad_ctl_ld2[7]	pull	—	PP	—	no
LD3	input	pulled up	NVCC7	regular	—	fast	—	no	sw_pad_ctl_ld3[2]	high	—	100k pull-up	sw_pad_ctl_ld3[7]	pull	—	PP	—	no
LD4	input	pulled up	NVCC7	regular	—	fast	—	no	sw_pad_ctl_ld4[2]	high	—	100k pull-up	sw_pad_ctl_ld4[7]	pull	—	PP	—	no
LD5	input	pulled up	NVCC7	regular	—	fast	—	no	sw_pad_ctl_ld5[2]	high	—	100k pull-up	sw_pad_ctl_ld5[7]	pull	—	PP	—	no
LD6	input	pulled up	NVCC7	regular	—	fast	—	no	sw_pad_ctl_ld6[2]	high	—	100k pull-up	sw_pad_ctl_ld6[7]	pull	—	PP	—	no
LD7	input	pulled up	NVCC7	regular	—	fast	—	no	sw_pad_ctl_ld7[2]	high	—	100k pull-up	sw_pad_ctl_ld7[7]	pull	—	PP	—	no
LD8	input	pulled up	NVCC7	regular	—	fast	—	no	sw_pad_ctl_ld8[2]	high	—	100k pull-up	sw_pad_ctl_ld8[7]	pull	—	PP	—	no

Table 4-12. i.MX31 and i.MX31L I/O Settings (continued)

Package Contact Name	Value After Reset		Supply	I/O Type	Slew Rate		Loop back?		Drive Strength		Pull Value and Direction		Pull/Keeper Control		Open-Drain/ Push-Pull		Schmitt Trigger?	
	Direction	State			Control Bit	Default	Control Bit	Default	Control Bits	Default	Control Bits	Default	Control Bit	Default	Control Bit	Default	Control Bit	Default
LD9	input	pulled up	NVCC7	regular	—	fast	—	no	sw_pad_ctl_ld9[2]	high	—	100k pull-up	sw_pad_ctl_ld9[7]	pull	—	PP	—	no
LD10	input	pulled up	NVCC7	regular	—	fast	—	no	sw_pad_ctl_ld10[2]	high	—	100k pull-up	sw_pad_ctl_ld10[7]	pull	—	PP	—	no
LD11	input	pulled up	NVCC7	regular	—	fast	—	no	sw_pad_ctl_ld11[2]	high	—	100k pull-up	sw_pad_ctl_ld11[7]	pull	—	PP	—	no
LD12	input	pulled up	NVCC7	regular	—	fast	—	no	sw_pad_ctl_ld12[2]	high	—	100k pull-up	sw_pad_ctl_ld12[7]	pull	—	PP	—	no
LD13	input	pulled up	NVCC7	regular	—	fast	—	no	sw_pad_ctl_ld13[2]	high	—	100k pull-up	sw_pad_ctl_ld13[7]	pull	—	PP	—	no
LD14	input	pulled up	NVCC7	regular	—	fast	—	no	sw_pad_ctl_ld14[2]	high	—	100k pull-up	sw_pad_ctl_ld14[7]	pull	—	PP	—	no
LD15	input	pulled up	NVCC7	regular	—	fast	—	no	sw_pad_ctl_ld15[2]	high	—	100k pull-up	sw_pad_ctl_ld15[7]	pull	—	PP	—	no
LD16	input	pulled up	NVCC7	regular	—	fast	—	no	sw_pad_ctl_ld16[2]	high	—	100k pull-up	sw_pad_ctl_ld16[7]	pull	—	PP	—	no
LD17	input	pulled up	NVCC7	regular	—	fast	—	no	sw_pad_ctl_ld17[2]	high	—	100k pull-up	sw_pad_ctl_ld17[7]	pull	—	PP	—	no
VSYNC0	input	pulled up	NVCC7	regular	—	fast	—	no	sw_pad_ctl_vsync0[2]	high	—	100k pull-up	sw_pad_ctl_vsync0[7]	pull	—	PP	—	no
HSYNC	input	floating	NVCC7	regular	—	fast	—	no	sw_pad_ctl_hsync[2]	high	—	100k pull-up	—	disable	—	PP	—	no

Table 4-12. i.MX31 and i.MX31L I/O Settings (continued)

Package Contact Name	Value After Reset		Supply	I/O Type	Slew Rate		Loop back?		Drive Strength		Pull Value and Direction		Pull/Keeper Control		Open-Drain/ Push-Pull		Schmitt Trigger?	
	Direction	State			Control Bit	Default	Control Bit	Default	Control Bits	Default	Control Bits	Default	Control Bit	Default	Control Bit	Default	Control Bit	Default
FPSHIFT	input	floating	NVCC7	regular	—	fast	—	no	sw_pad_ctl_fpshift[2]	high	—	100k pull-up	—	disable	—	PP	—	no
DRDY0	input	floating	NVCC7	regular	—	fast	—	no	sw_pad_ctl_drdy0[2]	high	—	100k pull-up	—	disable	—	PP	—	no
SD_D_I	input	floating	NVCC7	regular	—	fast	—	no	sw_pad_ctl_sd_d_o[2]	high	—	100k pull-up	—	disable	—	PP	—	no
SD_D_IO	input	floating	NVCC7	regular	—	fast	—	no	sw_pad_ctl_sd_d_io[2]	high	—	100k pull-up	—	disable	—	PP	—	no
SD_D_CLK	input	floating	NVCC7	regular	—	fast	—	no	sw_pad_ctl_sd_d_clk[2]	high	—	100k pull-up	—	disable	—	PP	—	no
LCS0	input	floating	NVCC7	regular	—	fast	—	no	sw_pad_ctl_lcs0[2]	high	—	100k pull-up	—	disable	—	PP	—	no
LCS1	input	floating	NVCC7	regular	—	fast	—	no	sw_pad_ctl_lcs1[2]	high	—	100k pull-up	—	disable	—	PP	—	no
SER_RS	input	floating	NVCC7	regular	—	fast	—	no	sw_pad_ctl_ser_rs[2]	high	—	100k pull-up	—	disable	—	PP	—	no
PAR_RS	input	floating	NVCC7	regular	—	fast	—	no	sw_pad_ctl_par_rs[2]	high	—	100k pull-up	—	disable	—	PP	—	no
WRITE	input	floating	NVCC7	regular	—	fast	—	no	sw_pad_ctl_write[2]	high	—	100k pull-up	—	disable	—	PP	—	no

Table 4-12. i.MX31 and i.MX31L I/O Settings (continued)

Package Contact Name	Value After Reset		Supply	I/O Type	Slew Rate		Loop back?		Drive Strength		Pull Value and Direction		Pull/Keeper Control		Open-Drain/ Push-Pull		Schmitt Trigger?	
	Direction	State			Control Bit	Default	Control Bit	Default	Control Bits	Default	Control Bits	Default	Control Bit	Default	Control Bit	Default	Control Bit	Default
READ	input	floating	NVCC7	regular	—	fast	—	no	sw_pad_ctl_read[2]	high	—	100k pull-up	—	disable	—	PP	—	no
VSYNC3	input	pulled up	NVCC7	regular	—	fast	—	no	sw_pad_ctl_vsync3[2]	high	—	100k pull-up	sw_pad_ctl_vsync3[7]	pull	—	PP	—	no
CONTRAST	input	floating	NVCC7	regular	—	fast	—	no	sw_pad_ctl_contrast[2]	high	—	100k pull-up	—	disable	—	PP	—	no
D3_REV	input	floating	NVCC7	regular	—	fast	—	no	sw_pad_ctl_d3_rev[2]	high	—	100k pull-up	—	disable	—	PP	—	no
D3_CLS	input	floating	NVCC7	regular	—	fast	—	no	sw_pad_ctl_d3_cls[2]	high	—	100k pull-up	—	disable	—	PP	—	no
D3_SPL	input	floating	NVCC7	regular	—	fast	—	no	sw_pad_ctl_d3_spl[2]	high	—	100k pull-up	—	disable	—	PP	—	no
SD1_CMD	input	floating	NVCC3	regular	sw_pad_ctl_sd1_cmd[0]	fast	sw_pad_ctl_sd1_cmd[9]	no	sw_pad_ctl_sd1_cmd[2:1]	std	—	100k pull-up	—	disable	—	PP	sw_pad_ctl_sd1_cmd[4]	no
SD1_CLK	input	floating	NVCC3	regular	sw_pad_ctl_sd1_clk[0]	fast	—	no	sw_pad_ctl_sd1_clk[2:1]	std	—	100k pull-up	—	disable	—	PP	—	no
SD1_DATA0 **note 9, 10**	input	floating	NVCC3	regular	sw_pad_ctl_sd1_data0[0]	fast	sw_pad_ctl_sd1_data0[9]	no	sw_pad_ctl_sd1_data0[2:1]	std	sw_pad_ctl_sd1_data2[8]	100k pull-up	sw_pad_ctl_sd1_data0[8]	disable	—	PP	—	no

Table 4-12. i.MX31 and i.MX31L I/O Settings (continued)

Package Contact Name	Value After Reset		Supply	I/O Type	Slew Rate		Loop back?		Drive Strength		Pull Value and Direction		Pull/Keeper Control		Open-Drain/ Push-Pull		Schmitt Trigger?	
	Direction	State			Control Bit	Default	Control Bit	Default	Control Bits	Default	Control Bits	Default	Control Bit	Default	Control Bit	Default	Control Bit	Default
SD1_DATA1 **note 9**	input	floating	NVCC3	regular	sw_pad_ctl_sd1_data1[0]	fast	sw_pad_ctl_sd1_data1[9]	no	sw_pad_ctl_sd1_data1[2:1]	std	sw_pad_ctl_sd1_data2[8]	100k pull-up	sw_pad_ctl_sd1_data1[8]	disable	—	PP	—	no
SD1_DATA2 **note 9, 10**	input	floating	NVCC3	regular	sw_pad_ctl_sd1_data2[0]	fast	sw_pad_ctl_sd1_data2[9]	no	sw_pad_ctl_sd1_data2[2:1]	std	sw_pad_ctl_sd1_data2[8]	100k pull-up	sw_pad_ctl_sd1_data0[8]	disable	—	PP	—	no
SD1_DATA3 **note 9**	input	floating	NVCC3	regular	sw_pad_ctl_sd1_data3[0]	fast	sw_pad_ctl_sd1_data3[9]	no	sw_pad_ctl_sd1_data3[2:1]	std	sw_pad_ctl_sd1_data2[8]	100k pull-dn	sw_pad_ctl_sd1_data3[8]	disable	—	PP	—	no
Note 9: SD1_DATA[3:0] are controlled by the same Pull Value and Direction bit. 0=100k pull-down, 1=100k pull-up. **Note 10:** SD1_DATA2 and SD1_DATA0 are controlled by the same Pull/Keeper control bit.																		
ATA_CS0	input	floating	NVCC3	regular	sw_pad_ctl_ata_cs0[0]	slow	—	no	sw_pad_ctl_ata_cs0[2:1]	std		100k pull-up	—	disable	—	PP	—	no
ATA_CS1	input	floating	NVCC3	regular	sw_pad_ctl_ata_cs1[0]	slow	—	no	sw_pad_ctl_ata_cs1[2:1]	std	—	100k pull-up	—	disable	—	PP	—	no
ATA_DIOR	input	pulled up	NVCC3	regular	sw_pad_ctl_ata_dior[0]	slow	—	no	sw_pad_ctl_ata_dior[2:1]	std	—	100k pull-up	sw_pad_ctl_ata_dior[8]	pull	—	PP	—	no
ATA_DIOW	input	pulled up	NVCC3	regular	sw_pad_ctl_ata_diow[0]	slow	—	no	sw_pad_ctl_ata_diow[2:1]	std	—	100k pull-up	sw_pad_ctl_ata_diow[8]	pull	—	PP	—	no
ATA_DMACK	input	pulled up	NVCC3	regular	sw_pad_ctl_ata_dmack[0]	slow	—	no	sw_pad_ctl_ata_dmack[2:1]	std	—	100k pull-up	sw_pad_ctl_ata_dmack[8]	pull	—	PP	—	no

Table 4-12. i.MX31 and i.MX31L I/O Settings (continued)

Package Contact Name	Value After Reset		Supply	I/O Type	Slew Rate		Loop back?		Drive Strength		Pull Value and Direction		Pull/Keeper Control		Open-Drain/ Push-Pull		Schmitt Trigger?	
	Direction	State			Control Bit	Default	Control Bit	Default	Control Bits	Default	Control Bits	Default	Control Bit	Default	Control Bit	Default	Control Bit	Default
ATA_RESET	input	pulled up	NVCC3	regular	sw_pad_ctl_ata_reset[0]	slow	—	no	sw_pad_ctl_ata_reset[2:1]	std	—	100k pull-up	sw_pad_ctl_ata_reset[8]	pull	—	PP	—	no
CE_CONTROL **note 11**	input	floating	NVCC8	regular	—	fast	—	no	—	std	—	100k pull-dn	—	disable	—	PP	—	no
Note 11: CE_CONTROL is a reserved input and must be externally tied to GND through a 1 kΩ resistor.																		
CLKSS	input	floating	NVCC1	regular	—	fast	—	no	—	std	—	100k pull-up	—	disable	—	PP	—	no
CSPI3_MOSI	input	pulled up	NVCC3	regular	sw_pad_ctl_cspi3_mosi[0]	slow	—	no	sw_pad_ctl_cspi3_mosi[2:1]	std	—	100k pull-up	sw_pad_ctl_cspi3_mosi[8]	pull	—	PP	—	no
CSPI3_MISO	input	pulled up	NVCC3	regular	sw_pad_ctl_cspi3_miso[0]	slow	—	no	sw_pad_ctl_cspi3_miso[2:1]	std	—	100k pull-up	sw_pad_ctl_cspi3_miso[8]	pull	—	PP	—	no
CSPI3_SCLK	input	pulled up	NVCC3	regular	sw_pad_ctl_cspi3_sclk[0]	slow	—	no	sw_pad_ctl_cspi3_sclk[2:1]	std	—	100k pull-up	sw_pad_ctl_cspi3_sclk[8]	pull	—	PP	sw_pad_ctl_cspi3_sclk[4]	yes
CSPI3_SPI_RDY	input	pulled up	NVCC3	regular	sw_pad_ctl_cspi3_spi_rdy[0]	slow	—	no	sw_pad_ctl_cspi3_spi_rdy[2:1]	std	—	100k pull-up	sw_pad_ctl_cspi3_spi_rdy[8]	pull	—	PP	—	no
TTM_PAD	—	pulled down	NVCC7	regular	—	slow	—	no	—	std	—	100k pull-dn [1]	—	disable[1]	—	PP	—	no

[1] TTM_PAD is for Freescale factory use only. Control bits indicate pull-up/down disabled. However, TTM_PAD is actually connected to an on-chip pull-down device. Users must float this signal or tie it to GND.

4.4.2 EMI Signal Multiplexing

The EMI signal multiplexing described in this section deals with multiplexing that is performed in the EMI. This type of multiplexing is performed automatically whenever the processor accesses a particular memory space that is controlled by one of the four memory controllers: EIM, ESDCTL (SDRAM-SDR or SDRAM-DDR), PCMCIA, or NFC. See Table 4-13 for details.

Table 4-13. EMI Signal Multiplexing

Contact Name	I/O Type	EIM	SDRAM SDR	PCMCIA	SDRAM DDR	NFC
A0	regular	A0	MA0	A0	MA0	—
A1	regular	A1	MA1	A1	MA1	—
A2	regular	A2	MA2	A2	MA2	—
A3	regular	A3	MA3	A3	MA3	—
A4	regular	A4	MA4	A4	MA4	—
A5	regular	A5	MA5	A5	MA5	—
A6	regular	A6	MA6	A6	MA6	—
A7	regular	A7	MA7	A7	MA7	—
A8	regular	A8	MA8	A8	MA8	—
A9	regular	A9	MA9	A9	MA9	—
A10	regular	A10	—	A10	—	—
MA10	regular	—	MA10	—	MA10	—
A11	regular	A11	MA11	A11	MA11	—
A12	regular	A12	MA12	A12	MA12	—
A13	regular	A13	MA13	A13	MA13	—
A14	regular	A14	—	A14	—	—
A15	regular	A15	—	A15	—	—
A16	regular	A16	—	A16	—	—
A17	regular	A17	—	A17	—	—
A18	regular	A18	—	A18	—	—
A19	regular	A19	—	A19	—	—
A20	regular	A20	—	A20	—	—
A21	regular	A21	—	A21	—	—
A22	regular	A22	—	A22	—	—
A23	regular	A23	—	A23	—	—
A24	regular	A24	—	A24	—	—
A25	regular	A25	—	A25	—	—

Table 4-13. EMI Signal Multiplexing (continued)

Contact Name	I/O Type	EIM	SDRAM SDR	PCMCIA	SDRAM DDR	NFC
SDBA1	regular	—	SDBA1	$\overline{CE1}$	SDBA1	—
SDBA0	regular	—	SDBA0	$\overline{CE2}$	SDBA0	—
SD0	ddr	—	SD0	—	SD0	—
SD1	ddr	—	SD1	—	SD1	—
SD2	ddr	—	SD2	—	SD2	—
SD3	ddr	—	SD3	—	SD3	—
SD4	ddr	—	SD4	—	SD4	—
SD5	ddr	—	SD5	—	SD5	—
SD6	ddr	—	SD6	—	SD6	—
SD7	ddr	—	SD7	—	SD7	—
SD8	ddr	—	SD8	—	SD8	—
SD9	ddr	—	SD9	—	SD9	—
SD10	ddr	—	SD10	—	SD10	—
SD11	ddr	—	SD11	—	SD11	—
SD12	ddr	—	SD12	—	SD12	—
SD13	ddr	—	SD13	—	SD13	—
SD14	ddr	—	SD14	—	SD14	—
SD15	ddr	—	SD15	—	SD15	—
SD16	ddr	—	SD16	—	SD16	—
SD17	ddr	—	SD17	—	SD17	—
SD18	ddr	—	SD18	—	SD18	—
SD19	ddr	—	SD19	—	SD19	—
SD20	ddr	—	SD20	—	SD20	—
SD21	ddr	—	SD21	—	SD21	—
SD22	ddr	—	SD22	—	SD22	—
SD23	ddr	—	SD23	—	SD23	—
SD24	ddr	—	SD24	—	SD24	—
SD25	ddr	—	SD25	—	SD25	—
SD26	ddr	—	SD26	—	SD26	—
SD27	ddr	—	SD27	—	SD27	—
SD28	ddr	—	SD28	—	SD28	—
SD29	ddr	—	SD29	—	SD29	—

Table 4-13. EMI Signal Multiplexing (continued)

Contact Name	I/O Type	EIM	SDRAM SDR	PCMCIA	SDRAM DDR	NFC
SD30	ddr	—	SD30	—	SD30	—
SD31	ddr	—	SD31	—	SD31	—
DQM0	ddr	—	DQM0	—	DQM0	—
DQM1	ddr	—	DQM1	—	DQM1	—
DQM2	ddr	—	DQM2	—	DQM2	—
DQM3	ddr	—	DQM3	—	DQM3	—
EB0	regular	EB0	—	REG	—	—
EB1	regular	EB1	—	IORD	—	—
OE	regular	OE	—	IOWR	—	—
CS0	regular	CS0	—	—	—	—
CS1	regular	CS1	—	—	—	—
CS2	regular	CS2	CSD0	—	CSD0	—
CS3	regular	CS3	CSD1	—	CSD1	—
CS4	regular	CS4	—	—	—	—
CS5	regular	CS5	—	—	—	—
ECB	regular	ECB	—	—	—	—
LBA	regular	LBA	—	$\overline{OE}$	—	—
BCLK	regular	BCLK	—	—	—	—
RW	regular	RW	—	WE	—	—
RAS	regular	—	RAS	—	RAS	—
CAS	regular	—	CAS	—	CAS	—
SDWE	regular	—	SDWE	—	SDWE	—
SDCKE0	regular	—	SDCKE0	—	SDCKE0	—
SDCKE1	regular	—	SDCKE1	—	SDCKE1	—
SDCLK	regular	—	SDCLK	—	SDCLK	—
$\overline{SDCLK}$	regular	—	—	—	$\overline{SDCLK}$	—
SDQS0	ddr	—	—	—	SDQS0	—
SDQS1	ddr	—	—	—	SDQS1	—
SDQS2	ddr	—	—	—	SDQS2	—
SDQS3	ddr	—	—	—	SDQS3	—
$\overline{NFWE}$	regular	—	—	—	—	WE
$\overline{NFRE}$	regular	—	—	—	—	RE

Table 4-13. EMI Signal Multiplexing (continued)

Contact Name	I/O Type	EIM	SDRAM SDR	PCMCIA	SDRAM DDR	NFC
NFALE	regular	—	—	—	—	ALE
NFCLE	regular	—	—	—	—	CLE
$\overline{\text{NFWP}}$	regular	—	—	—	—	WP
$\overline{\text{NFCE}}$	regular	—	—	—	—	CE
NFRB	regular	—	—	—	—	R/B
D15	regular	D15	—	D15	—	D15
D14	regular	D14	—	D14	—	D14
D13	regular	D13	—	D13	—	D13
D12	regular	D12	—	D12	—	D12
D11	regular	D11	—	D11	—	D11
D10	regular	D10	—	D10	—	D10
D9	regular	D9	—	D9	—	D9
D8	regular	D8	—	D8	—	D8
D7	regular	D7	—	D7	—	D7
D6	regular	D6	—	D6	—	D6
D5	regular	D5	—	D5	—	D5
D4	regular	D4	—	D4	—	D4
D3	regular	D3	—	D3	—	D3
D2	regular	D2	—	D2	—	D2
D1	regular	D1	—	D1	—	D1
D0	regular	D0	—	D0	—	D0
$\overline{\text{PC\_CD1}}$	regular	—	—	CD1_B	—	—
$\overline{\text{PC\_CD2}}$	regular	—	—	CD2_B	—	—
$\overline{\text{PC\_WAIT}}$	regular	—	—	WAIT_B	—	—
PC_READY	regular	—	—	READY	—	—
PC_PWRON	regular	—	—	PC_PWRON	—	—
PC_VS1	regular	—	—	VS1	—	—
PC_VS2	regular	—	—	VS2	—	—
PC_BVD1	regular	—	—	BVD1	—	—
PC_BVD2	regular	—	—	BVD2	—	—
PC_RST	regular	—	—	RST	—	—
IOIS16	regular	—	—	IOIS16/WP	—	—

Table 4-13. EMI Signal Multiplexing (continued)

Contact Name	I/O Type	EIM	SDRAM SDR	PCMCIA	SDRAM DDR	NFC
$\overline{\text{PC\_RW}}$	regular	—	—	RW_B	—	—
PC_POE	regular	—	—	POE	—	—

4.5 Special I/O Signal Considerations

The following I/O lines should be connected as described in the following sections.

4.5.1 Power Ready Input (GPIO1_5)

The i.MX31/31L uses the Power Ready input as a qualifier when exiting state retention mode. The power ready input, GPIO1_5, should be connected to an external power management IC power ready output signal. If not used, GPIO1_5 must either be (a) externally pulled-up to NVCC1 or (b) a no connect, internally pulled-up by enabling the on-chip pull-up resistor. GPIO1_5 is a dedicated input and cannot be used as a general-purpose input/output.

4.5.2 SJC_MOD

SJC_MOD must be externally connected to GND for normal operation. Termination to GND through an external pull-down resistor (such as 1 kΩ) is allowed, but the value should be much smaller than the on-chip 100 kΩ pull-up.

4.5.3 CE_CONTROL

CE_CONTROL is a reserved input and must be externally tied to GND through a 1 kΩ resistor.

4.5.4 TTM_PAD

TTM_PAD is for Freescale factory use only. Control bits indicate pull-up/down disabled. However, TTM_PAD is actually connected to an on-chip pull-down device. Users must either float this signal or tie it to GND.

4.5.5 M_REQUEST and M_GRANT

These two signals are not utilized internally. The user should make no connection to these signals.

4.5.6 External DMA Signals (EXTDMA)

Input signals EXTDMA_0, EXTDMA_1, and EXTDMA_2 (also called EXTDMAREQ1, EXTDMAREQ2, and EXTDMAREQ3) are used as external DMA request event signals and are designed to trigger internal DMA transactions. This type of functionality is analogous to interrupt requests. These signals should not be used to trigger memory-to-memory DMA transactions. The SDMA script for

memory-to-memory transfers is not designed to clear the SDMA request event when the DMA transaction is complete.

To trigger SDMA transfers to/from the WEIM, ESDCTL, or NAND Flash Controller, the external DMA request signals should be configured as a GPIO interrupt, where the interrupt service routine initiates the memory-to-memory DMA transfer. The i.MX31/31L do not provide external bus mastership. DMA-style transactions on the external bus (those requiring both a DMA request and DMA grant or acknowledge) are not supported. These external request signals are accessed by invoking Alternate Mode 1 on GPIO1_0, GPIO1_1, and GPIO1_2.

4.5.7 Tamper Detect Logic

Tamper detect logic is used to issue a security violation. This logic is activated if the tamper detect input is asserted.

The tamper detect logic is disabled after reset. After enabling the logic, it is impossible to disable it until the next reset. The GPR[16] bit functions as the tamper detect enable bit.

GPIO1_6 functions similarly to other I/O with GPIO capabilities regardless of the status of the tamper detect enable bit. (For example, the GPIO1_6 can function as an input with GPIO capabilities, such as sampling through PSR or generating interrupts.)

4.5.8 Clock Source Select (CLKSS)

The CLKSS is the input that selects the default reference clock source providing input to the DPLL. To select CKIH, tie CLKSS to NVCC1. To select CKIL, tie CLKSS to ground. After initialization, the reference clock source can be changed (initial setting is overwritten) by programming the PRCS bits in the CCMR.

Chapter 5
General Purpose Input/Output (GPIO)

The General Purpose Input/Output (GPIO) module provides 32 bits of bidirectional, general-purpose input and output signals. Figure 5-1 presents a block diagram of the GPIO.

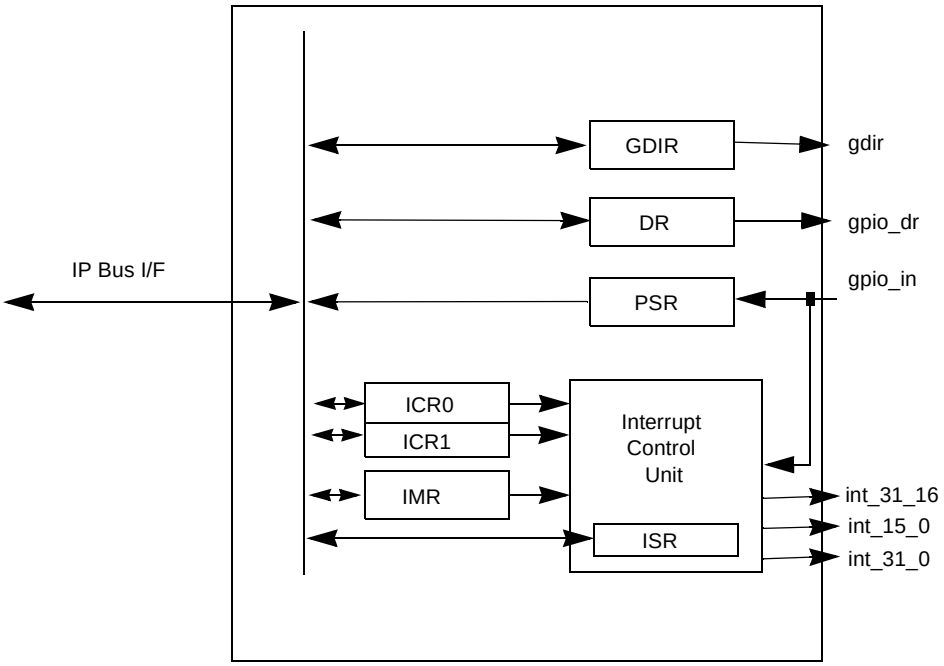


Figure 5-1. GPIO Block Diagram

5.1 Overview

The GPIO peripheral provides dedicated general-purpose pins that can be configured as either inputs or outputs. When configured as an output, you can write to an internal register to control the state driven on the output pin. When configured as an input, you can detect the state of the input by reading the state of an internal register.

The GPIO module is one of the modules controlling the IOMUX of the System on a Chip (SoC). Figure 5-2 shows the SoC muxing scheme.

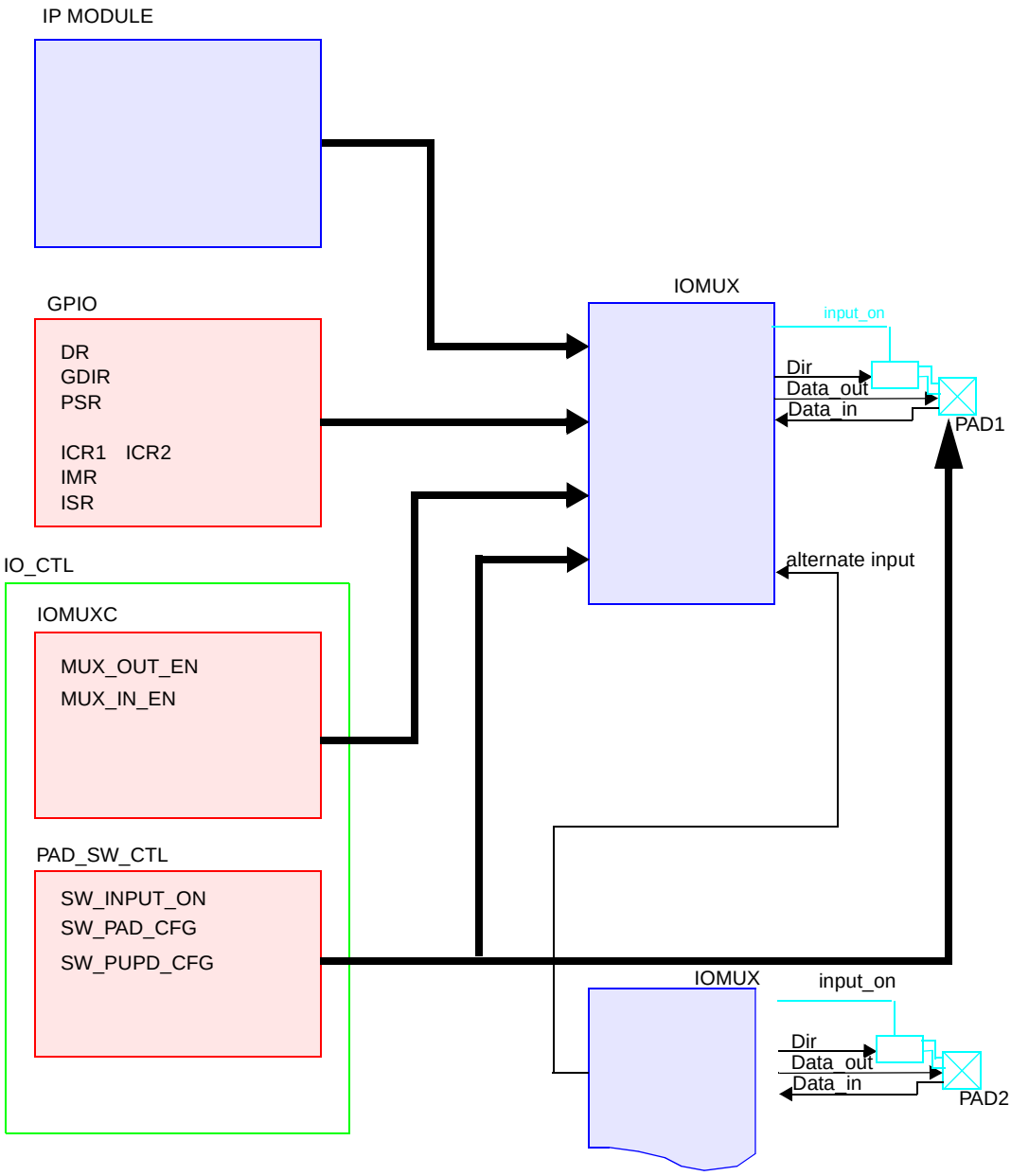


Figure 5-2. SoC IOMUX Scheme

The functionality is provided through seven registers, an edge-detect circuit, and an interrupt generation logic.

The seven registers include:

- DR—Data Register (32-Bit)
- GDIR—Data Direction Register (32-Bit)
- PSR—Pad Sample Register (32-Bit)
- ICR (ICR1, ICR2)—Two Interrupt Control Registers (2x32Bit)
- IMR—An Interrupt Mask Register (32-Bit)

- ISR—An Interrupt Status Register (32-Bit)

The data register is used to drive data from this register to I/O pins. Read access to the data register returns the value stored in the register or the value of the pad depending on the corresponding GDIR bit.

The data direction register controls the direction of the pin through the I/O multiplexer. Writing one to this register configures the pin as an output, while a zero configures it as an input.

Each GPIO input has a dedicated edge-detect circuit that can be configured through software to detect rising edges, falling edges, a logic low-level or a logic high-level on the input pin. Thirty two interrupts are supported. Two registers are used for this configuration, the interrupt configuration register (ICR1 and ICR2). Two bits are assigned to each GPIO pin, selecting one of the four possible detection methods.

The outputs of the edge detect circuits are optionally masked by setting the corresponding bit in the interrupt mask register (IMR). These qualified outputs are OR'ed together to generate three interrupt lines:

- interrupt_or_31_16 : 1 Bit Int OR of 16 High Int
- interrupt_or_15_0 : 1 Bit Int OR of 16 Low Int
- ipi_gpio_int : 1 Bit Int OR of 32
- ipi_gpio_int32 : 32 bit All Interrupt Out

5.1.1 Features

The GPIO includes the following features:

- General purpose input/output logic:
  - — Provides the ability to drive specific data to the pad using the DR register
  - — Provides the ability to control the direction of the pad using the GDIR register
  - — Enables the core to have the ability to sample the status of the corresponding pads by reading the PSR register
- GPIO interrupts:
  - — Provides the ability to support up to 32 interrupts
  - — Enables the ability to identify interrupt edges
  - — Generates three active high interrupts to the SoC interrupt controller

5.2 External Signal Description

This module has no usable external signals.

5.3 Memory Map and Register Definition

There are seven GPIO registers. All registers are byte-addressable and accessible from the IP interface.

5.3.1 Memory Map

Table 5-1 shows the GPIO memory map.

Table 5-1. GPIO Memory Map

Address	Register	Access	Reset Value	Section/Page
0x53FC_C000 GPIO1 (DR) 0x53FD_0000 GPIO2 (DR) 0x53FA_4000 GPIO3 (DR)	GPIO Data	R/W	0X0000_0000	5.3.3.1/5-6
0x53FC_C004 GPIO1 (GDIR) 0x53FD_0004 GPIO2 (GDIR) 0x53FA_4004 GPIO3 (GDIR)	GPIO Direction	R/W	0X0000_0000	5.3.3.2/5-7
0x53FC_C008 GPIOP1 (PSR) 0x53FD_0008 GPIOP2 (PSR) 0x53FA_4008 GPIOP3 (PSR)	GPIO Pad Status	R	0X0000_0000	5.3.3.3/5-8
0x53FC_C00C GPIO1 (ICR1) 0x53FD_000C GPIO2 (ICR1) 0x53FA_400C GPIO3 (ICR1)	GPIO Interrupt Configuration Register1	R/W	0X0000_0000	5.3.3.4/5-9
0x53FC_C010 GPIO1 (ICR2) 0x53FD_0010 GPIO2 (ICR2) 0x53FA_4010 GPIO3 (ICR2)	GPIO Interrupt Configuration Register2	R/W	0X0000_0000	5.3.3.5/5-10
0x53FC_C014 GPIO1(IMR) 0x53FD_0014 GPIO2 (IMR) 0x53FA_4014 GPIO3 (IMR)	GPIO Interrupt Mask Register	R/W	0X0000_0000	5.3.3.6/5-10
0x53FC_C018 GPIO1 (ISR) 0x53FD_0018 GPIO2 (ISR) 0x53FA_4018 GPIO3 (ISR)	GPIO Interrupt Status Register	R/W	0X0000_0000	5.3.3.7/5-11

5.3.2 Register Summary

The following definitions serve as a key for the GPIO register summary and individual register diagrams.

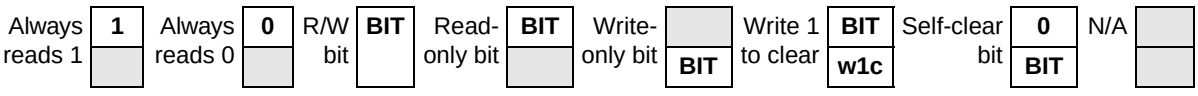


Figure 5-3. Key to Register Fields

Table 5-2 provides a key for register figures and the register summary table.

Table 5-2. Register Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writeable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
R	Read only. Writing this bit has no effect.
W	Write only.
R/W	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.

Table 5-2. Register Conventions (continued)

Convention	Description
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero (previously labeled slfclr).
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 5-3 shows the GPIO register summary.

Table 5-3. GPIO Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x53FC_C000 GPIO1 (DR) 0x53FD_0000 GPIO2 (DR) 0x53FA_4000 GPIO3 (DR)	R	DR[31:16]															
	W																
	R	DR[15:0]															
	W																
0x53FC_C004 GPIO1 (GDIR) 0x53FD_0004 GPIO2 (GDIR) 0x53FA_4004 GPIO3 (GDIR)	R	GDIR[31:16]															
	W																
	R	GDIR[15:0]															
	W																
0x53FC_C008 GPIOP1 (PSR) 0x53FD_0008 GPIOP2 (PSR) 0x53FA_4008 GPIOP3 (PSR)	R	PSR[31:16]															
	W																
	R	PSR[15:0]															
	W																
0x53FC_C00C GPIO1 (ICR1) 0x53FD_000C GPIO2 (ICR1) 0x53FA_400C GPIO3 (ICR1)	R	ICR1[31:16]															
	W																
	R	ICR1[15:0]															
	W																

Table 5-3. GPIO Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x53FC_C010 GPIO1 (ICR2) 0x53FD_0010 GPIO2 (ICR2) 0x53FA_4010 GPIO3 (ICR2)	R	ICR2[31:16]															
	W																
	R	ICR2[15:0]															
	W																
0x53FC_C014 GPIO1(IMR) 0x53FD_0014 GPIO2 (IMR) 0x53FA_4014 GPIO3 (IMR)	R	IMR[31:16]															
	W																
	R	IMR[15:0]															
	W																
0x53FC_C018 GPIO1 (ISR) 0x53FD_0018 GPIO2 (ISR) 0x53FA_4018 GPIO3 (ISR)	R	ISR[31:16]															
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
	R	ISR[15:0]															
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c

5.3.3 Register Descriptions

This section contains the detailed register descriptions for the GPIO registers.

5.3.3.1 GPIO Data Register (DR)

The GPIO DR register is a 32-bit register. Each bit stores a data to be driven to the pad at all times. If the IOMUX is in GPIO mode and the direction is output, this data will be driven there. If the direction is input, then a read action to DR bit reflects the value on the corresponded pad. Two wait states are required in read access for synchronization.

Reading of DR:

The data returned when reading the DR register is a function of the IOMUX input mode settings and the corresponding GDIR bit.

If GDIR == 1 && IOMUX input mode == GPIO: reading DR will return the content of the DR register

If GDIR == 0 && IOMUX input mode == GPIO: reading DR will return the pad’s value

If GDIR == 1 && IOMUX input mode != GPIO: reading DR will return the content of the DR register

If GDIR == 0 && IOMUX input mode != GPIO: reading DR will return zero.

Figure 5-4 shows the DR register, and Table 5-4 shows the register’s field descriptions.

0x53FC_C000 GPIO1 (DR)
0x53FD_0000 GPIO2 (DR)
0x53FA_4000 GPIO3 (DR)

Access: User read/write

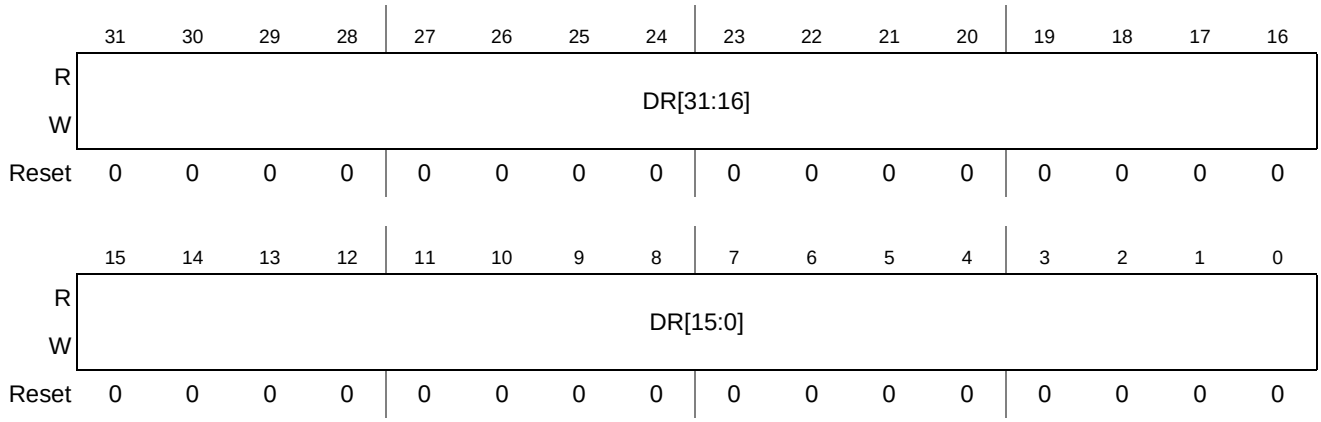


Figure 5-4. GPIO Data Register (DR)

Table 5-4. DR Field Descriptions

Field	Description
31–0 DR	Data bits. This register defines the value of the GPIO output when the pin is configured as an output (GDIR[n]=1). Writes to this register are stored in a register. Reading DR returns the value stored in the register if the pin is configured as an output (GDIR=1), or the state of the I/O pin if configured as an input (GDIR[n]=0). Settings: The I/O multiplexer associated with each bit must be configured for GPIO for the function to affect the state of the pin. Reading the data register with the input path disabled will always return a zero value.

5.3.3.2 GPIO Direction Register (GDIR)

The GPIO GDIR register is a 32-bit register that functions as direction control when the IOMUX direction is controlled by this bit. Each bit specifies the direction of a specific pad. The mapping of each DIR bit to a corresponding pad is determined on the SoC's pin assignment and IOMUX table. Figure 5-5 shows the GDIR register, and Table 5-5 shows the register's field descriptions.

0x53FC_C004 GPIO1 (GDIR)
0x53FD_0004 GPIO2 (GDIR)
0x53FA_4004 GPIO3 (GDIR)

Access: User read/write

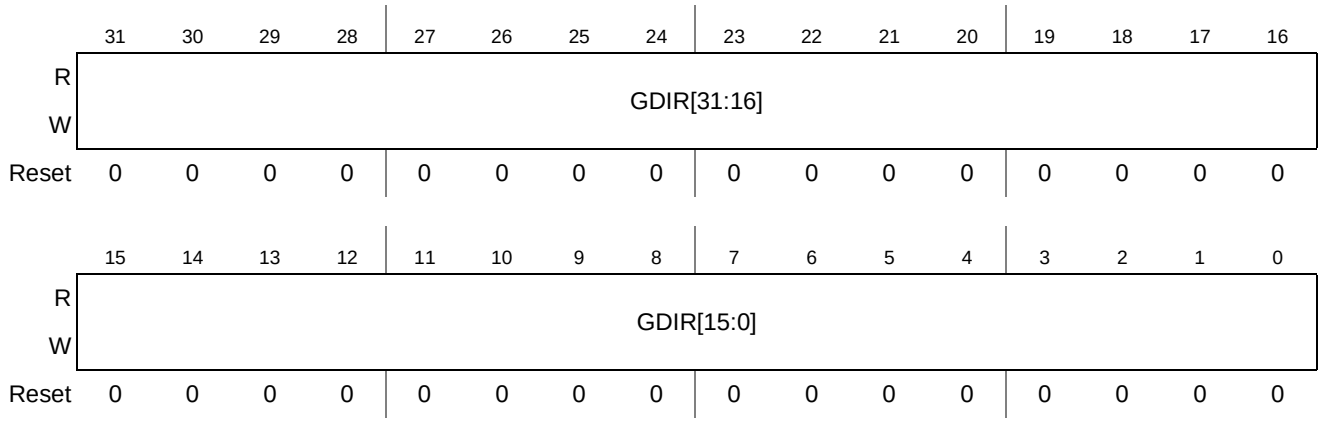


Figure 5-5. GPIO Direction Register (GDIR)

Table 5-5. GDIR Field Descriptions

Field	Description
31–0 GDIR	GPIO Direction bits. Bit i of this register defines the direction of the GPIO[i] signal. 0 GPIO is configured as input. 1 GPIO is configured as output. **Note:** GDIR affects only the direction of the I/O pin when the corresponding bit in the I/O MUX is configured for GPIO.

5.3.3.3 GPIO Pad Status Register (PSR)

The GPIO PSR register is a 32-bit read-only register. Each bit stores the value of the corresponding pad. This register is clocked with the ipg_clk_s clock, meaning that the value on the pad is sampled only when accessing this location. Two wait states are required any time this register is accessed for synchronization. Figure 5-6 shows the PSR register, and Table 5-6 shows the register's field descriptions.

NOTE

PSR[i]—pad sample register [i]

0x53FC_C008 GPIOP1 (PSR)
0x53FD_0008 GPIOP2 (PSR)
0x53FA_4008 GPIOP3 (PSR)

Access: User read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PSR[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PSR[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-6. GPIO Pad Status Register (PSR)

Table 5-6. PSR Field Descriptions

Field	Description
31–0 PSR	GPIO Pad Status bits (status bits). Reading PSR returns the state of the corresponding pad. Settings: The I/O multiplexer associated with each bit must be configured for GPIO for the function to affect the state of the pin.

5.3.3.4 GPIO Interrupt Configuration Register1 (ICR1)

The GPIO ICR1 register is a 32-bit register. Each set of 2 bits specifies the interrupt configuration for each corresponding interrupt line. There is total support for 16 interrupts. Figure 5-7 shows the ICR1 register, and Table 5-7 shows the register's field descriptions.

0x53FC_C00C GPIO1 (ICR1)
0x53FD_000C GPIO2 (ICR1)
0x53FA_400C GPIO3 (ICR1)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ICR1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ICR1[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-7. GPIO Interrupt Configuration Register1 (ICR1)

Table 5-7. ICR1 Field Descriptions

Field	Description
31–0 ICR1	Interrupt Configuration 1 bits. This register controls the active condition of the interrupt function for lines 15 to 0. Settings: interrupts (i) 0 to 15, when bits {ICR1[2*i+1],ICR1[2*i]} are as follows: 00 The interrupt i is low-level sensitive. 01 The interrupt i is high-level sensitive. 10 The interrupt i is rise-edge sensitive. 11 The interrupt i is fall-edge sensitive.

5.3.3.5 GPIO Interrupt Configuration Register2 (ICR2)

The GPIO ICR2 register is a 32-bit register. Each set of 2 bits specifies the interrupt configuration for each corresponding interrupt line. There is total support for 16 interrupts. Figure 5-8 shows the ICR2 register, and Table 5-8 shows the register's field descriptions.

0x53FC_C010 GPIO1 (ICR2)
0x53FD_0010 GPIO2 (ICR2)
0x53FA_4010 GPIO3 (ICR2)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ICR2[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ICR2[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-8. GPIO Interrupt Configuration Register2 (ICR2)

Table 5-8. ICR2 Field Descriptions

Field	Description
31–0 ICR2	Interrupt Configuration 2 bits. This register controls the active condition of the interrupt function for lines 31 to 15. Settings: interrupts (i) 0 to 15, when bits {ICR2[2*i+1],ICR2[2*i]} are as follows: 00 The interrupt i+16 is low-level sensitive. 01 The interrupt i+16 is high-level sensitive. 10 The interrupt i+16 is rise-edge sensitive. 11 The interrupt i+16 is fall-edge sensitive.

5.3.3.6 GPIO Interrupt Mask Register (IMR)

The GPIO IMR is a 32 bit register. Each bit is the interrupt masking bit for each interrupt line. Figure 5-9 shows the IMR register, and Table 5-9 shows the register's field descriptions.

0x53FC_C014 GPIO1(IMR)
0x53FD_0014 GPIO2 (IMR)
0x53FA_4014 GPIO3 (IMR)

Access: User read/write

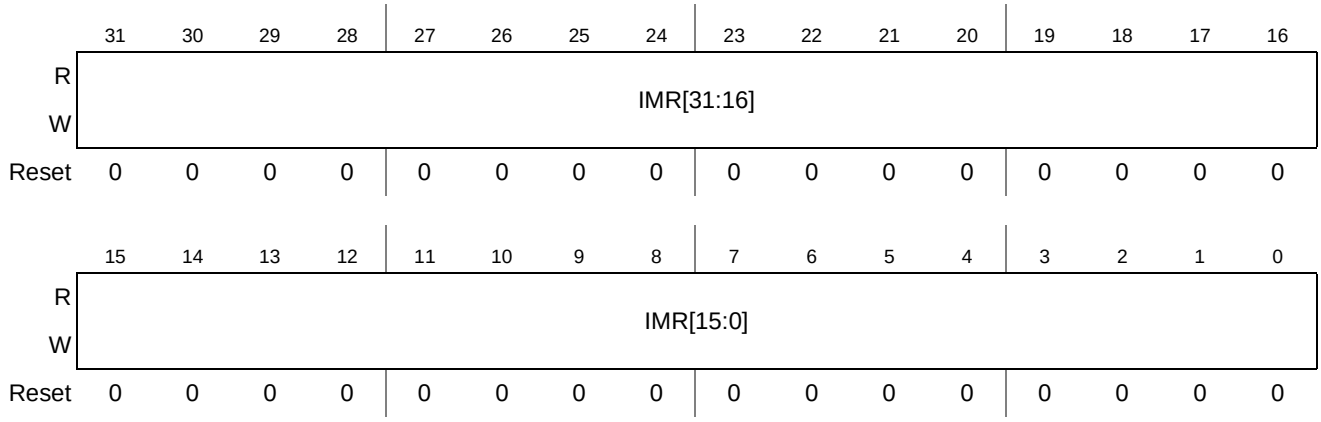


Figure 5-9. GPIO Interrupt Mask Register (IMR)

Table 5-9. IMR Field Descriptions

Field	Description
31–0 IMR	Interrupt Mask bits. This register is used to enable/disable the interrupt function on each of the 32 GPIO pins. Settings: For i from 0 to 31, when IMR[i] is as follows: 0 The interrupt i is disabled. 1 The interrupt i is enabled.

5.3.3.7 GPIO Interrupt Status Register (ISR)

The GPIO ISR is a 32- bit register that functions as interrupt. Each bit indicates whether an interrupt has occurred. When an interrupt event occurs, the bit in this register is set. The condition for setting of the bit is determined by the Interrupt Configuration Register (ICR) and the input that satisfies the configuration. Two wait states are required in read access for synchronization. One wait state is required for reset.

Figure 5-10 shows the ISR register, and Table 5-10 shows the register's field descriptions.

0x53FC_C018 GPIO1 (ISR)
0x53FD_0018 GPIO2 (ISR)
0x53FA_4018 GPIO3 (ISR)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ISR[31:16]															
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ISR[15:0]															
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-10. GPIO Interrupt Status Register (ISR)

Table 5-10. ISR Field Descriptions

Field	Description
31–0 IMR	Interrupt Status bits. Bit i of this register is asserted (active high) when the active condition is detected on the GPIO input and is waiting for service. The value of this register is independent of the value in the IMR register. When the active condition has been detected, the corresponding bit remains set until cleared by software. Status flags are cleared by writing a 1 to the corresponding bit position.

5.4 Functional Description

5.4.1 GPIO Function

A GPIO pin can operate as a general-purpose input/output when the IOMUX functions in GPIO mode. You can independently configure each GPIO pin as either an input or an output using the GPIO Direction Register (GDIR). When configured as an output (GDIR bit = 1), the value in the data bit in the GPIO Data Register (DR) is driven on the corresponding GPn pin. When configured as an input (GDIR bit = 0), the state of the input can be read from the corresponding PSR bit.

5.4.2 GPIO Programming

5.4.2.1 Read Value from Pad

Programming sequence should be as follows:

1. Configure IOMUXC to select GPIO mode.
2. Configure GPIO Direction Register to input.
3. Read value from Data reg*ister*/Pad Status register.

Pseudo code Description To Read [pad3:pad0] Values:

```
write sw_mux_ctl_<pad0>_<pad1>_<pad2>_<pad3> , 32'h00000000          // SET PADS TO GPIO MODE.
write GDIR[31:4,pad3_bit,pad2_bit, pad1_bit, pad0_bit,] 32'hxxxxxxx0 // SET GDIR TO INPUT.
read DR                                                             // READ PAD VALUE FROM DR.
read PSR                                                           // READ PAD VALUE FROM PSR.
```

NOTE

While GPIO direction is set to input (GDIR = 0), a read access to DR does not return DR data. Instead, it returns the PSR data, which is the corresponding pad value.

5.4.2.2 Write Value to Pad

Programming sequence should be as follows:

1. Configure IOMUXC to select GPIO mode.
2. Configure GPIO Direction Register to output.
3. Write value to Data Register (DR).

Pseudo code Description To Drive 4'b0101 on [pad3:pad0]:

```
write sw_mux_ctl_<pad0>_<pad1>_<pad2>_<pad3> , 32'h00000000          // SET PADS TO GPIO MODE.
write GDIR[31:4,pad3_bit,pad2_bit, pad1_bit, pad0_bit,] 32'hxxxxxxxF // SET GDIR TO OUTPUT.
write DR, 32'hxxxxxxx5                                              // WRITE PAD VALUE TO DR.
read_cmp PSR, 32'hxxxxxxx5                                       // READ PAD VALUE FROM PSR ONLY.
```

NOTE

While GPIO direction is set to output, you can verify real pad value only through PSR.

5.4.3 Interrupt Control Unit

In addition to the general-purpose input/output function, the edge-detect logic in the GPIO peripheral reflects whether a transition has occurred on a given GPIO signal that is configured as an input (GDIR bit = 0). The GPIO signal transition is reflected in the GPIO ICR registers. The GPIO ICR registers enables to configure each interrupt input to its sensitivity case (low-to-high transition; high-to-low transition; low; high).

The interrupt control unit is built of 32 interrupt control sub units. Each sub unit handles a single interrupt line.

Chapter 6
Debugging the i.MX31 and i.MX31L

The i.MX31 and i.MX31L debug features are the enablers for hardware/software (HW/SW) debug and validation of the silicon, either on an evaluation board, customer application board, or even a closed or opened radio device.

6.1 Overview

Debugging is used to identify and isolate causes of failure when running HW and SW in real applications. The source of failure could be the SW or HW (a race condition, for example).

The i.MX31 and i.MX31L debug hardware also supports system profiling. System profiling is used to improve overall system performance by identifying optimal system configurations.

Because of the multi-core nature of the i.MX31 and i.MX31L applications processors, all internal cores have their own dedicated debug features and ports to enable parallel debug of the MCU (ARM11) core and peripherals, the Smart Direct Memory Access (SDMA) core. The debug architecture of the i.MX31 and i.MX31L is therefore composed of the individual cores' debug components as well as shared debug components.

The aspects with which the individual cores share resources are as follows:

- The JTAG controller port, which is used to communicate with each of the multiple cores.
- The ECT module, which is used to control cross trigger events among the multiple cores.

In addition, secure JTAG options will be provided to protect debug resources from attacks by unauthorized users. The secure JTAG design will prevent the debug architecture from compromising security.

Figure 6-1 shows a block diagram of the i.MX31 and i.MX31L, along with the debug-related I/O.

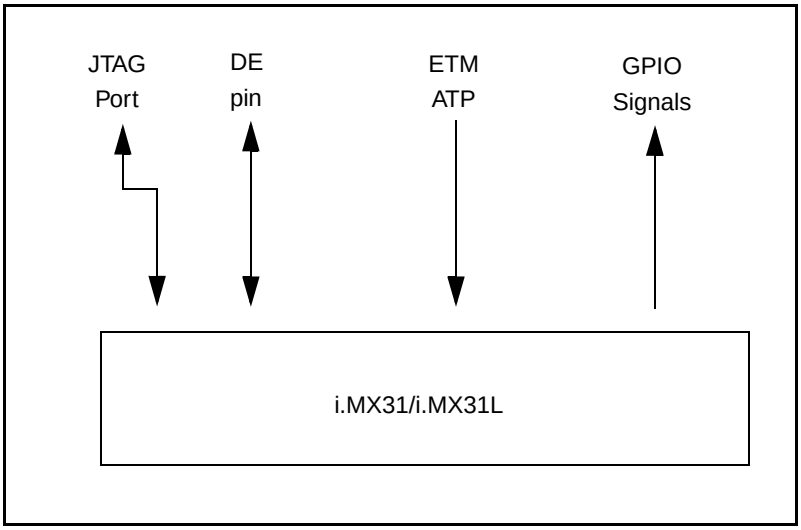


Figure 6-1. i.MX31 and i.MX31L Debug Port Scheme

6.1.1 Features

The i.MX31 and i.MX31L microprocessors provide a full set of features for multi-core debug. This section gives an overview of these features and the interconnections between different modules.

The overall debug system for the i.MX31 and i.MX31L is divided as follows:

- Multi-core debug support is provided by multi-core debuggers via the System JTAG Controller (SJC) and the extensive cross trigger support of the Embedded Cross Trigger (ECT) module.
- Static debug support is provided via the System JTAG Controller (SJC) and appropriate accesses to ICE/OnCE resources on the cores. Support is provided for debug start/stop, single-step, break points, access to CPU and system resources.
- Non-intrusive real-time instruction and data tracing is supported on the MCU (ETM11) processor.
- ROM patching is supported on the MCU processor. Patching provides the means to effectively replace data in a ROM memory location. It can also enable the SW to execute different instructions from those residing in ROM.
- Multiplexing of internal signals is possible using the IOMUX to chip IOs. Critical signals can be routed to the top-level SoC pads for external visibility.
- Limited time stamping support in the MCU domain is facilitated by use of three counters within the PMU of the ARM11 processor, used in conjunction with other resources such as the ETM and the ECT module.
- Performance profiling is supported for the MCU domain. For example, it is possible to count the number of processor stalls, L1 cache hits, L2 cache misses, or external memory accesses that have occurred. Profiling data is accessible by the SW and can be used to optimize system configurations for optimal performance.

The remainder of this chapter presents more details of the various debug partitions within the i.MX31 and i.MX31L:

- MCU domain (ARM11)
- SDMA domain
- System JTAG Controller
- I/O muxing scheme
- Embedded Cross Trigger configuration

6.2 AP Debug Support

This section describes the debug features specific to the applications processor ICs.

Figure 6-2 is a block diagram of the ARM11 Platform. Debug and performance profiling related components are indicated in blue dashed lines.

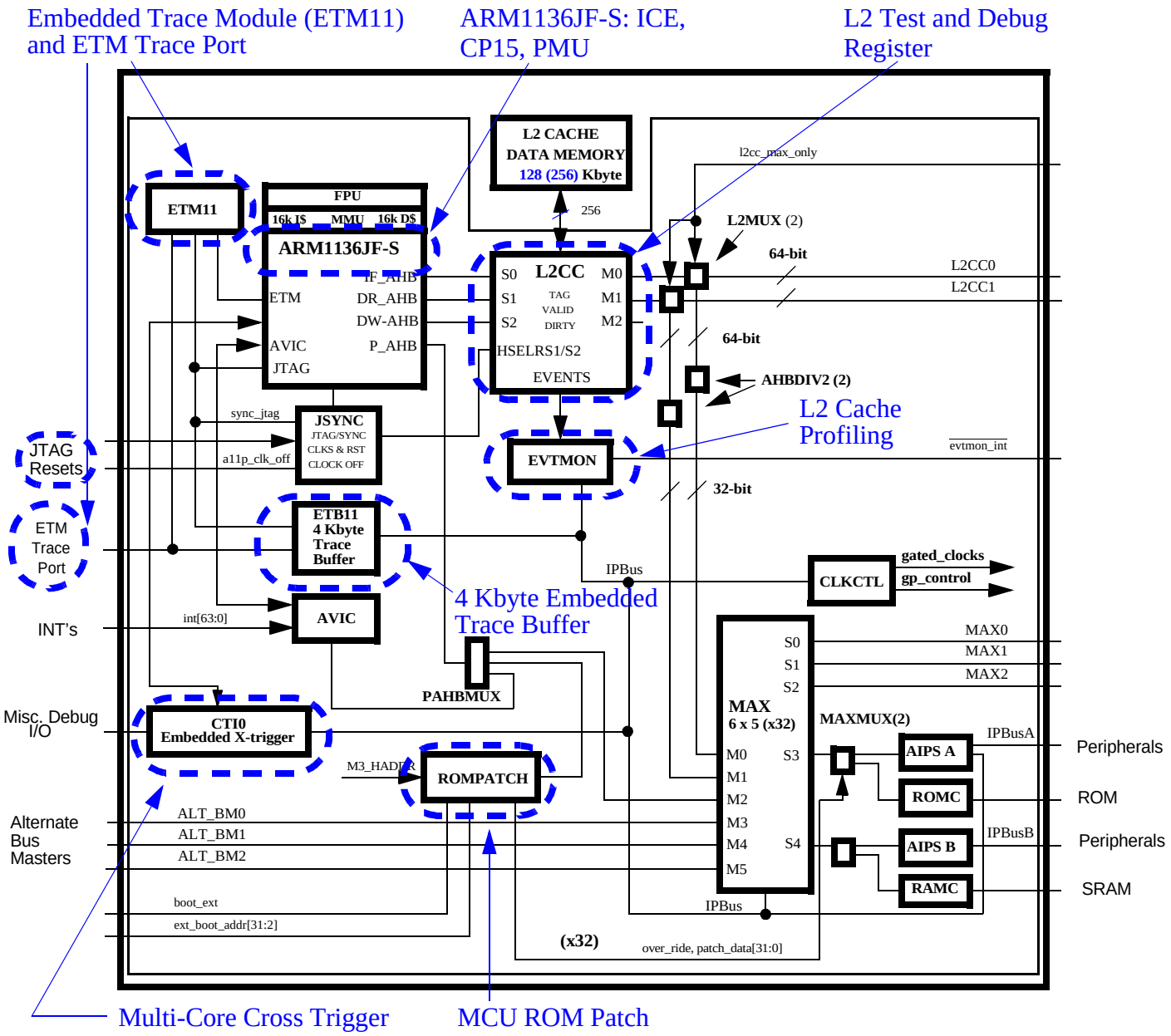


Figure 6-2. ARM11 Platform Debug Features (Highlighted in Blue)

All of the debug components shown can be accessed via JTAG or software running on the ARM11 core. The ETM is accessed through the coprocessor interface, and the ETB, ECT, and EVTMON modules reside on the IP Bus as memory mapped peripherals. The ROM Patch module resides on the peripheral AHB bus.

6.2.1 ARM1136JF-S

The ARM1136JF-S processor has robust support for debug and performance profiling, including:

- Dedicated ARM11 core, including ICE logic support by debuggers

- ICE-enabled ARM11 processor resource access; speeds system memory access
- Control of breakpoints and watchpoints
- Data communication channel between ARM core and host debugger via JTAG
- Control of debug status
- CP15 register for debugging the MMU, I, and D L1 cache, and TLB
- Performance Metrics Unit (PMU) used for system profiling and debug

6.2.1.1 PMU, L1 Caches, MMU, and TLB Debug Support via CP15 Registers

One of the first resources software programmers often use for debugging an ARM1136JF-S system are the processor's CP15 registers. The CP15 registers and associated logic have been designed specifically to support Memory Management Unit (MMU) debug, Translation Lookaside Buffer (TLB) debug and L1 instruction and data cache debug.

6.2.1.2 Performance Metrics Unit (PMU)

The Performance Metrics Unit (PMU) is part of the ARM1136JF-S processor. It includes four registers, which are implemented as part of the CP15 register:

- Performance Monitor Control Register (PMNC)
- Count Register 0, PMN0
- Count Register 1, PMN1
- Cycle Count Register, CCNT

The Performance Monitor Control Register controls the operation of all three counter registers. The PMN0 and PMN1 registers are 32-bit counters that can be programmed to count selected EVNTBUS (Table 6-1) occurrences. The cycle count register (CCNT) is a 32-bit counter used to count core clock cycles. All three counters can be enabled to generate interrupts on overflow.

System performance monitoring uses a series of system events to profile system performance. These events are described in Table 6-1.

Table 6-1. EVNTBUS Signals Used For Performance Modeling

Event Signal	Description
EVNTBUS[0]	Instruction cache miss to a cacheable location. This requires a fetch from external memory.
EVNTBUS[1]	Stall because instruction buffer cannot deliver an instruction. This could indicate an Instruction Cache miss or an Instruction MicroTLB miss. This event occurs every cycle in which the condition is present.
EVNTBUS[2]	Stall because of a data dependency. This event occurs every cycle in which the condition is present.
EVNTBUS[3]	Instruction MicroTLB miss.
EVNTBUS[4]	Data MicroTLB miss.

Table 6-1. EVNTBUS Signals Used For Performance Modeling (continued)

Event Signal	Description
EVNTBUS[5]	Branch instruction executed. **Note:** The branching action may or may not have changed program flow.
EVNTBUS[6]	Fixed at 0 in the current RTL revision.
EVNTBUS[7]	Branch miss-predicted.
EVNTBUS[8]	A valid instruction was executed (including either a branch phantom or a non-phantom instruction)
EVNTBUS[9]	Both a branch phantom and non-phantom instruction were executed in this cycle (for example, 2 instructions were executed).
EVNTBUS[10]	Data cache was accessed, not including internal cache operations. This event occurs for each non-sequential access to a cache line, for cacheable locations.
EVNTBUS[11]	Data cache access, not including cache operations. This event occurs for each non-sequential access to a cache line, regardless of whether or not the location is cacheable.
EVNTBUS[12]	Data cache miss, not including cache operations.
EVNTBUS[13]	Data cache Write-Back. This event occurs once for each half line of four words that are written back from the cache.
EVNTBUS[14]	Software changed the PC. This event occurs any time the PC is changed by software and there is not a mode change. For example, a MOV instruction with PC as the destination triggers this event. Executing an SWI from User mode does not trigger this event, because it incurs a mode change.
EVNTBUS[15]	Same as bit 14, but this bit indicates if it was an executed branch phantom
EVNTBUS[16]	Main TLB miss.
EVNTBUS[17]	External memory request (Cache Refill, Non-cacheable, Write-Through, Write-Back).
EVNTBUS[18]	Stall because the Load Store Unit request queue is full. This event takes place each clock cycle in which the condition is met. A high incidence of this event indicates the BCU is often waiting for transactions to complete on the external bus.
EVNTBUS[19]	The number of times the Write Buffer was drained because of a Drain Write Buffer command or Strongly Ordered operation. This signal will go high each time the write buffer needs to be drained.

The PMU counters can be used to generate an interrupt ($\overline{\text{pmu\_irq}}$). The PMU resources can then be accessed through an interrupt handler, and the resulting register reads can be made visible on the ETM11 trace port. One can also get periodic interrupts by using the cycle counter register of the PMU. When entering the interrupt routine, the core can read the counters, and using the address comparator, an ETM

packet can be generated that includes the PMU counter value. See Section 6.2.7, “Interrupts” for more details.

6.2.2 Embedded Trace Macrocell (ETM11)

The ARM1136 platform includes the ETM11 module, which supports real-time instruction and data tracing via ETM version 3.1. The ARM11 Platform enables access to the ETM registers via software, in addition to using the JTAG.

Tracing is controlled by specifying the exact set of triggering and filtering resources required for a particular application. Resources include four pairs of address comparators, two data comparators, eight address decoders and two counters, a three-stage sequencer, and four external inputs EXT_INT[3:0]. Three of those external triggers comes from the ECT/ARM Cross Trigger Interface (CTI) and one is accessible at the chip-level pads through IOMUX programing (for example, muxed with other signals on the pad).

The EXTIN inputs of ETM are connected to the CTI of the ARM11 Platform, but one can use an IO pad via the IOMUX to trigger the input of ECT and then trigger the CTI -> ETM through EXTIN.

The ETM includes two FIFOs (each holding 69 bytes) to store the compressed trace information. To prevent overflow, the user can program a FIFO full level register to behave as follows: When the FIFO is nearly full, the data trace is suppressed but instruction tracing is permitted to continue.

Two counters are located inside the ETM and their values can be traced. The counters can count events, including the EXTIN signals. The debugger tool can then reconstruct the global timing information. ETM can generate a debug request upon triggers.

ETM also has two outputs (EXTOUT) that are asserted to echo a trigger; those go to the CTI to activate or deactivate other debug functions in the i.MX31 and i.MX31L. These two bits are connected to the ARM11 core through the ECT to be monitored by the PMU.

The ETM also supports JAVA debug. The trace tool indicates when the core enters or exits JAVA state. The ARM tool then uses this information to calculate the percentage of time or cycles spent in and out of JAVA state, which enables one to quantify JAVA efficiency. In addition, the ARM11 “java_mode” signal can be muxed out to a chip pin for monitoring.

6.2.2.1 ETM11 Trace Port

The ETM11 module provides trace information to an external debugger through the ETM Trace Port interface at the SoC level.

The following signals are part of the ETM trace port interface:

- TRACECLK: clock signal, running at ARM_CLK/4 or ARM_CLK/8. Clock division is controlled by programming a register in the ETM.

NOTE

When ARM_CLK:TRACE_CLK ration is 1:1, TRACECLK is not used and the trace information automatically goes to the ETB.

- TRACEDATA[n-1:0]: data bus, bus width (n) is configurable by a programming register in the ETM. The maximum value of n in the i.MX31 and i.MX31L is 23.
- TRACECTL: control signal used in conjunction with TRACEDATA[0] to signal the debugger that a valid trace is available. See Table 6-2 for the decode information.

Table 6-2. Trace Signals Decode

TRACECTL	TRACEDATA[0]	Status
1	0	Trigger
1	1	Valid data trace
0	X	Do not capture current info

With ETM V3.1, data and control are updated on the rising and the falling edges of the TraceCLK signal so there is no real difference in terms of frequency of the CLK, DATA, and CTL signals; DDR is always enabled.

The ETM11 supports 3 port modes: dynamic, 1:2, and 1:4.

Dynamic mode is used for on-chip trace capture using the 4 Kbyte embedded trace buffer. When it is used, TRACECLK does NOT toggle, and the ETB11 should be setup to capture the trace data. The ETB11 captures the data on the rising edge of the ARM processor clock.

1:2 and 1:4 mode refer to the trace-port to arm clock ratio, *not* the TRACECLK to ARM clock ratio. Because the trace-port uses double data rate (clocking on both edges of TRACECLK) the TRACECLK ratio is actually twice the trace-port ratio (for example, 1:2 uses a 1:4 TRACECLK and 1:4 uses a 1:8 TRACECLK).

See Figure 6-3 and Figure 6-4 for trace port timing in 1:2 and 1:4 modes.

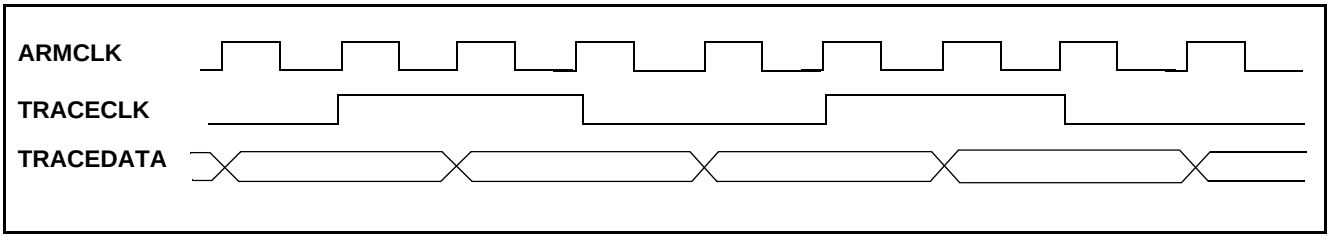


Figure 6-3. 1:2 Port Mode

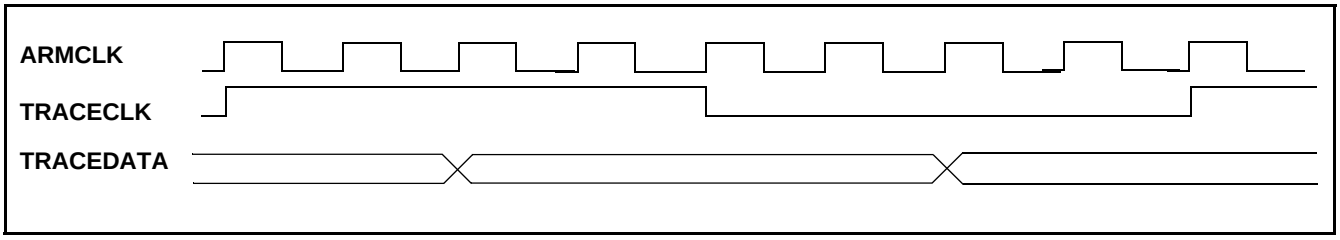


Figure 6-4. 1:4 Port Mode

6.2.3 Embedded Trace Buffer (ETB11)

The ARM11 Platform includes a 4-Kbyte embedded trace buffer (ETB11) to internally store the real-time trace data generated by the ETM11 module. The ETM writes 32 bits to the ETB at ARM11 processor clock speed. The ETB is memory mapped (IP Bus), but can also be accessed via JTAG. The memory array of the ETB is a compiled RAM. The ETB11 memory may be used by software as general purpose scratch memory when the ETM11 is not using it for debug. However, to prevent Endianness issues, only word accesses are enabled. Accesses to the ETB over the IP bus takes 7 hclk (‘per_clk’) clock cycles for write and read.

6.2.4 L2CC Debug Support

The L2CC design includes Register 15 for test and debug. This register enables the contents of the L2 cache to be read/written and forces specific behavior required for debug. The register supports mechanisms to force all cacheable accesses to be treated as write-through (no write-allocate) as well as preventing the cache from updating when performing a linefill on a miss.

6.2.4.1 ARM11 L2CC Event Monitor (EVTMON)

The L2CC event monitor (EVTMON) has profiling capabilities for the L2 cache. It is memory-mapped on IP Bus interface. It includes four 32-bit counters and 9 different events from the L2 cache are supported, as shown in Table 6-3. For more details, see Chapter 14, “L2 Cache Controller (L2CC).”

Table 6-3. ARM11 L2 Cache Events To EVTMON

Event Name	Description
BWABT	Buffered write abort
CO	Castout of a line from the L2 cache
DRHIT	Data read hit
DRREQ	Data read request
DWHIT	Data write hit
DWREQ	Data write request
IRHIT	Instruction read hit
IRREQ	Instruction read request
WA	Write allocate (write caused a linefill to the L2 cache)

An interrupt ($\overline{\text{evntmon\_interrupt}}$) can be generated on counter overflow conditions or on an increment. See Section 6.2.7, “Interrupts” for details.

There is no direct link from EVTMON counters to ETM but one can access the counter values through ETM by using interrupt routines. Refer to Section 6.2.1.2, “Performance Metrics Unit (PMU)” for more details.

6.2.5 Embedded Cross Trigger Interface (ECTCTI)

For details on the ECT module, see Section 6.3, "Embedded Cross Trigger (ECT)."

6.2.6 Debug Support Via Critical Signal Visibility

In addition to I/O signals required for functionality, several internal signals have been brought to the top-level of the ARM11 Platform specifically to support debug. These signals, along with any signal on the ARM11 Platform top-level, can be used by external logic or muxed out to an SoC pad, to gain visibility and assist in debugging the system. These signals are summarized in Table 6-4.

Table 6-4. ARM11 Platform Debug Centric Signal Visibility

Signal Name	Description	Visibility Details
standbywfi	ARM1136JF-S is in standby mode, waiting for an interrupt.	Visible on pads via OBS_INT register of IOMUXC
wfipending	ARM1136JF-S output to ETM on execution of Wait-for-Interrupt instruction. ETM must drain FIFO before shutting off clocks. standbywfi asserts after FIFO is drained.	
java_mode	Internal ARM1136JF-S J-bit (flopped by arm_clk)	
thumb_mode	Internal ARM1136JF-S T-bit (flopped by arm_clk)	
$\overline{\text{fiq}}$	AVIC fast interrupt to ARM1136JF-S	Not connected in the i.MX31 and i.MX31L
$\overline{\text{irq}}$	AVIC normal interrupt to ARM1136JF-S	Not connected in the i.MX31 and i.MX31L
$\overline{\text{evtmon\_interrupt}}$	L2CC Event Monitor Interrupt	Visible on pads via OBS_INT register of IOMUXC
etb_acqcomp	Embedded Trace Buffer Acquisition Complete	Connected to ECT, can be visible on pads via programming of ECT to either display ctm_lines or via cti_trig_out_1[5:2] connected to IOMUX.
etb_full	Embedded Trace Buffer Full	
$\overline{\text{pmu\_irq}}$	System Metrics Module Interrupt	
evntbus[19:0]	System Metrics Module Event Bus (refer to ARM1136JF-S Technical Reference Manual)	Visible on pads via alternate mode settings of IOMUX.

6.2.7 Interrupts

The ARM11 Platform has several interrupt outputs:

- $\overline{\text{pmu\_irq}}$
- $\overline{\text{evtmon\_interrupt}}$
- $\overline{\text{ect\_irq}}$[1:0]

Each of these interrupts are directly or indirectly tied to one or more of the $\overline{\text{a11p\_int}}$[63:0] inputs at the SoC level. In the i.MX31 and i.MX31L, the assignments are as follows:

Table 6-5. Interrupt Source Of Debug Signals

Signal Source	Interrupt Source
	i.MX31/i.MX31L
$\overline{\text{pmu\_irq}}$ and $\overline{\text{evtmon\_interrupt}}$	
$\overline{\text{ect\_irq}}$[0]& $\overline{\text{ect\_irq}}$[1]	

6.2.8 General Purpose Timer (GPT)

The i.MX31 and i.MX31L SoC includes the General Purpose Timer (GPT) module mapped to the MCU peripheral space. The GPT 32bit timer can be used during debug and profiling sessions according to debug/profiling specific needs. Programming of the GPT is done in debug session by executing ARM commands via JTAG interface.

6.3 Embedded Cross Trigger (ECT)

This section provides a summary of the Embedded Cross Trigger (ECT) debug scheme. An overview of the IP is given first, followed by specific i.MX31 and i.MX31L integration details. The Cross Trigger Interface (CTI) trigger signals are listed for the each CTI. Finally, some debug use cases using the ECT are presented.

6.3.1 ECT Overview

The ECT scheme is based on the ECT debugging hardware from ARM Ltd. The ECT is composed of three Cross Trigger Interfaces (CTIs) and one Cross Trigger Matrix (CTM). The ECT is key in the multi-core and multi-IP debug strategy. The outcome is a SW-controlled debug signal matrix that receives many signals from various sources (for example, cores and peripherals) and propagates/routes them to the different debug resources of the SoC. As seen in previous sections, those debug resources can include profiling capabilities, real-time trace (trace enabled or disabled), triggers, Soc level multiplexing, and debug interrupts.

The main advantages of using the ECT are as follows:

- Standardized debug scheme, in line with ARM RealView debugger, simplifies integration with ARM debug tools.
- Within a single debug domain, all the IPs can share the same debug resources. There is no need to duplicates counters or real-time trace resources. One trace port can be used with one tool to track the activity of the core and its peripherals.
  As the ECT should only be used during debug sessions, it is off (disabled) by default.

The ECT features are enabled by enabling the individual CTIs. It is only allowed in supervisor mode, after having presented a suitable key to the CTI's CTILOCK register (by writing 0x0ACCE550). Hardware input from the SJC module prevents use of the all CTIs based on the security mode of the IC. See Table 6-6 for details.

Table 6-6. ECT Functionality Depending on the i.MX31 and i.MX31L Security Mode

Security Mode	Security level	ECT Functionality
No Debug	Maximal	Use of ECT scheme is prevented by hardware.
Secure JTAG	High	Proper authentication of challenge-response sequence must be carried, prior to enabling the ECT.
JTAG Enable	Low	ECT is enabled by SW activation.
SCC JTAG	Un-secure	ECT is enabled by SW activation.

Figure 6-5 illustrates the construction of the ECT and how it interfaces with the CTI embedded in the ARM11 Platform.

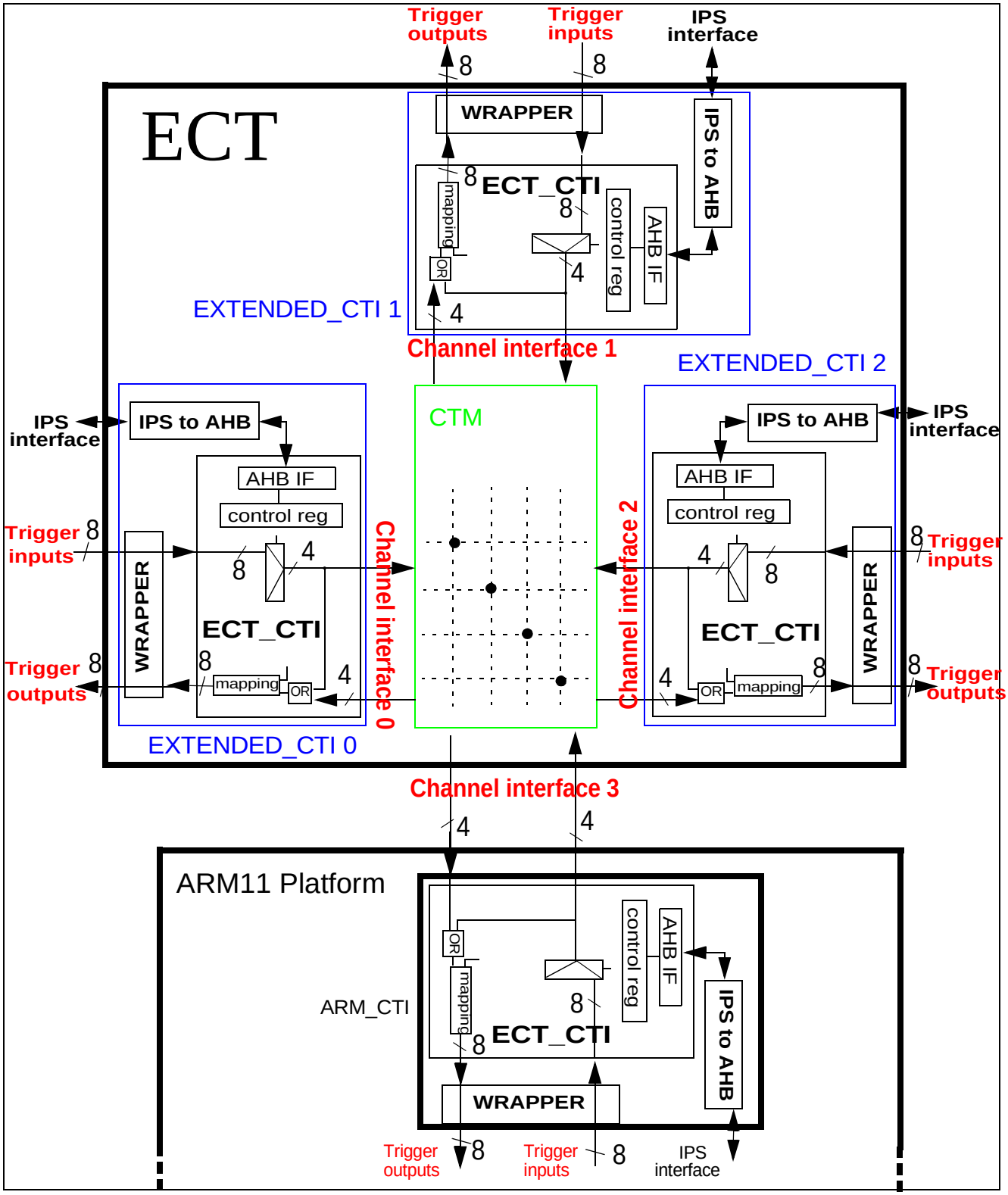


Figure 6-5. ECT Block Diagram

6.3.1.1 Cross Trigger Interface (CTI)

The CTI has eight inputs (trigger inputs) and eight outputs (trigger outputs) to the IC and four inputs and outputs to the CTM (channel triggers). The eight trigger inputs can come from a wrapper or from an IP or a core directly. Figure 6-6 shows the block diagram of a single CTI.

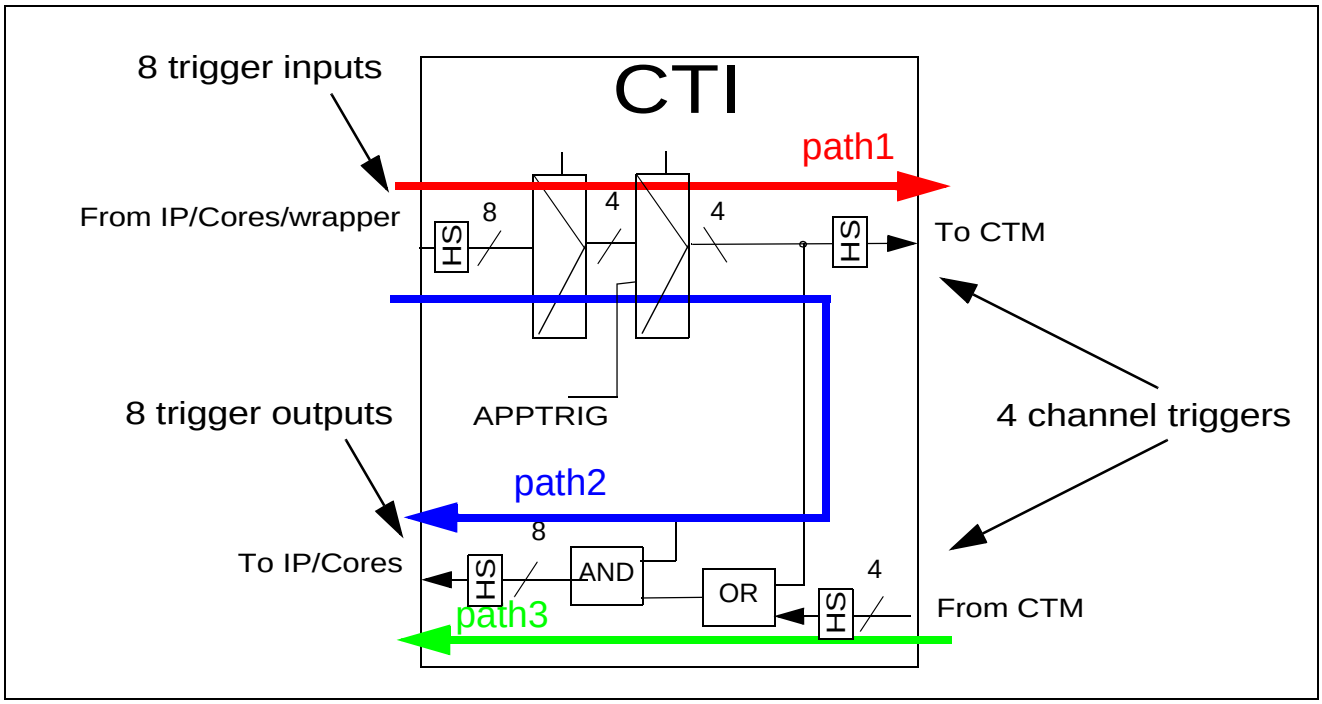


Figure 6-6. CTI Block Diagram

Each trigger input can be connected to any of the channel triggers to the CTM (path1) and/or propagate back to the eight trigger outputs (path2). The four channel triggers of the CTM to the CTI can be mapped to any of the eight trigger outputs of the CTI (path3). All of this is fully programmable using memory-mapped registers.

The muxing with APPTRIG shown on the input path is for software triggering support; by writing to some of the CTI registers, one can generate a trigger by software/assert one of the four channel triggers to CTM or/and one of the eight trigger outputs. A pulse can be generated by writing to the APPPULSE register or one can assert by writing to APPSET and then de-assert by writing to APPCLEAR.

The CTI supports four classes of trigger inputs:

- Conditioned—Keeps the output active until an acknowledge is received.
- Level/pulse—Used when the trigger destination is level-sensitive, the signal is active for only one clock cycle.
- Sticky—Trigger is active until the source is cleared.
- NOACK—Used when the output follows the input trigger and does not require an acknowledge.

The CTI includes handshaking/synchronization mechanism (represented as HS in Figure 6-6) on the eight trigger inputs/eight trigger outputs and 4 + 4 channel input and output triggers. This handshake mechanism is enabled at integration level to support the various classes of signals, clock domains, and frequencies.

6.3.1.2 Wrapper On CTI

The wrapper placed in front of the CTI is optional, depending on the nature of the signals connected to the input of CTI. The event signals in the ECT are all transmitted as a binary level, with “one” being active, and “zero” inactive.

The wrapper is used as a handshaking mechanism as well as for synchronization when necessary, especially in the case of multiple clock domains on trigger inputs or outputs or when timing is difficult to meet.

6.3.1.3 Cross Trigger Matrix (CTM)

The CTM logic serves as router logic between the system CTIs. The CTM consists of four interfaces of four inputs and four outputs each, referred to as the *channel triggers*.

Figure 6-7 presents the high-level CTM block diagram, while Figure 6-8 provides the logic details for the lower bit of the channel trigger signals.

The CTM includes handshake hardware (similar to the one used in the CTI) on its inputs and outputs to manage various clock domains, clock frequencies, and types of signals.

One can assign any trigger input to any trigger channel and to any trigger output. However, there are some timing and signal class considerations to take into account. For that reasons connectivity of ECT should follow the ECT design specifications. The ECT module provides additional visibility to the internal CTM logic. To track activity on each line, an OR4 function of the all four channel inputs is performed. These four system lines are available at the ECT boundaries for silicon debugging.

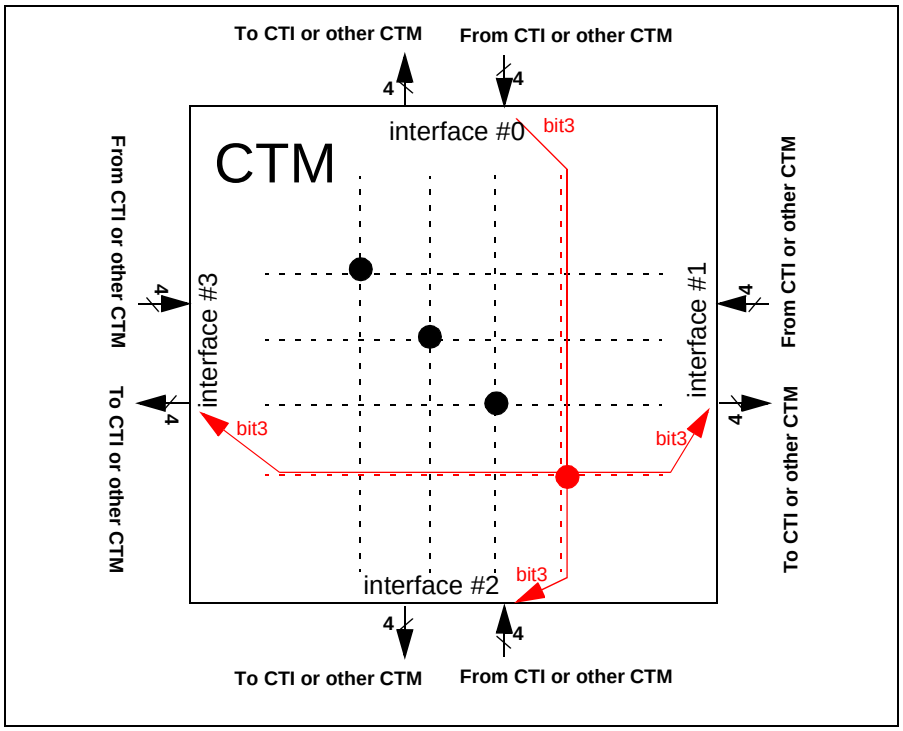


Figure 6-7. CTM Logic Bit Diagram

6.3.1.4 Clock Considerations

Each CTI has its own clock for its bus interface and memory-mapped registers but it also has its own clock for all the triggering. The CTM also has its own clock for handshaking logic (synchronization and flip-flop acknowledgement). Obviously one needs to properly use the wrapper/HS logic to manage the signals crossing those clock domains (including wrappers for CTI inputs and outputs).

When a processor clock is stopped (for example, waiting for an interrupt), the corresponding CTI can receive an event from the CTM. When the CTI clock is the same domain as the subsystem and the handshaking logic is not bypassed, the CTM keeps the signal active until an acknowledge is received (which only occurs when the clock is started again). In this situation, out-of-date events on the core can occur. This does not prevent the channel from being used by other processors.

However, if the CTI clock differs from the local processor clock (for example, gated differently), it is possible for the CTI to raise an event to the core using a trigger output while the processor clock is off. This generates a debug interrupt and wakes the core so the debug features connected to the CTI trigger outputs are available again.

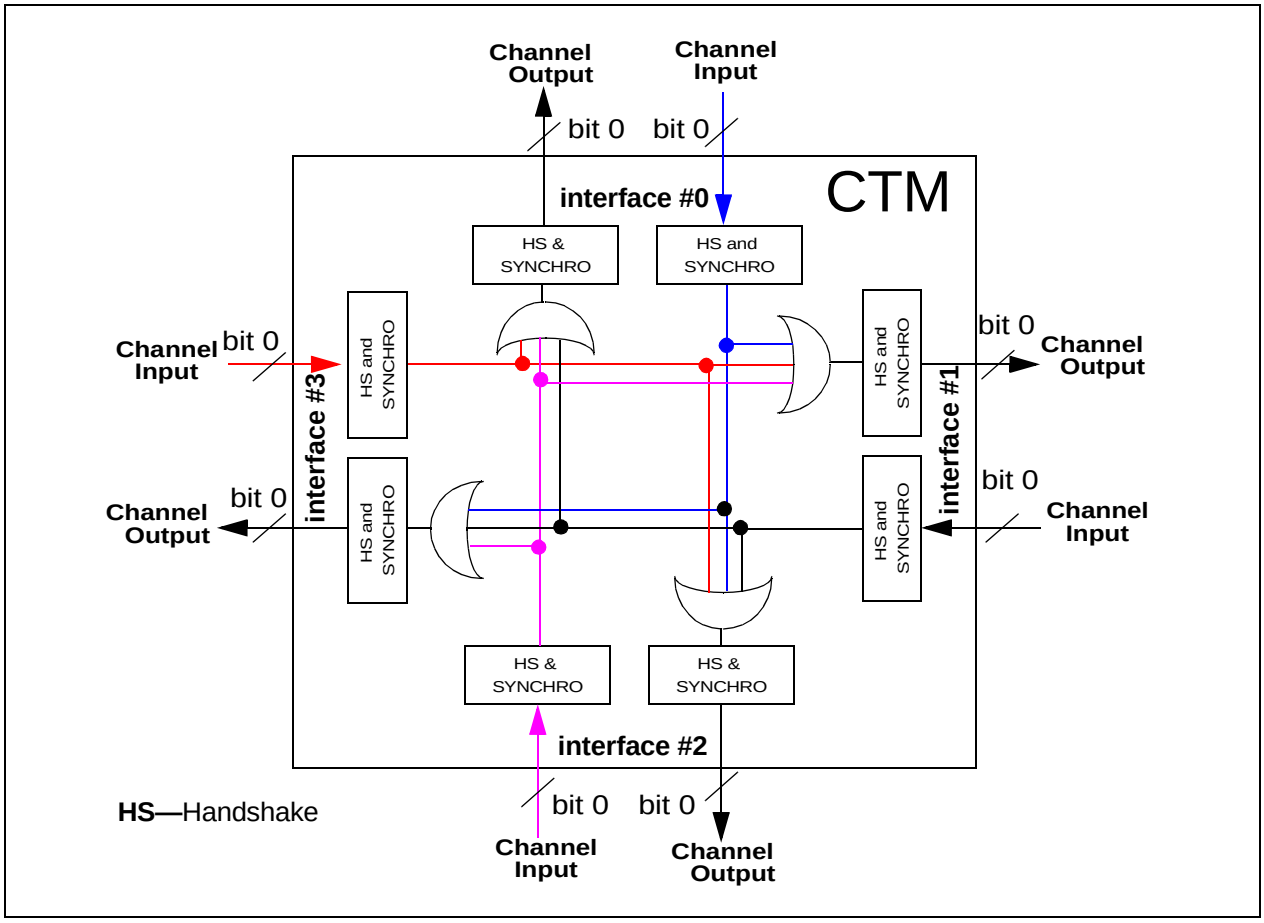


Figure 6-8. CTM Detailed Implementation for Channel 0

6.3.2 ECT Integration in the i.MX31 and i.MX31L

Figure 6-9 shows how the CTIs/CTM are connected to the other debug IPs. In the i.MX31 and i.MX31L, one CTI is located inside the ARM11 Platform while the other ECT components (including the CTM) reside in the ECT module (external to the ARM11 Platform). The ECT is connected in the following way:

- ECT CTI ‘0’ is used for SDMA signals. Its registers are memory-mapped to the ARM11 peripheral bus.
- ECT CTI ‘1’ (referred to as the MCU CTI) is used for ARM11 Platform peripherals. It is accessed via memory-mapped registers on the ARM peripheral bus.

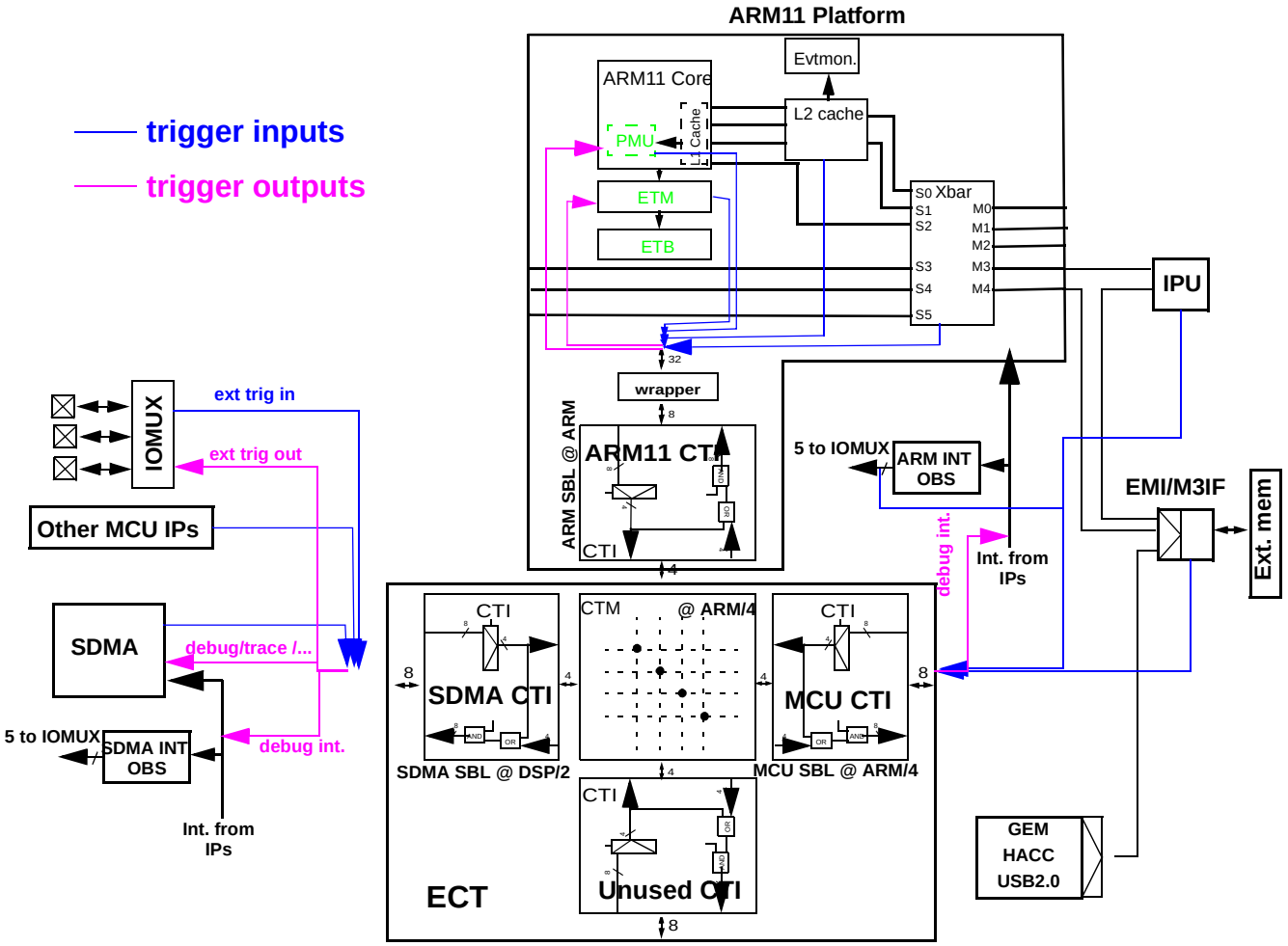


Figure 6-9. ECT in the i.MX31 and i.MX31L

6.3.2.1 CTI SJC Connectivity

In the i.MX31 and i.MX31L, debug signal routing is provided primarily by the ECT scheme. However, for simplicity of testing and debugging, an alternate way of propagating debug enable events (from DE_B pin) to the various cores exists in hardware. Figure 6-10 details the DE_B pin and debug request OR'ing logic in the SJC module.

The typical use case is setting the de_to_dsp, de_to_sdma, and de_to_mcu bits in the DCR register (a JTAG mapped register) of the SJC to pass the DE_B debug request event to one, two, or all three cores upon DE_B pin assertion. The following activities are possible:

- Enter debug and/or trigger CTI by the DE_B pad. In this mode, sjc_mode signal should be pulled low to allow DE_B input path.
- Enter debug by CTI trigger output. In this mode, sjc_mode signal goes high to strobe (active low) the debug lines inside the SJC module. sjc_mode also turns DE_B pin to output, and disable the DE_B input path from triggering debug events at the same time.

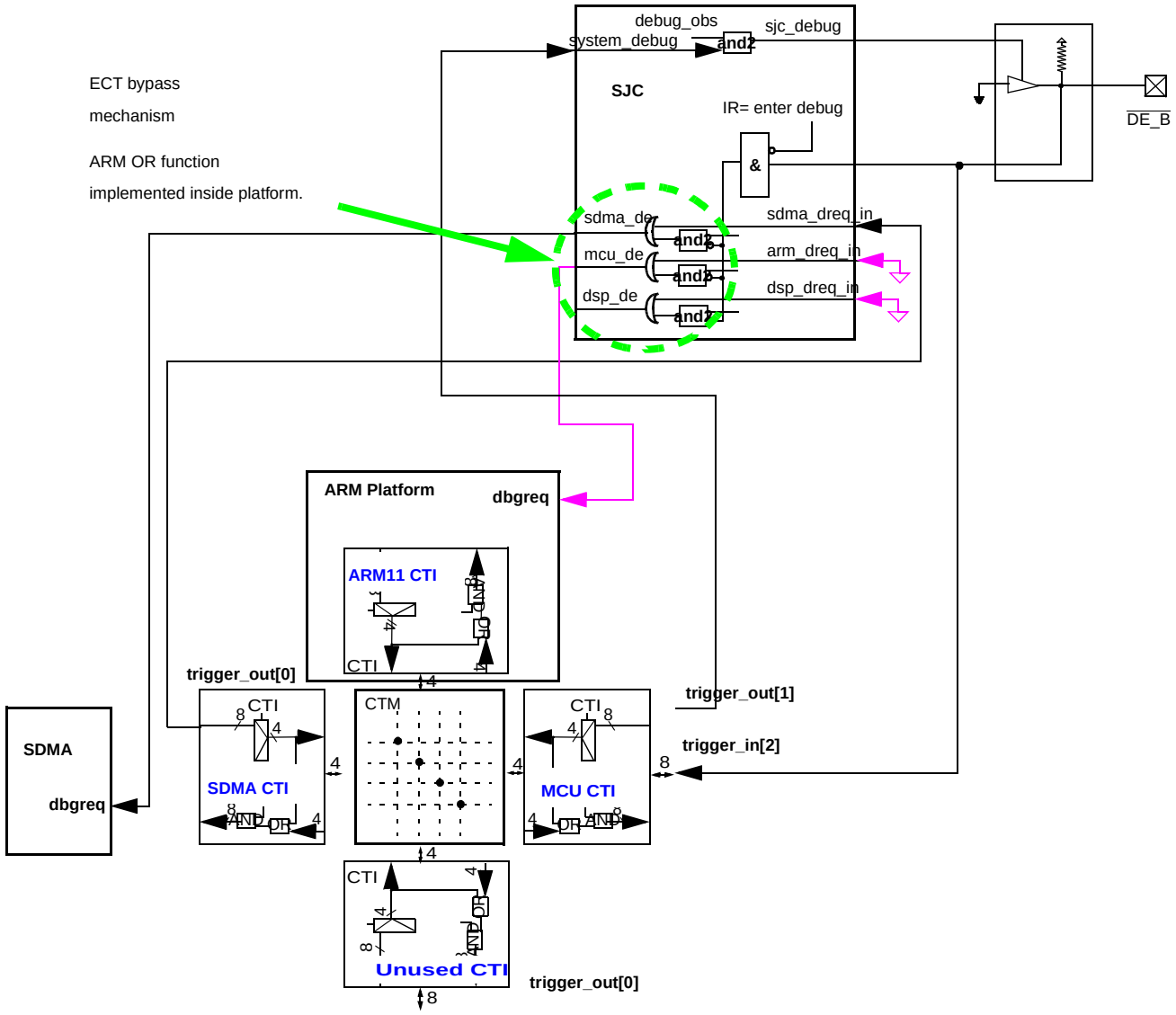


Figure 6-10. CTI-SJC Connectivity

6.3.3 Cross Trigger Input and Output Signals

Table 6-7 through Table 6-11 detail the CTI source connectivity. Signal directions are given with reference to the CTI modules.

6.3.3.1 ARM Cross Trigger Interface (CTI) Signal Assignments

Table 6-7 shows the ARM Cross Trigger Interface (CTI) signal input assignments.

Table 6-7. ARM CTI Signal Input Assignments

Trigger Input	Debug Resource Name	Channel Function	Comment
0	arm_dbgack	Debug Acknowledge	From ARM11
1	$\overline{\text{pmu\_irq}}$	PROFILING	System Metric interrupt request
2	etb_full	TRACING	ETB Buffer full
3	etb_acqcomp	TRACING	ETB Acquisition complete
4–5	extout[1:0]	TRACING	ETM11 external output, provides feedback to the ARM1136JF-S system metrics unit (allows EVNTBUS to be conditioned with ETM triggering facilities and fed back to the core)
6	etm_trigger	TRACING	ETM11 trigger event occurred
7	$\overline{\text{evtmon\_interrupt}}$	TRACING	L2CC event monitor interrupt request

Table 6-8 shows the cross trigger outputs for the ARM11 Platform's instantiation of CTI0.

Table 6-8. ARM CTI Signal Output Assignments

Trigger Output	Debug Resource Name	Channel Function	Comment
0	ect_edbgrq	DEBUG	Debug Request to ARM11
1	ect_etm_extin[2]	TRACE/DEBUG	External input to the ETM11. Used as a triggering mechanism
2	$\overline{\text{ect\_irq}}$[0]	TRACE/DEBUG	Interrupt request, can be connected to any platform interrupt input[1]
3	$\overline{\text{ect\_irq}}$[1]		
4	ect_etm_extin[0]	TRACING	External input to the ETM11. Used as a triggering mechanism.
5	ect_etm_extin[1]		
6	ect_arm_etmextout[0]	TRACE/DEBUG	Input to the PMU (performance monitoring unit) counter
7	ect_arm_etmextout[1]	TRACE/DEBUG	

[1] See interrupt tables for these signals connectivity.

6.3.3.2 SDMA Cross Trigger Signals (ECT CTI '0')

Table 6-9 shows the SDMA CTI input assignments.

Table 6-9. SDMA Cross Trigger Input Assignments

Trigger Input	Debug Resource Name	Channel Function	Comment
0	debug_mode	DEBUG	SDMA Debug acknowledge
1	debug_core_run	DEBUG	SDMA Run/sleep

Table 6-9. SDMA Cross Trigger Input Assignments (continued)

Trigger Input	Debug Resource Name	Channel Function	Comment
2	debug_evt_chain_lines[2:0]	DEBUG/TRACING	SDMA Trigger lines[1]
3			
4			
5	req_ma	DEBUG	SPBA status acknowledge (3): req_ma, req_mb and req_mc signals are asserted when a request (from a master: SDMA, ARM) is valid, and released when the request has been served. A request is determined to be valid when the SPBA address is valid and the peripheral access is authorized by the PRR (access registers).
6	req_mb	DEBUG	
7	req_mc	DEBUG	

1 SDMA Trig_line[7:0] can reflect any of the 32 events of SDMA event_bus and/or any of the 32 channels.

Table 6-10 shows the SDMA CTI output assignments.

Table 6-10. SDMA Cross Trigger Output Assignments

Trigger Output	Debug Resource Name	Channel Function	Comment
0	sdma_dreq_in	DEBUG	Debug request
1–2	Not-Used		
3	cti_trig_out_0[3]	SDMA event	Connected to SDMA channel #31
4–7	Not Used		

6.3.3.3 MCU Cross Trigger Signals (ECT CTI ‘1’)

Table 6-11 shows the MCU CTI input assignments.

Table 6-11. MCU Cross Trigger Input Assignments

Trigger Input	Debug Resource Name	Channel Function	Comment
0	ipu_mcu_dbgrq	DEBUG	Ext trigger inputs (IPU Debug Request): Connected to IPU Debug Request
1	ipi_int_epit_oc	DEBUG	EPIT #1 pgm clock (1): Periodic Interrupt Timer Interrupt generator. Active high signal.
2	$\overline{\text{ipp\_debug\_in}}$	DEBUG	$\overline{\text{DE}}$ pin input signal directly from Pad
3	Not Used		
4	cti_trig_in_1[4]	DEBUG	Input from PAD that connects to Visibility MUX #1 (See Chapter 4, “Signal Multiplexing”) Active low logic. Need to enable “loopback” in order for it to capture internal events.

Table 6-11. MCU Cross Trigger Input Assignments (continued)

Trigger Input	Debug Resource Name	Channel Function	Comment
5	cti_trig_in_1[5]	DEBUG	Input from PAD that connects to Visibility MUX #2 (See Chapter 4, "Signal Multiplexing") Active low logic. Need to enable "loopback" in order for it to capture internal events.
6	cti_trig_in_1[6]	DEBUG	Ext trigger inputs (4), (IOMUX): Connected to IOs via IOMUXs and allow monitoring of signals/events on the pads. Active high logic.
7	cti_trig_in_1[7]	DEBUG	

Table 6-12 shows the MCU CTI output assignments.

Table 6-12. MCU Cross Trigger Output Assignments

Trigger output	Debug resource name	Channel function	Comment
0	mcu_dbgack	DEBUG	MCU Debug Acknowledge—ARM debug acknowledge to IPU
1	system_debug	DEBUG	Connect to SJC system_debug input for passing interrupt to DE_B pin if pass enabled.
2	cti_trig_out_1[2]	DEBUG	External Trigger out (4, IOMUX): Connected to IOs via IOMUXs and allow monitoring of signals/events on the pads.
3	cti_trig_out_1[3]	DEBUG	
4	cti_trig_out_1[4]	DEBUG	
5	cti_trig_out_1[5]	DEBUG	
6	Unused		

6.3.3.4 Loopback of IOMUX Observability Signals to ECT

For extending the selection of trigger inputs using existing observability logic, the following scheme is implemented. If the loopback path is enabled in SW_PAD_CTL in IOMUXC then the input path is enabled, and in such a manner interrupts and DMA request lines can be used for generating ECT triggers. When loopback is disabled, the PADs can be used as trigger inputs driven from sources external to the i.MX31 and i.MX31L.

Figure 6-11 presents an overview graphic of the IOMUXC observability to ECT loopback logic.

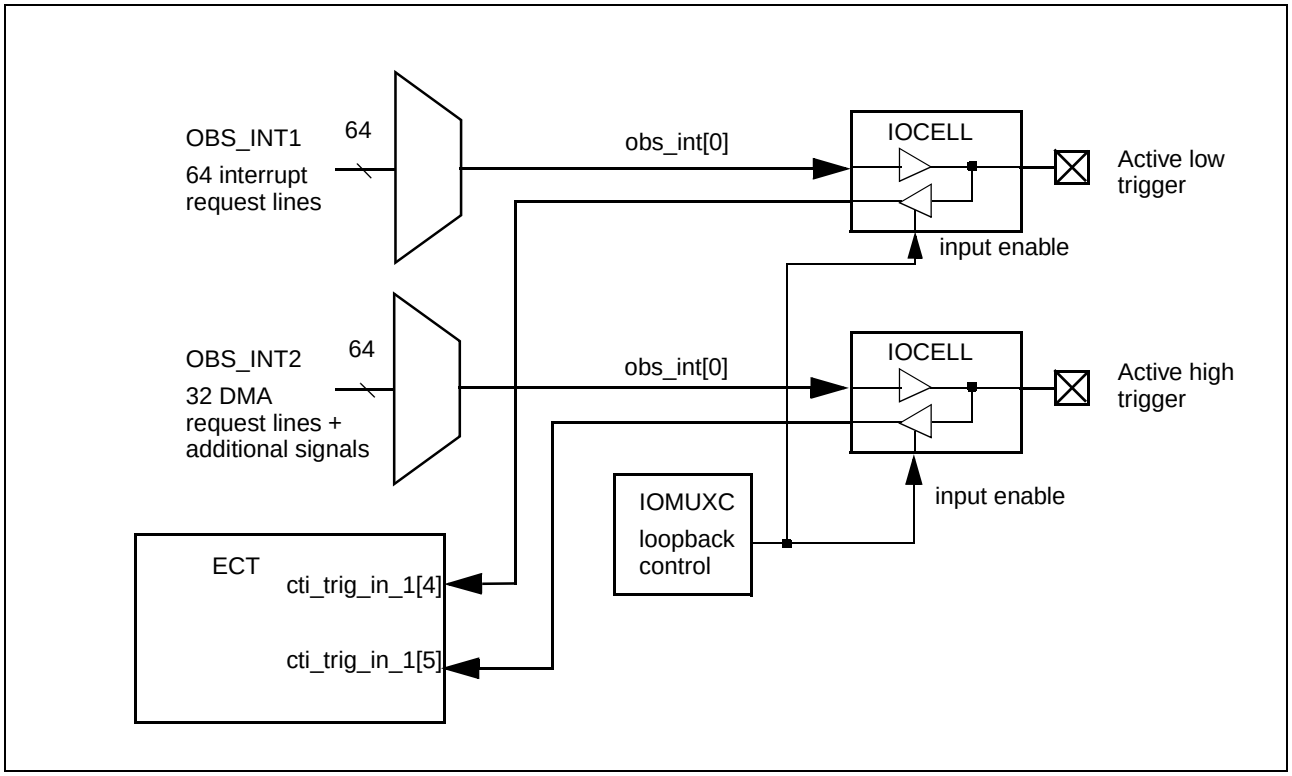


Figure 6-11. IOMUXC Observability To ECT Loopback Logic

6.3.4 Examples Of Debug Use Cases Using ECT Scheme

6.3.4.1 Debug Request/Debug Acknowledge

This section describes how the ECT is used with the System JTAG Controller (SJC) to handle all the debug requests and debug acknowledges to/from the cores, IP (debug request from the IPU), and the DE_B pad.

One of the first debug scenarios a user may want to put in place for the i.MX31 or i.MX31L is to place the SDMA and ARM into debug mode to access their ICE/OnCE.

Using the two debug IPs, this can be achieved in one of two ways: System JTAG Controller and Embedded Cross Trigger.

The System JTAG Controller controls the propagation of debug requests from DE_B to each of the cores and propagation of a system_debug signal from the Embedded Cross Trigger to the DE_B pad for debug observability.

6.3.4.2 SDMA Debug

NOTE

As the SDMA does not support any real-time trace, the ECT is used to provide access to real-time trace capabilities of the other cores (Nexus, ETM).

Eight trigger lines are provided at the SDMA boundaries and routed to the SDMA CTI. Due to pre-muxing before wrapper, one can choose which signals are applicable on the CTI. Each trigger line can either be mapped to a channel (channel_X_start) or to an event (event_Y) with correct programming of the registers within the SDMA.

Through the ECT, these signals can be traced for debug purposes.

For example, the ECIT can trace channel C activity depending on event E (for example, interrupt) on the ARM ETM.

6.3.4.3 IO Triggers

By using the ECT/SDMA CTI and pads properly setup with the IOMUX, one can observe a trigger on a pad. A trigger from MCU side can be observed, and the user can trigger the debug IP on the MCU side. Figure 6-12 shows IO triggers.

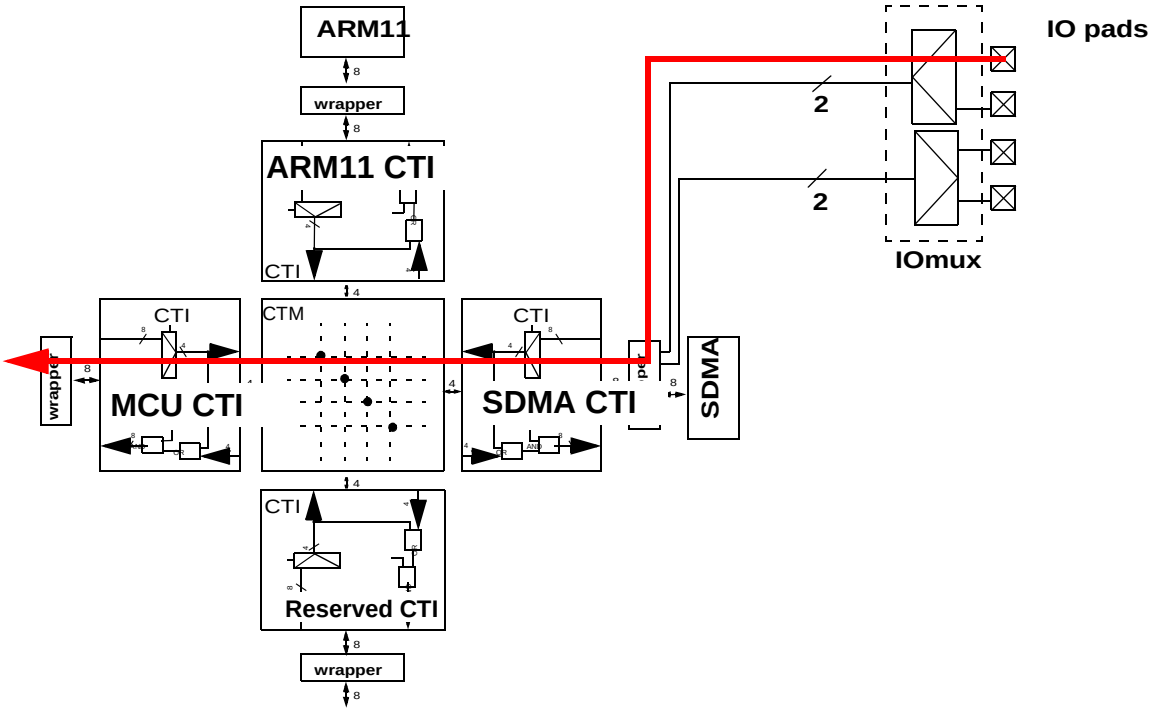


Figure 6-12. IO Triggers

6.3.4.4 Reconfiguration of the ECT

As only four channels are available in the CTM, only four different triggers can be managed separately at the same time (unless some of them can be OR'ed).

So an interesting use case is to reprogram the CTI during a debug session. This is intrusive but can be achieved quickly during a debug interrupt.

For example, one can start the debug session focusing on IP A on the MCU side by properly programming the MCU CTI to monitor IP A's input signals to the CTI. When IP A is done, then IP B processes the data

generated by IP A. An ‘IP A Done’ signal can be monitored through the CTI and used to generate a debug interrupt; when the interrupt is asserted, an interrupt routine reprograms the MCU CTI to now monitor the IP B’s signals.

6.4 System JTAG Controller (SJC)

The SJC module implements the IEEE1149.1 v2001 standard JTAG Test Access Port. The SJC supports the following:

- Access to the OnCE and ICE of each core
- Debug features to improve controllability and observability of the cores
- Manufacturing test features (special test modes, PLL bypass, memory BIST, burn-in, among others)

The SJC provides debug and test control with the defined security level. JTAG pins can be muxed to the bottom bus connector by the IO muxing logic.

See Chapter 4, “Signal Multiplexing” for more details on pin multiplexing. Figure 6-13 shows an overview of the SJC architecture.

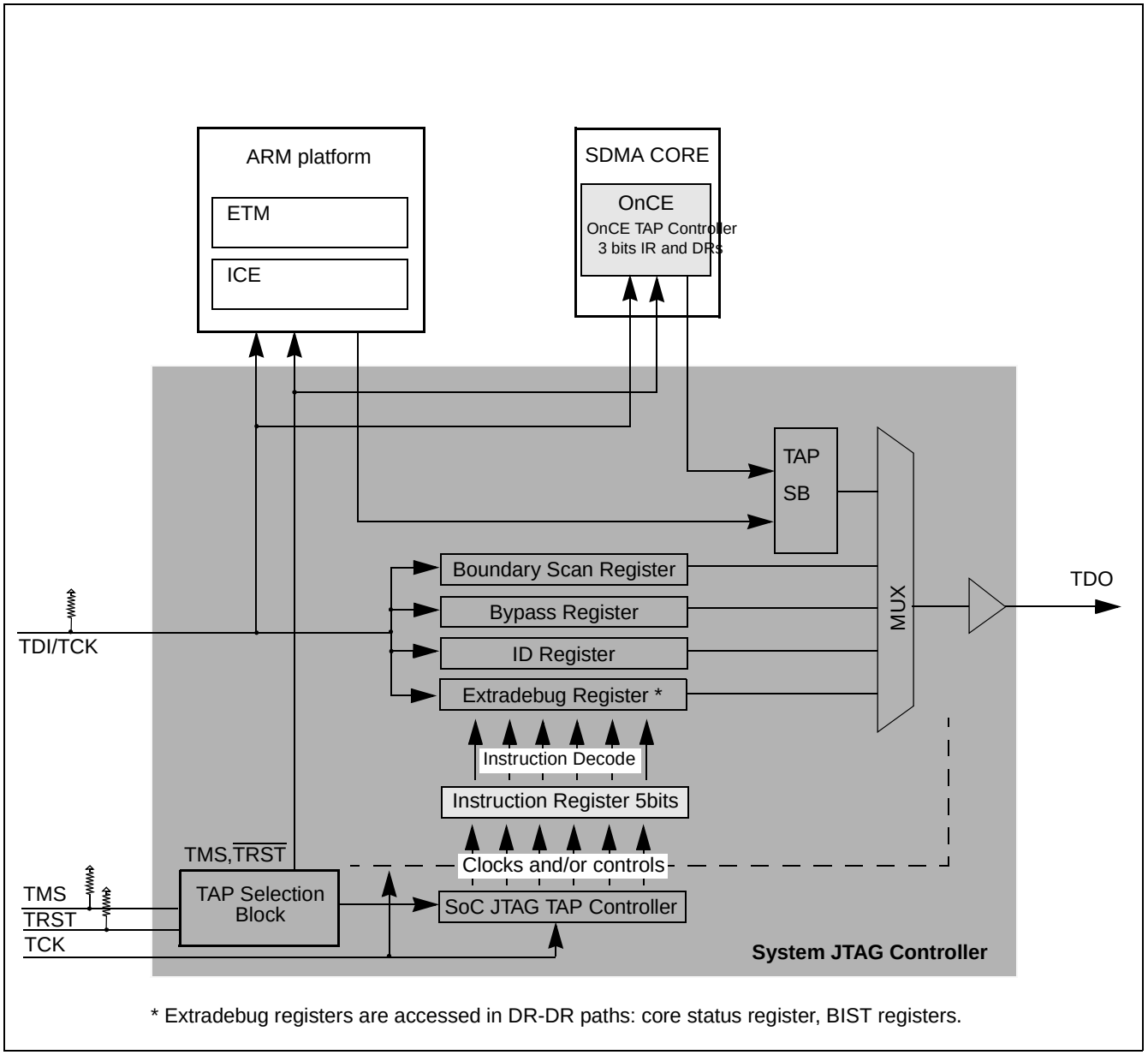


Figure 6-13. System JTAG Controller Block Diagram

6.4.1 SJC Main Features

The System JTAG Controller provides the following capabilities:

- Supports JTAG IEEE1149.1 mandatory instructions as well as optional ID_CODE and HIGHZ instructions
- Provides a means of accessing each OnCE/ICE TAP controller independently to control a target system.
- Provides means of accessing the ExtraDebug logic

- The System JTAG Controller operates at the higher rate possible of all cores in the chain. For example in normal operation (no core in low-power mode), this frequency will be 1/8 of the SDMA frequency if this core is present in the TDI-TDO chain (for example, serially connected with the other cores or standalone). The user also needs to take into account the 25 MHz frequency limitation on the CE Bus.
- Multi-core daisy-chained mode to support multi-core debuggers. The multi-core daisy-chained mode is the default configuration.

6.4.2 SJC TAP Port

The SJC in the i.MX31 and i.MX31L supports the following standard JTAG pins: TRSTB, TDI, TDO, TCK, and TMS with the addition of the RTCK (Return-TCK) pin as defined by ARM for utilizing the adaptive-clocking scheme. See Section 6.4.3, “Return-TCK (RTCK) Pin Support” for more details.

6.4.3 Return-TCK (RTCK) Pin Support

The i.MX31 and i.MX31L supports the Return-TCK pin (RTCK) as defined by ARM for using the “Adaptive clocking scheme.”

6.4.4 OnCE/ICE Accesses

Access to the OnCE/ICE is achieved using reserved IRs of the cores. Refer to Section 6.4, “System JTAG Controller (SJC)” for more information.

6.4.5 TAP Connectivity Scheme

The JTAG port by default has all cores’ TAPS controllers in a single daisy chain. Figure 6-14 presents this connectivity with the SDMA bypass TAP select mechanism, which controls whether the SDMA or the Alternate TAP are connected in the TDI-TDO chain.

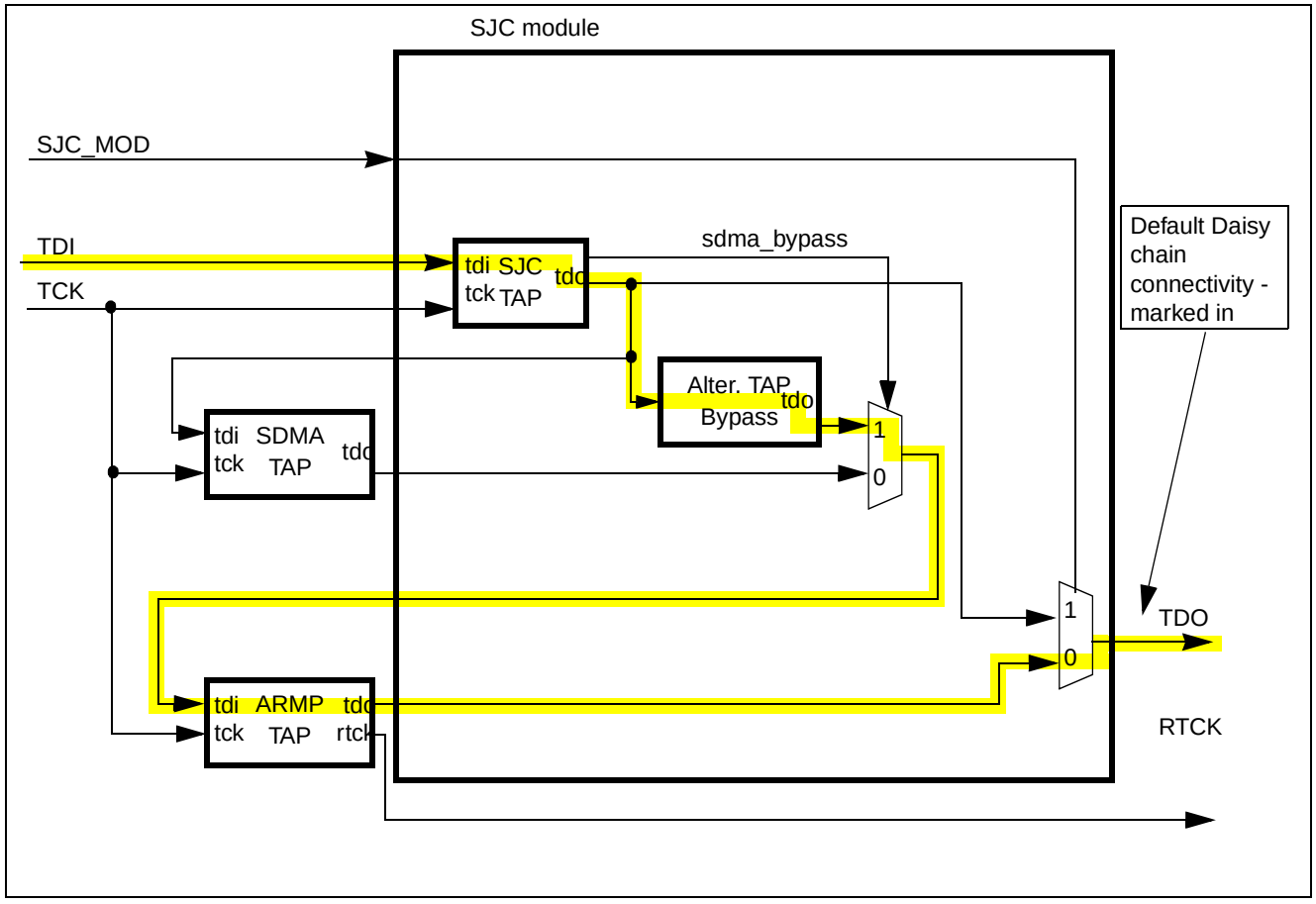


Figure 6-14. The i.MX31 and i.MX31L JTAG Default TAP Daisy Chain

6.4.6 SDMA TAP Bypass Mechanism

The TAP Select register controls the SDMA TAP connectivity, selecting between the SDMA TAP and an alternate/dummy TAP. This mechanism is used to enable high-speed TCK frequencies. The SDMA core has a design restriction of a 1:8 or greater ratio of the TCK and SDMA core clocks. Thus for a 66 MHz SDMA core clock, the TCK maximum frequency is limited to 8.25 MHz (= 66/8). In the case where TAPs are connected in a single daisy chain, this imposes a limit on the common TCK frequency of all the cores.

The outcome of this scheme, is that the number of cores in the chain stays the same whether the SDMA TAP is connected or bypassed, to not confuse the debuggers.

6.4.7 Out Of Reset Modes Of Operation

The System JTAG Controller (SJC) out-of-reset/wake-up mode is controlled by the input pin “sjc_mod”, which is sampled at TRST reset (JTAG reset). The “sjc_mod” port selects between the two connection modes of the TAPs, as specified by Table 6-13. Figure 6-15 presents the outcome connectivity based on the “sjc_mod” pin value.

Table 6-13. SJC Out Of Reset Modes

sjc_mod	Name	Description
0	Daisy chain ALL[1]	SDMA bypass is selected by default.
1	SJC only	IEEE 1149.1 JTAG compliant mode

[1] Although the SDMA TAP “pretends” to be in the chain, it is actually bypassed by the alternate TAP to allow faster a TCK frequency for debug.

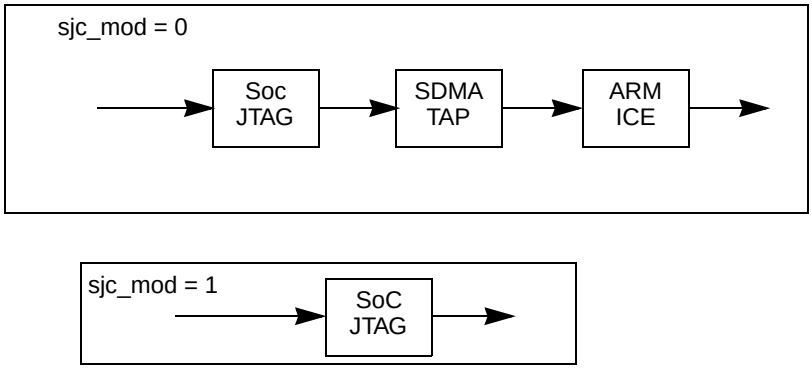


Figure 6-15. SJC Mode Selection Through Pin Sampling

6.4.8 Bottom Connector/CE Bus Support

The JTAG port is supported on the CE bus/bottom connector for closed radio debug, multiplexed with the UART, USB, and digital audio port. Using this feature, one can access OnCE/ICE, the SDMA trace buffer, and the ARM ETB. The ECT can also be programmed through the JTAG-CE bus.

NOTE

The RTCK (Return TCK) pin is not supported on CE bus.

CE bus supports both the TDI USBOTG specification with the new Freescale Power Management ICs, as well as legacy Power Management ICs (GCAP/PCAP), meaning that USB_DAT_VP and SUB_SE0_VM can be connected to either UART TxD or RxD. The same swapping exists for JTAG’s TDI and TDO pins. Pin sampling at reset (see Chapter 4, “Signal Multiplexing” for more details) is used to properly configure JTAG (in the case that MUXCTL is asserted, no software involvement is required) and UART on CE bus (through boot code/software programing of IOMUX).

NOTE

DE_B pad is dedicated for external debug request/acknowledge; there are no separate pads for debug_ack and debug_request.

Figure 6-16 shows bottom connector/CE supports.

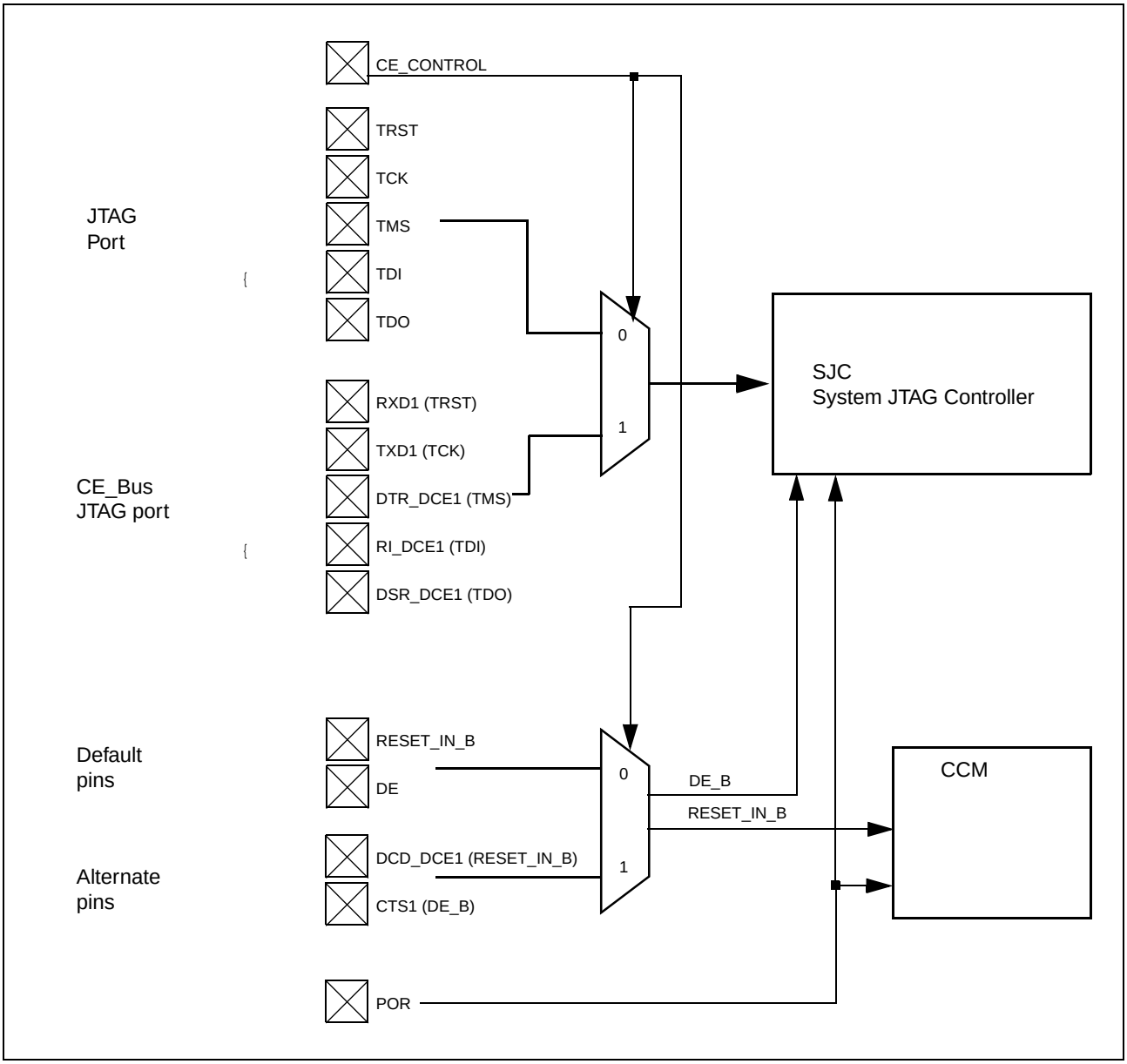


Figure 6-16. Bottom Connector/CE Supports

Chapter 7
i.MX31 and i.MX31L Boot

This chapter describes the system boot-up sequence for the i.MX31 and i.MX31L microprocessors.

7.1 Overview

The system boot-up is designed according to the configuration of external BOOT pins. The sequence is therefore divided into Boot-external, Boot-internal, Bootloader, and In-Factory security test modes.

The system boot flow defined in i.MX31 controls and assures that the system boots up in a pre-defined secure path and ensures that the software in the Flash ROM runs on the system for which it was intended. This helps prevent unauthorized modifications to the trusted OS. This also turns the Flash ROM into tamper-evident memory, protecting against unauthorized extraction of key(s) or downloads of hacking programs, as well as viruses.

The boot flow makes use of security hardware and software components to protect the flash image, and to create a well-bonded secure based platform for core and other functional modules. The boot components are shown in Figure 7-1.

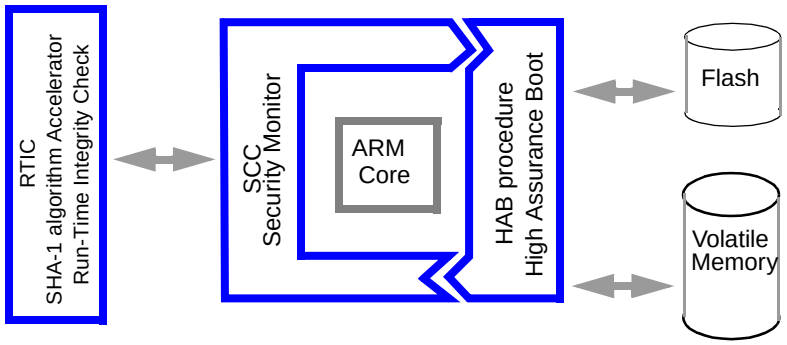


Figure 7-1. Boot Components

Figure 7-1 shows the boot flow components. The ARM core interfaces with the SCC and operates the HAB procedure. The HAB procedure is stored in Flash memory, and uses Volatile Memory for its operation. During boot, the RTIC is in operation, and performs its tasks.

7.1.1 Features

The following are boot features:

- System boots up in a pre-defined secure path
- Prevents unauthorized modifications to the trusted OS by ensuring that software in Flash ROM runs on the system for which it was intended

- Boot action effectively makes Flash ROM tamper-evident memory
- Protects against unauthorized accesses, against hacking, and against viruses
- Boot modes are divided into Boot-external, Boot-internal, Bootloader, and In-Factory security test

7.2 System Boot-Up Flow in i.MX31

Upon POR, boot pins are sampled. The state represented by the pins' values is stored in the CCM module RCSR register (refer to Chapter 3, "Clocks, Power Management and Reset (AP Clock Controller Module)"). The INT_BOOT fuse state is also sampled to check if an external boot is not restricted. If the fuse is blown (for example, external boot forbidden) the system begins execution of boot code starting from the address 0x0. Otherwise, the boot mode (reflected in BOOT[4:0]) is checked to determine if the core will execute from the iROM (boot internal), external memory (boot external), or if it will be booted for test purposes. Table 7-1 shows the boot execution addresses.

7.2.1 Supported Boot Modes

Table 7-1 lists the supported boot modes.

Table 7-1. System Boot Mode Selection

Inputs BOOT[4:0]	Output Signals Active Device	Boot Address	Comments	Type
00000	Bootloader USB/UART	32'h0000_0000	Via USB, UART and more	Internal
00001	8-bit NAND Flash (2 Kbytes per page)	32'h0000_0000		
00010	8-bit NAND Flash (512 bytes per page)	32'h0000_0000		
00011	16-bit NAND Flash (2 Kbytes per page)	32'h0000_0000		
00100	16-bit NAND Flash (512 bytes per page)	32'h0000_0000		
00101	16-bit CS0 at D[15:0]	32'h0000_0000		
00110–00111	Reserved	32'h0000_0000	Acts as bootloader UART/USB mode	
01001	M-Systems Disk On Chip	32'h0000_0000		
01001–01111	Reserved	32'h0000_0000	Acts as bootloader UART/USB mode	
10000	8-bit NAND Flash (2 Kbytes per page)	NANDFC base		External
10001	8-bit NAND Flash (512 bytes per page)	NANDFC base		
10010	16-bit NAND Flash (2 Kbytes per page)	NANDFC base		
10011	16-bit NAND (512 bytes per page)	NANDFC base		
10100	16-bit CS0 at D[15:0]	EMI base		
10101–10110	Reserved	External Memory	Acts as 16-bit CS0-External	

Table 7-1. System Boot Mode Selection (continued)

Inputs BOOT[4:0]	Output Signals Active Device	Boot Address	Comments	Type
10111	Reserved	N/A	Reserved for Manufacturing and Test	Internal
11xxx	Reserved	N/A	Manufacturing and Test	N/A

NOTE

If the INT_BOOT fuse of the IIM is blown, using an external boot (BOOT[4:0]=10000-10110) is forbidden, and any external boot mode, if selected, is ignored, and an internal UART/USB boot is performed instead.

7.3 Endian Boot Mode

The i.MX31 and i.MX31L boot in Little Endian mode. It is possible to boot the system in Big Endian mode *only for the external boot modes* (for example, BOOT[4:0] = 10000-10110). For these modes, the Endianness is configured by the BIG_ENDIAN fuse of the IC Identification Module (IIM). If this fuse is intact, the system will boot in Little Endian mode, if the fuse is blown, the system will boot in Big Endian mode.

When booting from external memory, the BIG_ENDINIT input of the ARM platform is driven by the logic described in Figure 7-2. If the BIG_ENDIAN fuse is blown, the ARM starts execution in Big Endian mode, otherwise it starts execution in Little Endian mode.

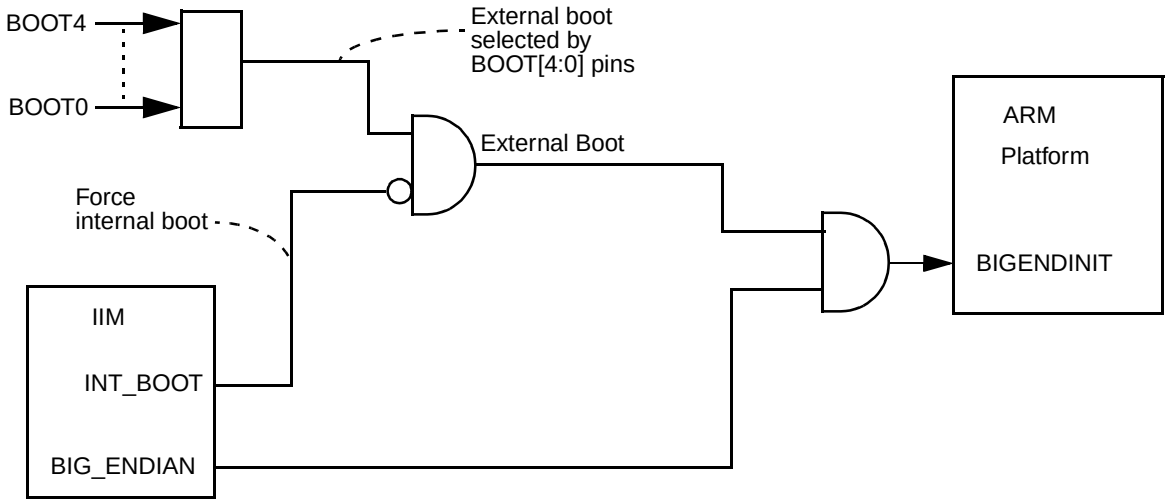


Figure 7-2. External Boot Endian Mode Logic

7.4 Special Boot Cases

This section discusses the special cases relevant to boot.

7.4.1 Development Parts

7.4.1.1 Use of RAM Loader to Download Flash Image onto System Flash

Upon POR, the system starts bootcode execution at address 0x0. It first reads the fuse HAB_TYPE as 001 to confirm it is a Development Part and BOOT[4:0] is set as 00000, indicating USB/UART boot loader. Boot loader code first checks the i.MX31 and i.MX31L security hardware and proceeds with all required initialization. Then it calls the bootloader code to download the required flash image for flashing. Boot loader code establishes a PC communication channel that configures SDRAM for download. After the SDRAM is set, flash loader and target flash images are download into SDRAM. Boot loader code then calls HAB to validate the flash loader and executes the flash loader to move the target flash image into system flash. Any errors, such as security hardware initialization failure or flash loader or image validation failure could be reported via the PC communication channel, but such errors will not stop system operation as it is a Development part.

7.4.1.2 iROM System Flash Bootup

Upon POR, the system begins execution of bootcode from address 0x0. It first reads the HAB_TYPE from the fuse to confirm it is a Development part (HAB_TYPE=001), then it reads BOOT[4:0], which indicates the kind of flash memory used and the starting address. Again, bootcode first checks the i.MX31 and i.MX31L security hardware and proceeds with all required initialization. It then calls HAB to validate the flash image. After it is validated, bootcode jumps to the starting address of the flash and continues to execute the target image. Any validation failure is ignored as it is a Development part.

7.5 High Assurance Boot (HAB)

The system boot flow defined in the i.MX31 and i.MX31L ensures that the system boots up in a pre-defined secure path and that the software in Flash ROM runs on the system for which it was intended. This helps prevent unauthorized modification to the trusted OS. This turns the Flash ROM into tamper-evident memory, protected against unauthorized key(s) extraction or hacking program downloads as well as viruses. The flow makes use of security hardware and software components to protect flash image, and to create a well-bonded secure based platform for core and other functional modules.

NOTE

Contact your Freescale Semiconductor sales office or distributor for additional information on HAB.

Book II:
Applications Processors' Core and Peripherals

Introduction

Book II comprises detailed information on the applications processors' core and peripherals. Book II includes the following chapters.

Book II, Part 1: ARM11 Core and Interrupts

Book II, Part 2: Security

Book II, Part 3: Memory Systems

Book II, Part 4: External Interfaces

Book II, Part 5: Connectivity Peripherals

Book II, Part 6: Timer Peripherals

Book II, Part 7: System Control Peripherals

Book II, Part 8: Multimedia Peripherals

Book II, Part 1:
ARM11 Core and Interrupts

Introduction

This part provides an overview of the modules that make up the ARM11 core and interrupts.

Chapter 8, “ARM11 Platform,” on page 8-1

Chapter 9, “ARM1136JF-S Vectored Interrupt Controller (AVIC),” on page 9-1

ARM11 Platform

The ARM1136JF-S processor platform is composed of a combination of intelligent integrated peripherals and an advanced processor core that meet the processing and power management needs of the i.MX31 and i.MX31L applications processors.

The ARM1136JF-S incorporates an integer unit that implements the ARM v6 architecture. It supports the ARM and Thumb instruction sets, Jazelle technology to enable direct execution of Java bytecodes, and a range of SIMD DSP instructions that operate on 16-bit or 8-bit data values in 32-bit registers. The ARM11 Platform also includes a vector floating point coprocessor, a Level 2 Cache Controller (L2CC), and ROM and RAM controllers, in addition to the sub-systems necessary to communicate with internal buses and peripherals.

ARM1136JF-S Interrupt Controller (AVIC)

The ARM1136JF-S Interrupt Controller (AVIC) is a 32-bit peripheral that collects interrupt requests from up to 64 sources and provides an interface to the ARM1136JF-S core. The AVIC includes hardware acceleration of normal interrupts. The AVIC also includes software-controlled priority levels for normal interrupts.

Chapter 8
ARM11 Platform

The ARM1136JF-S processor platform is built from a combination of intelligent integrated peripherals and an advanced processor core that meet the processing and power management needs of the i.MX31 and i.MX31L Multimedia Applications Processors.

The ARM1136JF-S incorporates an integer unit that implements the ARM v6 architecture. It supports the ARM and Thumb instruction sets, Jazelle technology to enable direct execution of Java bytecodes, and a range of SIMD DSP instructions that operate on 16-bit or 8-bit data values in 32-bit registers. The ARM11 Platform also includes a vector floating point coprocessor, a Level 2 Cache Controller (L2CC), ROM and RAM controllers, in addition to the sub-systems necessary to communicate with internal buses and peripherals.

Since the processor is at the heart of this platform, this chapter begins with a brief overview of the ARM1136JF-S. A block diagram of the ARM11 is shown in Figure 8-1, followed by brief functional descriptions of each submodule within the platform. Overall platform functionality in terms of bus interfaces, clocks and resets, and power management is also described.

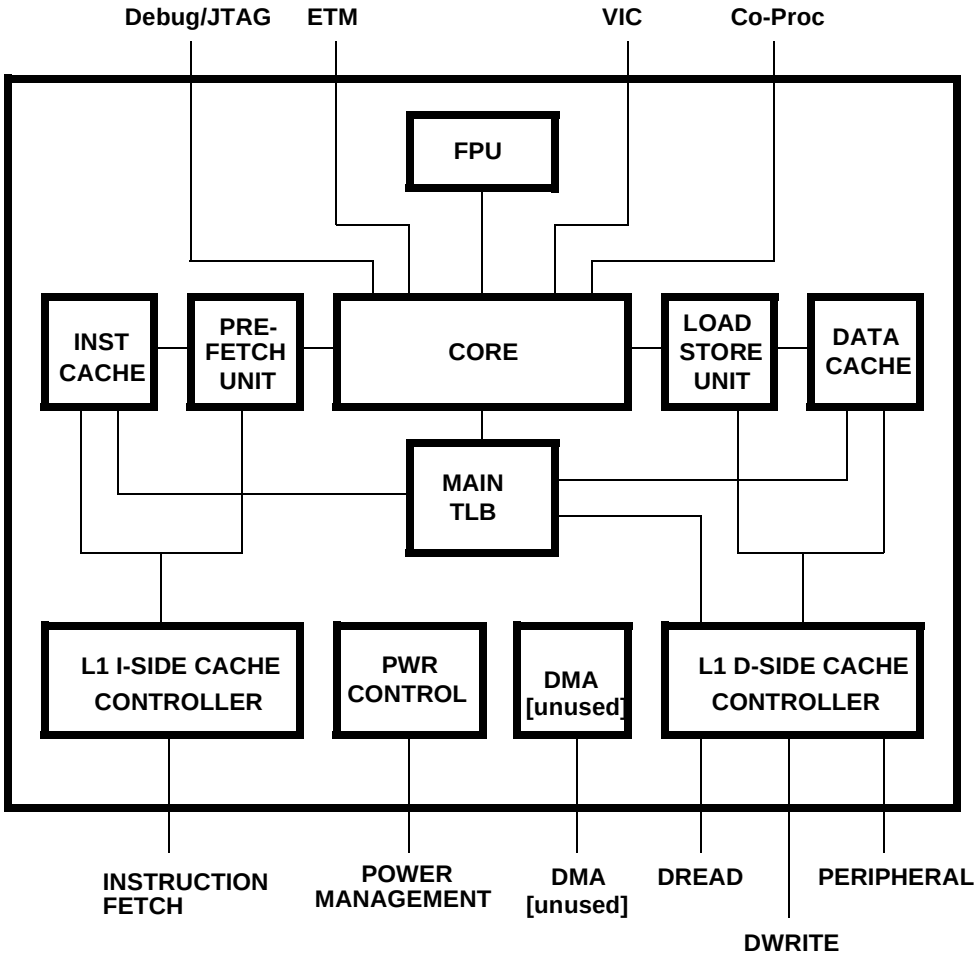


Figure 8-1. The ARM1136JF-S Block Diagram

8.1 Overview

This section presents a brief overview of the ARM1136JF-S processor. Not all features that are listed are available in the ARM11 Platform's implementation of the ARM1136JF-S processor, but are mentioned for completeness. Refer to Section 8.5, "Configuration of ARM1136JF-S in the ARM11 Platform" for more information on unused features.

The core of the ARM11 Platform is the ARM1136JF-S processor, which is referred to in this chapter as simply the "ARM11." The ARM11 Platform incorporates an integer unit that implements the ARM v6 architecture. It supports the ARM and Thumb instruction sets; Jazelle technology enables direct execution of Java byte codes; and a range of SIMD DSP instructions that operate on 16-bit or 8-bit data values in 32-bit registers. With the exception of memories related to the caches and MMU, the ARM11 is a fully synthesizable design.

8.1.1 Features

The ARM1136JF-S processor includes the following features:

- Integer unit with Embedded ICE logic
- 8-stage pipeline
- Branch prediction with return stack
- Low interrupt latency mode
- External coprocessor interface
- Instruction and Data MMUs, managed using micro TLB structures backed by a unified main TLB
- Instruction and Data Caches including a non-blocking data cache with Hit-Under-Miss
- Caches are virtually indexed/physically addressed
- 64-bit interface to both caches
- Write Buffer (bypassable)
- AMBA L2 interface supporting multiprocessor implementations
- JTAG-based debug
- Floating Point coprocessor

The following features of the ARM1136JF-S processor are not implemented in the ARM11 Platform:

- Instruction and Data TCMs (tightly-coupled memories) are not implemented.
- The DMA port is not implemented (the DMA port is used to transfer data to/from TCMs).
- The external coprocessor port is not implemented.

8.2 ARM11 Interfaces

This section gives short descriptions of the main interfaces to the ARM11 processor.

8.2.1 Debug/JTAG

This interface provides connection to external tools for the development of application software, operating systems, and hardware. The ARM11 JTAG interface must be synchronized externally to the processor's CLK domain as with previous ARM cores.

NOTE

JTAG synchronization is done internal to the ARM11 Platform by the JSYNC module. See Section 8.5.14, "JTAG Synchronization Module (JSYNC)."

8.2.2 ETM

The ETM interface provides connections to the ETM11 real-time trace hardware.

8.2.3 Vectored Interrupt Controller (VIC)

The Vectored Interrupt Controller (VIC) port is provided to support vectored interrupts. An external interrupt controller supplies the starting address (vector address) of the interrupt handler corresponding to the highest priority interrupt request.

8.2.4 Level Two Interface

The ARM11 level two memory interface exists to provide a high-bandwidth interface to second level caches, on-chip RAM, peripherals, and interfaces to external memory. It provides a high bandwidth mechanism for filling the caches on a cache miss. The ARM11 processor level two interconnect system uses four 64-bit wide generic AMBA 2.v6 (AHB-Lite) interfaces:

- Instruction Fetch Interface
- Data Read Interface
- Data Write Interface

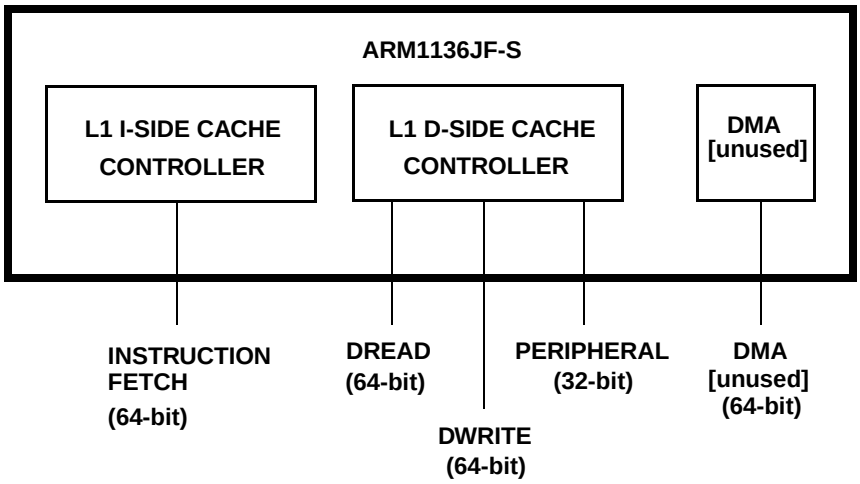


Figure 8-2. Instruction Fetch Interface

8.2.4.1 Instruction Fetch Interface

The Instruction Fetch Interface is a read-only interface that services the Instruction Cache on cache misses, including the fetching of instructions for the prefetch unit of instructions that are held in memory marked as non-cacheable. The interface is optimized for cache line fills rather than individual requests.

8.2.4.2 Data Read Interface

The Data Read Interface performs reads and swap writes. It services the Data Cache on cache misses, handles TLB misses on hardware page table walks, and reads uncacheable locations. While cache miss handling is important, the latency between outstanding uncacheable loads is minimized. The same address never appears on the Data Read Interface and the Data Write Interface simultaneously.

8.2.4.3 Data Write Interface

The Data Write Interface is a write-only interface that services the writes out of the write buffer. Multiple writes can be queued up as part of this interface.

8.2.4.4 Peripheral Interface

The Peripheral Interface is an AHB-Lite interface that services peripheral devices. The peripheral bus is a single master bus with the ARM11's Peripheral Interface being the only master. In the ARM11 processor, this interface is intended for peripherals that are private to the ARM11 core, such as the vectored interrupt controller or a watchdog timer.

Accesses to regions of memory that are marked as Device and Non-Shared are routed to the Peripheral Interface in preference to the Data Read or Data Write Interface. The Peripheral Port Memory Remap Register enables memory regions to be remapped to the Peripheral Interface.

8.2.5 ARM11 Symbol

For purposes of this document, a simplified symbol of the ARM11 is used. Figure 8-3 shows the ARM11 symbol.

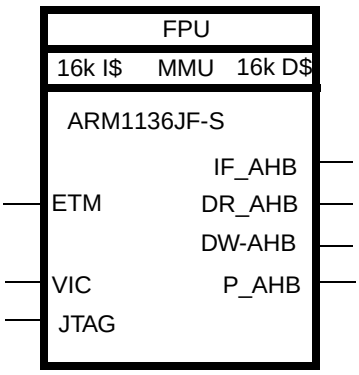


Figure 8-3. A Simplified ARM11 Symbol

The following are the signals referenced in Figure 8-3:

- ETM—Embedded Trace Macrocell Interface
- JTAG Interface
- VIC—Vectored Interrupt Controller Interface
- IF_AHB—Instruction Fetch Interface (64-bit)
- DR_AHB—Data Read Interface (64-bit)
- DW_AHB—Data Write Interface (64-bit)
- P_AHB—Peripheral Interface (32-bit)

8.3 ARM11 Platform Block Diagram

Figure 8-4 shows a block diagram of the ARM11 Platform.

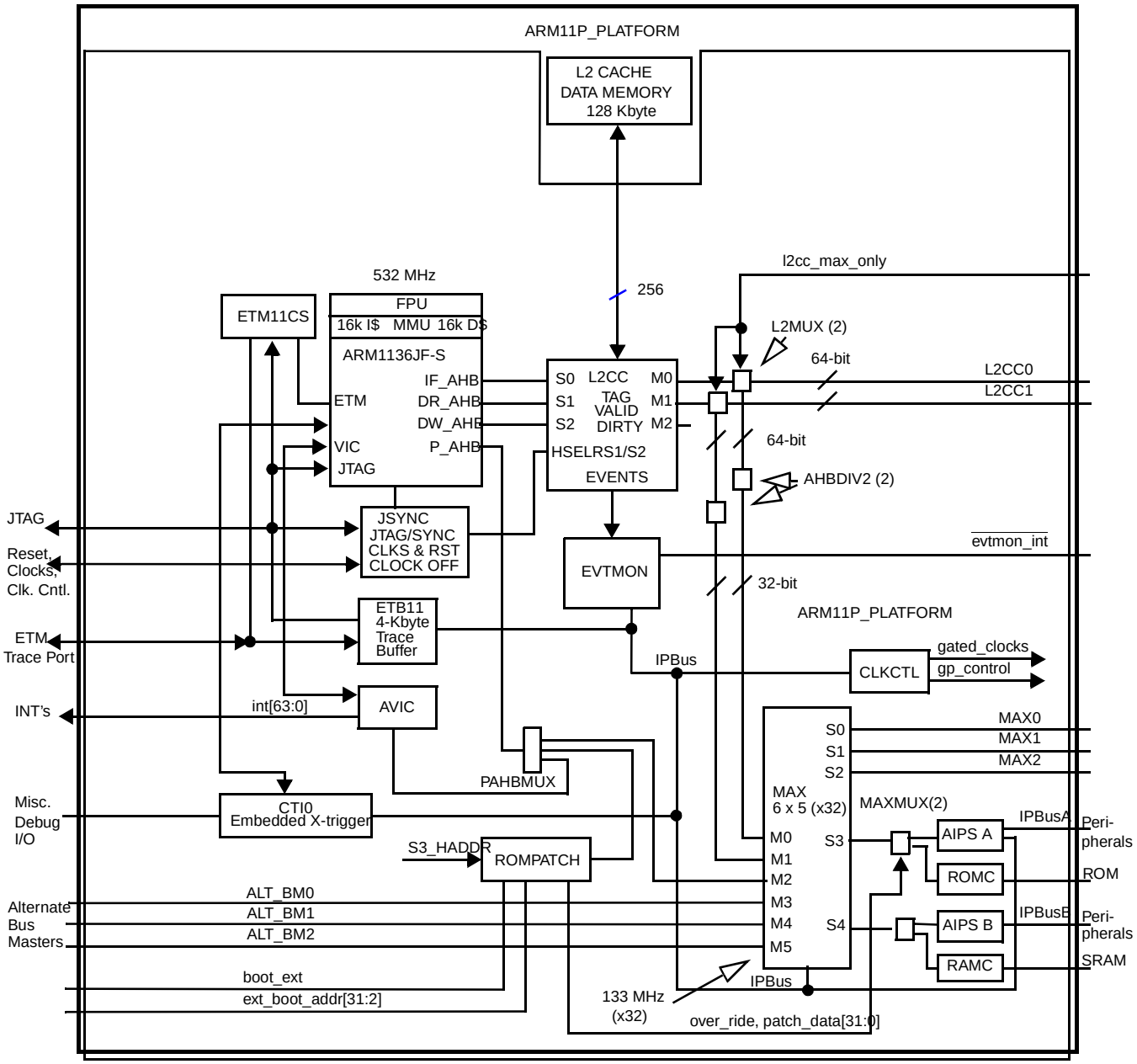


Figure 8-4. ARM11 Platform Block Diagram

8.4 Overview of Platform Submodules

As in Figure 8-4, the platform consists of the ARM1136JF-S processor, a Level two (L2) cache controller (L2CC), L2 cache memory, an L2 event monitor (EVTMON), two AMBA bus down-sizers (AHBDIV2), a 6x5 multi-layer crossbar switch (MAX), an SRAM controller (RAMC), two AHB to IP-Bus interface gaskets (AIPS), a ROM controller (ROMC), an ARM11 vectored interrupt controller (AVIC), a clock control module (CLKCTL), a ROMPATCH module, and a JTAG synchronization module (JSYNC). In addition, the L2MUX, PAHBMUX, and MAXMUX modules handle AHB infrastructure support.

In addition, the ARM11 Platform contains Embedded Trace Kit (ETK), from ARM. The ETK modules consist of the Embedded Trace Macrocell (ETM), the Embedded Trace Buffer (ETB), and the Cross Trigger Interface (CTI) module.

8.5 Configuration of ARM1136JF-S in the ARM11 Platform

The ARM11 was discussed in Section 8.1, "Overview." The information presented in this section focuses on the ARM1136JF-S implementation within the ARM11 Platform. Refer to the *ARM1136JF-S Technical Reference Manual* for more detailed information.

8.5.1 VFP11—Vector Floating Point Coprocessor

The ARM1136JF-S processor comes bundled with a floating point unit that is fully compliant with the *ANSI/IEEE Std 754-1985, IEEE Standard for Binary Floating-Point Arithmetic*. The VFP11 coprocessor supports all addressing modes described in the *ARM Architecture Reference Manual*.

The VFP11 is optimized for high data transfer bandwidth through 64-bit split load and store buses. Divide and square root operations run in parallel with other arithmetic operations to reduce the impact of long-latency operations. In addition to full IEEE 754 standard support, near IEEE 754 standard compatibility is provided in RunFast mode without support code assistance.

The VFP11 coprocessor provides full support of single-precision and double-precision add, subtract, multiply, divide, multiply with accumulate and square root operations. Conversions between floating-point data formats and the ARM integer word format are provided, with special operations to perform the conversion in round-toward-zero mode for high-level language support. Throughput and latency cycle counts for some commonly used VFP11 instructions are shown in Table 8-1.

Table 8-1. VFP11 Instruction Throughput and Latency Cycle Counts

Instructions	Single-Precision Throughput/Latency	Double-Precision Throughput/Latency
FABS, FNEG, FCVT, FCPY	1 5	1 5
FCMP, FCMPE, FCMPZ, FCMPEZ	1 5	1 5
FADD, FSUB	1 9	1 9
FMUL, FNMUL	1 9	1 9
FMAC, FNMAC, FMSC, FNMSC	1 9	2 10
FDIV, FSQRT	15 19	29 33

The VFP11 coprocessor is integrated with the ARM11 CPU through a dedicated VFP coprocessor interface and uses coprocessor ID number 10 for single-precision instructions and coprocessor ID number 11 for double-precision instructions. In some cases, such as mixed-precision instructions, the coprocessor ID represents the destination precision. In the ARM11 Platform, which contains the VFP11 coprocessor, these coprocessor ID numbers must not be used by another coprocessor.

Access to the VFP11 coprocessor is controlled by the ARM11 Coprocessor Access Control Register. The coprocessor access rights must be configured correctly before any VFP11 instructions may be executed. Refer to the *ARMVFP11 Technical Reference Manual* for detailed information.

8.5.2 ARM11 Instruction and Data Caches (L1)

The ARM11 is configured with a 16-Kbyte Instruction Cache and a 16-Kbyte Data Cache. Each cache is implemented as four-way set associative and each is both virtually indexed and physically addressed. Both the Instruction Cache and Data Cache are capable of providing two words per cycle for all requesting sources. The number of cache ways is fixed at four. The line length is not configurable and is fixed at eight words per line.

Refer to the *ARM1136EJ-S Technical Reference Manual* for more detailed information.

8.5.3 L2 Interface (IF_AHB, DR_AHB, DW_AHB)

The three 64-bit L2 interfaces described earlier: Instruction Fetch, Data Read, and Data Write, are connected directly to the tightly coupled L2 cache controller.

8.5.4 Vectored Interrupt Controller Interface (VIC Interface)

The Vectored Interrupt Controller (VIC interface) of the ARM11 is connected directly to the ARM11 Vectored Interrupt Controller (AVIC) module. The AVIC module is accessed by the ARM11 via its P_AHB bus and the MAXMUX module. Refer to Section 8.5.11, “ARM11 Vectored Interrupt Controller (AVIC)” for more details.

NOTE

The AVIC module used on the ARM11 Platform is not the VIC module supplied by ARM. The AVIC module is one designed by Freescale to support the vectored interrupt interface (VIC) on the ARM11.

8.5.5 JTAG Interface

The JTAG interface on the ARM1136 and the ETK11 modules are connected to the platform’s JSYNC module. The JSYNC module performs the standard ARM JTAG synchronization logic. The JSYNC module also contains some miscellaneous debug related logic in addition to JTAG and power-on-reset control. Refer to Section 8.5.14, “JTAG Synchronization Module (JSYNC)” for more details.

8.5.6 Level Two Cache Controller (L2CC)

This section focuses on the use and configuration of the L2CC within the ARM11 Platform. First, the general features of the L2CC is listed.

Refer to Chapter 14, “L2 Cache Controller (L2CC)” for more details.

8.5.6.1 L2CC Configuration on the ARM11 Platform

The ARM11 Platform Level 2 Cache Controller (L2CC) is optimized to sit between the ARM11 (L1) and main memory (L3, or the external EMI). The L2 cache is a unified, physically indexed, physically tagged 8-way cache. The replacement algorithm can be locked on a way basis, allowing the associativity to be reduced from 8 way down to 1 way (direct mapped), and the locked ways to be used as physically mapped memory. The L2CC does not have snooping hardware to maintain coherency between caches, so coherency must be maintained via software.

The L2 cache sizes supported by the ARM11 Platform is 128 Kbyte. A 256-bit data path is implemented to the L2 data arrays. The L2 cache latency is 8 clocks on hits assuming a single cycle RAM access. However, the actual latency of the L2 data RAMs on the ARM11 Platform is 4 clocks (three wait-states), and therefore the range on the ARM11 Platform is predicted to be 11–12 clocks. The L2 tag, valid, and dirty RAMs are zero wait-state compiled memories.

8.5.7 L2CC Performance

The L2CC on the ARM11 Platform implement zero wait-state tag, valid and dirty memory arrays. These memories reside inside the common ARM11 Platform module. The L2 data memories reside outside of the common ARM11 Platform and use three-wait states for both reads and writes. The total access time to the first word of read data is calculated to be 11 or 12 clocks. This number increases from the ARM11 perspective due to L1 pipeline delays. Table 8-2 shows the ARM11 Platform's L2CC access times for a 4 word burst read hit from the perspective of the ARM1136 L2 interface AHB's. These are best case access times, as there are other scenarios when access times on L2 hits could be longer (concurrent requests for example).

Table 8-2. L2CC Burst Read Access Time

Burst of 4	Access Time in Clock Cycles
1st 64 bits of burst	11
2nd 64 bits of burst	12
3rd 64 bits of burst	13
4th 64 bits of burst	14

Refer to ARM's *L2 Cache Controller (L2CC) Engineering Specification* for more detailed information on the L2CC design and the L2CC programmer's model.

8.5.8 Level Two AHB MUX (L2MUX)

The 64-bit L2CC master ports M0 and M1 are routed directly to the top-level of the ARM11 Platform. Additionally, the 64-bit L2CC master ports M0 and M1 are each connected to an AHBDIV2 module before connecting to the MAX module. The AHBDIV2 modules downsize the 64-bit L2CC data paths to the 32-bit data path of the MAX. There are two Level Two AHB MUX (L2MUX) modules instantiated, one for each L2CC master (one for M0 and one for M1), to handle the AHB infrastructure. Each L2MUX

performs address decoding (HSEL generation) to select either the EMI or AHBDIV2 as the target of the requested transaction. The l2cc_max_only input (intended to be tied off at the SoC level) feeds into the HSEL logic in the L2MUX. When asserted, l2cc_max_only causes all transactions from the L2CC master ports to be routed only to the MAX. Each L2MUX also performs the HRDATA muxing and HREADY generation back to the L2CC master port.

8.5.9 AHB Downsizer (AHBDIV2)

The purpose of the AHBDIV2 module is to translate the L2CC master ports' 64-bit data AHB accesses into a 32-bit data AHB access for the MAX. To alleviate a critical timing path in the 133 MHz per_clk domain, a wait-state is taken on transactions destined for the MAX, which immediately follow a previous access to the L2CC0/1 direct connect ports. This wait-state is not observed on designs that do not use the direct connect ports (l2cc_max_only=1).

8.5.10 Multi-Layer 6 X 5 AHB Crossbar Switch (MAX)

The L2CC master ports, the off-platform alternate bus masters, and the ARM11 P_AHB arbitrates memory and peripherals via a 6X5 Multi-Layer AHB Crossbar switch (a 6x5 MAX). The MAX port connections for the ARM11 Platform are shown in Table 8-3.

Table 8-3. MAX Port Connections

MAX Port Name	AHB Function	ARM11 Platform Connection
M0	Slave	L2CC Master M0 (through an AHBDIV2)
M1	Slave	L2CC Master M1 (through an AHBDIV2)
M2	Slave	P_AHB
M3	Slave	ALT_BM0
M4	Slave	ALT_BM1
M5	Slave	ALT_BM2
S0	Master	Off Platform AHB-Lite 2.v6
S1	Master	Off Platform AHB-Lite 2.v6
S2	Master	Off Platform AHB-Lite 2.v6
S3	Master	AIPS A and ROMC (ROM Controller)
S4	Master	AIPS B and RAMC (RAM Controller)

8.5.10.1 Peripheral Bus Timeout Monitors

The ARM11 Platform's MAX module returns an AHB hresp=ERROR termination status on attempted accesses to undefined regions of memory. Likewise, the AIPS module returns an AHB hresp=ERROR termination status on attempted accesses to unpopulated regions of the peripheral space. ."

8.5.11 ARM11 Vectored Interrupt Controller (AVIC)

The ARM11 Vectored Interrupt Controller (AVIC) hardware exists to prioritize interrupts and to supply the interrupt vector address of the highest priority interrupt to the ARM11 core automatically via an interrupt sideband bus (the ARM11 “VIC” interface). The AVIC module has an AHB-Lite interface, and is programmed via the ARM11’s peripheral AHB bus (P_AHB). The AVIC contains a 63x30 register array to store the vector table.

Synchronization of the VIC interface to the ARM11 takes place in the ARM11 itself. This is why the INTSYNCEN and IRQADDRVSYCNEN inputs on the ARM11 are tied low.

NOTE

The ARM11 TRM states that the synchronizers in the ARM11 are bypassed when INTSYNCEN and IRQADDRVSYNCEN are asserted. This is not intuitively obvious from the signal names (one would normally consider synchronization to take place when the signals are asserted since the signals have the SYNCEN suffix attached). The AVIC interface to the ARM1136 is shown in Figure 8-5.

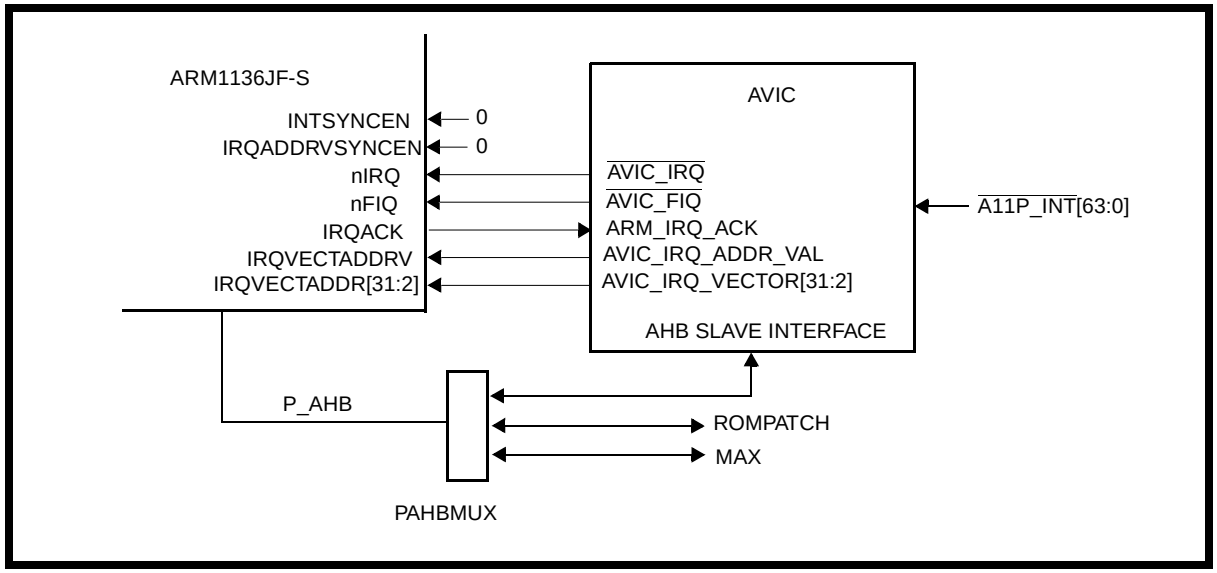


Figure 8-5. The AVIC Module Connected to the ARM1136

8.5.12 Clock Control Module (CLKCTL)

The Clock Control Module (CLKCTL) module performs module level clock gating, when possible, for modules within the platform. The CLKCTL module is also a 32-bit peripheral connected to AIPS A’s on platform peripheral bus. The IP-Bus slave interface allows access to a general purpose control register and a general purpose status register. T

8.5.13 CLKCTL Registers

The CLKCTL module is connected to IPBus A, slot 3 as a 32-bit peripheral. The registers residing in the CLKCTL module are shown in Table 8-4.

Table 8-4. CLKCTL Registers

AIPS A_ADDR[3:0]	Register Name	Register Definition	Width
5'h00	GP_CTRL	General Purpose Control Register	[15:0]
5'h04	GP_SER	Set Enable Register	[15:0]
5'h08	GP_CER	Clear Enable Register	[15:0]
5'h0C	GP_STAT	General Purpose Status Register	[15:0]
5'h10	L2_MEM_VAL	L2 Data Memory VAL Settings	[11:0]

8.5.13.1 GP_CTRL, GP_SER, and GP_CER Registers

The GP_CTRL register is a write-read register. All bits in the GP_CTRL are cleared on reset. All bits of the GP_CTRL register may be written concurrently by writing directly to the GP_CTRL register. However, to simplify software (prevent the need for a read-modify-write), a single bit may be set in the GP_CTRL register by writing a one to the relative bit location in the Set Enable Register (GP_SER). Similarly, writing a one to the relative bit location in the Clear Enable Register (GP_CER) clears the associated GP_CTRL bit. Reading the GP_SER or GP_CER registers returns all zeros.

Table 8-5 shows the functions of all bits in the CLKCTL module's General Purpose Control Register.

Table 8-5. CLKCTL General Purpose Control Bit (GP_CTRL) Usage

GP_CTRL Register Bit	Reset Value	Description
[3:0]	0	Driven off-platform for use at SoC level.
[4]	0	When set, enables the peripheral bus timeout monitors.Refer to Section 8.5.10.1, "Peripheral Bus Timeout Monitors."
[5]	0	When set, this bit enables clocks to the CTI module. Clocks to the CTI module are automatically issued when a debugger is connected (dbgen asserted). However, if it is desirable to enable cross trigger entry into debug mode prior to connecting a debugger, this bit must be set, and the disable_trace input must be negated. This bit should not be set in a non-debug mode environment.
[6]	0	When set, this bit enables the clocks to the ETB. Clocks to ETB are automatically issued when a debugger is connected (dbgen asserted). However if system wants to use the ETB trace buffer as a general purpose memory, this bit must be set, and the disable_trace input must be negated. This bit should not be set in a non-debug mode environment if the ETB memory is not going to be used.
[7]	0	When set, this bit enables the high speed DVFS signals to propagate through the platform boundary flip-flop.
[8]	0	When set, this bit enables module level clock gating of the L2CC. When low, module level clock gating of the L2CC is disabled.

Table 8-5. CLKCTL General Purpose Control Bit (GP_CTRL) Usage (continued)

GP_CTRL Register Bit	Reset Value	Description
[9]	1	Connected to AIPS A's aips_byte_config[0] input. The default setting ("1") is the intended use. This control bit is for flexibility only in case of a possible peripheral Endian-related bug.
[10]	1	Connected to AIPS B's aips_byte_config[0] input. The default setting ("1") is the intended use. This control bit is for flexibility only in case of a possible peripheral Endian-related bug.
[11]	0	0 MAX slave port 0 and 1 is disabled. 1 MAX slave port 0 and 1 is enabled.
[15:10]	0	Spares.

Refer to the *ARM11 Platform Clock Controller Specification* for more detail.

8.5.13.2 GP_STAT Register

The GP_STAT register is read only. All bits are cleared on reset. Reading the GP_STAT register while reset is negated will return the values on the ARM11 Platform's inputs.

8.5.13.3 L2_MEM_VAL Register

The L2_MEM_VAL register enables the user to modify the default value of the L2 data memory's "VAL" settings. The L2_MEM_VAL register is reset to the values on the l2_rval_rst[3:0], l2_rval2_rst[3:0], and l2_wval_rst[3:0] inputs. Subsequent to system reset, this register may be written to change the VAL settings driven to the L2 data memory. The register's bit assignments are shown in Table 8-6.

Table 8-6. L2_MEM_VAL Register Bit Assignments

L2_MEM_VAL Bit	Reset Value	Description
[3:0]	l2_rval_rst[3:0]	These bits are reset to the value of the l2_rval_rst[3:0] platform inputs. The register's outputs are driven directly to the L2 data memory's RVAL inputs. After reset, these bits may be written by the user to modify the reset value.
[7:4]	l2_rval2_rst[3:0]	These bits are reset to the value of the l2_rval2_rst[3:0] platform inputs. The register's outputs are driven directly to the L2 data memory's RVAL2 inputs. After reset, these bits may be written by the user to modify the reset value.
[11:8]	l2_wval_rst[3:0]	These bits are reset to the value of the l2_wval_rst[3:0] platform inputs. The register's outputs are driven directly to the L2 data memory's WVAL inputs. After reset, these bits may be written by the user to modify the reset value.
[31:12]	Unimplemented	Unspecified.

8.5.14 JTAG Synchronization Module (JSYNC)

The JTAG Synchronization Module (JSYNC) synchronizes the external JTAG interface to the ARM11's clock (arm_clk). The inputs and outputs of the synchronization circuit is connected to the JTAG interface

on the ARM11, ETM11, and ETB11 modules. The standard ARM11 JTAG synchronization circuit is used. The JSYNC module also performs any synchronization requirements for all resets on the platform. In addition, the JSYNC module monitors the ARM11's DR and DW AHB HADDR[31:28] bits to decode the HSELRS1 and HSELRS2 inputs to the L2CC module. These inputs assert on accesses to the L2CC configuration registers. Refer to

8.5.15 ARM1136JF-S Embedded Trace Macrocell (ETM11)

The Embedded Trace Module (ETM11) is directly connected to the ARM11's ETM interface and provides real time trace debug capability. ETM11 is capable of both instruction and data tracing. ETM11 provides a trace protocol that is an integral part of the ARM Real Time Debug solution (*RealView*). Real time tracing is controlled by specifying a set of triggering and filtering resources that include address and data comparators, counters, and sequencers. Figure 8-6 shows the main functional blocks and external interfaces of the ETM11 module.

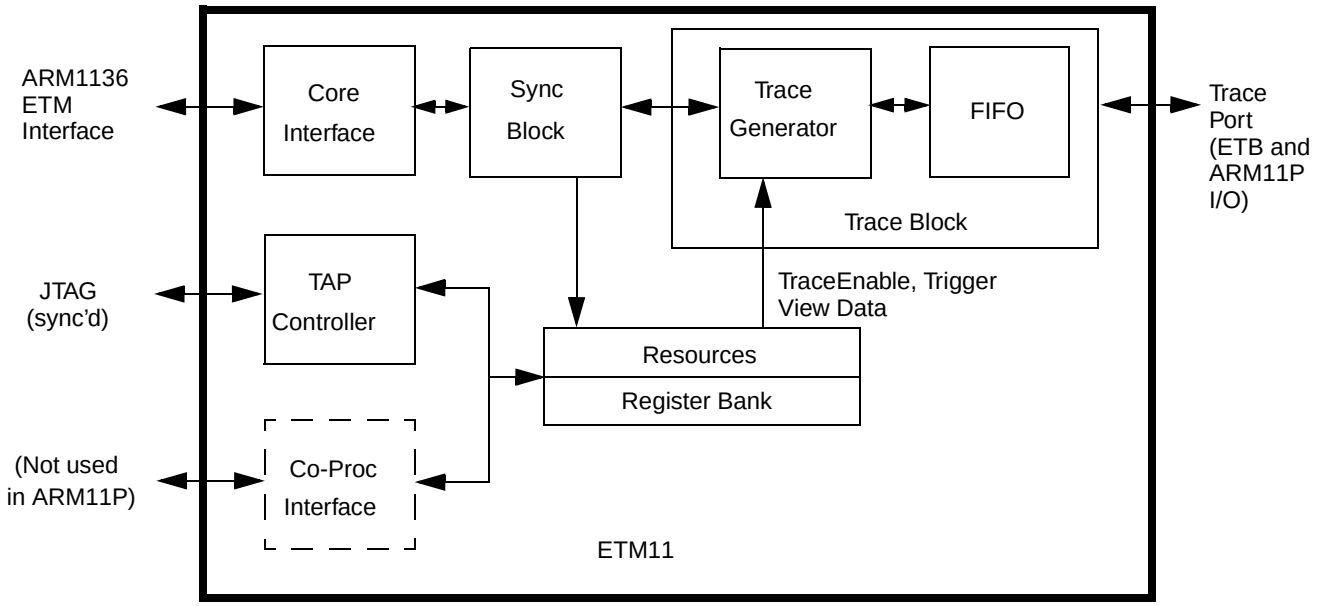


Figure 8-6. ETM11 Functional Block Diagram

The ETM11 is delivered with a fixed FIFO size (69 bytes) optimized for use with the Embedded Trace Buffer (ETB). The trace data packet width is selectable at 4/8/16/24/32 using the ETM control register. ETM11 supports two data comparators and 4 pairs of address comparators.

Trace port timing is a critical issue and the way in which ARM Ltd. named the port modes is not intuitive. The ETM11 supports only 3 modes: dynamic, 1:2 and 1:4. Dynamic is used for on-chip trace (ETB11). 1:2 and 1:4 modes refer to the traceport data to ARM_CLK ratio, not the ARM_CLK to TRACECLK ratio. The 1:2 mode corresponds to a 1:4 clock ratio; the 1:4 mode corresponds to a 1:8 clock ratio.

The system metrics module within ARM1136 can be used to count events, such as cache hits for example, and indicate such events on the ARM11's eventus[19:0] outputs. The ETM11 can then be configured to monitor these events and use them as additional trace filters. Alternately, the ARM1136 system metrics unit can count the two ETM11 outputs as additional inputs.

Refer to ARM's *ETM11 Technical Reference Manual* and *Embedded Trace Macrocell Architecture Specification* for more detailed information.

8.5.16 Embedded Trace Buffer (ETB11)

The ETB11 module, or Embedded Trace Buffer, stores trace data from the trace port of the ETM11 module. The buffered data can then be accessed via the JTAG interface or through the ETB11's slave interface on the IP Bus. Providing an on-chip buffer alleviates the pin count, bandwidth, and pad design requirements associated with sending trace data to a debugger directly through package pins in a real-time fashion. The ETB11 is designed with a standard compiled RAM interface. The ARM11 Platform has implemented a 4-Kbyte compiled memory for the trace buffer. The 4-Kbyte buffer can be used by the system as a general purpose memory, however the clocks must be manually enabled if the platform is not in debug mode. To simplify Endian issues, the ETB module supports only 32-bit accesses to the 4-Kbyte buffer. The access time of the ETB memory over the IP Bus is 7 per_clks for both writes and reads. The 4-Kbyte ETB memory begins at address 32'h43F1_4000. Figure 8-7 shows a simplified block diagram of the main functional blocks and external interfaces of the ETB11 module.

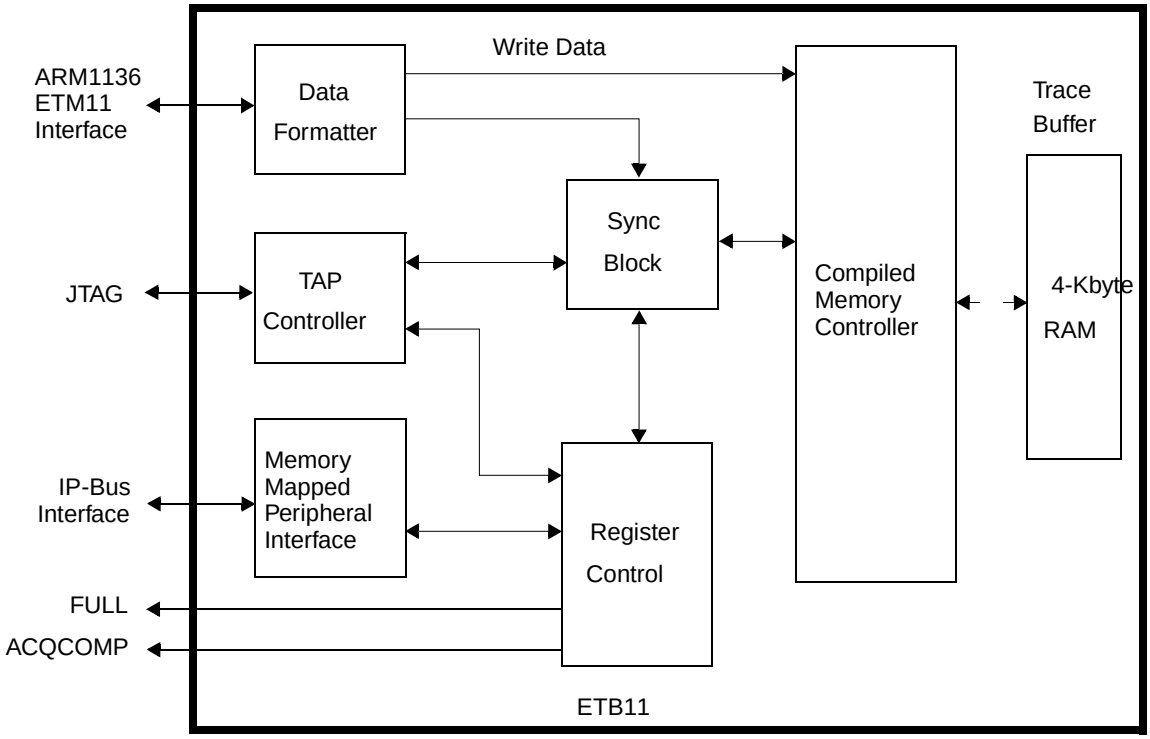


Figure 8-7. ETB11 Functional Block Diagram

Refer to ARM's *ETB11 Technical Reference Manual* for more detailed information.

8.5.17 Embedded Cross-Trigger (ECT)

The ECT module, or Embedded Cross-Trigger, passes debug events from one processor to another. For example, the ECT can be programmed to allow communication of debug state information from one core to another so that program execution on both processors can be stopped at the same time. The ECT module

contains two main components: the CTI (Cross Trigger Interface) unit provides a common programmers model for use by the debug tools, controls the trigger sources, and interfaces to the CTM component. The CTM (Cross Trigger Matrix) combines the trigger requests from the CTI's and broadcasts them to all other CTI's as triggers, enabling one CPU to trigger with another. Figure 8-8 shows a simplified block diagram of the main functional blocks and external interfaces of the ECT module.

8.5.18 ECT Implementation in the ARM11 Platform

For SoC partitioning reasons, only a single CTI of the ECT module (CTI0) is instantiated on the platform, as shown in blue shade in Figure 8-8. The full ECT design is shown for completeness and to understand how the ARM11 communicates cross-trigger information with other processors.

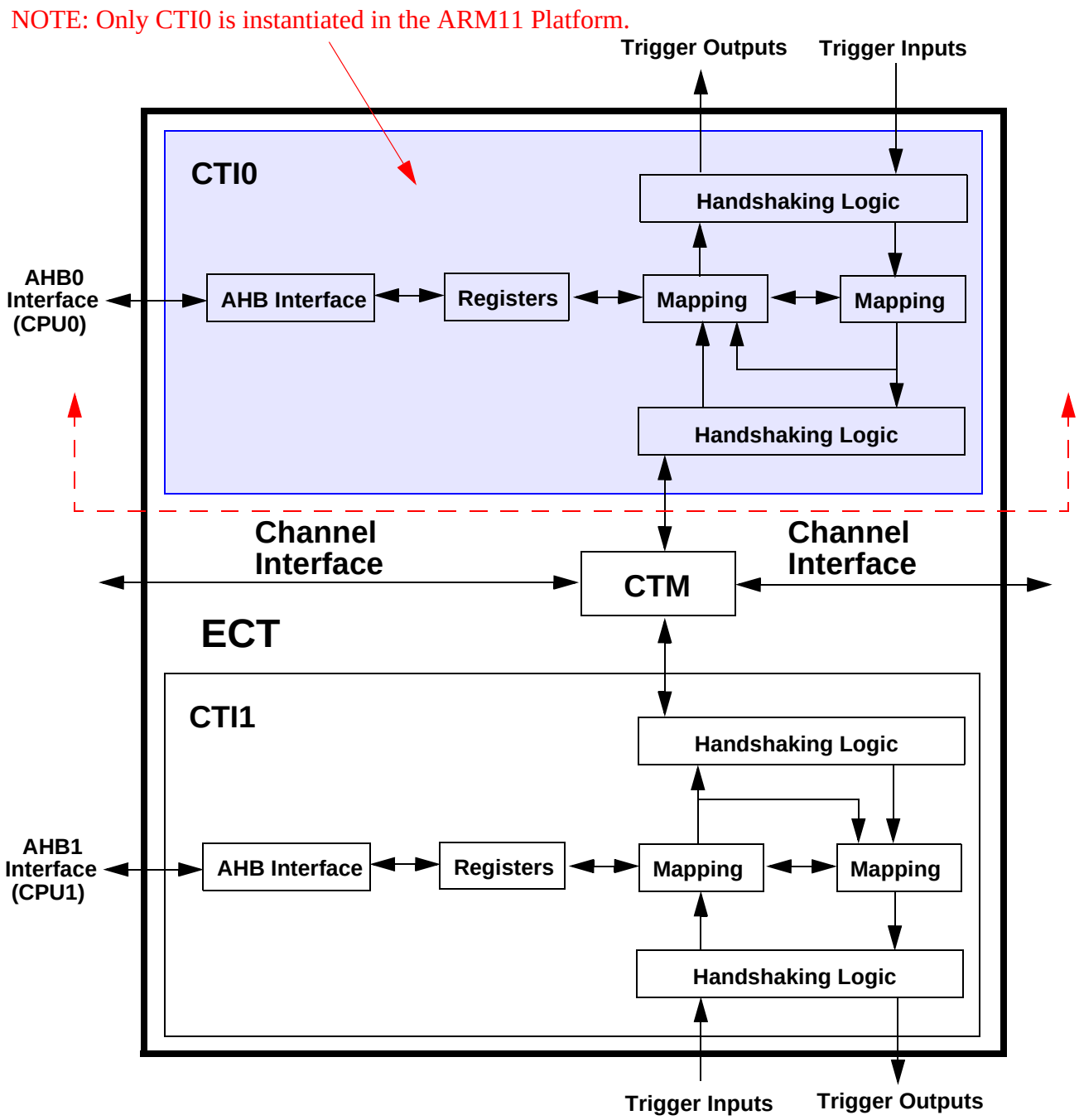


Figure 8-8. ECT Functional Block Diagram

Refer to ARM's *ECT Technical Reference Manual* for more detailed information.

8.6 Security Summary

The ARM11 Platform works in conjunction with a variety of components and mechanisms—some of which are external to the platform—to address security concerns. These are summarized here:

- ARM1136JF-S MMU
- AIPS Peripheral Access Control Registers
- AHB hmaster[3:0] Encoding
- disable_trace input to halt external instruction and data tracing via ETM11
- Secure JTAG (external to platform)
- Security Controller Module (SCC)—external to platform

8.6.1 ARM1136JF-S MMU

The ARM11's MMU provides access permission checks for the instruction and data ports of the ARM1136JF-S processor.

8.6.2 AIPS Access Control Registers

The AIPS module contains several security related features. Refer to the Chapter 38, "AHB-Lite 2.v6 to IP Bus Interface (AIPS)" for more detailed information.

8.6.3 AIPS Master Privilege Registers

The AIPS module contains Master Privilege Registers, which define the access privilege level associated with all bus masters in the system. The registers specify for each master:

- If read accesses are trusted
- If write accesses are trusted
- If accesses are forced to user-mode regardless of the hprot[1] attribute

8.6.4 AIPS Peripheral Access Control Registers

The AIPS module contains Peripheral Access Control Registers, which define the access levels supported by each peripheral. The registers specify for each peripheral if:

- The peripheral requires supervisor mode for accesses
- The peripheral allows write accesses
- Accesses from an untrusted master are allowed.

Appropriate action is taken on unallowed accesses.

8.6.5 ipsa_cacheable, ipsb_cacheable

Each AIPS module (A and B) has implemented an IP Bus sideband signal to indicate to peripherals residing external to the ARM11 Platform that the current IP Bus transfer is the result of a cacheable AHB

transaction. These signals, the ipsa_cacheable and ipsb_cacheable outputs, indicate transaction cacheability as determined by the AHB hprot[3] signal. Although any peripheral may find the cacheable indicator useful, the outputs were specifically implemented so that the SCC can prevent its secure memory from being cached. It is important to realize accesses via the P_AHB bus are by definition not cacheable as all P_AHB accesses are of type “Non-shared Device.” However, an SCC module residing on an IP Bus could potentially be accessed by an L2 master port or an alternate bus master, any of which could master a cacheable transaction.

8.6.6 hmaster[3:0] Encodings

Each master in the system will drive the hmaster[3:0] encoding assigned to it for each AHB transaction. The reserved hmaster[3:0] encodings for the ARM11 Platform are shown in Table 8-7. The available encodings shown in Table 8-7 can be used by external bus masters.

Table 8-7. HMASTER Encodings

HMASTER[3:0]	Master
0000	MAX
0001	ARM1136 (All AHB Masters) and L2CC
0010–1111	Available for External Alternate Bus Masters

The ARM1136JF-S will have the same hmaster encodings for all AHB interfaces used on the platform (IF, DR, DW, P_AHB, L2CC0, L2CC1). The hmaster encodings can be used by AHB slave devices to forbid accesses by untrusted masters. The hmaster encodings will also be used by exclusive access monitors at the memory slaves to identify exclusive access ownership and the success or failure of an exclusive access.

The AIPS peripheral interfaces indicate master information on the ipsa_master[3:0] and ipsb_master[3:0] signals. These signals can be used by any peripheral, the SCC for instance, as an additional qualifier for access protection security.

The L2CC0 and L2CC1 buses do not have hmaster[3:0] signals on their platform interface to save power. The hmaster[3:0] signals for these buses are tied to 0001 at the slaves (same as ARM1136JF-s).

NOTE

The L2CC design will drive hmaster[3:0]=0xF on L2CC internally generated bus cycles (write allocate and buffered writes for example). However, since the hmaster signals are not routed from the L2CC module and are tied off at the slave to 0001, the 0xF value for hmaster is still available for other bus masters’ use.

8.6.7 Secure JTAG

The ARM11 Platform has been designed assuming there is JTAG port security implemented at the top-level of the SoC. Nothing specific with regards to securing the JTAG port has been implemented on the ARM11 Platform. The platform’s assumption is that jtag_tms and/or jtag_tdi and jtag_tdo are gated in some fashion at the SoC level if a security violation has been detected.

8.6.8 disable_trace

The ARM11 Platform's real-time trace port (ETM11) can be disabled by assertion of the disable_trace input. Disabling the ETM11 real-time trace port is done by gating of the ETM's arm_clk2.

8.6.9 Security Controller Module (SCC)

The SCC module assumed to reside on one of the ARM11 Platform's peripheral IP-bus interfaces. As such, it is downstream of the AIPS security registers, and can receive the master and cacheable indicators previously discussed (see above).

Chapter 9
ARM1136JF-S Vectored Interrupt Controller (AVIC)

The ARM1136JF-S Vectored Interrupt Controller (AVIC) is a 32-bit peripheral that collects interrupt requests from up to 64 sources and provides an interface to the ARM1136JF-S core. The simplified block diagram of the AVIC is shown in Figure 9-1.

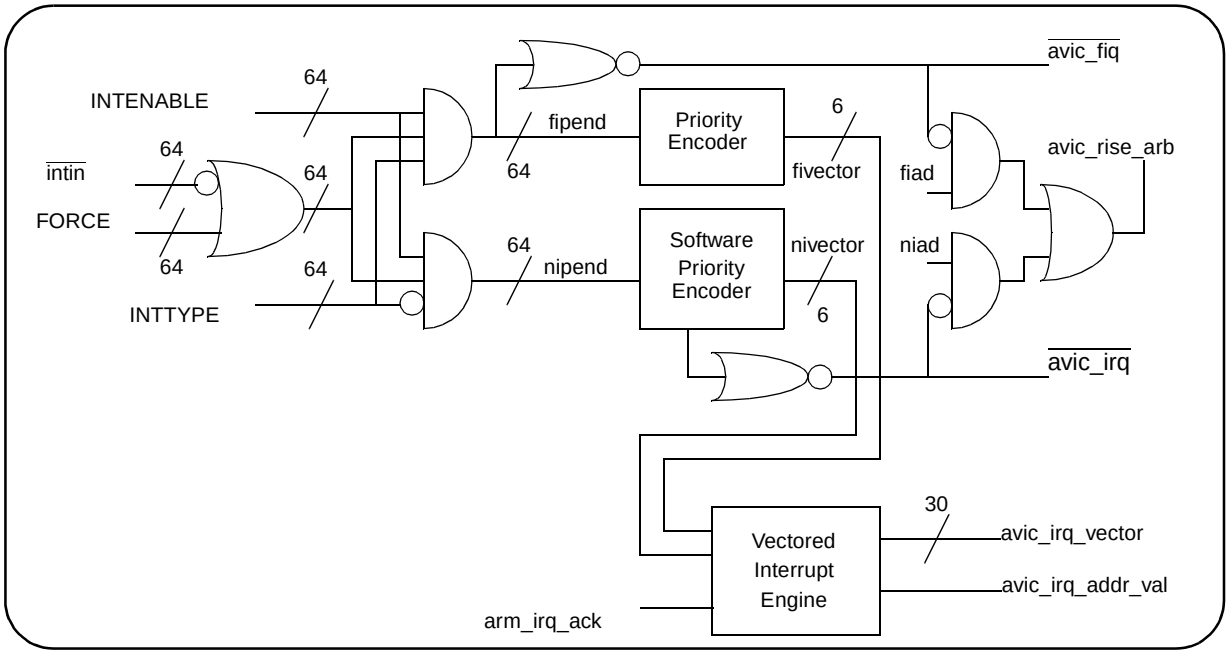


Figure 9-1. AVIC Block Diagram

9.1 Overview

The AVIC provides hardware acceleration of NORMAL interrupt service routine entries. If enabled when a normal interrupt occurs, the ARM1136JF-S core uses the integrated vector table and the ARM1136JF-S's AVIC interface to quickly jump to the interrupt service routine. This core mechanism is controlled by the vector enable bit (VE bit) in ARM's CP15 control register, and normal interrupt mode control bit of AVIC's INCNTL register (bit 18).

9.1.1 Features

The AVIC performs the following functions:

- Supports up to 64 interrupt sources
- Supports fast and normal interrupts
- Selects normal or fast interrupt request for any interrupt source

- Supports hardware accelerated vectoring to service routines for all normal interrupts
- Indicates pending interrupt sources via a register for normal and fast interrupts
- Indicates highest priority interrupt number via register (can be used a table index)
- Independently enable or disable any interrupt source
- Provides a mechanism for software to schedule an interrupt
- Provides integrated vector table for hardware acceleration of normal interrupt service routine entry
- Supports up to 16 software controlled priority levels for normal interrupts and priority masking
- Single bit disabling of all normal interrupts and of all fast interrupts, used in enabling of secure operations

9.1.2 Modes of Operation

The AVIC includes hardware acceleration of NORMAL interrupt service routine entry. If enabled and a normal interrupt occurs, the ARM1136JF-S core will use the integrated vector table and the

ARM1136JF-S's AVIC interface to quickly jump to the interrupt service routine. This core mechanism is controlled by the vector enable bit (VE bit) in ARM's CP15 control register, and normal interrupt mode control bit of AVIC's INCNTL register ((bit 18).

The interrupt controller consists of a set of control registers and associated logic to perform interrupt masking, priority support, and hardware acceleration of normal interrupts.

The interrupt source registers (INTSRCH/INTSRCL) are a pair of 32-bit status registers with a single interrupt source associated with each of the 64 bits. An interrupt line or set of interrupt lines are routed from each interrupt source to the INTSRCH or INTSRCL register. This allows up to 64 distinct interrupt sources in an implementation. Interrupt requests may be forced to be asserted by way of the interrupt force registers (INTFRCH/INTFRCL). Each bit in this register is logically “OR’ed” with the corresponding hardware request line prior to feeding the INTSRCH or INTSRCL register inputs. See Table 9-7 for a listing of the interrupt vector number assignments.

There is a corresponding set of interrupt enable registers (INTENABLEH/INTENABLEL), also 32-bits wide which allow individual bit masking of the INTSRCH/INTSRCL registers. There is also a corresponding set of interrupt type register (INTTYPEH/INTTYPEL) which selects whether an interrupt source will generate a normal or fast interrupt to the ARM1136JF-S core.

There is a corresponding set of normal interrupt pending registers (NIPNDH/NIPNDL) which indicate pending normal interrupt requests. These registers are equivalent to the logical AND of the interrupt source registers (INTSRCH/INTSRCL), the interrupt enable registers (INTENABLEH/INTENABLEL), and the NOT of the interrupt type registers (INTTYPEH/INTTYPEL). (Refer to Figure 9-1.) The NIPNDH/NIPNDL register bits are bit-wise “NOR’ed” together to form the nIRQ signal routed to the ARM1136JF-S core. This core input signal is maskable by the normal interrupt disable bit (I bit) in the processor status register (CPSR). The normal interrupt vector register (NIVECSR) indicates the vector index of highest priority pending normal interrupt.

There is a corresponding set of fast interrupt pending registers (FIPNDH/FIPNDL) which indicate pending fast interrupt requests. These registers are equivalent to the logical AND of the interrupt source registers (INTSRCH/INTSRCL), the interrupt enable registers (INTENABLEH/INTENABLEL), and the interrupt

type registers (INTTYPEH/INTTYPEL). (Refer to Figure 9-1.) The FIPNDH/FIPNDL register bits are bit-wise “NORed” together to form the nFIQ signal routed to the ARM1136JF-S core. This core input signal is maskable by the fast interrupt disable bit (F bit) in the CPSR. The fast interrupt vector register (FIVECSR) indicates the vector index of highest priority pending fast interrupt.

The AVIC includes hardware acceleration of normal interrupt service routine entry. If enabled and a normal interrupt occurs, the ARM1136JF-S core will use the integrated vector table and the ARM1136JF-S’s AVIC interface to quickly jump to the interrupt service routine.

All interrupt controller registers are readable and writable in privileged mode only. Writes attempted to read-only registers will be ignored. These registers can be modified only using 32-bit writes.

The INTFRCH/INTFRCL registers are provided for software generation of interrupts. By enabling interrupts for these bit positions, software can force an interrupt request. This register can also be used to debug hardware interrupt service routines by providing an alternate method of interrupt assertion.

The interrupt requests are prioritized in the following sequence:

1. Fast interrupt requests, in order of highest number
2. Normal interrupt requests, in order of highest priority level, then highest source number with the same priority

The AVIC provides 16 software controlled priority levels for normal interrupts. Every interrupt can be placed in any priority level. The AVIC also provides a normal interrupt priority level mask (NIMASK) which disables any interrupt with a priority level lower than or equal to the mask. If a level 0 normal interrupt and a level 1 normal interrupt are asserted at the same time, the level 1 normal interrupt will be selected assuming that NIMASK has not disabled level 1 normal interrupts. If two level 1 normal interrupts are asserted at the same time, the level 1 normal interrupt with the highest source number will be selected, also assuming that NIMASK has not disabled level 1 normal interrupts.

9.2 Memory Map and Register Definition

The AVIC module has 26 registers. All of these registers are single cycle access as the AVIC sits on the native bus of the ARM1136JF-S core. Section 9.2.3, “Register Descriptions” provides the detailed descriptions for all of the AVIC registers.

9.2.1 Memory Map

Table 9-2 shows the AVIC memory map.

Table 9-2. AVIC Memory Map

Address	Register	Access	Reset Value	Section/Page
General Registers				
0x6800_0000 (INTCNTL)	Interrupt Control Register	R/W	0x0000_0000	9.2.3.1/9-10
0x6800_0004 (NIMASK)	Normal Interrupt Mask Register	R/W	0x0000_001F	9.2.3.2/9-11

Table 9-2. AVIC Memory Map (continued)

Address	Register	Access	Reset Value	Section/Page
0x6800_0008 (INTENNUM)	Interrupt Enable Number Register	R/W	0x0000_0000	9.2.3.3/9-12
0x6800_000C (INTDISNUM)	Interrupt Disable Number Register	R/W	0x0000_0000	9.2.3.4/9-13
0x6800_0010 (INTENABLEH)	Interrupt Enable Register High	R/W	0x0000_0000	9.2.3.5/9-14
0x6800_0014 (INTENABLEL)	Interrupt Enable Register Low	R/W	0x0000_0000	9.2.3.5/9-14
0x6800_0018 (INTTYPEH)	Interrupt Type Register High	R/W	0x0000_0000	9.2.3.6/9-15
0x6800_001C (INTTYPEL)	Interrupt Type Register Low	R/W	0x0000_0000	9.2.3.6/9-15
0x6800_0020 (NIPRIORITY7)	Normal Interrupt Priority Level Register 7	R/W	0x0000_0000	9.2.3.7/9-16
0x6800_0024 (NIPRIORITY6)	Normal Interrupt Priority Level Register 6	R/W	0x0000_0000	9.2.3.7/9-16
0x6800_0028 (NIPRIORITY5)	Normal Interrupt Priority Level Register 5	R/W	0x0000_0000	9.2.3.7/9-16
0x6800_002C (NIPRIORITY4)	Normal Interrupt Priority Level Register 4	R/W	0x0000_0000	9.2.3.7/9-16
0x6800_0030 (NIPRIORITY3)	Normal Interrupt Priority Level Register 3	R/W	0x0000_0000	9.2.3.7/9-16
0x6800_0034 (NIPRIORITY2)	Normal Interrupt Priority Level Register 2	R/W	0x0000_0000	9.2.3.7/9-16
0x6800_0038 (NIPRIORITY1)	Normal Interrupt Priority Level Register 1	R/W	0x0000_0000	9.2.3.7/9-16
0x6800_003C (NIPRIORITY0)	Normal Interrupt Priority Level Register 0	R/W	0x0000_0000	9.2.3.7/9-16
0x6800_0040 (NIVECSR)	Normal Interrupt Vector and Status Register	R	0xFFFF_FFFF	9.2.3.8/9-24
0x6800_0044 (FIVECSR)	Fast Interrupt Vector and Status Register	R	0xFFFF_FFFF	9.2.3.9/9-25
0x6800_004C (INTSRCL)	Interrupt Source Register High	R	0x0000_0000	9.2.3.10/9-26
0x6800_004C (INTSRCL)	Interrupt Source Register Low	R	0x0000_0000	9.2.3.10/9-26
0x6800_0050 (INTFRCH)	Interrupt Force Register High	R/W	0x0000_0000	9.2.3.11/9-27
0x6800_0054 (INTFRCL)	Interrupt Force Register Low	R/W	0x0000_0000	9.2.3.11/9-27

Table 9-2. AVIC Memory Map (continued)

Address	Register	Access	Reset Value	Section/Page
0x6800_0058 (NIPNDH)	Normal Interrupt Pending Register High	R	0x0000_0000	9.2.3.12/9-28
0x6800_005C (NIPNDL)	Normal Interrupt Pending Register High	R	0x0000_0000	9.2.3.12/9-28
0x6800_0060 (FIPNDH)	Fast Interrupt Pending Register High	R	0x0000_0000	9.2.3.13/9-29
0x6800_0064 (FIPNDL)	Fast Interrupt Pending Register Low	R	0x0000_0000	9.2.3.13/9-29
0x6800_0100 (VECTOR0)–0x6800_01FC (VECTOR63) through 0x6800_0100 (VECTOR0)–0x6800_01FC (VECTOR63)	Vector Register 0 through Vector Register 63	R/W	0x0000_0000	9.2.3.14/9-31

9.2.2 Register Summary

Figure 9-2 shows the key to the register fields and Table 9-3 shows the register figure conventions.

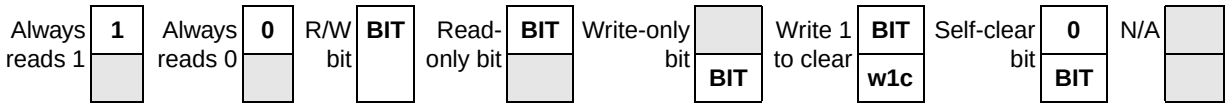


Figure 9-2. Key to Register Fields

Table 9-3. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	

Table 9-3. Register Figure Conventions (continued)

Convention	Description
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 9-4 shows the AVIC register summary.

Table 9-4. AVIC Detailed Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x6800_0000 (INTCNTL)	R	0	0	0	0	0	0	ABFLAG	ABFEN	0	NIDIS	FIDIS	NIAD	FIAD	NM	0	0
	W							r/wfc									
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
0x6800_0004 (NIMASK)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	NIMASK				
	W																
0x6800_0008 (INTENNUM)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	ENNUM					
	W											slfclr	slfclr	slfclr	slfclr	slfclr	slfclr
0x6800_000C (INTDISNUM)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	DISNUM					
	W											slfclr	slfclr	slfclr	slfclr	slfclr	slfclr
0x6800_0010 (INTENABLEH)	R	INTENABLE[63:32]															
	W																
	R	INTENABLE[63:32]															
	W																

Table 9-4. AVIC Detailed Register Summary (continued)

Name		31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0			
0x6800_0014 (INTENABLEL)	R	INTENABLE[31:16]			
	W				
	R	INTENABLE[15:1]			
	W				
0x6800_0018 (INTTYPEH)	R	INTTYPE[63:32]			
	W				
	R	INTTYPE[63:32]			
	W				
0x6800_001C (INTTYPEL)	R	INTTYPE[31:0]			
	W				
	R	INTTYPE[31:0]			
	W				
0x6800_0020 (NIPRIORITY7)	R	NIPR63	NIPR62	NIPR61	NIPR60
	W				
	R	NIPR59	NIPR58	NIPR57	NIPR56
	W				
0x6800_0024 (NIPRIORITY6)	R	NIPR55	NIPR54	NIPR53	NIPR52
	W				
	R	NIPR51	NIPR50	NIPR49	NIPR48
	W				
0x6800_0028 (NIPRIORITY5)	R	NIPR47	NIPR46	NIPR45	NIPR44
	W				
	R	NIPR43	NIPR42	NIPR41	NIPR40
	W				
0x6800_002C (NIPRIORITY4)	R	NIPR39	NIPR38	NIPR37	NIPR36
	W				
	R	NIPR35	NIPR34	NIPR33	NIPR32
	W				
0x6800_0030 (NIPRIORITY3)	R	NIPR31	NIPR30	NIPR29	NIPR28
	W				
	R	NIPR27	NIPR26	NIPR25	NIPR24
	W				

Table 9-4. AVIC Detailed Register Summary (continued)

Name		31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0x6800_0034 (NIPRIORITY2)	R	NIPR23	NIPR22	NIPR21	NIPR20
	W				
	R	NIPR19	NIPR18	NIPR17	NIPR16
	W				
0x6800_0038 (NIPRIORITY1)	R	NIPR15	NIPR14	NIPR13	NIPR12
	W				
	R	NIPR11	NIPR10	NIPR9	NIPR8
	W				
0x6800_003C (NIPRIORITY0)	R	NIPR7	NIPR6	NIPR5	NIPR4
	W				
	R	NIPR3	NIPR2	NIPR1	NIPR0
	W				
0x6800_0040 (NIVECSR)	R	NIVECTOR			
	W				
	R	NIPRILVL			
	W				
0x6800_0044 (FIVECSR)	R	FIVECTOR[31:16]			
	W				
	R	FIVECTOR[15:0]			
	W				
0x6800_0048 (INTSRCH)	R	INTIN[63:32]			
	W				
	R	INTIN[63:32]			
	W				
0x6800_004C (INTSRCL)	R	INTIN[31:0]			
	W				
	R	INTIN[31:0]			
	W				
0x6800_0050 (INTFRCH)	R	FORCE[63:32]			
	W				
	R	FORCE[63:32]			
	W				

Table 9-4. AVIC Detailed Register Summary (continued)

Name		31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0x6800_0054 (INTFRCL)	R	FORCE[31:0]
	W	
	R	FORCE[31:0]
	W	
0x6800_0058 (NIPNDH)	R	NIPEND[63:32]
	W	
	R	NIPEND[63:32]
	W	
0x6800_005C (NIPNDL)	R	NIPEND[31:0]
	W	
	R	NIPEND[31:0]
	W	
0x6800_0060 (FIPNDH)	R	FIPEND[63:32]
	W	
	R	FIPEND[63:32]
	W	
0x6800_0064 (FIPNDL)	R	FIPEND[31:0]
	W	
	R	FIPEND[31:0]
	W	
0x6800_0100 (VECTOR0)–0x6800_01FC (VECTOR63)	R	VECTOR0
	W	
	R	VECTOR0 0 0
	W	
0x6800_0100 (VECTOR0)–0x6800_01FC (VECTOR63)	R	VECTOR63
	W	
	R	VECTOR63 0 0
	W	

9.2.3 Register Descriptions

This section contains the detailed register descriptions for the AVIC registers.

9.2.3.1 Interrupt Control Register

The interrupt control register (INTCNTL) controls the hardware acceleration done by the AVIC. Both normal interrupts and fast interrupts can be enabled to jump directly to the interrupt service routine.

This register is located on the ARM1136JF-S native bus, accessible in 1 cycle, and can be accessed only in privileged mode. This register can be modified only using 32-bit writes. Figure 9-3 shows the register; Table 9-5 provides its field descriptions.

0x6800_0000 (INTCNTL) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	ABFLAG	ABFEN	0	NIDIS	FIDIS	NIAD	FIAD	NM	0	0
W							w1c									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-3. INTCNTL Register

Table 9-5. INTCNTL Field Descriptions

Field	Description
31–26	Reserved
25 ABFLAG	Core Arbitration Prioritization Risen Flag. This status bit indicates that the AVIC is asserting the avic_rise_arb signal to rise the priority level of the ARM core in the bus arbitration logic. This signal is directly tied to the avic_rise_arb output signal. When the ABFEN bit is cleared, this bit will indicate the current value of the avic_rise_arb signal to the ARM core. This bit will be set if the nFIQ is asserted and the FIAD bit is set or if the nIRQ is asserted and the NIAD bit is set. When the ABFEN bit is set, this bit will be keep the “1” until written to with a “1”. This bit will remain set on either of the above conditions normally set the ABFLAG bit. 0 AVIC is not affecting the bus arbitration priority levels 1 AVIC is rising ARM core priority level in bus arbitration logic.
24 ABFEN	ABFLAG Sticky Enable. This bit controls whether the ABFLAG bit is “sticky.” This allows the arbitration logic to keep the ARM core at the higher priority level until the ABFLAG bit is written. 0 ABFLAG bit is normal. 1 ABFLAG bit is “sticky” and requires a write of “1” to clear the bit.
23	Reserved

Table 9-5. INTCNTL Field Descriptions (continued)

Field	Description
22 NIDIS	Normal Interrupt Disable. This bit, when set, disables the generation of the normal interrupt signal. This bit is similar to the I bit of the ARM1136JF-S core. This bit along with the FIDIS bit is used to enable secure operations. 0 Does not affect the normal interrupt generation 1 Disable all normal interrupts
21 FIDIS	Fast Interrupt Disable. This bit, when set, disables the generation of the fast interrupt signal. This bit is similar to the F bit of the ARM1136JF-S core. This bit along with the NIDIS bit is used to enable secure operations. 0 Does not affect the fast interrupt generation 1 Disable all fast interrupts
20 NIAD	Normal Interrupt Arbiter Rise ARM Level. This bit, when asserted, increases the bus arbitration priority of the ARM core when the normal interrupt signal (nIRQ) is asserted. If an alternate master has ownership of the bus when a normal interrupt occurs, the bus will be given back to the processor core *after* the DMA device has completed its accesses. The NIAD bit does not affect alternate master accesses that are in progress. To prevent an alternate master from accessing the bus during an interrupt service routine, the interrupt flag should not be cleared until the end of the service routine. Another option is to use the ABFEN and ABFLAG bits. 0 Disregard the normal interrupt flag when evaluating bus requests 1 Normal interrupt flag increases the bus arbitration priority of the ARM core to decrease the latency of the interrupt service routine.
19 FIAD	Fast Interrupt Arbiter Rise ARM Level. This bit functions the same as the NIAD bit except for the fast interrupts (nFIQ). 0 Disregard the fast interrupt flag when evaluating bus requests 1 Fast interrupt flag increases the bus arbitration priority of the ARM core to decrease the latency of the interrupt service routine.
18 NM	Normal Interrupt Mode Control. Selects the hardware acceleration mode for normal interrupt. 0 AVIC will not accelerate nIRQ servicing, that is, complete software control 1 Normal interrupt hardware acceleration enabled, that is, enable AVIC interface for Normal interrupts
17–0	Reserved

9.2.3.2 Normal Interrupt Mask Register

The normal interrupt mask register (NIMASK) controls the normal interrupt mask level. All normal interrupts with a priority level lower than or equal to the NIMASK will be disabled. The priority level of normal interrupts are determined by the normal interrupt priority level registers (NIPRIORITY7, NIPRIORITY6, NIPRIORITY5, NIPRIORITY4, NIPRIORITY3, NIPRIORITY2, NIPRIORITY1, and NIPRIORITY0). The reset state of this register will not disable any normal interrupts.

Writing all ones, or –1, to the NIMASK will set the normal interrupt mask to –1, which will not disable any normal interrupt priority levels.

This hardware mechanism can be used to create reentrant normal interrupt routines by disabling lower priority normal interrupts. Refer to Section 9.3.6, “Writing Reentrant Normal Interrupt Routines” for more details on the use of the NIMASK register.

This register is located on the ARM1136JF-S native bus, accessible in 1 cycle, and can be accessed only in privileged mode. This register can be modified only using 32-bit writes. Figure 9-4 shows the register; Table 9-6 provides its field descriptions.

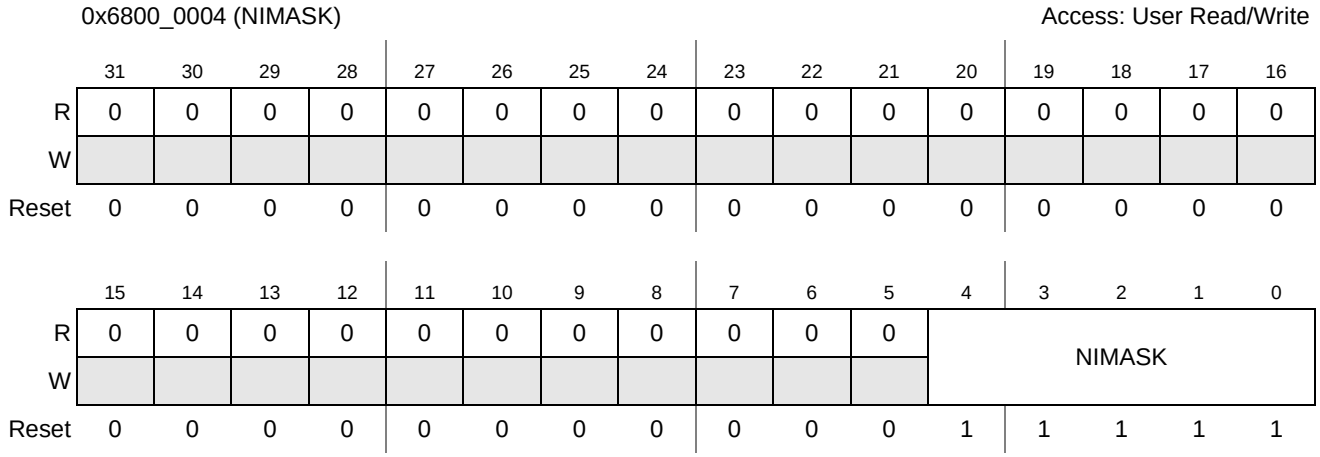


Figure 9-4. Normal Interrupt Mask Register

Table 9-6. NIMASK Descriptions

Field	Description
31–5	Reserved
4–0 NIMASK	Normal Interrupt Mask. Controls normal interrupt mask level. All normal interrupts of priority level lower than or equal to the NIMASK will be disabled. –1 Do not disable any normal interrupts 0 Disable priority level 0 normal interrupts 1 Disable priority level 1 and lower normal interrupts —— 15 Disable all normal interrupts 16+ Do not disable any normal interrupts

9.2.3.3 Interrupt Enable Number Register

The interrupt enable number register (INTENNUM) provides a hardware accelerated enabling of interrupts. Any write to this register will enable one interrupt source. If the six LSBs are equal 000000, then interrupt source 0 is enabled; if the six LSBs equal 000001, then interrupt source 1 is enabled; and so forth. This register is decoded into a one hot mask which will be logically OR'ed with the INTENABLEH/INTENABLEL register.

This hardware mechanism alleviates the need for an atomic read/modify/write sequence to enable an interrupt source. To enable interrupts 10 and 20, the software need only preform two writes to the AVIC: first write 10 to INTENNUM register, then write 20 to INTENNUM register (the order of the writes is irrelevant).

This register is located on the ARM1136JF-S native bus, accessible in 1 cycle, and can be accessed only in privileged mode. This register can be modified only using 32-bit writes. This register always reads back all 0s. Figure 9-5 shows the register; Table 9-7 provides its field descriptions.

0x6800_0008 (INTENNUM) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W											ENNUM					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-5. Interrupt Enable Number Register

Table 9-7. INTENNUM Field Descriptions

Field	Description
31–6	Reserved
5–0 ENNUM	Interrupt Enable Number. Writing to this register will enable the interrupt source associated with this value. 0 Enable interrupt source 0 1 Enable interrupt source 1 —— 63 Enable interrupt source 63

9.2.3.4 Interrupt Disable Number Register

The interrupt disable number register (INTDISNUM) provides a hardware accelerated disabling of interrupts. Any write to this register will disable one interrupt source. If the six LSBs are equal 000000, then interrupt source 0 is disabled; if the six LSBs equal 000001, then interrupt source 1 is disabled; and so forth. This register is decoded into a one hot mask which will be inverted and logically AND'ed with the INTENABLEH/INTENABLEL register.

This hardware mechanism alleviates the need for an atomic read/modify/write sequence to disable an interrupt source. To disable interrupts 10 and 20, the software need only preform two writes to the AVIC: first write 10 to INTDISNUM register, then write 20 to INTDISNUM register (the order of the writes is irrelevant).

This register is located on the ARM1136JF-S native bus, accessible in 1 cycle, and can be accessed only in privileged mode. This register can be modified only using 32-bit writes. This register will always read back all 0s. Figure 9-6 shows the register; Table 9-8 provides its field descriptions.

0x6800_000C (INTDISNUM) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W											DISNUM					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-6. Interrupt Disable Number Register

Table 9-8. INTDISNUM Field Descriptions

Field	Description
31–6	Reserved
5–0 ENNUM	Interrupt Disable Number. Writing to this register will disable the interrupt source associated with this value. 0 Disable interrupt source 0 1 Disable interrupt source 1 —— 63 Disable interrupt source 63

9.2.3.5 Interrupt Enable Registers

The interrupt enable register high (INTENABLEH) and the interrupt enable register low (INTENABLEL) are used to enable pending interrupt requests to the core. Each bit in this register corresponds to an interrupt source available in the system. The reset state of these registers are all interrupts masked.

This register can be updated by various methods: writing directly to the INTENABLEH/INTENABLEL registers, setting bits with the INTENNUM register, or clearing bits with the INTDISNUM register.

These registers are located on the ARM1136JF-S native bus, accessible in 1 cycle, and can be accessed only in privileged mode. These registers can be modified only using 32-bit writes. Figure 9-7 and Figure 9-8 show the registers; Table 9-9 provides their field descriptions.

0x6800_0010 (INTENABLEH) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	INTENABLE[63:48]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INTENABLE[47:32]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-7. Interrupt Enable Register High

0x6800_0014 (INTENABLEL) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	INTENABLE[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INTENABLE[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-8. Interrupt Enable Register Low

Table 9-9. INTENABLEH/L Field Descriptions

Field	Description
31–0 INTENABLEH 31–0 INTENABLEL	Interrupt Enable. This bit enables the corresponding interrupt source to request a normal interrupt or a fast interrupt. A reset operation clears this bit. If an enable bit is set and the corresponding interrupt source is asserted, the interrupt controller asserts a normal or a fast interrupt request depending on associated INTTYPEH/INTTYPEL setting. 0 Interrupt disabled 1 Interrupt enabled and will generate a normal or fast interrupt upon assertion

9.2.3.6 Interrupt Type Registers

The interrupt type register high (INTTYPEH) and the interrupt type register low (INTTYPEL) are used to select whether a pending interrupt source—when enabled with the INTENABLEH/INTENABLEL—will create a normal interrupt or a fast interrupt to the core. Each bit in this register corresponds to an interrupt source available in the system. The reset state of these registers will cause all enabled interrupt sources to generate a normal interrupt.

These registers are located on the ARM1136JF-S native bus, accessible in 1 cycle, and can be accessed only in privileged mode. These registers can be modified only using 32-bit writes. Figure 9-9 and Figure 9-10 show the registers; Table 9-10 provides their field descriptions.

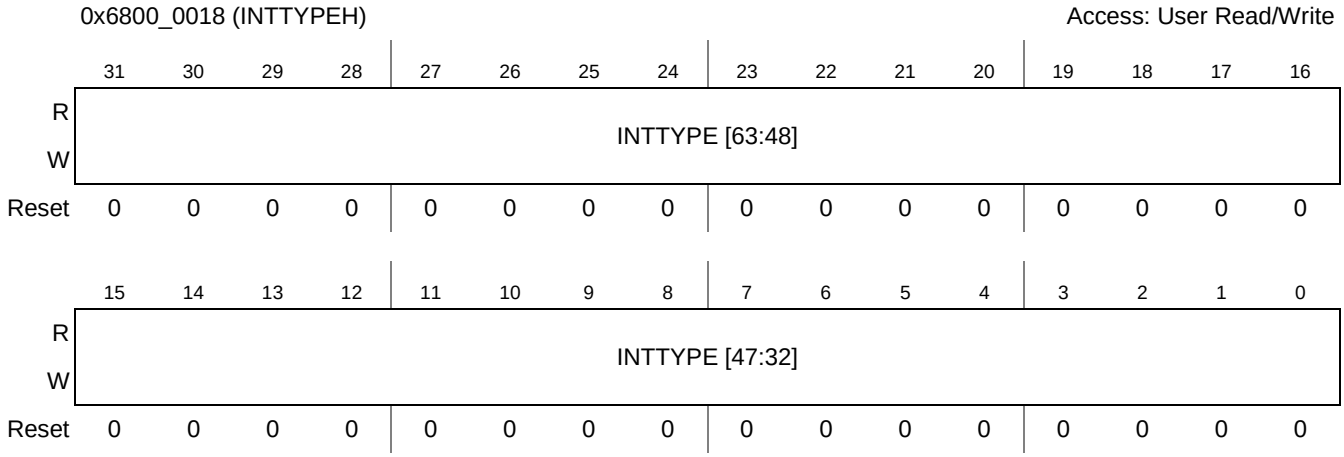


Figure 9-9. Interrupt Type Register High

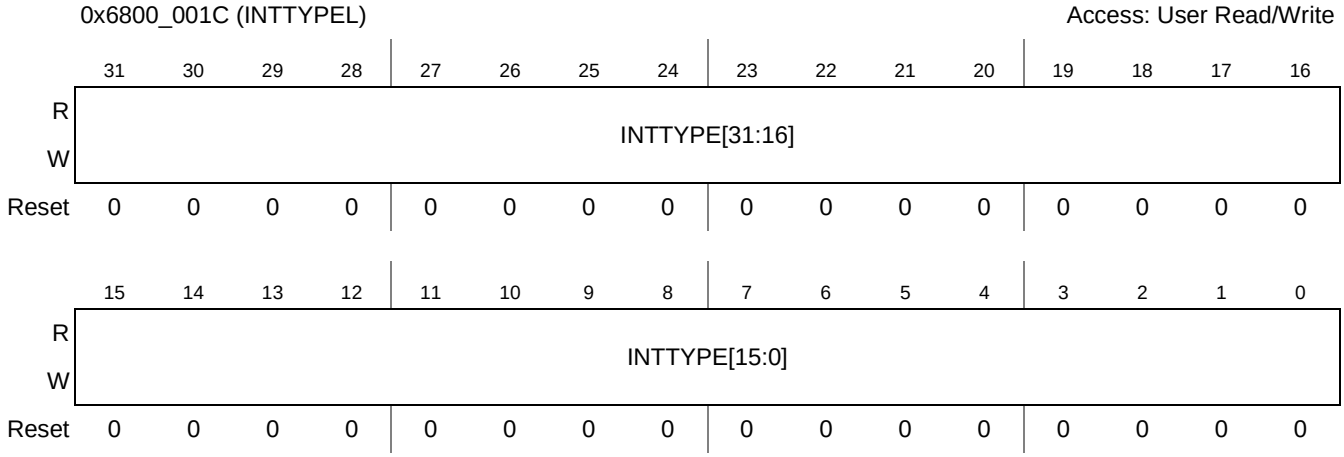


Figure 9-10. Interrupt Type Register Low

Table 9-10. INTTYPEH/INTTYPEL Field Descriptions

Field	Description
31–0 INTTYPEH 31–0 INTTYPEL	Interrupt Type. This bit controls whether the corresponding interrupt source will request a normal interrupt or a fast interrupt. If a INTTYPE bit is set and the corresponding interrupt source is asserted, the interrupt controller will assert a fast interrupt request. 0 Interrupt source will generate a normal interrupt (nIRQ). 1 Interrupt source will generate a fast interrupt (nFIQ).

9.2.3.7 Normal Interrupt Priority Level Registers

The normal interrupt priority level registers (NIPRIORITY7, NIPRIORITY6, NIPRIORITY5, NIPRIORITY4, NIPRIORITY3, NIPRIORITY2, NIPRIORITY1, and NIPRIORITY0) provide a

software controllable prioritization of normal interrupts. Normal interrupts with a higher priority level will preempt normal interrupts with a lower priority. The reset state of these registers forces all normal interrupts to the lowest priority level.

If a level 0 normal interrupt and a level 1 normal interrupt are asserted at the same time, the level 1 normal interrupt will be selected assuming that NIMASK has not disabled level 1 normal interrupts. If two level 1 normal interrupts are asserted at the same time, the level 1 normal interrupt with the highest source number will be selected, also assuming that NIMASK has not disabled level 1 normal interrupts.

These registers are located on the ARM1136JF-S native bus, accessible in 1 cycle, and can be accessed only in privileged mode. These registers can be modified only using 32-bit writes. Figure 9-11 through Figure 9-18 show the registers; Table 9-11 through Table 9-18 provide their field descriptions.

0x6800_0020 (NIPRIORITY7) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	NIPR63				NIPR62				NIPR61				NIPR60			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	NIPR59				NIPR58				NIPR57				NIPR56			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-11. Normal Interrupt Priority Level 7 Register

Table 9-11. NIPRIORITY7 Register Field Descriptions

Field	Description
31–28 NIPR63 27–24 NIPR62 23–20 NIPR61 19–16 NIPR60	Normal Interrupt Priority Level. Selects the software controlled priority level for the associated normal interrupt source. These registers do not affect the prioritization of fast interrupt priorities. 0 Lowest priority normal interrupt —— 15 Highest priority normal interrupt
15–12 NIPR59 11–8 NIPR58 7–4 NIPR57 3–0 NIPR56	Normal Interrupt Priority Level. Selects the software controlled priority level for the associated normal interrupt source. These registers do not affect the prioritization of fast interrupt priorities. 0 Lowest priority normal interrupt —— 15 Highest priority normal interrupt

0x6800_0024 (NIPRIORITY6) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	NIPR55				NIPR54				NIPR53				NIPR52			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	NIPR51				NIPR50				NIPR49				NIPR48			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-12. Normal Interrupt Priority Level 6 Register

Table 9-12. NIPRIORITY6 Register Field Descriptions

Field	Description
31–28 NIPR55 27–24 NIPR54 23–20 NIPR53 19–16 NIPR52	Normal Interrupt Priority Level. Selects the software controlled priority level for the associated normal interrupt source. These registers do not affect the prioritization of fast interrupt priorities. 0 Lowest priority normal interrupt —— 15 Highest priority normal interrupt
15–12 NIPR51 11–8 NIPR50 7–4 NIPR49 3–0 NIPR48	Normal Interrupt Priority Level. Selects the software controlled priority level for the associated normal interrupt source. These registers do not affect the prioritization of fast interrupt priorities. 0 Lowest priority normal interrupt —— 15 Highest priority normal interrupt

0x6800_0028 (NIPRIORITY5) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	NIPR47				NIPR46				NIPR45				NIPR44			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	NIPR43				NIPR42				NIPR41				NIPR40			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-13. Normal Interrupt Priority Level 5 Register

Table 9-13. NIPRIORITY5 Register Field Descriptions

Field	Description
31–28 NIPR47 27–24 NIPR46 23–20 NIPR45 19–16 NIPR44	Normal Interrupt Priority Level. Selects the software controlled priority level for the associated normal interrupt source. These registers do not affect the prioritization of fast interrupt priorities. 0 Lowest priority normal interrupt —— 15 Highest priority normal interrupt
15–12 NIPR43 11–8 NIPR42 7–4 NIPR41 3–0 NIPR40	Normal Interrupt Priority Level. Selects the software controlled priority level for the associated normal interrupt source. These registers do not affect the prioritization of fast interrupt priorities. 0 Lowest priority normal interrupt —— 15 Highest priority normal interrupt

0x6800_002C (NIPRIORITY4) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	NIPR39				NIPR38				NIPR37				NIPR36			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	NIPR35				NIPR34				NIPR33				NIPR32			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-14. Normal Interrupt Priority Level 4 Register

Table 9-14. NIPRIORITY4 Register Field Descriptions

Field	Description
31–28 NIPR39 27–24 NIPR38 23–20 NIPR37 19–16 NIPR36	Normal Interrupt Priority Level. Selects the software controlled priority level for the associated normal interrupt source. These registers do not affect the prioritization of fast interrupt priorities. 0 Lowest priority normal interrupt —— 15 Highest priority normal interrupt
15–12 NIPR35 11–8 NIPR34 7–4 NIPR33 3–0 NIPR32	Normal Interrupt Priority Level. Selects the software controlled priority level for the associated normal interrupt source. These registers do not affect the prioritization of fast interrupt priorities. 0 Lowest priority normal interrupt —— 15 Highest priority normal interrupt

0x6800_0030 (NIPRIORITY3) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	NIPR31				NIPR30				NIPR29				NIPR28			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	NIPR27				NIPR26				NIPR25				NIPR24			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-15. Normal Interrupt Priority Level 4 Register

Table 9-15. NIPRIORITY3 Register Field Descriptions

Field	Description
31–28 NIPR31 27–24 NIPR30 23–20 NIPR29 19–16 NIPR28	Normal Interrupt Priority Level. Selects the software controlled priority level for the associated normal interrupt source. These registers do not affect the prioritization of fast interrupt priorities. 0 Lowest priority normal interrupt —— 15 Highest priority normal interrupt
15–12 NIPR27 11–8 NIPR26 7–4 NIPR25 3–0 NIPR24	Normal Interrupt Priority Level. Selects the software controlled priority level for the associated normal interrupt source. These registers do not affect the prioritization of fast interrupt priorities. 0 Lowest priority normal interrupt —— 15 Highest priority normal interrupt

0x6800_0034 (NIPRIORITY2) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	NIPR23				NIPR22				NIPR21				NIPR20			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	NIPR19				NIPR18				NIPR17				NIPR16			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-16. Normal Interrupt Priority Level 2 Register

Table 9-16. NIPRIORITY2 Register Field Descriptions

Field	Description
31–28 NIPR23 27–24 NIPR22 23–20 NIPR21 19–16 NIPR20	Normal Interrupt Priority Level. Selects the software controlled priority level for the associated normal interrupt source. These registers do not affect the prioritization of fast interrupt priorities. 0 Lowest priority normal interrupt —— 15 Highest priority normal interrupt
15–12 NIPR19 11–8 NIPR18 7–4 NIPR17 3–0 NIPR16	Normal Interrupt Priority Level. Selects the software controlled priority level for the associated normal interrupt source. These registers do not affect the prioritization of fast interrupt priorities. 0 Lowest priority normal interrupt —— 15 Highest priority normal interrupt

0x6800_0038 (NIPRIORITY1) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	NIPR15				NIPR14				NIPR13				NIPR12			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	NIPR11				NIPR10				NIPR9				NIPR8			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-17. Normal Interrupt Priority Level 1 Register

Table 9-17. NIPRIORITY1 Register Field Descriptions

Field	Description
31–28 NIPR15 27–24 NIPR14 23–20 NIPR13 19–16 NIPR12	Normal Interrupt Priority Level. Selects the software controlled priority level for the associated normal interrupt source. These registers do not affect the prioritization of fast interrupt priorities. 0 Lowest priority normal interrupt —— 15 Highest priority normal interrupt
15–12 NIPR11 11–8 NIPR10 7–4 NIPR9 3–0 NIPR8	Normal Interrupt Priority Level. Selects the software controlled priority level for the associated normal interrupt source. These registers do not affect the prioritization of fast interrupt priorities. 0 Lowest priority normal interrupt —— 15 Highest priority normal interrupt

0x6800_003C (NIPRIORITY0) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	NIPR7				NIPR6				NIPR5				NIPR4			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	NIPR3				NIPR2				NIPR1				NIPR0			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-18. Normal Interrupt Priority Level 0 Register

Table 9-18. NIPRIORITY0 Register Field Descriptions

Field	Description
31–28 NIPR7 27–24 NIPR6 23–20 NIPR5 19–16 NIPR4	Normal Interrupt Priority Level. Selects the software controlled priority level for the associated normal interrupt source. These registers do not affect the prioritization of fast interrupt priorities. 0 Lowest priority normal interrupt —— 15 Highest priority normal interrupt
15–12 NIPR3 11–8 NIPR2 7–4 NIPR1 3–0 NIPR0	Normal Interrupt Priority Level. Selects the software controlled priority level for the associated normal interrupt source. These registers do not affect the prioritization of fast interrupt priorities. 0 Lowest priority normal interrupt —— 15 Highest priority normal interrupt

9.2.3.8 Normal Interrupt Vector and Status Register

The normal interrupt vector and status register (NIVECSR) provides the priority of the highest pending normal interrupt and provides the vector index of the interrupt's service routine. This hardware mechanism replaces the previous necessity for core support of the FF1 command. This number can be directly used as an index into a vector table to select the highest pending normal interrupt source.

This read-only register is located on the ARM1136JF-S native bus, accessible in 1 cycle, and can be accessed only in privileged mode. Figure 9-19 shows the register; Table 9-19 provides its field descriptions.

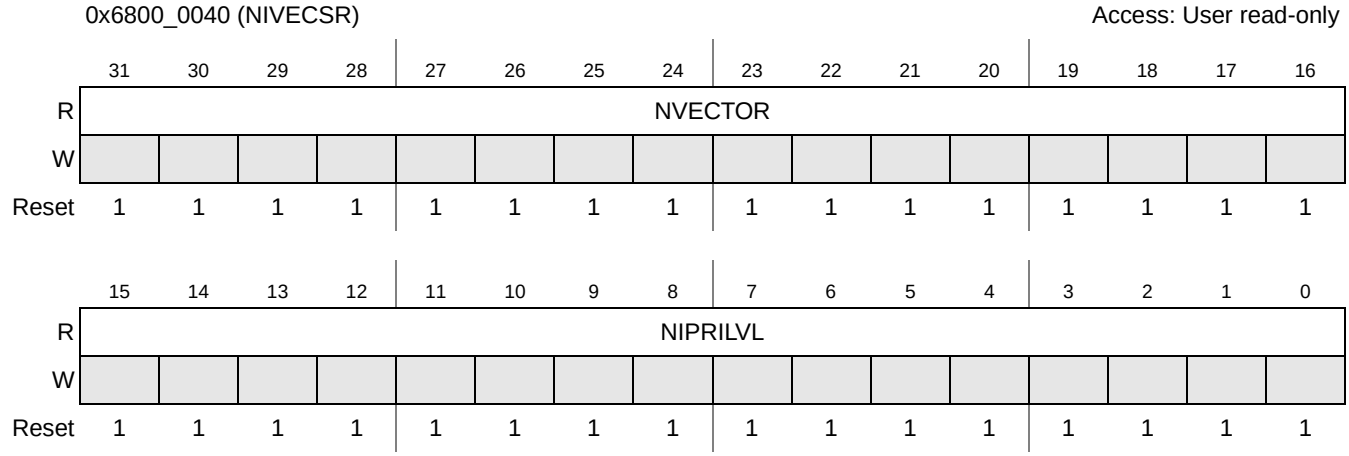


Figure 9-19. Normal Interrupt Vector and Status Register

Table 9-19. NIVECSR Register Field Descriptions

Field	Description
31–16 NIVECTOR	Normal Interrupt Vector. Indicates vector index for the highest pending normal interrupt. –1 No normal interrupt request pending 0 Interrupt 0 highest priority pending normal interrupt 1 Interrupt 1 highest priority pending normal interrupt —— 63 Interrupt 63 highest priority pending normal interrupt 64+ (not -1)unused, will not occur
15–0 NIPRILVL	Normal Interrupt Priority Level. Indicates the priority level of the highest priority normal interrupt. This number can be written to the NIMASK to disable the current priority normal interrupts to build a reentrant normal interrupt system. –1 No normal interrupt request pending 0 Highest priority normal interrupt is level 0 1 Highest priority normal interrupt is level 1 —— 15 Highest priority normal interrupt is level 15 16+ (not –1)unused, will not occur

9.2.3.9 Fast Interrupt Vector and Status Register

The fast interrupt vector and status register (FIVECSR) provides the vector index for the highest priority active fast interrupt's service routine (the higher the source number of the fast interrupt, the higher the priority level). This hardware mechanism replaces the previous necessity for core support of the FF1 command. This number can be directly used as an index into a vector table to select the highest pending fast interrupt source.

This read-only register is located on the ARM1136JF-S native bus, accessible in 1 cycle, and can be accessed only in privileged mode. Figure 9-20 shows the register; Table 9-20 provides its field descriptions.

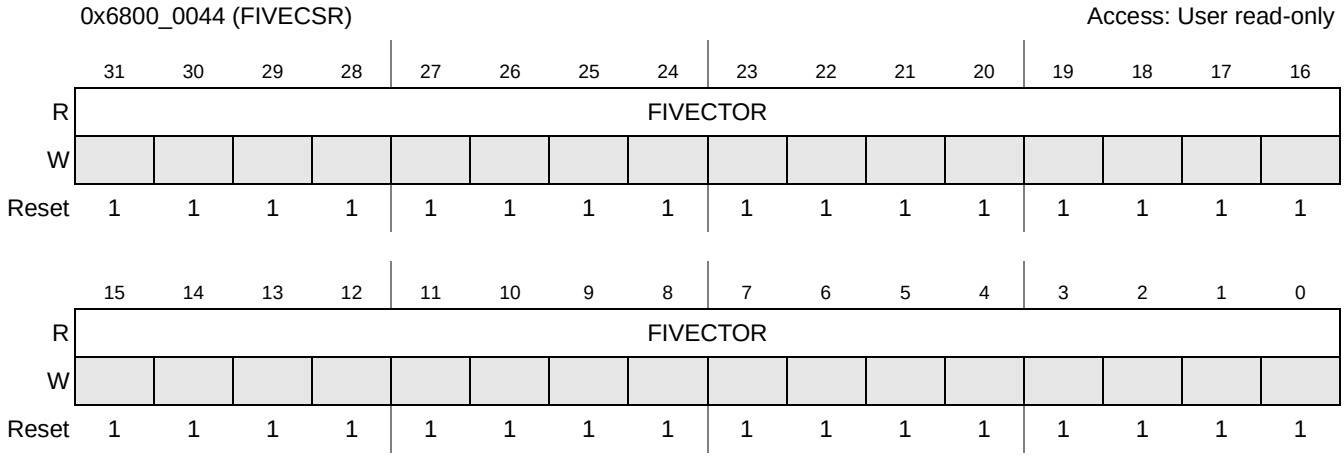


Figure 9-20. Fast Interrupt Vector and Status Register

Table 9-20. FIVECSR Register Field Descriptions

Field	Description
31–0 FIVECTOR	Fast Interrupt Vector. Indicates vector index for the highest pending fast interrupt. –1 No fast interrupt request pending 0 Interrupt 0 highest pending fast interrupt 1 Interrupt 1 highest pending fast interrupt —— 63 Interrupt 63 highest pending fast interrupt 64+ (not –1)unused, will not occur

9.2.3.10 Interrupt Source Registers

The interrupt source register high (INTSRCH) and the interrupt source register low (INTSRCL) are each 32-bits wide. INTSRCH and INTSRCL reflect the status of all interrupt request inputs into the interrupt controller (see Table 2-4 in Chapter 2, “System Memory Map, Interrupts, and SDMA Events” for details). Unused bit positions always read zero (no request pending). The state of this register out of reset is determined by the peripheral circuits generating the requests; normally, the requests would be inactive.

These read-only registers are located on the ARM1136JF-S native bus, accessible in 1 cycle, and can be accessed only in privileged mode. Figure 9-21 and Figure 9-22 show the registers; Table 9-21 provides their field descriptions.

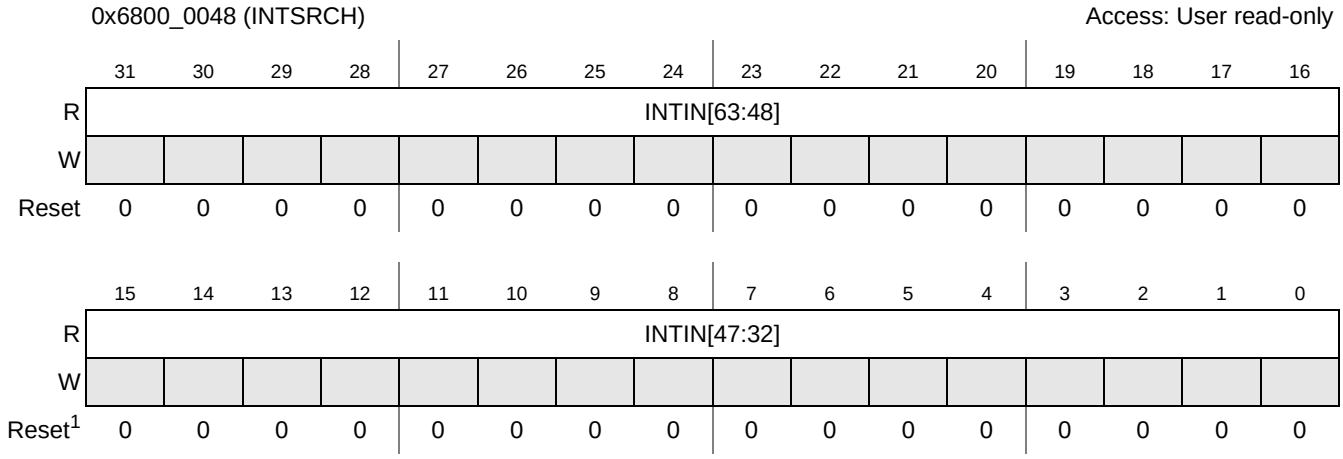


Figure 9-21. INTSRCH Register

1 The state of the register out of reset is determined by the peripheral circuits generating the requests; normally, the requests would be inactive. This read-only register should be only accessed with 32-bit reads.

0x6800_004C (INTSRCL) Access: User read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	INTIN[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	INTIN[15:0]															
W																
Reset[1]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-22. INTSRCL Register

1 The state of the register out of reset is determined by the peripheral circuits generating the requests; normally, the requests would be inactive. This read-only register should be only accessed with 32-bit reads.

Table 9-21. INTSRCH/INTSRCL Field Descriptions

Field	Description
31–16 INTIN[63:48] INTIN[31:16] 15–0 INTIN[47:30] INTIN[15:0]	Interrupt Source. Indicates the state of the corresponding hardware interrupt source. 0 Interrupt source negated 1 Interrupt source asserted

9.2.3.11 Interrupt Force Registers

The interrupt force register high (INTFRCH) and the interrupt force register low (INTFRCL) are each 32-bits wide. The interrupt force registers allow for software generation of interrupts for each of the possible interrupt sources for functional or debug purposes. The system level design may reserve one or more sources for software purposes to allow software to self-schedule interrupts by forcing one or more of these “sources” in the appropriate interrupt force register(s).

These registers are located on the ARM1136JF-S native bus, accessible in 1 cycle, and can be accessed only in privileged mode. These registers can be modified only using 32-bit writes. Figure 9-23 and Figure 9-24 show the registers; Table 9-22 provides their field descriptions.

0x6800_0050 (INTFRCH) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	FORCE[63:48]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	FORCE[47:32]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-23. INTFRCH Register

0x6800_0054 (INTFRCL) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	FORCE[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	FORCE15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-24. INTFRCL Register

Table 9-22. INTFRCH/INTFRCL Field Descriptions

Field	Description
31–16 FORCE[63:48] FORCE[31:16] 15–0 FORCE[47:30] FORCE[15:0]	Interrupt Source Force Request. Used to force a request for the corresponding interrupt source. 0 Standard interrupt operation 1 Interrupt forced asserted

9.2.3.12 Normal Interrupt Pending Register

The normal interrupt pending register high (NIPNDH) and the normal interrupt pending register low (NIPNDL) are 32-bit wide registers used to monitor the outputs of the enable and masking operations. These registers are actually only a set of buffers; therefore, the reset state of these registers are determined by the normal interrupt enable registers, the interrupt mask register, and the interrupt source registers. The value reflected in these registers is unaffected by the value of the NIMASK register.

These read-only registers are located on the ARM1136JF-S native bus, accessible in 1 cycle, and can be accessed only in privileged mode. Figure 9-25 and Figure 9-26 show the registers; Table 9-23 provides their field descriptions.

0x6800_0058 (NIPNDH) Access: User read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	INPEND[63:48]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	INPEND[47:32]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-25. NIPNDH Register

0x6800_005C (NIPNDL) Access: User read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	INPEND[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	INPEND[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-26. NIPNDL Register

Table 9-23. NIPNDH/NIPNDL Field Descriptions

Field	Description
31–16 INPEND[63:48] INPEND[31:16] 15–0 INPEND[47:30] INPEND[15:0]	Normal Interrupt Pending Bit. If a normal interrupt enable bit is set and the corresponding interrupt source is asserted, the interrupt controller will assert a normal interrupt request. The normal interrupt pending bits reflect the interrupt input lines which are asserted and are currently enabled to generate a normal interrupt. 0 No normal interrupt request 1 Normal interrupt request pending

9.2.3.13 Fast Interrupt Pending Register

The fast interrupt pending register high (FIPNDH) and the fast interrupt pending register low (FIPNDL) are 32-bit wide registers used to monitor the outputs of the enable and masking operations. These registers

are actually only a set of buffers; therefore, the reset state of these registers are determined by the fast interrupt enable registers, the interrupt mask register, and the interrupt source registers.

These read-only registers are located on the ARM1136JF-S native bus, accessible in 1 cycle, and can be accessed only in privileged mode. Figure 9-27 and Figure 9-28 show the registers; Table 9-24 provides their field descriptions.

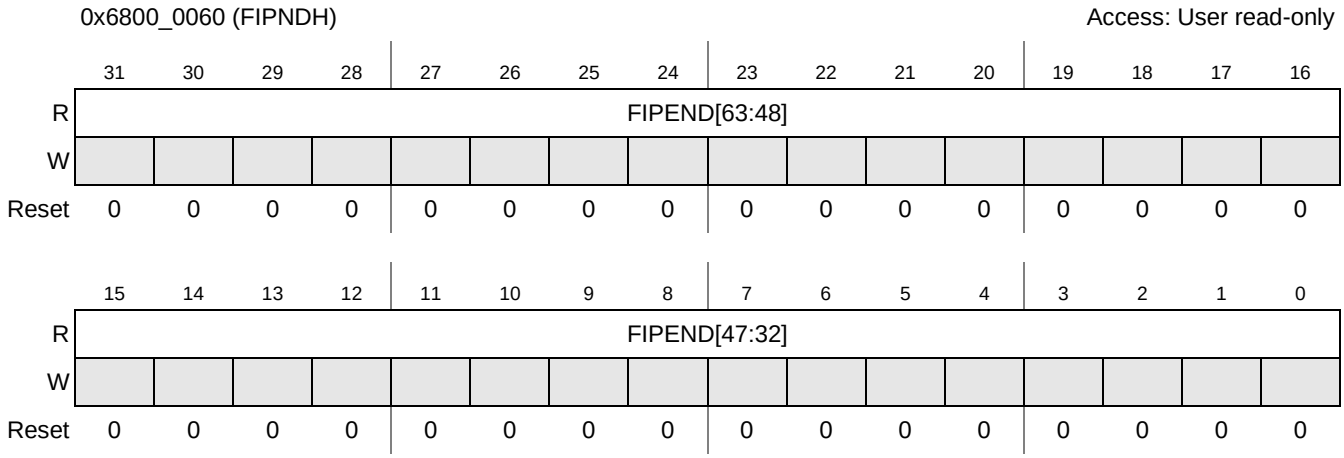


Figure 9-27. FIPNDH Register

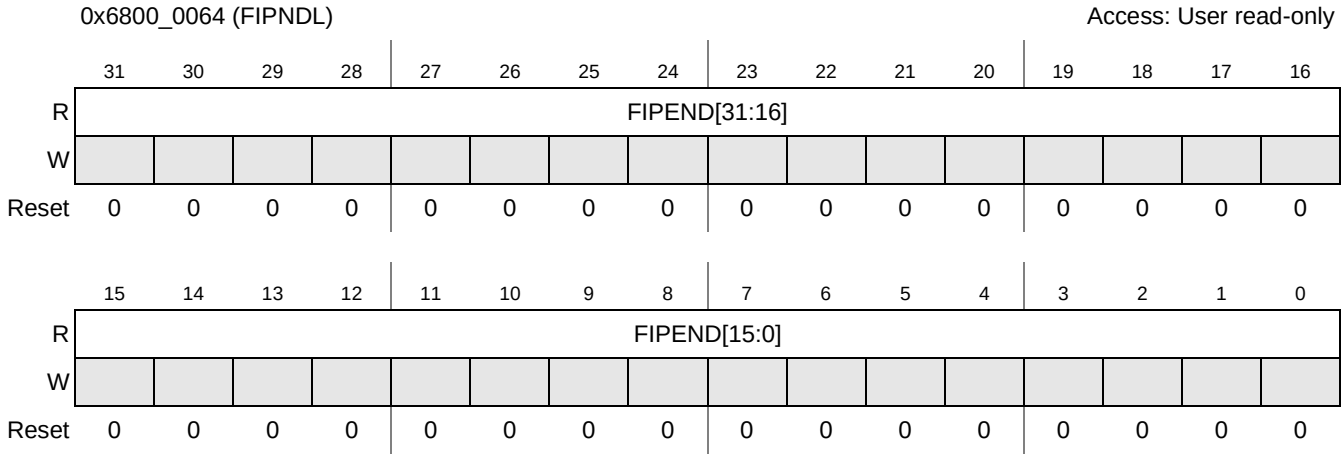


Figure 9-28. FIPNDL Register

Table 9-24. FIPNDH/FIPNDL Field Descriptions

Field	Description
31–16 FIPEND[63:48] FIPEND[31:16] 15–0 FIPEND[47:30] FIPEND[15:0]	Fast Interrupt Pending Bit. If a fast interrupt enable bit is set and the corresponding interrupt source is asserted, the interrupt controller will assert a fast interrupt request. The fast interrupt pending bits reflect the interrupt input lines which are asserted and are currently enabled to generate a fast interrupt. 0 No fast interrupt request 1 Fast interrupt request pending

9.2.3.14 AVIC Vector Registers

The AVIC vector registers (VECTOR0 through VECTOR63) store the memory addresses where the interrupt service routine for the associated interrupt source is located. The vector registers are each 30 bits wide and can point to any memory location.

These registers are located on the ARM1136JF-S native bus, accessible in 1 cycle, and can be accessed only in privileged mode. These registers can be modified only using 32-bit writes. Figure 9-29 shows the register; Table 9-25 provides its field descriptions.

0x6800_0100 (VECTOR0)–0x6800_01FC (VECTOR63) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	VECTOR[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	VECTOR[15:2]														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-29. AVIC Vector Registers

Table 9-25. VECTOR0–VECTOR63 Field Descriptions

Field	Description
31–2 VECTOR0 through VECTOR63	AVIC Vector Registers. These control registers contain the address of the interrupt service routine for the associated interrupt source. These addresses are used to vector the ARM1136JF-S core to the interrupt service routine quickly. The two LSBs are always 0 because all interrupt service routines are WORD aligned.
1–0	Reserved

9.3 ARM1136JF-S Interrupt Controller Operation

9.3.1 ARM1136JF-S Prioritization of Exception Sources

The ARM1136JF-S core imposes the following priority among the various exceptions:

- Reset (highest priority)
- Data Abort
- Fast Interrupt
- Normal Interrupt
- Prefetch Abort
- Undefined Instruction and SWI (lowest priority)

9.3.2 AVIC Prioritization of Interrupt Sources

The AVIC module prioritizes the various interrupt sources by source number where higher source numbers have higher priority. Fast interrupt always have higher priority over normal interrupts.

The interrupt requests are prioritized in the following sequence:

1. Fast interrupt requests, in order of highest source number
2. Normal interrupt requests, in order of highest priority level, then in order of highest source number with the same priority level

9.3.3 Controlling Bus Arbitration With AVIC

The AVIC has some logic to raise the priority level of the ARM core when either a fast or normal interrupt occurs. When a fast interrupt occurs and the FIAD bit is set, the AVIC will assert the avic_rise_arb signal. When a normal interrupt occurs and the NIAD bit is set, the AVIC will assert the avic_rise_arb signal. This signal will raise the priority level of the ARM core, so that it has priority of all other alternate masters. This signal will not stop the current alternate master transfer instead will prevent/limit future bus arbitration away from the ARM core.

The AVIC includes some logic in the ABFLAG and ABFEN bits. The ABFLAG bit indicates the AVIC is currently asserted the avic_rise_arb signal to the bus arbitration logic. The ABFEN bit changes the ABFLAG from a read-only status bit to a “sticky” write-1-to-clear status bit.

9.3.4 The AVIC Interface To The ARM1136JF-S Core

The AVIC utilizes the AVIC interface of the ARM1136JF-S core to vector the processor quickly to the interrupt service routine. The interface consists of four controls, which follow this sequence:

1. The AVIC asserts IRQ to the ARM1136JF-S core.
2. The ARM1136JF-S acknowledges interrupt and requests the vector via the “arm_irq_ack” signal.
3. The AVIC latches the current vector source and selects an interrupt service routine vector to place on the “avic_irq_vector[31:2]” bus.
4. The AVIC then asserts the “avic_irq_addr_val” signal indicating that the “avic_irq_vector[31:2]” bus is stable.
5. The ARM1136JF-S captures the vector information and negates the “arm_irq_ack” signal
6. The AVIC then negates the “avic_irq_addr_val” signal.

9.3.5 AVIC Interface and Fast Interrupts

The AVIC interface of the ARM1136JF-S core does not support hardware acceleration of fast interrupts; therefore, software must determine how to process the incoming fast interrupt requests. Below is a sample piece of the assembly code that reads the FIVECSR register, reads the vector out of the VECTORx register, and then jumps to the interrupt service routine.

```
ldr     r10, =0x6C000000          @ load AVIC base address
ldr     r9, [r10,#0x44]           @ read FIVECSR of AVIC
add     r10, r10, #100            @ move pointer to base of VECTOR table
```

```
ldr      r8, [r10,r9,lsl #2]        @ read FIQ vector from VECTOR table
bx       r8                         @ jump to FIQ service routine
```

9.3.6 Writing Reentrant Normal Interrupt Routines

The AVIC can be used to create a reentrant normal interrupt system. This enables preempting of lower priority level interrupts by higher priority level interrupts. This requires a small amount of software support and overhead.

1. Push the link register (LR_irq) on to the stack (SP_irq).
2. Push the saved status register (SPSR_irq) on to the stack.
3. Read the current value of NIMASK and push this value on to the stack.
4. Read current priority level via NIVECSR.
5. Interrupts of the equal or lesser priority than the current priority level should be masked via the NIMASK register by writing value from NIVECSR.
6. Clear the I bit in the ARM1136JF-S core via a MSR/MRS command sequence (now a higher priority normal interrupt can preempt a lower priority one).
   Also change the operating mode of the core to System Mode from IRQ mode.
7. Push System Mode link register (LR) on to the stack (SP_user).
8. The traditional interrupt service routine is now included.
9. Pop System Mode link register (LR) from the stack (SP_user).
10. Set I bit in the ARM1136JF-S core via a MSR/MRS command sequence (thus disabling all normal interrupts).
    Also change the operating mode of the core to IRQ Mode from System mode.
11. Pop the original value of normal interrupt mask and write to the NIMASK register.
12. The saved status register should be popped from the stack (SP_irq).
13. The link register should be popped from the stack into the PC.
14. Return from nIRQ.

NOTE

Steps 1, 2, 13, and 14 are automatically done by most C compilers and are only included for completeness.

9.4 Interrupt Usage

The following sections provides examples about how the AVIC processes interrupts within the IC.

9.4.1 Simple Steps to Enable Interrupts

For both for the following cases the exact sequence is determined by user's system. These examples are given only as example.

9.4.1.1 Normal or Fast Interrupt

The following is the sequence of events for both normal or fast interrupts:

a) Unmask interrupt generation at module level.
b) Unmask interrupt generation at AVIC level.
c) Example: UART1 is source number 45 from Table 2-3 in Chapter 2, "System Memory Map, Interrupts, and SDMA Events, so register INTENNUM = 45 will automatically enable this source in INTENABLEH register.
d) Optional: Set interrupt type in INTTYPE register (default is normal).
e) Optional: Set priority for normal interrupt in NIPRIORITYx (default is lowest priority).
f) Enable interrupt at core level: nFIQ or nIRQ in ARM's CPSR register.

NOTE

By default, the ARM core and AVIC do not work in accelerated mode; therefore the NM bit in INTCNTL register, as well as VE bit of ARM CP15 control registers clear.

9.4.1.2 Accelerated Normal Interrupt

The following is the sequence of events for both normal or fast interrupts:

a) Unmask interrupt generation at module level.
b) Unmask interrupt generation at AVIC level.

Example: UART1 is source number 45 from Table 2-3 in Chapter 2, "System Memory Map, Interrupts, and SDMA Events, so register INTENNUM = 45 will automatically enable this source in INTENABLEH register.

c) Set interrupt type as Normal in INTTYPE register.
d) Optional: Set priority for normal interrupt in NIPRIORITYx (default is lowest priority).
e) Set NM bit in INTCNTL register.
f) Enable interrupt at core level: nFIQ or nIRQ in ARM's CPSR register.
g) Set VE bit of ARM CP15 control register.

9.4.2 Enabling Interrupts, Code Examples

The actual code sequence in the examples shown is determined by user's system. These examples are given only for purposes of illustration. By default, the ARM core and AVIC do not operate in accelerated mode; therefore the NM bit in INTCNTL register is clear, as well as the VE bit of the ARM CP15 control register.

9.4.3 Normal Interrupt Mechanism

The normal interrupt mechanism is shown in Figure 9-30. The ARM core jumps to the address (0x18 + MMU_OFFSET), where it expects to find the address of the global interrupt handler.

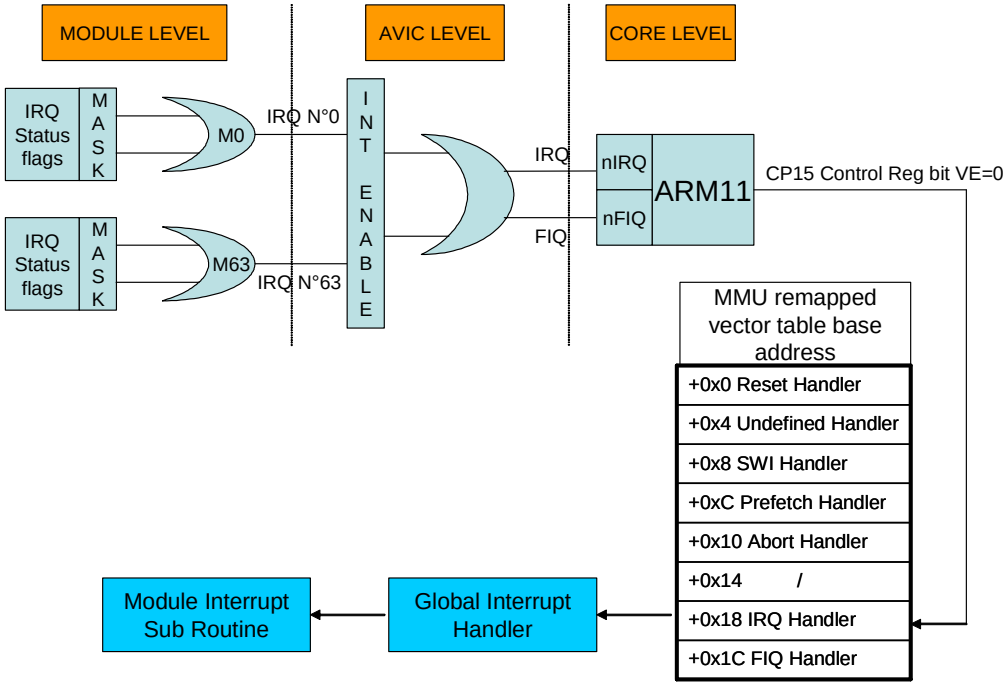


Figure 9-30. Normal Interrupt Mechanism

9.4.4 Vector Accelerated Normal Interrupt Mechanism

The Vector Accelerated Normal Interrupt Mechanism is shown in Figure 9-31. The ARM core jumps to the address of the module interrupt sub routine, which has been set inside the corresponding VECTORx register.

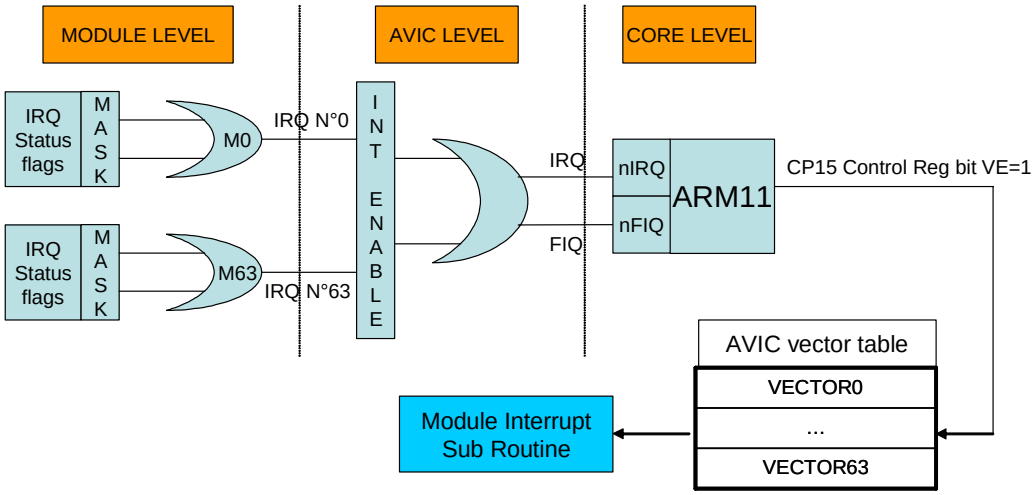


Figure 9-31. Vector Accelerated Normal Interrupt Mechanism

Book II, Part 2: Security

Introduction

This part provides an overview of the modules that make up the i.MX31 and i.MX31L security systems.

Security Controller (SCC)

The Security Controller (SCC) is a hardware component of the Platform Independent Security Architecture (PISA) baseline, and is itself composed of two sub-blocks, the Secure RAM and the Security Monitor.

The primary functionality of the SCC is associated with establishing the following:

- A centralized security state controller and hardware security state with a hardware configured, unalterable security policy
- An uninterruptable hardware mechanism to detect and respond to threat detection signals (specifically platform test access signals)
- A device-unique data protection/encryption resource to enable off chip storage of security sensitive data
- An internal storage resource that automatically and irrevocably destroys plain text security sensitive data upon threat detection

Security Random Number Generator Accelerator (RNGA)

The Security Random Number Generator Accelerator (RNGA) module is a digital integrated circuit capable of generating 32-bit random numbers (RNGA_32IP). This module connects to the IP bus (SkyBlue) interface.

Run-Time Integrity Checker (RTIC)

The Run-Time Integrity Checker (RTIC) function helps to ensure the integrity of the peripheral memory contents and assist with boot authentication. The RTIC has the ability to verify the memory contents during

system boot and during run-time execution. If the memory contents at run-time fail to match the hash signature, an error in the security monitor is triggered.

IC Identification (IIM)

The IC Identification Module (IIM) provides an interface for reading and in some cases programming and/or overriding identification and control information stored in on-chip fuse elements. This module supports electrically-programmable poly fuses (e-Fuses).

The IIM also provides a set of volatile software-accessible signals that can be used for software control of hardware elements, not requiring non-volatility.

Chapter 10
Security Controller (SCC)

The Security Controller (SCC) is a hardware component of the Platform Independent Security Architecture (PISA) baseline, and is itself composed of two sub-blocks, the Secure RAM and the Security Monitor (see Figure 10-1).

The primary functionality of the SCC is associated with establishing the following:

- A centralized security state controller and hardware security state with a hardware configured, unalterable security policy
- An uninterruptible hardware mechanism that detects and responds to threat detection signals (specifically, platform test access signals)
- A device-unique data protection/encryption resource that enables off-chip storage of security-sensitive data
- An internal storage resource that automatically and irrevocably destroys plain text security sensitive data upon threat detection

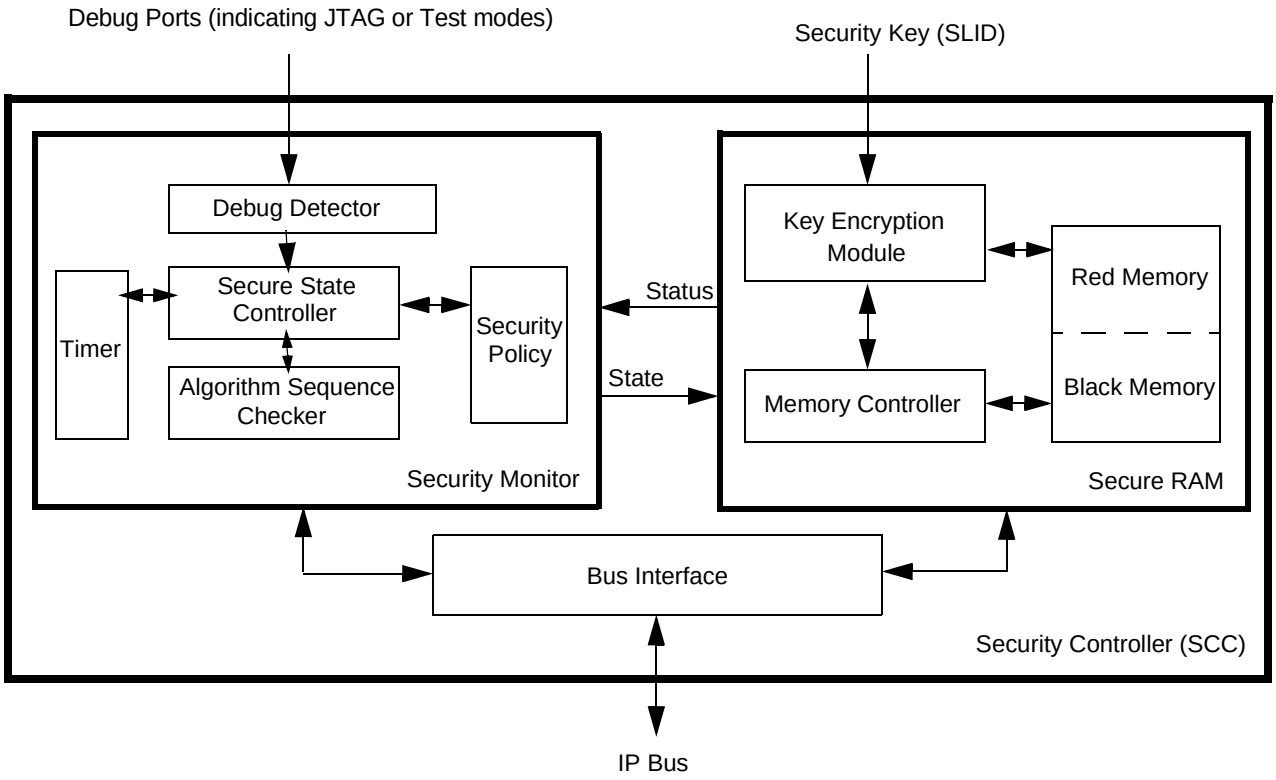


Figure 10-1. Security Controller Block Diagram

10.1 Overview

Security and security services, in an embedded or data processing platform, refer to the platform's ability to provide mandatory and optional information protection services. Information in this context refers to all embedded data, both program store and data load. Therefore, a secure platform is intended to protect information/data from unauthorized access in the form of inspection (read), modification (write), or execution (use).

10.2 External Signal Description

The SCC has no external signals.

NOTE

Contact your Freescale Semiconductor sales office or distributor for additional information about this module.

Chapter 11
Security Random Number Generator Accelerator (RNGA)

The Security Random Number Generator Accelerator (RNGA) module is a digital integrated circuit capable of generating 32-bit random numbers (RNGA_32IP). Figure 11-1 shows a top-level block diagram of the RNGA_32IP module. This module connects to the IP bus (SkyBlue) interface. See Chapter 38, "AHB-Lite 2.v6 to IP Bus Interface (AIPS)" and Chapter 8, "ARM11 Platform" for IP bus details.

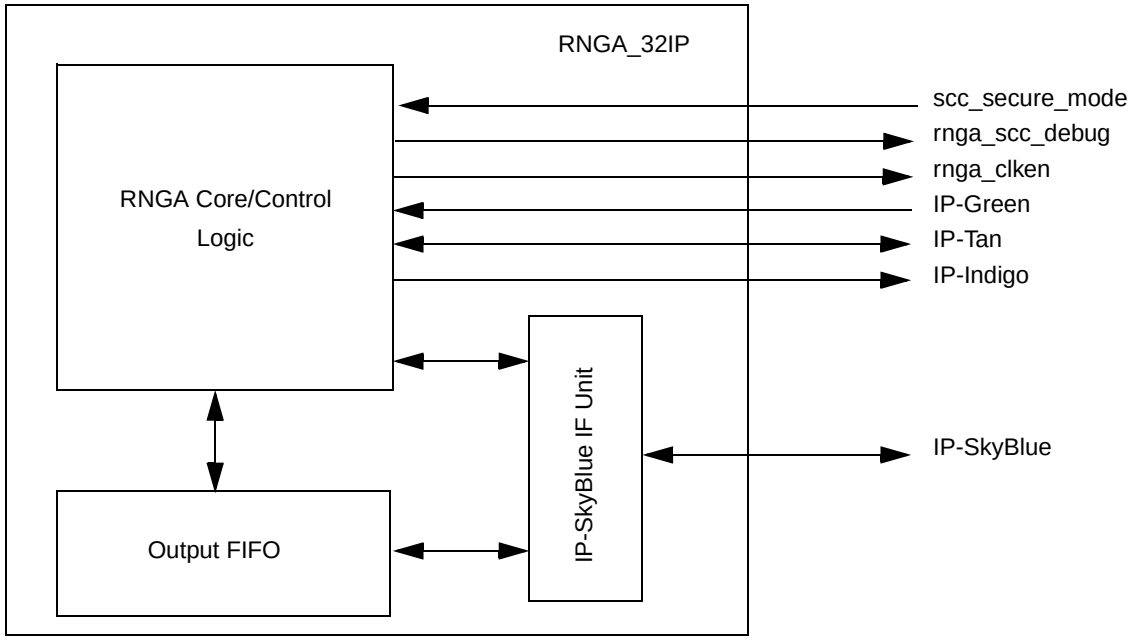


Figure 11-1. RNGA_32IP Block Diagram

11.1 Overview

The RNGA is designed to comply with FIPS-140 standards for randomness and non-determinism. The random bits are generated by clocking shift registers with clocks derived from ring oscillators. The configuration of the shift registers ensures statistically good data (that is, data that looks random). The oscillators with their unknown frequencies provide the required entropy needed to create random data.

NOTE

There is no known cryptographic proof showing that this is a secure method of generating random data. In fact, there may be an attack against the random number generator described in this document if its output is used directly in a cryptographic application. (The attack is based on the linearity of the internal shift registers.)

Due to the lack of a secure method and the potential for attacks, Freescale Semiconductor recommends that the random data produced by this module be used as an input seed to a NIST-approved (based on DES or SHA-1) or cryptographically secure (RSA Generator or BBS Generator) random number generation algorithm. It is also recommended that other sources of entropy be used along with the RNGA to generate the seed to the pseudo-random algorithm, but this is optional. It is better to combine as many random sources as possible to create the seed.

The following is a list of sources that could be easily combined with the output of this module:

- Current time using highest precision possible
- Mouse and keyboard motions (or equivalent if these motions are used on a cell phone or PDA)
- Other entropy supplied directly by the user (that is, a coin toss, sampled RF noise, radioactive substance decay, or any other random source which can be measured)

11.2 Features

The RNGA includes these distinctive features:

- 32-bit IP bus slave interface
- 16 x 32 FIFO
- Secure mode
- Power saving mode
- Optional external entropy value

11.3 Modes of Operation

The RNGA has several modes of operations, but only one is intended for use during normal operation. The other modes aid in verification and testability of the module except sleep modes, which could overlap with other modes.

The normal and oscillator frequency test modes are entered by setting the appropriate bits in the RNGA mode register. Sleep modes are entered by setting the appropriate bit in the RNGA control register, or by asserting ipg_doze (or while FIFO is full). Scan mode is entered by driving module input ipt_mode active. Finally, secure mode is entered by driving the module input scc_secure_mode high. This mode is functionally equivalent to normal mode and is provided for applications requiring higher assurance.

The following are descriptions for the modes for RNGA (these are high-level descriptions only):

- Full sleep

  This mode is entered by writing to the sleep bit in the Control register or by asserting ipg_doze. While in this mode, RNGA oscillator clocks are shut off, the FIFO cannot be loaded, and rnga_clken is de-asserted, which can be used to gate and turn off ipg_clk.

- Interface sleep

  While in normal or secure mode and when the FIFO is full and until no read FIFO is issued, rnga_clken is de-asserted, which can be used to gate and turn off ipg_clk. However, in this mode, the RNGA oscillator clocks continue to run.

- Normal

  In this mode, the RNGA generates random data. Since this is the default mode of operation, the user is not required to change the mode before requesting random data. This is also the only valid mode when in the secure state (that is, the input scc_secure_mode is high). While in this mode, the internal shift registers are driven by internally generated clocks with unknown frequency. Depending on the internal state of the RNGA, these clocks are derived from either the oscillators of the RNGA or a deterministic clock (based on the system clock). For simplicity sake, throughout the rest of this document, these clocks will be referred to as the oscillator clocks.

  The low power full sleep mode and interface sleep mode can be entered while in the normal mode (or secure mode). In normal mode (or secure mode) and not in full sleep mode and while the output FIFO is not full, the generated random data is transferred into the output FIFO.

- Secure

  In this mode, the RNGA is forced into the normal mode described above. Secure mode is equivalent to the condition where the RNGA is in normal mode and is unable to exit that mode. The mode is entered by driving the block input scc_secure_mode high. In this mode, all low power considerations as described in normal mode persist.

- Verification

  This mode is provided for verification and testing of the module. While in this mode, the random output is generated by a counter rather than the usual shift registers. The deterministic result allows for easy verification of the surrounding RNGA control logic.

- Oscillator frequency test

  This mode is provided for testing of the ring oscillators of the RNGA. While in this mode, the shift registers are clocked exclusively and continuously by the oscillator clocks (may not be the case in the Normal mode where at certain instances the shift register clocks are off), allowing the oscillator frequency counters to accurately count the pulses received from the oscillator clocks during a given amount of time.

- Scan

  In this mode, the RNGA reconfigures much of its untestable logic so it is testable by scan. The mode is entered by driving the block input ipt_mode active. This mode should be used only when scan is used to test the module.

11.4 External Signal Description

The RNGA has no external signals.

NOTE

Contact your Freescale Semiconductor sales office or distributor for additional information about this module.

Chapter 12
Run-Time Integrity Checker (RTIC)

The Run-Time Integrity Checker (RTIC) ensures the integrity of the peripheral memory contents, and assists with boot authentication. The RTIC has the ability to verify the memory contents during system boot and during run-time execution. If the memory contents at run-time fail to match the hash signature, an error in the security monitor is triggered. Figure 12-1 is a block diagram of the RTIC.

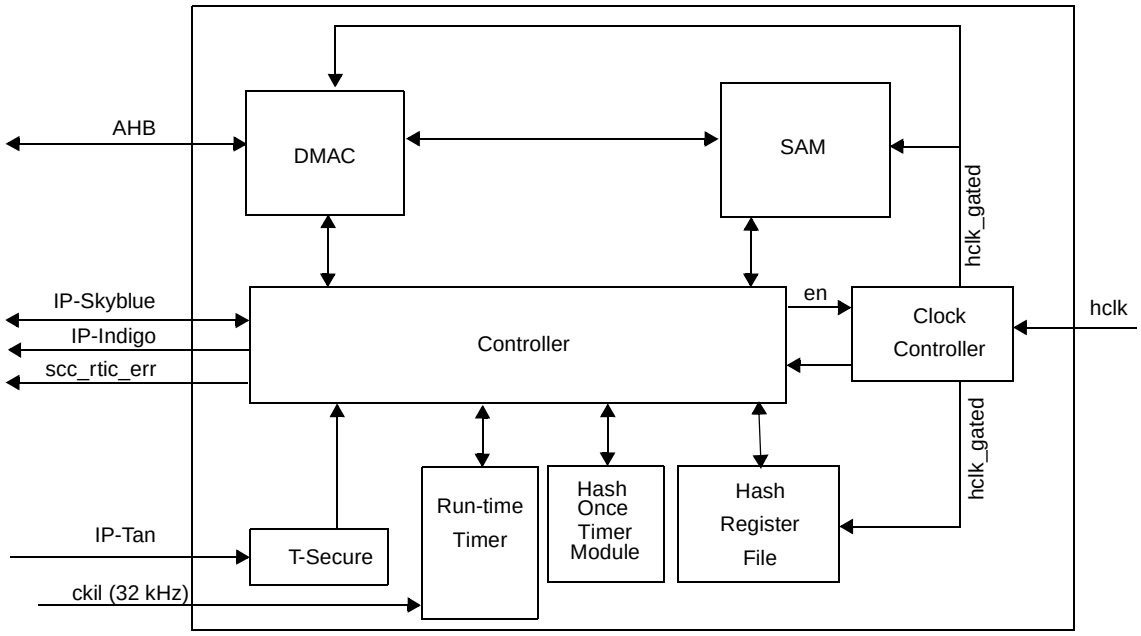


Figure 12-1. RTIC Block Diagram

12.1 Features

The RTIC offers the following features:

- SHA-1 message authentication
- Input DMA (AMBA-AHB Lite bus master) interface
- Segmented data gathering to support non-contiguous data blocks in memory (up to two segments per block)
- Works with high assurance boot process
- Secure-scan DFT security
- Support for up to four independent memory blocks
- Programmable DMA bus duty cycle timer and watchdog timer

- Power-saving clock gating logic
- Hardware configurable Big/Little-Endian data format
- Full word memory reads (word-aligned addresses, multiple of 32-bit lengths)

12.1.1 Modes of Operation

The RTIC operates in two primary modes:

- One-time hash mode
  - — Is used during high assurance boot for code authentication or one time integrity checking
  - — Stores hash result internally and signals interrupt to host
- Continuous hash mode
  - — Is used at run-time to continuously to verify integrity of memory contents
  - — Checks re-generated hash against internally stored values and interrupts host only if error occurs

12.1.2 Overview

The RTIC communicates through two interfaces: IP SkyBlue (slave) and AHB Lite (master). The IP-slave interface is used to read/write to the RTIC address space. The RTIC contains a DMA controller to perform reads of the peripheral memory block(s) on the AHB bus.

12.2 Initialization/Application Information

12.2.1 System Application

The RTIC is intended to serve as a single-use hash accelerator to assist with code authentication and other services at boot time, and as an autonomous/passive memory integrity checker during run-time. It is programmed through the IP-slave interface, and scans the peripheral memory contents over the AHB interface using direct memory access. A typical system configuration using the RTIC is shown in Figure 12-2.

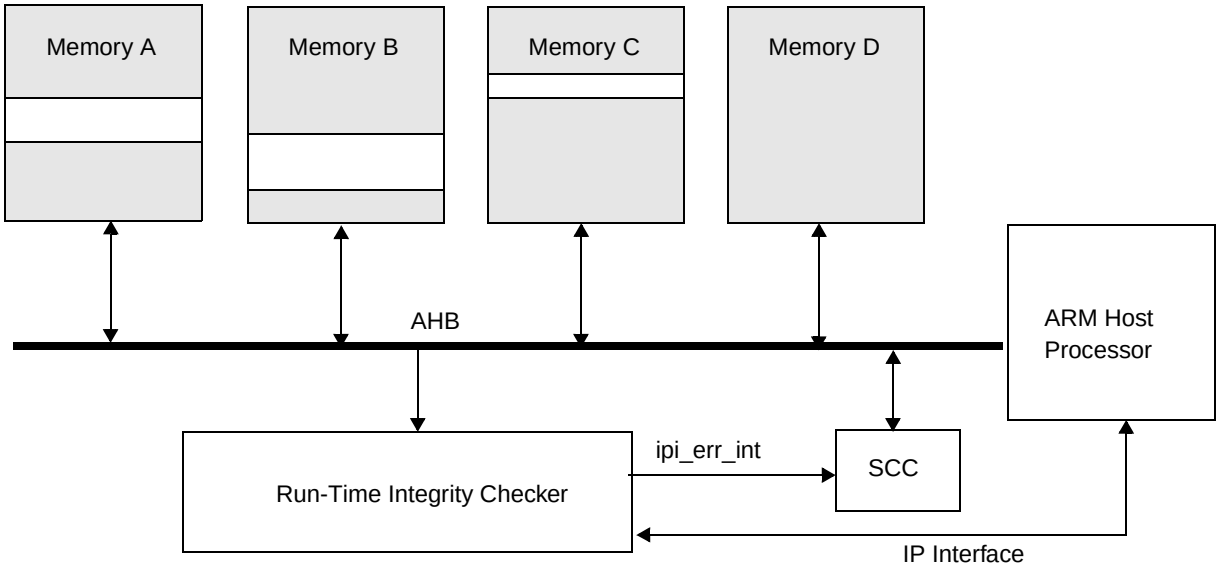


Figure 12-2. System Diagram

In this example, there are four independent memory blocks that can be checked by the RTIC. Memories A,B, and C have their contents partitioned over non-contiguous spaces. Memory D does not contain any physical partitioning. The host would program the RTIC with the starting address and length of each partition inside memory A, B, and C. For memory D, only one starting address and length would be specified, with the second start address/length fields for memory D being set to 0.

After setting the A/B/C/D hash once memory enable bits in the RTIC control register and hash once bit in the RTIC command register, the RTIC hashes each memory and stores the result in its hash register file to be read by the host. If the RTIC is used to verify that the memories are not corrupted during run-time, the A/B/C/D run-time memory enable bits in the control register must be set, followed by the tun time check bit in the RTIC command register. The RTIC re-hashes each enabled memory in a continuous loop until either an error occurs or the RTIC is reset.

12.2.2 RTIC Usage Diagram

Figure 12-3 shows an example usage flowchart for the RTIC.

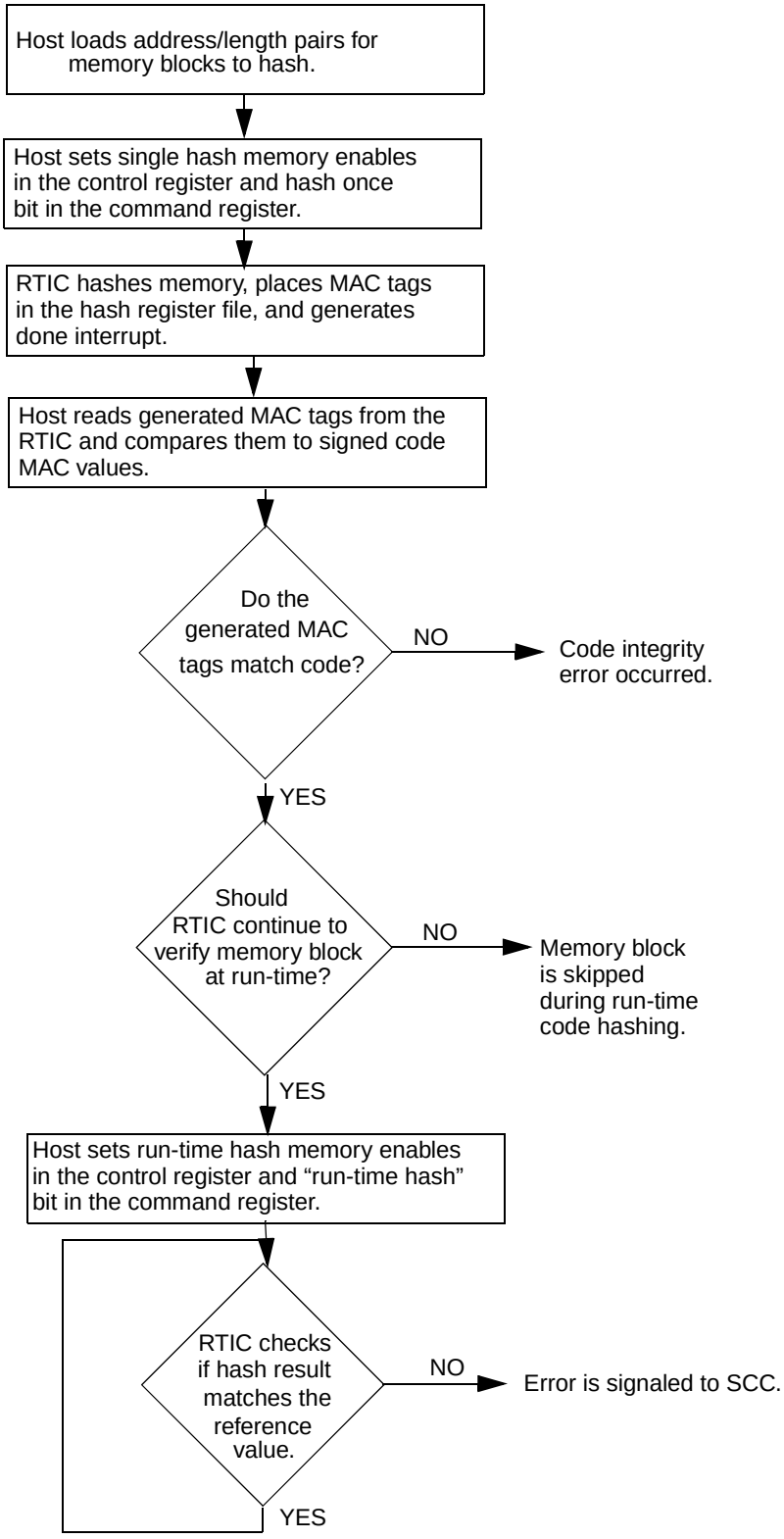


Figure 12-3. RTIC Flow Diagram

NOTE

Contact your Freescale Semiconductor sales office or distributor for additional information about this module.

Chapter 13
IC Identification (IIM)

The IC Identification Module (IIM) provides an interface for reading and, in some cases, programming and/or overriding identification and control information stored in on-chip fuse elements.

The IIM also provides a set of volatile software-accessible signals that can be used for software control of hardware elements, not requiring non-volatility.

13.1 Overview

The IIM provides the primary user-visible mechanism for interfacing with on-chip fuse elements. Among the uses for the fuses are unique chip identifiers, mask revision numbers, cryptographic keys, and various control signals requiring permanent non-volatility. The IIM also provides up to 28 volatile control signals and the means to generate a second 168-bit SCC key.

The IIM consists of a master controller, a software fuse value shadow cache, and a set of registers to hold the values of signals visible outside the module. Up to eight arrays of fuses (e-Fuses) are associated with the IIM.

The IIM is accessible using an 8-bit SkyBlue IP bus interface. An 8-bit interface is used because it matches the natural width of the fuse arrays. All registers are 32-bit aligned to allow the module to be instantiated on IP buses supporting only 32-bit peripherals.

13.1.1 Features

The following are the IIM features:

- Up to eight independent fuse banks (number of fuse bank and size of the bank are parameterized)
- Maximum usable fuse bank size is 2048 bits
- e-Fuse banks may be intermixed on a per bank basis
- Support for driving secure JTAG challenge and response values to the SJC (size of each field configurable using RTL parameter; challenge default size is 64 bits, response default size is 56 bits)
- Ability to provide up to two distinct 168-bit 3DES keys from a single set of fuses
- Ability to override fuse values in software (does not affect the fuse element); override capability can be permanently disabled on a per-bank basis
- Ability to write-protect e-Fuses on a per-bank basis
- Ability to scan-protect (read and program) on a per-bank basis
- Fuses may be programmed by software, directly by JTAG, or indirectly by JTAG using a processor.
- Recommended signal assignments to maximize software re-use

13.2 Memory Map and Register Definition

Section 13.2.2.1, “Silicon Revision (SREV)” provides the detailed register description for the SREV register.

13.2.1 Memory Map

The IIM register is eight bits wide, but addressable on 32-bit boundaries. Only the bottom eight bits (the usable bits) of the register is shown in Figure 13-2. The top 24 bits is always read as 0 and writes to them are ignored.

13.2.2 Register Summary

Figure 13-1 shows the key to the register fields

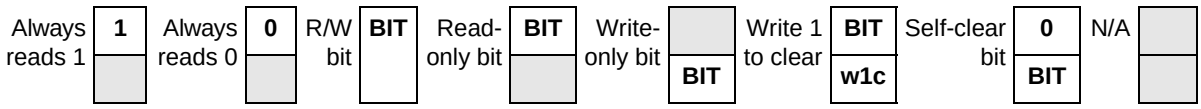


Figure 13-1. Key to Register Fields

13.2.2.1 Silicon Revision (SREV)

SREV is the silicon revision (that is, mask revision) register, which corresponds to the bottom eight bits of the deprecated HW_REV register. See Figure 13-2 for an illustration of valid bits in the Silicon Revision Register, Table 13-1 for its field descriptions, and Table 13-2 for the SILICON_REV settings.

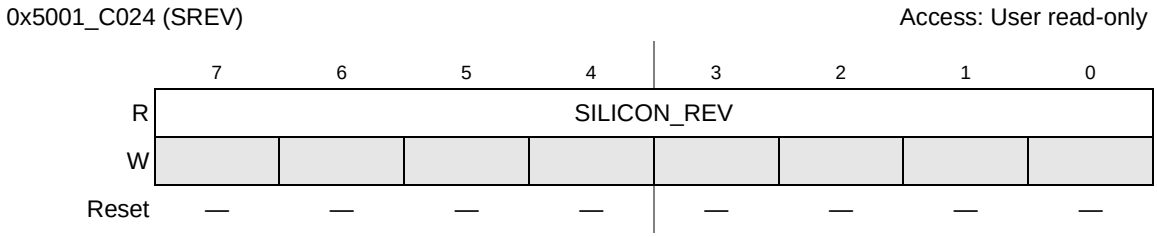


Figure 13-2. Silicon Revision Register

Table 13-1. Silicon Revision Register Field Descriptions

Field	Description
7–0 SILICON_REV	Mask Set Revision. The mask set revision used in the fabrication of the part. The value changes with each change to the mask set. See Table 13-2 for the revision settings.

Table 13-2. SILICON_REV Settings

SREV	Device	Silicon Revision	Device Marking Wafer Fab 1	Device Marking Wafer Fab 2
0x00	i.MX31 and i.MX31L	1.0	L38W	—
0x10	i.MX31	1.1	2L38W	—
0x11	i.MX31L	1.1	2L38W	—
0x12	i.MX31	1.15	2L38W 3L38W[1]	—
0x13	i.MX31L	1.15	2L38W 3L38W[1]	—
0x14	i.MX31	1.2	3L38W[2]	M45G
0x15	i.MX31L	1.2	3L38W[3]	M45G

1 Misidentified device. IC stamped 3L38W, SREV register reads correct value: 2L38W. The device marking has the initial characters MCIMX31.

2 Non-production part used for population of ADS boards. The device marking has the initial characters PCIMX31.

3 Non-production part and is not available. The device marking has the initial characters PCIMX31.

NOTE

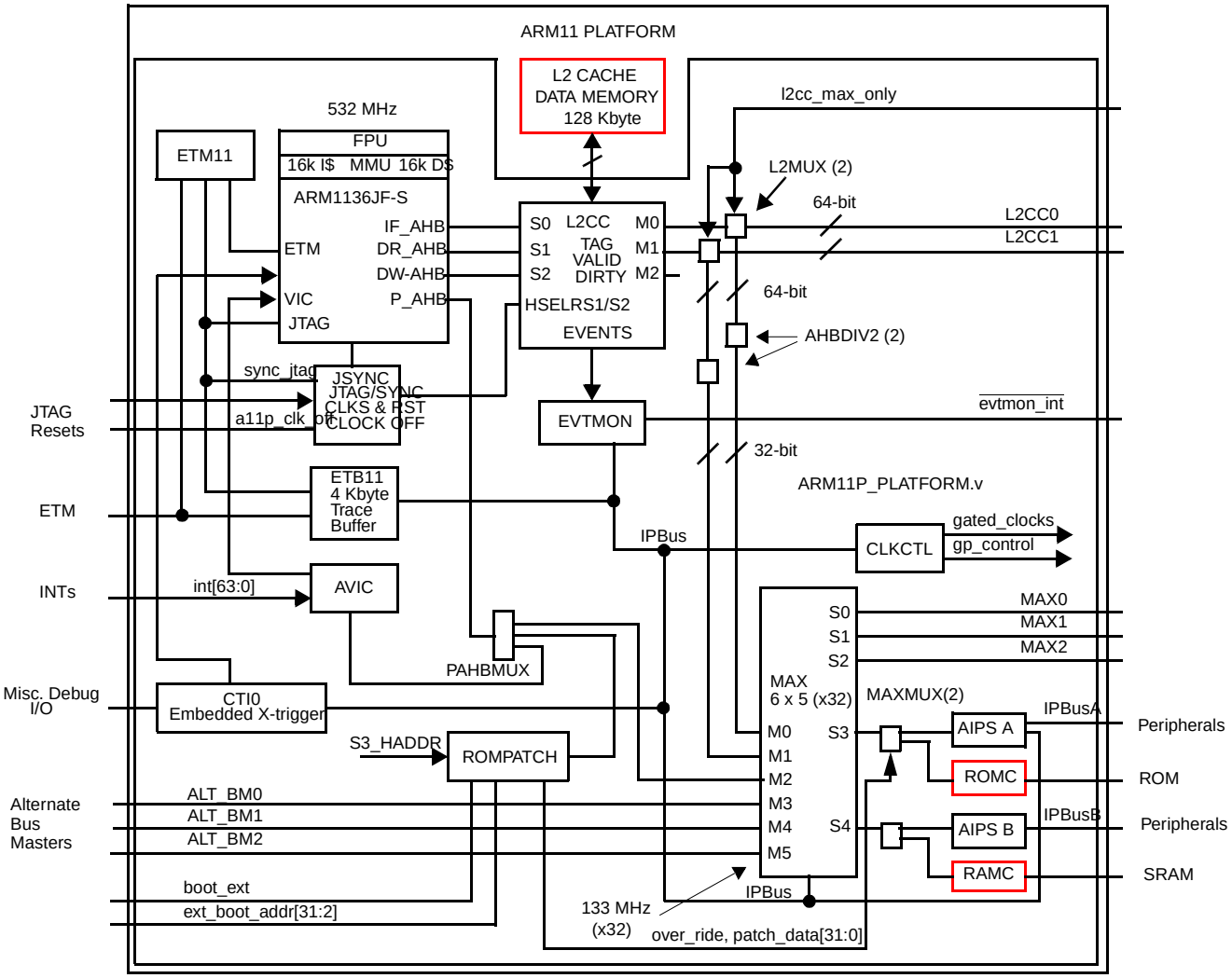
Contact your Freescale Semiconductor sales office or distributor for additional information about this manual.

Book II, Part 3: Memory Systems

Introduction

This section describes the modules that make up the i.MX31 and i.MX31L memory systems, and includes the following chapters:

The figure, ARM11 Platform, shows a block diagram of the ARM11 Platform. The major modules of the i.MX31 and i.MX31L memory systems are shown in red. The platform consists of the ARM1136JF-S processor, a Level two (L2) cache controller (L2CC), L2 cache memory, an L2 event monitor (EVTMON), two AMBA bus down-sizers (AHBDIV2), a 6x5 multi-layer AHB 2.v6 crossbar switch (MAX), an SRAM controller (RAMC), two AHB to IP-Bus interface gaskets (AIPS), a ROM controller (ROMC), an ARM11 vectored interrupt controller (AVIC), a clock control module (CLKCTL), a ROMPATCH module, and a JTAG synchronization module (JSYNC).



ARM11 Platform

Memory System Modules

The i.MX31 and i.MX31L memory system uses the following modules in the ARM11 Platform:

- Level Two Cache Controller (L2CC)
- L2CC Event Monitor (EVTMON)
- ROM Controller (ROMC)
- SRAM Controller (RAMC)
- ROM Patch Module (ROMPATCH)

Level Two Cache Controller (L2CC)

The ARM11 Platform Level 2 Cache Controller (L2CC) is optimized to sit between the ARM11 (L1) and main memory (L3, or the external EMI). The L2 cache is a unified, physically indexed, physically tagged

8-way cache. The replacement algorithm can be locked on a way basis, allowing the associativity to be reduced from 8-way down to 1-way (direct mapped), and the locked ways to be used as physically mapped memory. The L2CC does not have snooping hardware to maintain coherency between caches, so coherency must be maintained in via software.

The L2 cache sizes supported by the ARM11 Platform is 128 Kbytes. A 256-bit data path is implemented to the L2 data arrays. The L2 cache latency is 8 clocks on hits assuming a single cycle RAM access. However, the actual latency of the L2 data RAMs on the ARM11 Platform is 4 clocks (three wait-states), and therefore the L2 cache latency on the ARM11 Platform is 11 clocks.The L2 tag, valid, and dirty RAMs are zero wait-state compiled memories. All slave and master ports of the L2CC are 64-bit AHB-Lite 2.v6 compliant. Refer to Chapter 14, "L2 Cache Controller (L2CC)" for more details.

L2CC Event Monitor (EVTMON)

The L2CC design provides a number of L2 cache event monitoring output signals. These signals are routed from the L2CC to the L2CC's Event Monitor (EVTMON) module, which contains a control register, a status register, and event counters. Access to EVTMON registers are provided via an IP-Bus connection. The evtmon_interrupt output signal is generated based upon enabling various L2CC events. This signal is made available at the top-level of the ARM11 Platform in order to provide flexible interrupt assignment and muxing at the SoC level. Although some of the events monitored by the EVTMON module are used for L2 operational profiling (cache requests, hits, and so on), some of the event signals monitored are necessary to resolve error conditions not signaled on the ARM1136/L2CC AHB interfaces (bwabt for example).

ROM Controller (ROMC)

The ARM platform's ROM controller (ROMC) shares MAX master port S3 with the AIPS A module. The ROMC interface is AHB-Lite 2.v6 compliant. The rom_connect input on the ARM11 Platform must be tied high if ROM is attached to the ROMC interface. The ROMC module supports ROM sizes between 16 Kbyte and 4 Mbyte, in 1 Kbyte increments, by strapping the rom_size[11:0] inputs appropriately. Any actual ROM size smaller than 16 Kbytes is not fully supported as it is treated as a 16 Kbyte ROM. A configurable BIST engine is provided.

SRAM Controller (RAMC)

The SRAM controller (RAMC) shares MAX port S4 with the AIPS B module. The ram_connect input on the ARM11 Platform must be tied high if RAM is connected to the RAMC RAM interface. The RAMC module supports a minimum of 1 Kbyte of RAM and a maximum of 1 Mbyte. Non power-of-two sizes between 1 Kbyte and 1 Mbyte are supported by strapping the ram_size[9:0] inputs, which correspond to HADDR[19:10]. RAMC address space starts at $1000_0000 and ascends to a maximum of $100F_FFFF (1 Mbyte). RAMC space is aliased between $1000_0000 and $1FFF_FFFF (a 256-Mbyte region) based on the ram_size inputs.

SRAM Wait State Control

Internal RAM accesses from the ARM11 and/or the L2CC have significantly better timing than external RAM accesses (via alternate bus masters). For this reason, the platform has two RAM related wait-state control inputs. The ram_read_wait_arm11 input should be tied high at integration time if a wait state is required to make read data timing on RAM accesses from the ARM11 (or L2CC). A separate input, ram_wait_alt_mstr, is provided to control RAM wait-states for alternate bus masters. Typically, ram_read_wait_arm11 could be tied low (ARM11 accesses RAM at zero wait-states) while ram_wait_alt_mstr could be tied high (more difficult timing for alternate bus masters requires a wait-state on RAM accesses). The solution uses the hmaster encodings to identify the ARM11/L2CC as the access requester, and is optimized for critical ARM11 zero-wait state RAM timing.

All RAM write accesses are zero wait-state. The RAM interface supports single clock-edge non late-write style compiled memories, and implements an internal write buffer to mimic the late-write capability for improved performance. Both polarities of RAM control signals are provided to support Freescale and non-Freescale RAM solutions. A configurable BIST engine is provided. The RAMC module also supports a single outstanding AHB 2.v6 exclusive access.

ROM Patch Module (ROMPATCH)

The ROM patch module (ROMPATCH) is used to patch errant ROM code. The registers of the ROMPATCH are programmed by the ARM11 over the P_AHB bus. However, the ROM only patches accesses to the ROMC (not accesses to AIPS A, which shares MAX master port S3 with the ROMC module). The ROMPATCH module can be used to patch source code or data tables. The module supports 16 patches. The ROMPATCH module also supports external booting by over-riding the reset vector fetch.

Chapter 14
L2 Cache Controller (L2CC)

The ARM11 Platform Level 2 Cache Controller (L2CC) controls the L2 cache, which is an unified, physically indexed, physically tagged 8-way cache or both instructions and data. The purpose of this chapter is to provide a high-level description of the L2CC. For additional detailed information about the L2CC, refer to the references noted at the end of the chapter.

14.1 Overview

The L2CC, shown in Figure 14-1, is optimized to sit between the ARM11 (L1) and main memory (L3, or the external EMI). The L2 cache replacement algorithm can be locked on a way basis, allowing the associativity to be reduced from 8 way down to 1 way (direct mapped), and the locked ways to be used as physically mapped memory. The L2CC does not have snooping hardware to maintain coherency between caches, so coherency must be maintained via software.

The L2 cache sizes supported by the ARM11 Platform is 128 Kbyte. A 256-bit data path is implemented to the L2 data arrays. The L2 cache latency is 8 AHB HCLK clocks on hits assuming a single cycle RAM access. However, the actual latency of the L2 data RAMs on the ARM11 Platform is 4 AHB HCLK clocks (three wait-states), and therefore the range on the ARM11 Platform is predicted to be 11–12 AHB HCLK clocks. The L2 tag, valid, and dirty RAMs are zero wait-state compiled memories.

The L2CC is tightly coupled to the ARM11 and serves as a private cache. The ARM11's instruction fetch, data read, and data write buses are connected directly to slave ports S0, S1, and S2, respectively, of the L2CC. The S0 slave port only services reads (instruction fetches) and the S2 slave port only services writes. The S1 port services both reads and writes, allowing exclusive accesses and swaps.

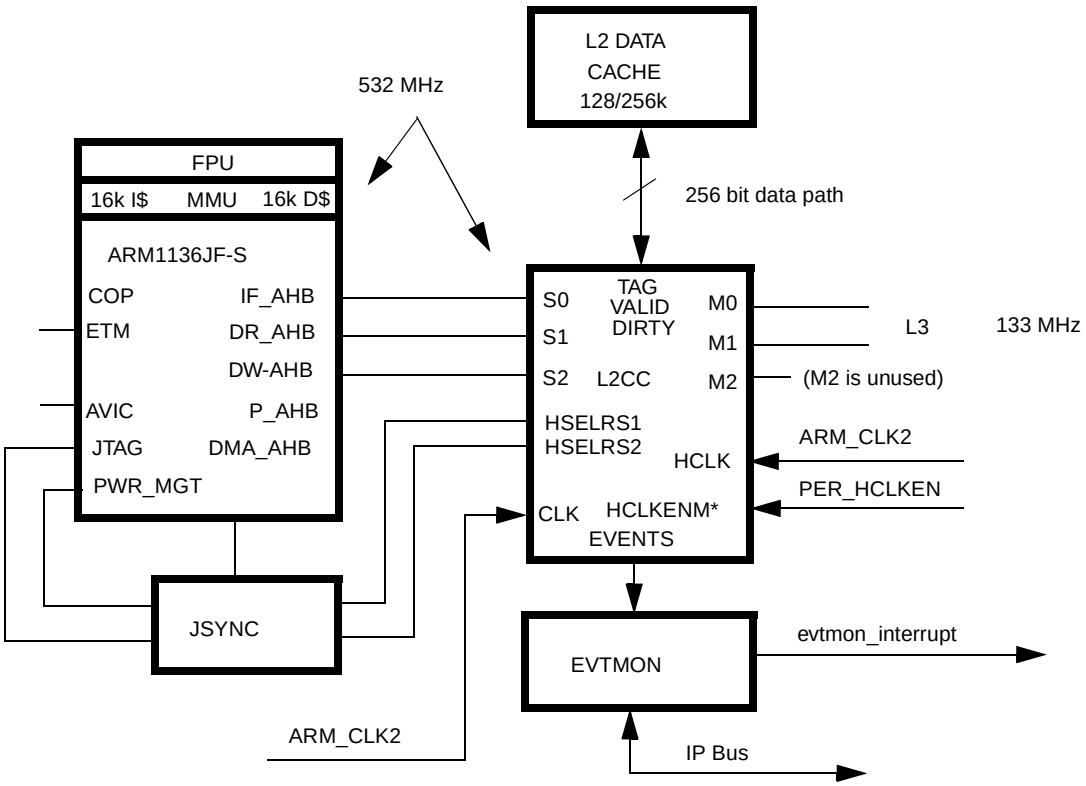


Figure 14-1. Tightly Coupled ARM11 and L2CC

14.1.1 L2CC Feature Set

The L2CC module has the following features:

- Physically addressed and physically tagged
- Lockdown format C supported (way locking) for data and instructions
- L2 cache size is 128 Kbyte
- Direct mapped to 8-way associativity, depending on the use of lockdown registers
- Fixed line length of 32 bytes (8 words)
- Data RAM is byte writable.
- Supports these cache modes:
  - — Write-through, read allocate
  - — Write-back, read allocate, write no allocate
  - — Write back, read and write allocate
  - — Write allocate override option to always have cacheable writes allocated to the L2 cache
  - — Performs critical word first refilling, with the option of refilling starting with word 0
  - — Pseudo-random victim selection policy—can be made deterministic with use of lockdown registers

- Two 32-byte linefill buffers (LFB). These buffers capture linefill data from main memory, waiting for a complete line before writing to L2 memory. This makes the L2 cache non-blocking for requests from the other slave ports.
- Two 32-byte line read buffers (LRB). These buffers hold a line from the L2 cache for subsequent requests that hit on the line.
- One castout buffer (CB). This holds a line from the L2 cache, to be written back to main memory.
- One 32 byte write buffer (WB). This holds ordered writes to be written to main memory, as well as pending writes to the L2 cache. The write buffer has a 16-word data buffer and a 4-address buffer.
- Parity error support available but not supported on the ARM11 Platform
- Memory error support
- MBIST support
- L2 cache event monitor port

14.1.2 L2CC Configuration

The slave ports, as well as all memory and memory controllers within the L2CC will run on the high- speed arm_clk2. The L2CC master ports are throttled (the L2CC's HCLKENM* inputs are tied to per_hclken) and will run at the per_clk frequency. The L2CC issues evictions and write allocates as supervisor mode transactions.

Also shown in Figure 14-1 is the connection to the L2CC of the Event Monitor (EVTMON) module. The EVTMON allows access to L2CC event monitoring logic via a connection to an IP Bus. See Chapter 15, "ARM11 Event Monitor (EVTMON)" for more details on the EVTMON module. Figure 14-2 shows the top-level L2CC architecture.

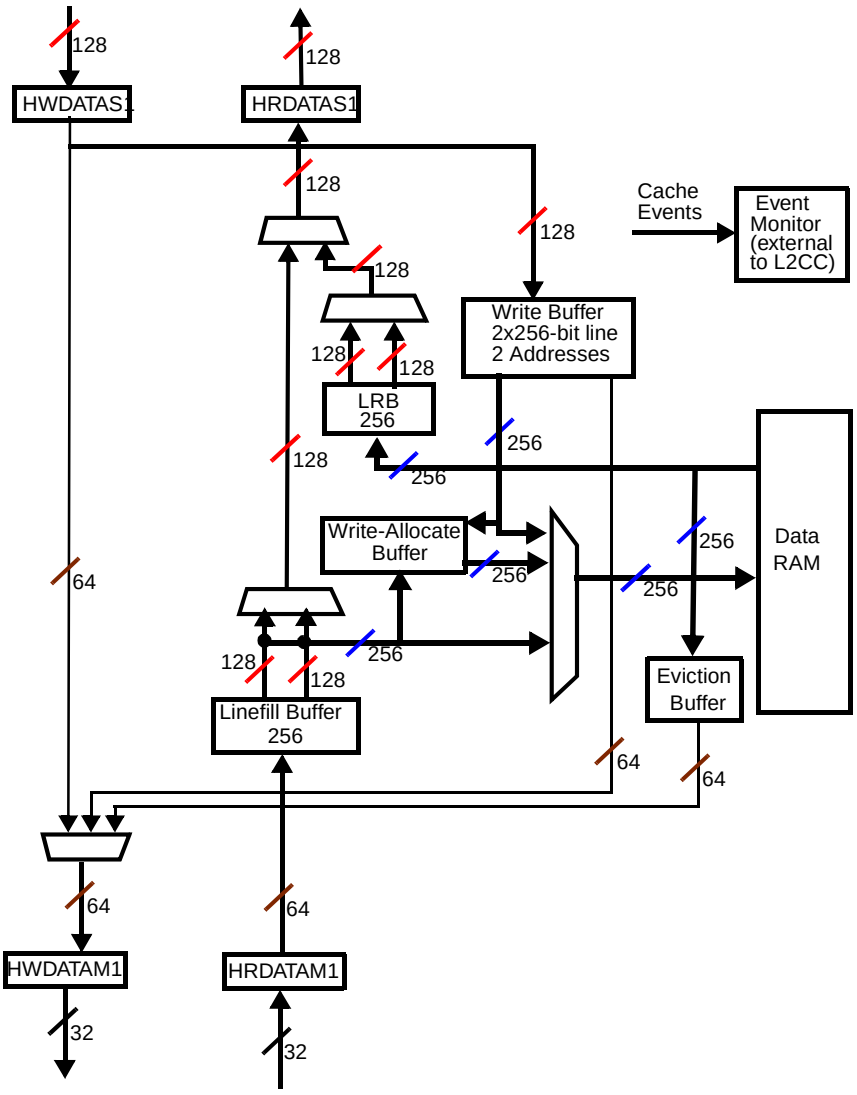


Figure 14-2. Top Level L2CC Architecture

The L2CC controller designed by ARM can be configured for three master port operation. To ensure optimum performance, power, routeability, and external memory controller complexity impact of multiple 64-bit AHB buses, the ARM11 Platform is implemented in a 2 master port configuration (M2 being unconnected). Table 14-1 shows the assignments for each master port in the 2 master port configuration.

Table 14-1. L2CC Master Port Transactions for 2 Master Port System

Master Port M0	Master Port M1	Master Port M2
Linefills with LF Buffer 0 Non-cached Reads	Linefills with LF Buffer 1 Non-cached Reads Swaps (Non-cached Read + NB Write) Buffered Stores with Castout Buffer Buffered Stores with Write Buffer Non-Buffered Stores	UNUSED

14.1.3 AHB Slave Port

The L2CC S1 slave port services both reads and writes, and can also access the internal L2CC registers. Write access to locations designated as INSTRUCTION/OPCODE (HPROTS1[0]=0) is not supported.

The slave port supports AMBA2.0. It does not support AMBA2.0 split accesses or retry responses. HRESPS1[1] is tied low internally. HSIDEBAND signals are not supported. Of the ARM v6 extensions to AMBA2.0, only HPROT[4] (ALLOCATE) is supported to indicate write-allocate policies. No other v6 extensions are supported on the slave port.

The slave port also supports a WRAP2 burst. This is controlled with the HBURST[3] bit, which is not part of the AMBA2.0 standard. A 2-beat WRAP2 burst is formed by the ARM11 Platform as HTRANSS1=NSEQ/NSEQ. This is in contrast to all other bursts which are formed as HTRANSS1=NSEQ/SEQ. Because the ARM11 never terminates a burst early, the L2CC interprets the second NSEQ beat of the WRAP2 burst as a sequential transfer.

The L2CC slave port is optimized for SNGL and WRAP2 transfers. Transfers of type SNGL, INC4, INC8, INC16, and WRAP16 are internally remapped into individual NSEQ/SNGL transfers. Register bits control whether WRAP4 and WRAP8 bursts are internally remapped into individual NSEQ/SNGL transfers.

HTRANSS1 may be NSEQ, SEQ, or IDLE. HTRANSS1=BUSY is not supported. The Q2MA does not produce BUSY cycles. The L2CC requires that once a fixed-length burst begins, it must complete. Early terminated bursts are not supported. HMASTERS1 should be tied to a constant, and not change from transfer to transfer.

14.1.4 AHB Master Port

The AHB master port in L2 cache is used for interfacing L2 cache and external memory.

The AHB master port is AMBA2.0 compliant and is AHB-Lite compatible. The only ARM v6 extensions that are supported on the master port are HPROT[4] (ALLOCATE), HUNALIGNM1, and HBSTRBM1[3:0]. Although WRAP2 bursts are supported on the slave port, WRAP2 bursts are never produced by the master port.

When the cache is enabled, cacheable transfers on the slave port initiate HBURSTM1=INC8 or WRAP8 transfers on the master port with HSIZEM1 indicating 32-bit transfers. When the cache is disabled or when a non-cacheable access is performed, the L2CC is capable of producing all burst types on the master port, except WRAP2. For these bursts, HSIZEM1 may indicate 8-bit, 16-bit, or 32-bit.

NOTE

Sparse transfers can be produced on the master port. These transfers will have one or more bits of HBSTRBM1[3:0] low. Furthermore, the L2CC master port can perform bursts for which some beats have all bits of HBSTRBM1 low (that is, a dummy beat). All devices connected to the L2CC master port must properly handle this condition.

The L2CC will set any byte of HWDATAM1 which has its corresponding HBSTRBM1 bit low to 0xff. Although byte write data with low byte strobes should be *Don't Cares*, this is done to help debug modules

which may not support byte strobes. For example, if an AHB Nexus block which does not support byte strobes monitors the master port, any 0xFF byte in a write can be interpreted to be a byte whose byte strobe was likely 0 (255/256>99% chance, assuming random write data). Bytes of HWDATAM1 are forced to 0xFF when the corresponding HBSTRBM1 bit is low only if at least one of the four HBSTRBM1 bits is high. If all 4 HBSTRBM1 bits are low (this occurs with missing 32-bit words when the write buffer is drained), HWDATAM1 instead retains the value from the previous transfer.

The master port does not support split access or retry responses. HRESPM1[1] is ignored by the design of the L2CC requirements on L3 memory for reporting errors on HRESPM1[0].

The value on HMASTERM1 reflects the value on HMASTERS1 for non-cacheable or non-bufferable accesses, and for linefills. For evictions, bufferable writes, and write allocation reads, HMASTERM1 will indicate the value on DEFHMASTER, regardless of the value of HMASTERS1 for the initiating access. DEFHMASTER[3:0] should be tied to a constant at the SoC level.

The value on HPROTM1 reflects the value on HPROTS1 for noncacheable or nonbufferable accesses, and for linefills. For write allocation reads and evictions, the value ‘b11111 is always used on HPROTM1, indicating OWBNWA privileged data (no write allocate, cacheable, bufferable, privileged data access). This means that OWBWA accesses can result in accesses to L3 memory that are marked as privileged even if the original slave port access was in user mode. Thus, user access to privileged-only memory must be killed at the L1 MMU level for OWBWA accesses, and cannot rely solely on L3 memory protection mechanisms. For bufferable writes, HPROTM1[0] is always high, indicating a data access, and the remaining HPROTM1 bits are defined by HPROTS1 of the initiating access.

The master port may produce NSEQ, SEQ, IDLE, or BUSY values for HTRANSM1.

14.1.5 Write Buffer (WB)

The write buffer has two slots, each with a 256-bit data line and one address per slot. The write buffer has merging capabilities, so that successive writes to the same line address are merged in the same buffer slot. Two buffered write accesses to the same address cause the first one overridden if the write buffer has not be drained in between writes.

Merging capability means that lines are not treated as soon as they contain data. Write buffer draining policy is as follows:

- The write buffer is drained at each non-cacheable read occurrence.
- The write buffer is drained at each non-bufferable write occurrence.
- If the two slots of the write buffer contain data, the least recently accessed is drained.
- If a hazard is detected with one write buffer slot, it is drained to resolve the hazard.

There is one exception to these rules for write buffer draining. The write buffer is not drained on a non-cacheable or non-bufferable access if it is locked with HMASTLOCKS1 and if the prior transfer was also locked with HMASTLOCKS1. This “prior transfer” could have been just prior to the L2CC shutting its own clocks down (because there were no transfers to it), and that the L2CC is not aware of the status of HMASTLOCK while its clocks are shut off during periods when it is not being accessed. This is not expected to be an issue, and is only documented here to describe the behavior.

Each slot contains a byte-valid field that allows the control logic to know the data linefill level. If a drained slot is drained while its data line is not full, the write buffer must request correct transactions to the master port in the form of bursts or linear access requests.

14.1.6 Write-Allocation Buffer

Allocation to the cache in case of write-allocate transaction is not performed directly by the write buffer. In cases of a write miss, when a write buffer slot is drained while its L2 memory attributes correspond to a write-allocate region (or write allocation behavior is forced by the corresponding control bit in the Auxiliary configuration register), the data, address, and byte-valid information is sent to the write allocate buffer.

Based on byte-valid information, the write-allocation buffer requests, through the M1 port to L3, the data required to fill its line if not full:

- If the line is full, no request is made
- If one to eight sequential bytes within the same double-word boundary are missing, two 32-bit transaction are performed on L3.
- In all other cases, a full linefill request is performed

Data from the write buffer and M1 master port are then merged in the write-allocation buffer that can then request an allocation to the cache.

If M1 receives an ERROR response during a read transfer for the write-allocation buffer, the allocation from the write-allocate buffer to the cache is not performed.

14.1.7 Eviction Buffer (EB)

The eviction buffer is incorporated for holding write-back data for L2 cache line evictions or cleaning of dirty cache lines. It holds two halves of L2 cache line (32 bytes total) and two address entries.

14.1.8 Line Read Buffer (LRB)

The line read buffer (LRB) holds a line from the L2 cache, for subsequent requests that hit on the line.

14.1.9 Linefill Buffer (LFB)

The Linefill buffer (LFB) captures linefill data from L3 memory, waiting for a complete line before writing to L2 memory.

14.1.10 The ARM11 Event Monitor

The L2CC supplies the event signals for monitoring of the L2 cache. The events will be held high for one cycle of CLK each time the event occurs. An event monitor external to the L2CC can be connected to this event bus. It can be read, as a memory mapped peripheral through IP bus. The event monitoring port of the L2CC is enabled by writing to the L2CC Auxiliary Control register.

14.2 Modes of Operation

The following sections provide the L2CC modes of operation.

14.2.1 L2CC Clocking

The L2CC can be clocked using the same clock as the ARM11 Platform. It receives a single clock input CLK. For minimum run-time power, this clock should be gated externally to the L2CC with the L2CC_clken signal.

The AHB master and slave ports use a single clock enable, CLKENAHB, to enable the transfer rate on AHB ports to be a synchronous sub-multiple of CLK. For a rising edge of CLK to be recognized by the AHB ports, CLKENAHB must be asserted in the cycle before that rising edge. This is illustrated in Figure 14-3.

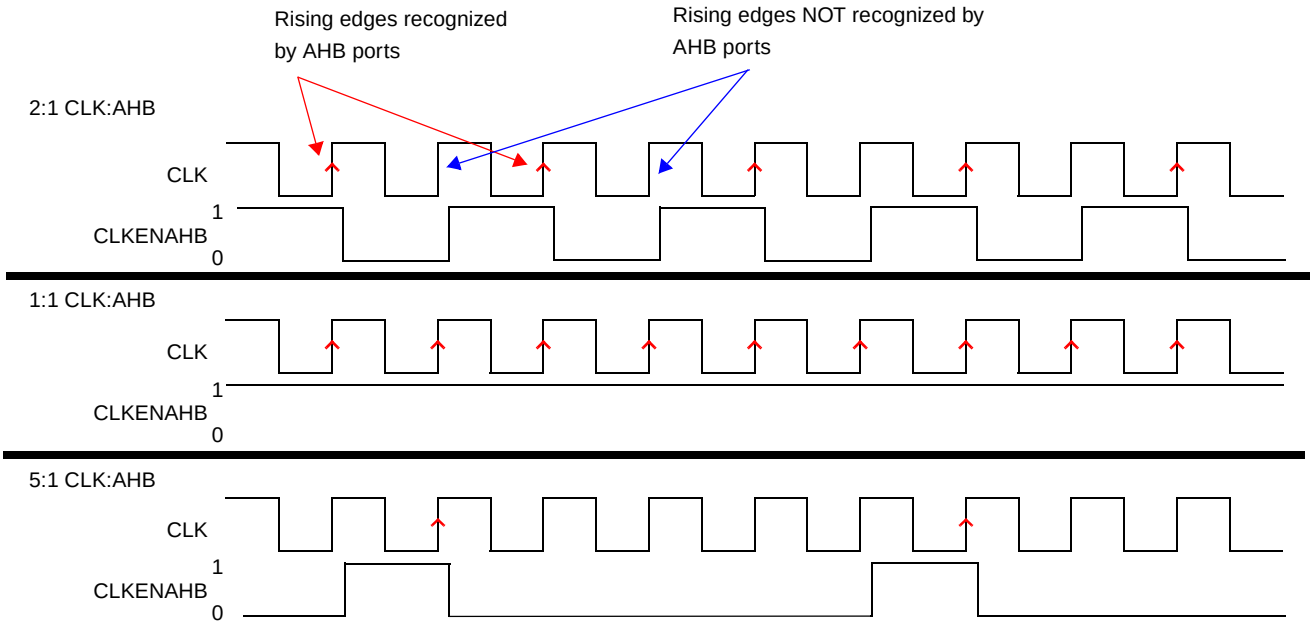


Figure 14-3. CLKENAHB Usage

14.2.2 L2CC Idle

The output signal IDLE indicates when the L2CC is not currently performing any internal processing. However, it does not accurately reflect whether any L2CC operations are pending. When IDLE is asserted and there is no slave port activity, the L2CC may autonomously de-assert the IDLE pin to perform operations such as a master port write allocation or write-buffer draining. For this reason, the following conditions are required to ensure that the L2CC is IDLE and remains:

- IDLE pin is high, and has been high for 16 clocks
- No transactions on the slave port

In DSM mode, the clock to both the ARM11 and L2CC are stopped. The L2CC must later be awoken at the same time or before the L1 core, so that any AHB requests from the ARM11 are now handled by the L2CC.

The L2CC responds to an ipg_stop assertion by asserting ipg_stop_ack when it can guarantee that the L2CC is idle and remains idle. This is illustrated in the Figure 14-4. The L2CC waits for IDLE to be high for 16 clocks before asserting ipg_stop_ack, via the L2CC_idle_forsure signal. Figure 14-4 also illustrates the L2CC clock gating. The L2CC automatically shuts down its clocks to save power when possible.

NOTE

To be able to count IDLE clocks (state machine in Figure 14-4), the L2CC clocks must be on. This is the reason that the L2CC forces its clocks on when there is an unacknowledged stop request (ipg_stop &&!ipg_stop_ack).

Additional combinational logic (not shown) is present in the L2CC_clken path that ensures synchronous operation of L2CC_clken to the 208 MHz clock, even though the AHB signals that enter this path are generated on a 104 MHz clock, and are thus multi-cycle paths.

There is also logic in the L2CC_clken path (not shown), which gates the L2CC clock with AHBCLKEN for debug purposes when the clken_core register bit is set.

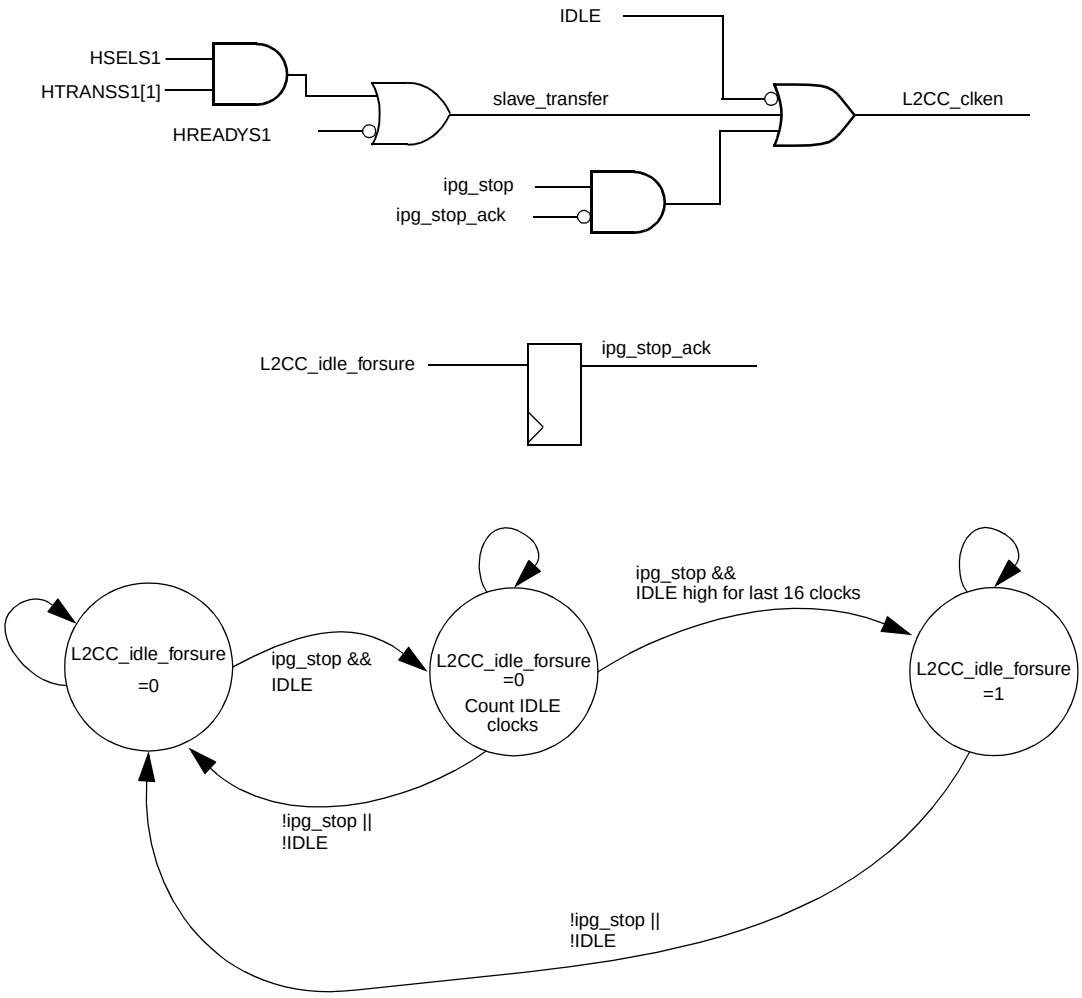


Figure 14-4. L2CC Clock Gating and ipg_stop_ack Function

Timing of ipg_stop_ack is illustrated in Figure 14-5. The first is a typical stop request while the L2CC is busy. The second scenario illustrates the case where the L2CC is idle when the stop request is issued.

NOTE

L2CC_clken is initially low. It is asserted so that IDLE clocks can be counted and ipg_stop_ack can be asserted. In this case, the CCM should honor ipg_stop_ack, even when L2CC_clken is initially low.

There should be no transfers on the L2CC slave port after ipg_stop has been asserted. Otherwise, there would be a possibility that the L2CC would assert ipg_stop_ack, the CCM would latch this value, and then the L2CC would begin operating on a subsequent slave port transfer. If this were to occur, and the clock to the L2CC were shut off based on the prior ipg_stop_ack assertion, data corruption could occur. No slave port transfers occur after ipg_stop is asserted.

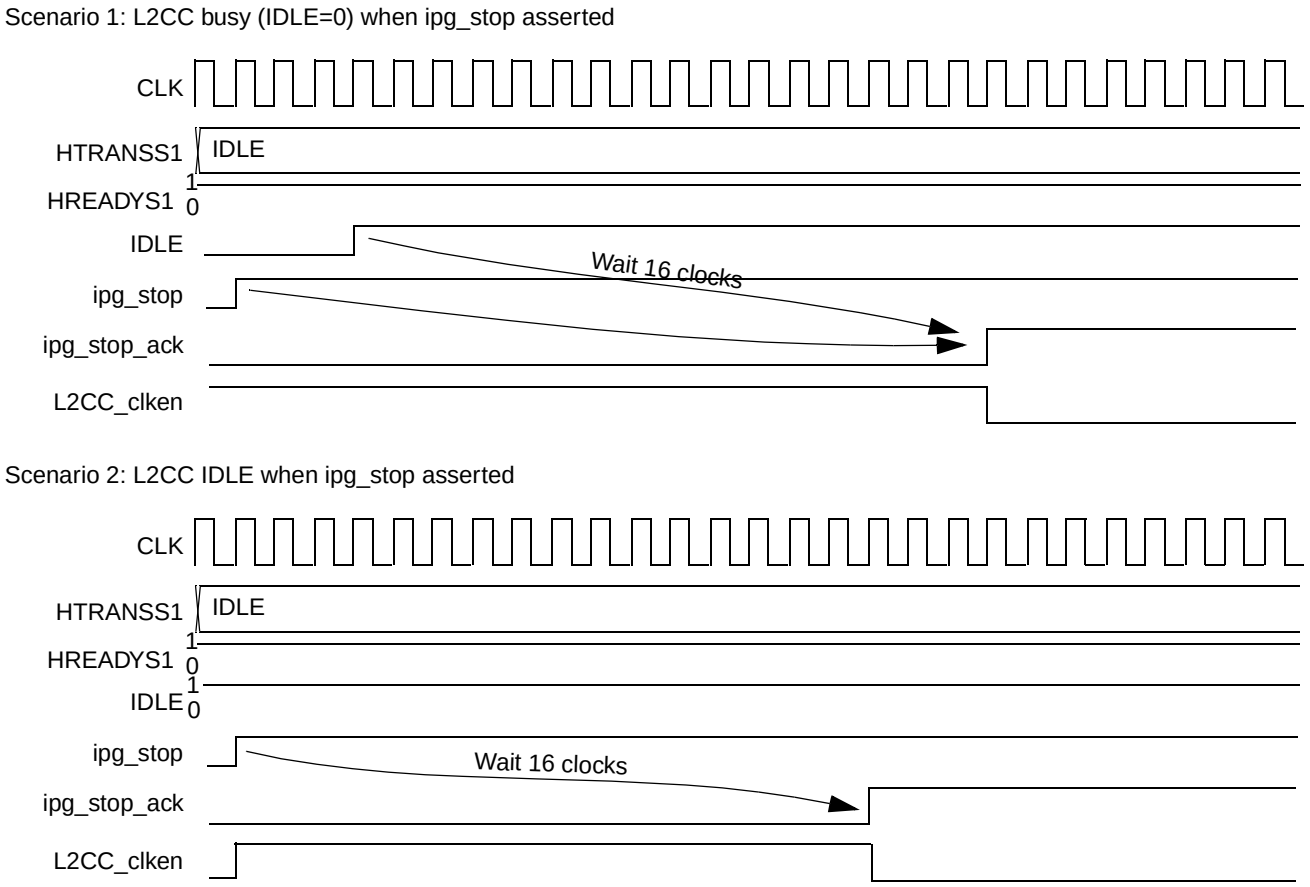


Figure 14-5. ipg_stop_ack Timing Diagrams

14.2.3 L2CC Disabled

When the L2CC block is present but not enabled, transactions will be passed through to the external main memory system on the L2CC master port output. The penalty introduced by disabled L2CC is twice the depth of internal L2CC pipeline (once for slave to master, and the other for master to slave). The minimum L2CC latency is 1 CLK cycle on the slave port, 1 CLK cycle on the master port.

14.2.4 L2CC Target Speed

The L2CC slave interface and all memory cache operations are designed to run in 1:1 mode with the ARM11 clock (arm_clk). The master side of the L2CC is throttled with per_hclken. Therefore the L2CC master ports always run at the same frequency as the MAX and the rest of the AHB domain.

14.2.5 L2CC Power Management

The L2CC uses the following techniques for saving power:

- 256-bit data RAM port to reduce the number of RAM transactions

- Serial Tag and Data lookup
- Serial Tag and Data lookup on L2 hits
- No Data RAM lookup on L2 misses

The l2cc_idle signal is used in conjunction with AHB snoop logic to perform module level clock gating of the L2CC within the JSYNC module.

14.2.6 L2CC Performance

The L2CC on the ARM11 Platform implements zero wait-state tag, valid and dirty memory arrays. These memories reside inside the common ARM11 Platform module. The L2 cache size can be up to 256 Kbyte. The L2 data memories reside outside of the common ARM11 Platform, is 128 Kbyte in size, and uses three-wait states for both reads and writes. The total access time to the first word of read data is calculated to be 11 or 12 AHB HCLK clocks. This number increases from the ARM11 perspective due to L1 pipeline delays. Table 14-2 shows the ARM11 Platform's L2CC access times for a 4 word burst read hit from the perspective of the ARM1136 L2 interface AHB's. These are best case access times, as there are other scenarios when access times on L2 hits could be longer (concurrent requests for example). Refer to ARM's *L2 Cache Controller (L2CC) Engineering Specification* for more detailed information on the L2CC design and the L2CC programmer's model.

Table 14-2. L2CC Burst Read Access Time

Burst of 4	Access Time in Clock Cycles
1st 64 bits of burst	11
2nd 64 bits of burst	12
3rd 64 bits of burst	13
4th 64 bits of burst	14

14.3 L2CC Memories

14.3.1 Latency Configuration

The memory latency is configured using the L2CC Auxiliary Control register. It resets to the slowest latency of eight cycles. Memory latencies can be programmed independently for the TAG/VALID, DIRTY and DATA memories, with separate read and write latency configuration bits for the DATA RAM. The L2CC must be disabled before configuring these bits.

14.3.1.1 L2 TAG/VALID and DIRTY Memories

The L2CC TAG/VALID and DIRTY memories are implemented using kbit compiled memories (the VALID bits are combined with the TAG bits to form a combined memory). There exist eight TAG/VALID memories and one DIRTY memory. Therefore, for the best performance the Associativity should be set to 8 way. The way size must be set to 16 Kbyte to correspond to 128 Kbyte cache size. The L2 cache architecture does not allow flexibility in setting way size in conjunction with associativity (for example,

32k way size and 4 way associativity). The Associativity and Way size are configured in the Auxiliary control register. Again the L2CC must be disabled before configuring these fields.

The latency for the TAG/VALID and DIRTY memories should be set to 0 wait-state.

14.3.1.2 L2 DATA Memory and Clock Stretch Circuit

The DATA memory is implemented using the Single Port Mega Bit SRAM. The Mega Bit memory has the requirement of a minimum high pulse clock width of approximately 4 ns, and a further restriction of only 1 clock pulse during the entire memory access cycle. To achieve this, a special Clock Stretch Circuit is employed. This circuit will generate the single clock pulse to the DATA memory upon receiving a valid enable signal and is held low at all other times for power savings. The Mega Bit memory has the following access time:

8ns (memory access time) + approximately 1ns (L2CC read path) = 9ns[1] total access time.

NOTE:

The “early write” version of the Mega Bit memory’s access times are the same for both reads and writes. For this reason, only the read latency setting is used by the Clock Stretch Circuit for controlling the memory clock. This mandates that the read and write latency should be set to the same value. The write latency should never be set faster than the read latency.

[1] Based on WCS PVT corner for both platform and memory.

The L2CC DATA memory latency should be configured based on the number of wait states required for the **9ns** access time, which is a function of the operating frequency. For example:

100 MHz clk speed = 10ns period, configure latency for 0 wait states
400 MHz clk speed = 2.5ns period, configure latency for 3 wait states, and others.

14.3.2 Mega Bit rval/wval Programmable Control Bits

The ARM11 Platform has reset inputs (l2_rval_rst, l2_rval2_rst, and l2_wval_rst) for the L2 DATA RAM rval, rval2, and wval configuration pins. These input pins are routed to the CLKCTL block’s RAM CTL register (bits 11:0) and sets the initial value of these register bits upon reset. Refer to Chapter 3, “Clocks, Power Management and Reset (AP Clock Controller Module)” for details. After reset, the RAM CTL register can be programmed (in supervisor mode only) to set the L2 DATA RAM rval, rval2, and wval control bits to different values during characterization.

CAUTION

These control bits should never be changed during functional mode of operation.

14.3.3 L2CC Clock-Gating Logic

The L2CC employs block-level clock-gating for power savings. There exists a clock-gate enable bit, bit 8 of the GP_CTRL register located in the CLKCTL block, to enable or disable L2CC clock-gating. The reset state of this is to NOT enable clock-gating. Refer to Chapter 3, “Clocks, Power Management and Reset

(AP Clock Controller Module)" for details. The L2CC also sends out a clock enable signal to the EVTMON (which is connected to the L2CC via the Event Bus) for gating off its high-speed clock when the L2CC clock is gated off.

14.4 Configuration and Control Registers

The L2CC is controlled via a set of memory-mapped registers that occupy a re-locatable 4 kilobyte of memory. The registers can be written using the AHB slave port when HSELRS1 is high. The registers are re-locatable by producing HSELRS1 with a combinational logic decode of HADDRS1 external to the L2CC.

All registers, except the test L2 data register, only accept 32-bit accesses. The results of other access sizes (8-bit, 16-bit, 64-bit, and 128-bit) are unpredictable. The test L2 data register only supports 32-bit and 64-bit accesses. The results of other access sizes (8-bit, 16-bit, and 128-bit) are unpredictable. Register r1, the Auxiliary control register, must only be written when the cache is turned off. You must disable the L2CC by writing to Control Register, and perform a Cache Sync operation before writing to any of the other internal registers.

Figure 14-6 shows the key to the register fields and Table 14-3 shows the register figure conventions.

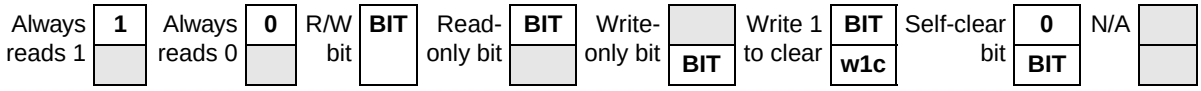


Figure 14-6. Key to Register Fields

Table 14-3. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.

Table 14-3. Register Figure Conventions (continued)

Convention	Description
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 14-4 shows L2CC register summary.

Table 14-4. L2CC Memory Map

Address	Register Number	Register	Access	Reset Value	Section/Page
0x3000_0000	0	L2 Cache ID	R	0xD500_0041	14.4.1.1/14-17
0x3000_0004	0	L2 Cache Type	R	0x1C10_0100	14.4.1.2/14-18
0x3000_0100	1	L2 Control	R/W	0x0000_0000	14.4.1.3/14-19
0x3000_0104	1	L2 Auxiliary Control Register 1	R/W	0xE402_0FFF	14.4.1.4/14-20
0x3000_0730	7	L2 Cache Sync	R	0x0000_0000	14.4.1.5/14-23
0x3000_077C	7	L2 Invalidate By Way	—	—	14.4.1.5/14-23
0x3000_0770	7	L2 Invalidate Line By PA	R	0x0000_0000	14.4.1.5/14-23
0x3000_07B8	7	L2 Clean Line by Index/Way	R	0x0000_0000	14.4.1.5/14-23
0x3000_07BC	7	L2 Clean by Way	—	—	14.4.1.5/14-23
0x3000_07B0	7	L2 Clean Line by PA	R	0x0000_0000	14.4.1.5/14-23
0x3000_07F8	7	L2 Clean and Invalidate Line by Index/Way	R	0x0000_0000	14.4.1.5/14-23
0x3000_07FC	7	L2 Clean and Invalidate by Way	—	—	14.4.1.5/14-23
0x3000_07F0	7	L2 Clean and Invalidate Line by PA	R	0x0000_0000	14.4.1.5/14-23
0x3000_0900	9	L2 Lockdown by Way-D side	—	—	14.4.1.6/14-25
0x3000_0904	9	L2 Lockdown by Way-I side	—	—	14.4.1.6/14-25
0x3000_0F00	15	L2 Test Operation	R	0x0000_0000	14.4.1.8/14-27
0x3000_0F10	15	L2 Line Data (8 x word)	—	—	14.4.1.8/14-27
0x3000_0F30	15	L2 Line Tag {Tag,V,D0,D1,VP}	R/W	0– – – –_– – – –	14.4.1.8/14-27
0x3000_0F40	15	L2 Debug control register	R/W	0– – – –_– – – –	14.4.1.8/14-27
0x3000_0104 (L2CCAUXCR)		L2CC Auxiliary Control Register 2 (L2CCAUXCR)			

NOTE

All addresses not defined in previous table are RESERVED, and must not be accessed as this can result in unpredictable behavior of the device.

All reserved or unused bits of registers must be written as zero, and ignored on read unless otherwise stated in the relevant text.

Table 14-5 shows the L2CC register summary.

Table 14-5. L2CC Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x3000_0000	R	RTL_IMPL								0	0	0	0	0	0	0	0
	W																
	R	CACHEID						PARTNUM				RTL_REL					
	W																
0x3000_0004	R	0	0	0	CTYPE				H	L2CACHEWAY				L2ASSOC[19:15]			
	W																
	R	L2	SBZ	L2CACHLEN		L2CACHEWAY				L2ASSOC					SBZ	L2CACHLEN	
	W																
0x3000_0100	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CACHEN
	W																

Table 14-5. L2CC Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x3000_0104	R	WRAP8 DISABLE	WRAP4 DISABLE	INCR DISABLE	CLKEN_CORE	IDLE8 DISABLE	IDLE PIN FORCE	WRAP2 DISABLE	EXCLUSIVE ABORT DISABLE	WRITE-ALLOCATE OVERRIDE	SHARED ATTRIBUTE OVERRIDE	PARITY ENABLE	EVENT MONITOR BUS ENABLE	WAY SIZE			ASSOCIATIVITY[16]
	W																
	R	ASSOCIATIVITY[15:13]			WRAP ACCESSES DISABLE	DIRTY RAM CYCLES OF LATENCY			TAG RAM CYCLES OF LATENCY			DATA RAM WRITE CYCLES OF LATENCY			DATA RAM READ CYCLES OF LATENCY		
	W																

14.4.1 Register Descriptions

This section contains the detailed description for the L2CC registers.

14.4.1.1 Register 0: L2CC Cache ID Register

This read only register returns the 32-bit device ID code. The device ID is specified by the value tied on the CACHEID[5:0] input. The format of the ID register is shown in Figure 14-7. The field descriptions of ID Register are shown in Table 14-6.

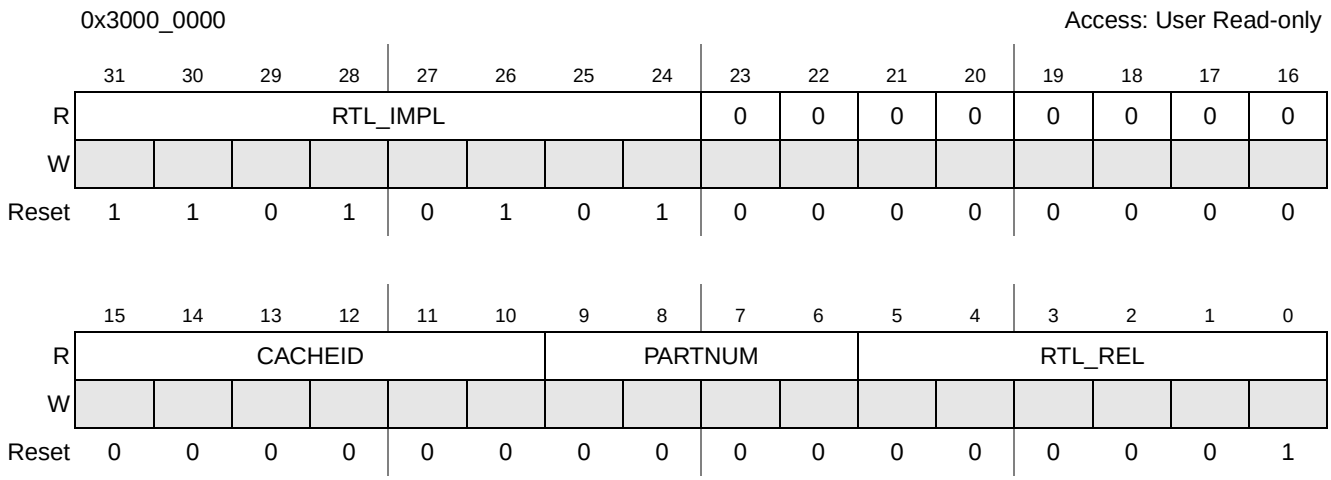


Figure 14-7. ID Register

Table 14-6. ID Register Field Descriptions

Field	Description
31–24 RTL_IMPL	RTL Implementor
23–16	Reserved. Reads as zero
15–10 CACHEID	CACHEID. Input pins for layout implemented
9–6 PARTNUM	Part number index
5–0 RTL_REL	RTL release index

14.4.1.2 Register0: L2CC Cache Type Register

This read-only register will return the 32-bit Cache Type, which makes the cache size a product of *L2 cache way size* and the *L2 associativity.* The format of the cache type register is shown in Figure 14-8. Table 14-7 provides the register's field descriptions.

0x3000_0004 Access: User Read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	CTYPE				H	L2CACHEWAY				L2ASSOC[19:15]			
W																
Reset	0	0	0	1	1	1	0	0	0	0	0	1	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	L2	SBZ	L2CACHLEN		L2CACHEWAY				L2ASSOC				SBZ	L2CACHLEN		
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

Figure 14-8. Cache Type Register

Table 14-7. Cache Type Register Field Descriptions

Field	Description
31–29	Reserved. Reads as zero
28–25 CTYPE	Lockdown Format C
24 H	Unified
23–20 L2CACHEWAY	L2 cache-way size. The L2 way size and associativity is read from the Auxiliary Control register.
19–15 L2ASSOC	L2 associativity. The L2 way size and associativity is read from the Auxiliary Control register.
14 SBZ	
13–12 L2CACHLEN	L2 cache line length. 32 bytes.
11–8 L2CACHEWAY	L2 cache-way size. The L2 way size and associativity is read from the Auxiliary Control register.
7–3 L2ASSOC	L2 associativity. The L2 way size and associativity is read from the Auxiliary Control register.
2 SBZ	
1–0 L2CACHLEN	L2 cache line length. 32 bytes.

14.4.1.3 Register 1: L2CC Control Register

The format of the L2CC control register (Register 1) is shown in Figure 14-9. Table 14-8 provides the field descriptions of control Register.

0x3000_0100 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CACH EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14-9. L2CC Control Register 1

Table 14-8. L2CC Control Register Field Descriptions

Field	Description
31–1	Reserved.
0 CACHEN	Cache Enable. Unified L2 Cache enable. 0 L2 Cache is in bypass mode (default). 1 L2 Cache is enabled.

14.4.1.4 Register 1: L2CC Auxiliary Control register

The format of the auxiliary control register is shown in Figure 14-10. The encoding of the Auxiliary Control Register is shown in Table 14-10. As mentioned earlier, the L2CC must be disabled when setting this register.

Table 14-9 provides a few L2CC Auxiliary Control register configuration examples:

Table 14-9. Aux Ctrl Configuration Examples

Frequency	Cache Size	Event Monitor Bus	Aux Control Value
400 MHz	128k	Disabled	0x0003001B
100 MHz	128k	Enabled	0x00130000

NOTE

The ARM11 Platform does *not* support RAM Parity or error. Therefore, the Parity Enable field (bit 21) of the Auxiliary Control register should always be 0 (disabled). Also, because the L2 cache size is128k, the way size must always be set to the 16k size (no other size setting is allowable) and the associativity should be set to 8 way for maximum performance.

0x3000_0104 Access: User Read/Write

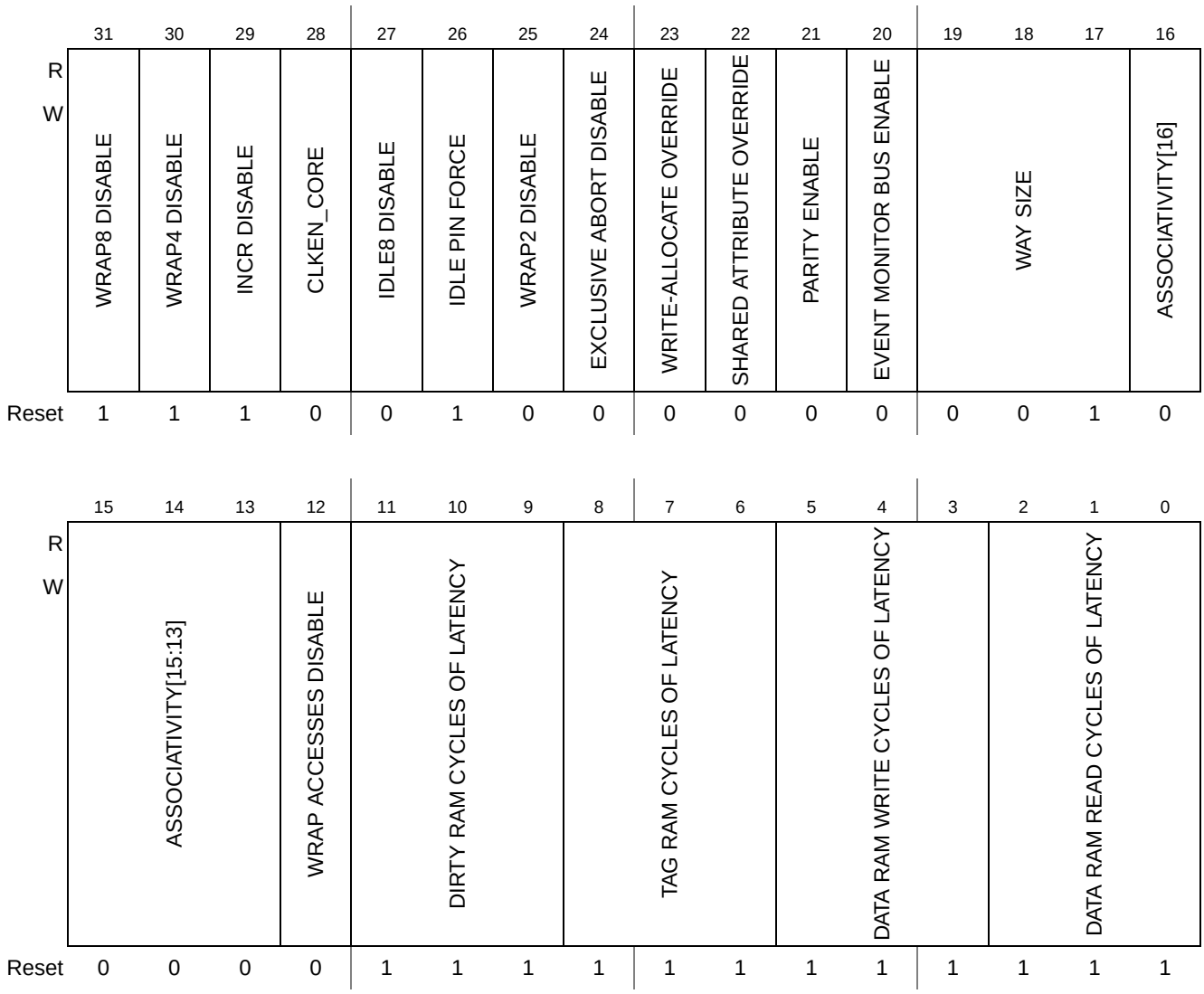


Figure 14-10. L2CC Control Register

Table 14-10. L2CC Auxiliary Control Register Field Descriptions

Field	Description
31 WRAP8DISABLE	0 WRAP8 accesses are supported on the slave port. 1 All WRAP8 slave port transfers are treated as if they were NONSEQ SINGLE transfers. (Default)
30 WRAP4 DISABLE	0 WRAP4 accesses are supported on the slave port. 1 All WRAP4 slave port transfers are treated as if they were NONSEQ SINGLE transfers. (Default)

Table 14-10. L2CC Auxiliary Control Register Field Descriptions (continued)

Field	Description
29 INCRDISABLE	0 INCR accesses are supported on the slave port. 1 All INCR slave port transfers are treated as if they were NONSEQ SINGLE transfers. (Default) All verification performed with this setting. **Note:** This bit applies only to unspecified length bursts (INCR). INCR4, INCR8, and INCR16 bursts are always treated by the L2CC as if they are NONSEQ SINGLE transfers.
28 CLKEN_CORE	Clock Enable Core. Used for debug of L2CC clock gating 0 Use CLKENAHB only for gating AHB ports. Provides highest performance. 1 Use CLKENAHB to gate both AHB ports and core L2CC logic.
27 IDLE8 DISABLE	Idle 8 Disable. Used for debug of L2CC IDLE pin. 0 IDLE pin de-assertion is extended for 8 clocks. All verification performed with this setting. (Default) 1 IDLE pin de-assertion is not extended. **Note:** The L2CC has a known bug when IDLE8_disable=1. Always program IDLE8_disable=0.
26 IDLE PIN FORCE	Idle pin force. Used for debug of L2CC IDLE pin 0 Normal operation 1 IDLE pin is forced low, and is never asserted high. This means that the L2CC will never internally turn off its clocks. (Default) **Note:** The L2CC is intended to be operated with IDLE_pin_force=0, for minimum power consumption. The default value is 1 for a fail-safe mechanism. Program to 0 upon boot.
25 WRAP2 DISABLE	0 WRAP2 accesses are supported on the slave port. (Default) 1 All WRAP2 slave port transfers are treated as if they were NONSEQ SINGLE transfers
24 EXCLUSIVE ABORT DISABLE	0 Sends an ERROR response back to exclusive access in a cacheable, shared memory region with shared override bit set. (Default) 1 Abort generation for exclusive access disable. Treat as cacheable non-shared accesses. **Note:** Exclusive access not supported. Bit must be programmed to 0.
23 WRITE ALLOCATE OVERRIDE	0 Use HPROT attributes (default) 1 Override HPROT attributes (All WT and WB accesses are read and write allocate)
22 SHARED ATTRIBUTE OVERRIDE ENABLE	0 Shared accesses treated as NC (default). 1 Shared attribute internally ignored (still forwarded to L3). **Note:** Shared accesses not supported. Bit must be programmed to 0.
21 PARITY ENABLE	0 Disabled (default) 1 Enabled **Note:** Parity not supported. Bit must be programmed to 0.
20 EVENT MONITOR BUS ENABLE	0 Disabled (default) 1 Enabled
19–17 WAY-SIZE	000 8 Kbyte 001 16 Kbyte (default) 010 32 Kbyte 011 64 Kbyte 100 128 Kbyte 101 256 Kbyte 110–111Reserved, and internally mapped to 256 kilobytes

Table 14-10. L2CC Auxiliary Control Register Field Descriptions (continued)

Field	Description
16–13 ASSOCIATIVITY	0000 Cache absent (default). 0001 Direct-mapped cache 0010 2-way cache 0011 3-way cache 0100 4-way cache 0101 5-way cache 0110 6-way cache 0111 7-way cache 1000 8-way cache 1001–1111 Reserved, and internally mapped to 8-way associativity
12 WRAP ACCESSES DISABLE	0 Master port can perform Wrap accesses (default). 1 Master port will not perform Wrap accesses (only SNGL and incrementing bursts will be used on the master port)
11–9 CYCLES OF LATENCY FOR DIRTY RAM	The output signals DIRTYLAT[2:0], TAGLAT[2:0], WDATALAT[2:0], and RDATALAT[2:0] reflect the value set in the Auxiliary Control Register in the respective fields. 000 1 cycle of latency, no additional latency 001 2 cycles of latency 010 3 cycles of latency 011 4 cycles of latency 100 5 cycles of latency 101 6 cycles of latency 110 7 cycles of latency 111 8 cycles of latency (default)
8–6 CYCLES OF LATENCY FOR TAG RAM	
5–3 CYCLES OF LATENCY FOR DATA RAM WRITES	
2–0 CYCLES OF LATENCY FOR DATA RAM READS	

14.4.1.5 Register 7: L2CC Cache Maintenance Operations

Table 14-11 shows L2CC cache maintenance operations.

Table 14-11. L2CC Cache Maintenance Operations

Address	Operation	Description
0x3000_07B0	Clean Line by PA (physical address)	Write the specific L2 cache line to main memory if the line is marked as valid and dirty. The line is marked as not dirty. The valid bit is unchanged.
0x3000_07B8	Clean Line by Index/Way	Write the specific L2 cache line within the specified way to main memory if the line is marked as valid and dirty. The line is marked as not dirty. The valid bit is unchanged.
0x3000_0770	Invalidate Line by PA	Specific L2 cache line is marked as not valid.
0x3000_07F0	Clean and Invalidate Line by PA	Write the specific L2 cache line to main memory if the line is marked as valid and dirty. The line is marked as not valid.
0x3000_07F8	Clean and Invalidate Line by Index/Way	Write the specific L2 cache line within the specified way to main memory if the line is marked as valid and dirty. The line is marked as not valid.

Table 14-11. L2CC Cache Maintenance Operations (continued)

Address	Operation	Description
0x3000_0730	Cache Sync	Drain eviction buffer (EB) to L3. Drain write buffer. Drain write-allocate buffer (WA) to the L2 data RAM. Invalidate LRB.
0x3000_077C	Invalidate by Way	Invalidate all data in specified ways, including dirty data. Completes as a background task. Invalidate All operation is equivalent to Invalidate by way while selecting all cache ways. Completes as a background task with the selected way or ways locked preventing allocation.
0x3000_07BC	Clean by Way	Writes each line of the specified L2 cache ways to main memory if the line is marked as valid and dirty. The lines are marked as not dirty. The valid bits are unchanged. Completes as a background task with the selected way or ways locked preventing allocation.
0x3000_07FC	Clean and Invalidate by Way	Writes each line of the specified L2 cache ways to main memory if the line is marked as valid and dirty. The lines are marked as not valid. Completes as a background task with the selected ways locked preventing allocation.

14.4.1.5.1 Atomic Operations

The following operations will stall the requesting slave port until they are complete:

- Clean Line
- Invalidate Line
- Clean and Invalidate Line
- Cache Sync

The time to complete these operations depends on the L3 memory speed. A clean operation requires 4 master port transfers if only one half of the cache line is dirty, or 8 master port transfers if both halves are dirty. In the worst case, a cache sync can initiate approximately 50 AHB master port transfers if all the internal buffers are full.

14.4.1.5.2 Background Operations

The following operations are run as background tasks.

- The appropriate R7 sub-register must be polled to see when a background cache operation has completed.
  - — Invalidate by Way
  - — Clean by Way
  - — Clean and Invalidate by Way
- The requesting slave port is not stalled during the operation. During these operations, the software should not access these ways. Any access to a way that is subject to maintenance operation must be regarded as unpredictable because you do not know whether it is a hit or a miss in the L2 cache.
- If an atomic or background task is requested before a previously requested task has completed, the second request is ignored.

- During background operation the targeted way is considered as being locked, which means that no allocation will occur to that way on read or write misses.
  - Line-based and way-based cache maintenance operations
- The cache maintenance operations are executed by writing to the cache operations register r7. For cache maintenance operations on a line, the line may be accessed using: Physical Address or Index/Way.

14.4.1.5.3 Line Based Operations

For line based operations, the PA (Figure 14-11) or Index/Way (Figure 14-12) is given on HWDATAS1.

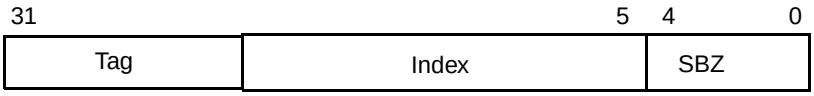


Figure 14-11. PA Format

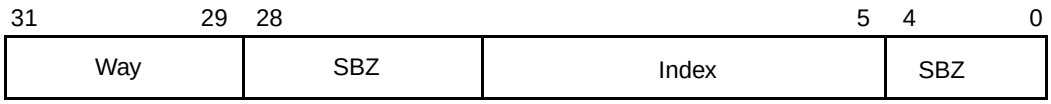


Figure 14-12. Index/Way Format

These operations are atomic operations, and writing to the register will start the operation, on the line specified by either {Tag, Index} or {Way, Index}. Tag and Index fields sizes depend on cache way size. The requesting slave port will remain blocked until the operation has completed. Those registers are always read as 0.

14.4.1.5.4 Way-Based Operations

For way based operations, the way is given on HWDATAS1, using the lockdown format C format shown in Figure 14-13. Multiple ways can be selected at the same time, by setting the Way bits to 1.

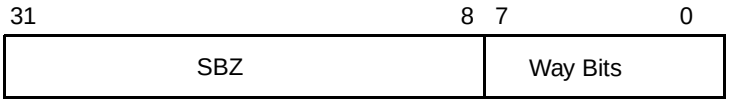


Figure 14-13. Format C Format

Way-based operations are background operations, so writing to the register will start the operation, on the Ways set to 1 in bits [7:0]. The Way bits will be reset to 0 as the respective Ways are cleaned and/or invalidated, so the register 7 sub-register should be polled to see when the operation has completed.

14.4.1.6 Register 9: L2CC Cache Lockdown

Cache lockdown is controlled by the cache lockdown register R9. It is implemented as 2 lockdown sub-registers, 1 for data and 1 for instructions.

If a cache lookup misses and a cache linefill is required, there are eight possible locations where the new line can be placed. Lockdown format C restricts the cache replacement algorithm to only use a subset of eight possible locations. The choice of an 8-way L2CC increases hit rate and increase effectiveness of Lockdown Format C.

If all ways are locked, a linefill is performed on a cache miss, reading one line from external memory, but the cache is not updated with the linefill data. In the same way, Write-Though write allocate and Write-Back write allocate accesses are treated as normal Write-Though and Write-Back accesses respectively, if all ways are locked.

14.4.1.6.1 Uses of Lockdown Format C

Lockdown format C, defined by the ARM caches, provides a method to restrict the replacement algorithm used on cache linefills, to only use selected cache WAYS within a SET. Using this method, code can be fetched or loaded into the L2 cache and protected from being evicted.

- Use lockdown format C to restrict the available ways for cache filling to n-ways, where n is less than the total number of ways, for example by only enabling filling to way 0.
- Use lockdown format C to fetch code into the cache:
  - — Executing the routine for the first time
  - — Loading the code into the cache, using a non-cacheable load routine.
- Use lockdown format C to load data into the cache by:
  - — Loading data into the cache for the first time
  - — Loading data into cache, using a non-cacheable load routine
- Write to the lockdown register to prevent filling to way 0, but enable filling to ways 1-7. The code and data is now protected in the cache and is not evicted on linefill.

14.4.1.6.2 Preventing or Reducing Cache Pollution

Consider an operating system or application using the entire cache for performance. A context switch to another routine occurs, which is too big to lock into the cache, but will benefit from the cache filling behavior to have high performance. In this case, restrict the number of WAYS which can now be used, effectively reducing the cache size. Then when context is returned to the operating system/application, allow the entire cache to be used. Because cache pollution was restricted, some of this code will still reside in the cache, so the impact in having switched context will have been reduced:

1. Context switch. Use lockdown format C to restrict the number of WAYS that can be used to 0-3, by locking out WAYS 4-7.
2. Execute new code. The cache will only use WAYS 0–3.
3. Context switch back. Unlock all ways, so WAYS 0–7 can be used.

14.4.1.6.3 Using Lockdown Format C for Processing Frame Buffers

There may be benefits in processing large frame buffers in the L2 cache, and making them appear as if there is a large amount of restricted physically addressed space available in fast memory. As the L2CC is 8-way set associative, using lockdown format C provides a simple method to load data into the L2CC using

the linefill mechanism, lock the cache memory to prevent eviction, then use the L2CC cache maintenance operations to efficiently clean the data to the main memory system.

Consider the example of requiring a 32 Kbyte Data Buffer in the 128 K 8-way set associative L2CC:

1. Set ARM11 Platform MMU attributes so the L1 is non-cacheable and the L2 is cacheable.
2. Set the L2CC lockdown to only fill to WAYS 0-1 (=32 Kbyte).
3. The 32 Kbyte will now be contiguously mapped over 2 WAYS of 16 Kbyte each.
4. Load data into the L2CC cache by executing a MOVE routine of ARM11 Platform, where a series of MOVE's are issued, a cache line apart from one another. The L1 attribute is non-cacheable, so the L1 cache will not be polluted. The L2 cache attribute is cacheable, so the L2CC will perform linefills, filling into WAYS 0-1. The linefill buffer in the L2CC will provide efficient filling.
5. When finished, set the L2CC lockdown to lock WAYS 0-1, and only allow filling to WAYS 2-7 (=96 Kbytes).

The L2 cache now contains a 32 Kbyte data buffer, and 96 Kbyte of 6-way associative cache. Lookups and reads/writes can occur to the entire 128 Kbyte, but the Data buffer is prevented from being evicted. L2CC cache maintenance operations can be used to efficiently clean to main memory. When the Data buffer is no longer required, WAYS 0-1 can be unlocked, and will be naturally overwritten.

Lockdown format C has a pattern matching field, so any of the 8 ways can be locked. There is no need to start at 0 and work up. If all ways are marked as being locked, then nothing will be allocated.

14.4.1.7 L2CC Replacement Strategy

The L2CC uses a pseudo-random replacement strategy. When used in combination with the lockdown registers, a deterministic replacement strategy can be achieved.

The pseudo-random replacement strategy will fill empty, unlocked ways first. If a set is completely full, the victim is chosen as the next unlocked way.

If a deterministic replacement strategy is desired, the lockdown registers are used to prevent ways from being allocated. For example, if the L2 size is 256 Kbyte (each way is 32 Kbyte), and the user wants a piece of code to reside in two ways (64 Kbyte space) with a deterministic replacement strategy, ways 1-7 must be locked before the code is filled into the L2 cache. The first 32 Kbyte of code will be so allocated into way 0 only. Then, way 0 should be locked and way 1 unlocked so that the second half of the code will be allocated in way 1.

There are two lockdown registers, one for data and the other for instructions, so you can also separate data and instructions into separate ways of the L2 cache, if desired.

14.4.1.8 Register 15: Test and Debug

14.4.1.8.1 Test registers

The test operation Register enables the contents of the L2 cache to be read and written.

For cache line read, test operation is written with the nRW bit cleared so that the content of the line designated by the Index/Way fields is put in the line data registers and its attributes put in the Line Tag

Register. All information needed about the cache line can then be retrieved by Line Data and Line Tag Registers.

For cache line writes, all line data, tag, and attributes must be written first in both the Line Data Registers and Line Tag Registers. At this time, by writing to the Test Operation Register, the cache line designated by the Index/Way fields is updated with register contents.

The Test Operation Register is stored in Index/Way format as shown in Figure 14-14. Index field size depends on cache way size.

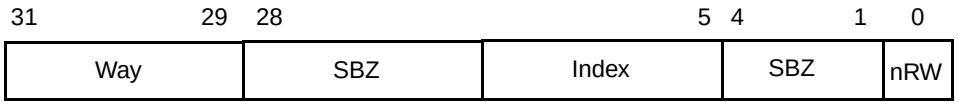


Figure 14-14. Index/Way Format of Test Operation Register, 0xF00

Words (32 bits) are stored in the following order in the cache line as shown in Figure 14-15.

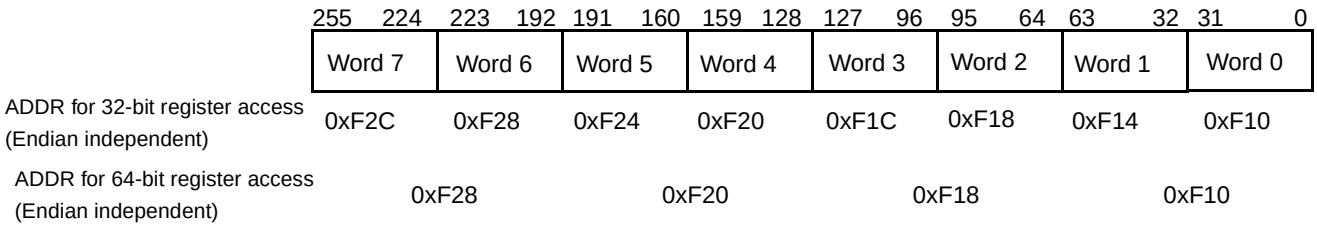


Figure 14-15. Word Order of L2 Cache Lines in Line Data Registers

When BIGEND=1, data stored in functional location 0x0C are stored in “Word 0” in Figure 14-15. When BIGEND=0, data stored in functional location 0x0 is stored in “Word 0” in Figure 14-15. When performing a 64-bit read of location 0xF10, {word1,word0} will be returned, for both Big and Little Endian. For 32-bit access, reading 0xF10 will return word0, while reading 0xf14 will return word1, for both Big and Little Endian.

14.4.1.9 L2 Line Tag Register

The Line Tag Register is shown in Figure 14-16. Table 14-12 shows the Line Tag Register field descriptions.

0x3000_0F30 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TAG[31:16]															
W																
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TAG[15:13]			VALID	DIRTY1	DIRTY0	VICTIM POINTER			0	0	0	0	0	0	0
W																
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Figure 14-16. Line TAG Register

Table 14-12. Line Tag Register Field Descriptions

Field	Description
31–13 TAG	Tag. An invalid line always has its tag set to zero until it has been written through a test operation
12 VALID	Valid.
11 DIRTY1	Dirty 1. Defines state of the last four words in the cache line, words 4-8.
10 DIRTY0	Dirty 0. Defines state of the first four words in the cache line, words 0-3.
9–7 VICTIMPOINTER	Victim pointer. Defines the last allocated way. 111 represents way 7 ——— 000 Represents way 0
6–0	Reserved

Address Base + 0xF10 stands for Word 0 in the line, up to Base+0xF2C which stands for Word 7.

14.4.1.10 L2CC Debug Control Register

The L2CC debug control register is used to force specific cache behavior required for debug. Figure 14-17 lists the L2CC debug control register bit assignments, and Table 14-13 provides its field descriptions.

0x3000_0F40 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DWB	DCL
W																
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0	0

Figure 14-17. L2CC Debug Control Register

Table 14-13. L2CC Debug Control Register

Field	Description
31–2	Reserved.
1 DWB	Disable write-back (force WT). 0 Enable write-back behavior (default). 1 Force write-through behavior
0 DCL	Disable Cache Linefill. 0 Enable cache linefills 1 Disable cache linefills

14.4.2 Forcing Write-Through Behavior

Setting the DWB bit to 1 forces the L2CC to treat all cacheable accesses as though they were in a Write-through no write-allocate region of memory. Setting of the DWB bit overrides access attributes. If the cache contains dirty cache lines, these remain dirty while the DWB bit is set, unless they are written back because of a write-back eviction after a linefill, or because of an explicit clean operation. Lines that are clean are not marked as dirty if they are updated while the DWB is set. This functionality allows a debugger to download code or data to external memory, without the requirement to clean part or all of the cache to ensure that the code or data being downloaded has been written to external memory.

If the DWB is set, and a write is made to a cache line that is dirty, then both the cache line and the external memory are updated with the write data. Other entries in the cache line still have to be written back to main memory to achieve coherency.

14.4.2.1 L2CC Auxiliary Control Register 2 (L2CCAUXCR)

See Figure 14-18 for an illustration of valid bits in the SIM Port1 Control Register, and Table 14-14 for its field descriptions.

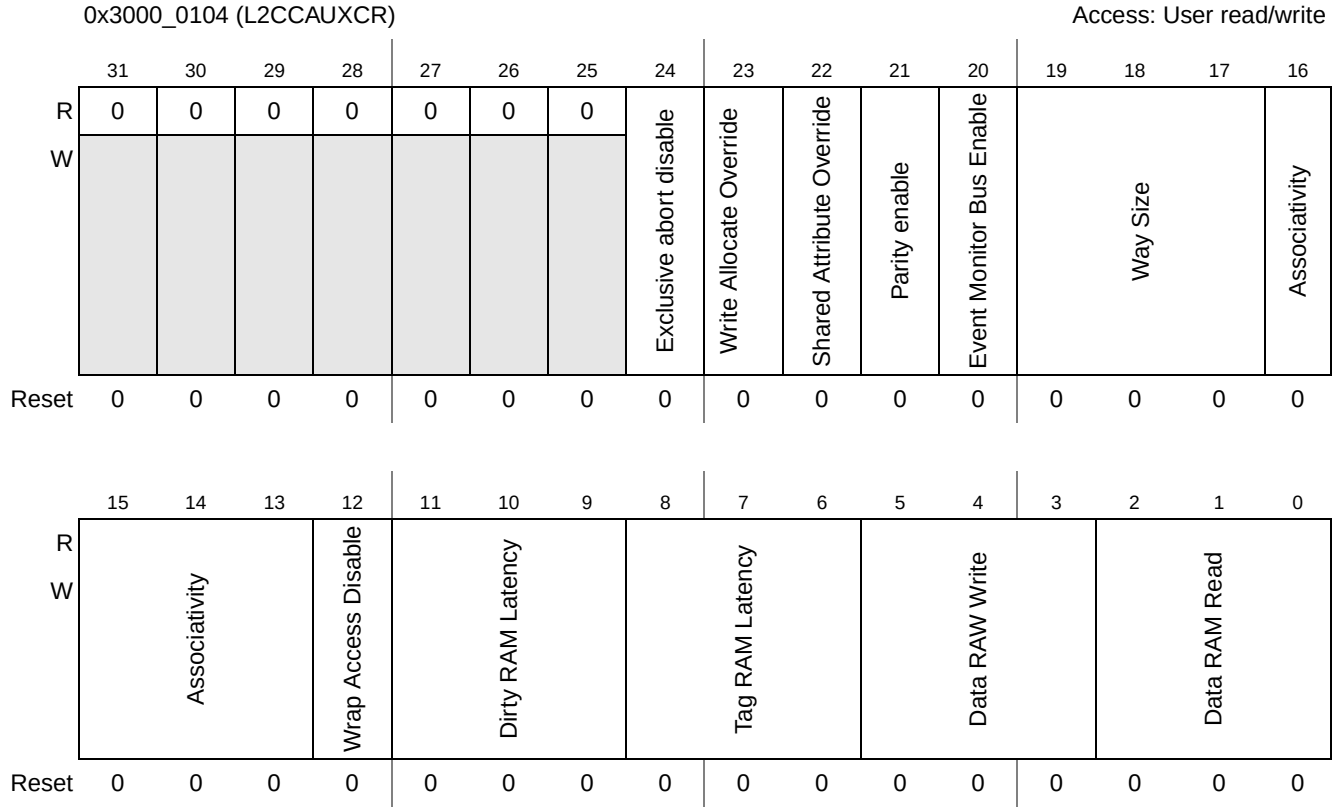


Figure 14-18. L2CC Auxiliary Control Register 2 (L2CCAUXCR)

Table 14-14. L2CC Auxiliary Control Register Field Descriptions

Field	Description
31–25	Reserved. These bits are reserved and should read zero.
24 Exclusive abort disable	0 L2CC sends an ERROR response back to exclusive access in a cacheable, shared memory region with shared override bit set (default)). 1 Abort generation for exclusive access disabled. Treated as cacheable non-shared accesses.
23 Write Allocate Override	Write Allocate Override. When enabled all Write-Through and write-back accesses are Read/Write-allocate. 0 Use of HPROT attributes (default) 1 Override HPROT attributes.
22 Shared Attribute Override	Shared Attribute Override. This bit determines if the shared attribute internally ignored but still forwarded to L3 memory 0 Shared accesses treated as non-cacheable (default) 1 Shared attribute internally ignored but still forwarded to L3 memory.
21 Parity	Parity. 0 Disabled (default) 1 Parity cannot be enabled
20 Event Monitor Bus Enabled	Event Monitor Bus Enabled 0 Disabled (default) 1 Enabled.

Table 14-14. L2CC Auxiliary Control Register Field Descriptions (continued)

Field	Description
19–17 Way Size	Way Size. 000 Reserved, and internally mapped to 16 Kbyte 001 16 Kbyte default 010 32 Kbyte 011 64 Kbyte 100 128 Kbyte 101 256 Kbyte 110–111 Reserved, and internally mapped to 256 Kbyte.
16–13 Associativity	Associativity 0000 Cache absent (default) 0001 direct-mapped cache 0010 2-way cache 0011 3-way cache 0100 4-way cache 0101 5-way cache 0110 6-way cache 0111 7-way cache 1000 8-way cache 1001-1111Reserved, and internally mapped to 8-way associativity.
12 Wrap Access Disable	Wrap Access Disable 0 Master ports can perform wrap accesses (default). 1 Wrap accesses requested on slave ports are converted to linear accesses on master ports.
11–9 Dirty RAM Latency	Dirty RAM Latency 000 1 cycle of latency, no additional latency 001 2 cycles of latency 010 3 cycles of latency 011 4 cycles of latency 100 5 cycles of latency 101 6 cycles of latency 110 7 cycles of latency 111 8 cycles of latency (default). **Note:** The output signals DIRTYLAT[2:0], TAGLAT[2:0], WDATALAT[2:0], and RDATALAT[2:0] reflect the values set in the Auxiliary Control Register in the respective fields.
8–6 Tag RAM Latency	Tag RAM Latency. 000 1 cycle of latency, no additional latency 001 2 cycles of latency 010 3 cycles of latency 011 4 cycles of latency 100 5 cycles of latency 101 6 cycles of latency 110 7 cycles of latency 111 8 cycles of latency (default). **Note:** The output signals DIRTYLAT[2:0], TAGLAT[2:0], WDATALAT[2:0], and RDATALAT[2:0] reflect the values set in the Auxiliary Control Register in the respective fields.

Table 14-14. L2CC Auxiliary Control Register Field Descriptions (continued)

Field	Description
5–3 Data RAW Write	Data RAW Write 000 1 cycle of latency, no additional latency 001 2 cycles of latency 010 3 cycles of latency 011 4 cycles of latency 100 5 cycles of latency 101 6 cycles of latency 110 7 cycles of latency 111 8 cycles of latency (default). **Note:** The output signals DIRTYLAT[2:0], TAGLAT[2:0], WDATALAT[2:0], and RDATALAT[2:0] reflect the values set in the Auxiliary Control Register in the respective fields.
2–0 Data RAM Read	Data RAM Read 000 1 cycle of latency, no additional latency 001 2 cycles of latency 010 3 cycles of latency 011 4 cycles of latency 100 5 cycles of latency 101 6 cycles of latency 110 7 cycles of latency 111 8 cycles of latency (default). **Note:** The output signals DIRTYLAT[2:0], TAGLAT[2:0], WDATALAT[2:0], and RDATALAT[2:0] reflect the values set in the Auxiliary Control Register in the respective fields.

14.5 L2 Initialization/Application Information

This section provides initialization and application information for the L2CC. To cache instructions/data into the L2CC, the cache must first be configured and invalidated. The way size, associativity, and cache memory latencies are specified using the Auxiliary Control register. Then, the cache is invalidated using the Invalidate By Way Cache Maintenance Operations register.

Next, it is recommended that the user initialize the MMU. Next, the L2CC is enabled by writing to the L2 Control register before the MMU is enabled.

14.5.1 Configuring the ARM11P L2CC

The ARM11 Platform memory map places the L2 registers at the base address of 0x3000_0000. The registers are listed and described in the *L2 Cache Controller Technical Reference Manual*.

The following steps are required to configure and initialize the L2CC:

1. Configure the Auxiliary Control Register.
2. Invalidate the ways.
3. Enable the L2 cache.

For all of ZATS the Auxiliary Control Register, located at address: 0x3000_0104, should be written with the value: 0x0003_001B. This value corresponds to the following configuration:

- 128k cache size (16k way size)
- 8-way associativity

- Event Monitor Bus disabled (set bit 20 to enable the Event Bus for monitoring by the EVTMON)
- 1 clk latency (0 wait state) TAG/VALID and DIRTY memories.
- 4 clk latency (3 wait state) DATA memory for Read and Write.

NOTE:

The L2CC must be disabled when accessing internal registers.

14.5.1.1 Invalidating the L2CC Cache Memory

The cache memory is invalidated using the Cache Invalidate by Way Maintenance Operations register. This register uses the Format C way-based background operation. Writing 0x0000_00FF to this register, located at address: 0x3000_077C, will invalidate the entire L2 cache.

NOTE

The cache way-based maintenance operations, and specifically the Invalidate by Way operation, run as background tasks. This requires polling the register to determine when the operation is complete.

The Cache Invalidate by Way Maintenance Operations register will transition to zero upon completion of the operation.

The following code sequence accomplishes the invalidation process:

```
ldr      r0, =0x3000_077C
ldr      r1, =0x0000_00FF
str      r1, [r0]
@@ Poll Invalidate By Way register for completion of operation
loop: ldr        r2, [r0]
         cmp     r2, #0
         bne     loop
```

When the MMU is enabled, the L2 cache will function according to the caching policy established by the MMU Page table(s) for the various memory types and regions. Refer to Table 14-6 in the L2CC TRM for a detailed description of the L2CC behavior with ARMv6 memory types.

Chapter 15
ARM11 Event Monitor (EVTMON)

The ARM1136 Platform Event Monitor (EVTMON) is a 32-bit IP-bus peripheral used for monitoring the Level 2 Cache Controller (L2CC) events via the L2CC event bus interface.

15.1 Overview

The EVTMON contains four event counters (EMC0–EMC3) that may be used to count four different events selected from a list of 10 possible external events (through the L2CC event bus) or five internal events (overflows or clock edges). Each event counter is a 32-bit counter which has its own interrupt generation logic. By combining different statistics, a variety of useful performance matrix can be obtained.

15.2 Features

The EVTMON has the following features:

- One 32-bit Monitor Control Register (EMMC)
- One 32-bit Counter Status Register (EMCS)
- Four 32-bit Counter Configuration Registers (EMCC0–EMCC3)
- Four 32-bit read-only event counters (EMC0–EMC3)
- Ability to track the following events:
  - CPU instruction read request to level 2
  - CPU instruction read hit in level 2 cache
  - CPU data read request to level 2
  - CPU data read hit in level 2 cache
  - CPU data write request to level 2 (not write through)
  - CPU data write request to level 2 (write through)
  - CPU data write hit in level 2 cache
  - Level 2 buffered write abort
  - L2 cache half-line eviction (2 events for one full-line eviction)
  - Allocation to the L2 cache caused by write transaction (write allocate)
- Intended to run at same speed as L2CC block

15.3 Memory Map and Register Definition

The EVTMON has 10 registers which are accessed through the IP-Bus interface. All transactions to these registers must be 32-bit word accesses or a transfer error will occur. All registers may be accessed in either

User or Supervisor mode. Section 15.3.3, "Register Descriptions" provides the detailed descriptions for all of the EVTMON registers.

15.3.1 Memory Map

Table 15-2 shows the EVTMON memory map.

Table 15-2. EVTMON Memory Map

Address	Register	Access	Reset Value	Section/Page
0x43F0_8000 (EMMC)	Event Monitor Control	R/W	0x0000_0000	15.3.3.1/15-4
0x43F0_8004 (EMCS)	Counter Status Register	R/W	0x0000_0000	15.3.3.2/15-5
0x43F0_8008 (EMCC0)	Counter 0 Configuration	R/W	0x0000_0000	15.3.3.3/15-6
0x43F0_800C (EMCC1)	Counter 1 Configuration	R/W	0x0000_0000	
0x43F0_8010 (EMCC2)	Counter 2 Configuration	R/W	0x0000_0000	
0x43F0_8014 (EMCC3)	Counter 3 Configuration	R/W	0x0000_0000	
0x43F0_8020 (EMC0)	Counter Register 0	R	0x0000_0000	15.3.3.4/15-8
0x43F0_8024 (EMC1)	Counter Register 1	R	0x0000_0000	
0x43F0_8028 (EMC2)	Counter Register 2	R	0x0000_0000	
0x43F0_802C (EMC3)	Counter Register 3	R	0x0000_0000	

The EVTMON module decodes sufficient addresses for the register block to be unique within an 8 K memory space. If the selected space is larger than 8 K, the module registers will alias accordingly.

15.3.2 Register Summary

Figure 15-1 shows the key to the register fields and Table 15-3 shows the register figure conventions.

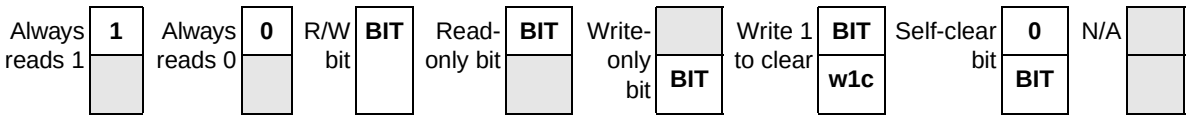


Figure 15-1. Key to Register Field

Table 15-3. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writeable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).

Table 15-3. Register Figure Conventions (continued)

Convention	Description
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a 1.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to 0.
1	Resets to 1.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 15-4 shows the EVTMON register summary.

Table 15-4. EVTMON Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x43F0_8000 (EMMC)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	INTPULDUR			POL	TYPE	EVTMONEN
	W					EMC3RST	EMC2RST	EMC1RST	EMC0RST								
0x43F0_8004 (EMCS)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	EMC3 Flag	EMC2 Flag	EMC1 Flag	EMC0 Flag
	W													w1c	w1c	w1c	w1c
0x43F0_8008 (EMCC0) 0x43F0_800C (EMCC1) 0x43F0_8010 (EMCC2) 0x43F0_8014 (EMCC3)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	SOURCE					FLAG	INTEN
	W																

Table 15-4. EVTMON Register Summary (continued)

Name		31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0x43F0_8020 (EMC0) 0x43F0_8024 (EMC1) 0x43F0_8028 (EMC2) 0x43F0_802C (EMC3)	R	COUNT
	W	
	R	COUNT
	W	

15.3.3 Register Descriptions

This section contains the detailed register descriptions for the EVTMON registers.

15.3.3.1 Monitor Control Register (EMMC)

The EMMC control register is used to provide the following:

- Enable an event monitor block
- Control interrupt generation
- Reset counters to zero

The EMMC register should only be accessed using a 32-bit read-modify-write sequence. Figure 15-2 shows the register; Table 15-5 provides its field descriptions.

0x43F0_8000 (EMMC) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	INTPULDUR			POL	TYPE	EVTMONEN
W					EMC3RST	EMC2RST	EMC1RST	EMC0RST								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 15-2. EMMC Register

Table 15-5. EMMC Register Field Descriptions

Field	Description
31–12	Reserved.
11–8 EMCxRST	Counter Resets. Used for resetting the event counters to zero. 0 No effect 1 Clears the corresponding counter register
7–6	Reserved.
5–3 INTPULDUR	Interrupt Pulse Duration. When configured for Edge Sensitive interrupts, this field encodes how many clocks the interrupt line should remain active. 000 1 CLK cycle 001 2 CLK cycles 010 4 CLK cycles 011 8 CLK cycles 100 16 CLK cycles 101 32 CLK cycles 110 64 CLK cycles 111 128 CLK cycles
2 POL	Interrupt Polarity. Configures the interrupt to be either active high or active low. 0 Interrupt signal is active low (default) 1 Interrupt signal is active high
1 TYPE	Interrupt Type. Configures the interrupt to be either level or edge sensitive. 0 Level sensitive (default), interrupt line remains active until all Counter Flags are cleared 1 Edge sensitive, interrupt line is active for n CLK cycles (n defined by bits 5–3)
0 EVTMONEN	Event Monitor Enable. Enable or disable the Event Monitor. 0 Disabled 1 Enabled

15.3.3.2 Counter Status Register (EMCS)

The Counter Status Register contains a flag for each counter and indicates if a counter set condition has occurred. It indicates which counter(s) caused an interrupt, if interrupt generation is enabled. If the interrupt is enabled and programmed as level sensitive, the interrupt remains active until all flags in the Counter Status register are cleared.

Figure 15-3 shows the register; Table 15-6 provides its field descriptions.

0x43F0_8004 (EMCS) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	EMC3 Flag	EMC2 Flag	EMC1 Flag	EMC0 Flag
W													w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 15-3. EMCS Register

Table 15-6. EMCS Field Descriptions

Field	Description
31–4	Reserved
3 EMC3	Event Counter 3 Flag. Indicates whether or not event counter 3 set condition has occurred. 0 Counter set condition has not occurred. 1 Counter set condition has occurred.
2 EMC2	Event Counter 2 Flag. Indicates whether or not event counter 2 set condition has occurred. 0 Counter set condition has not occurred. 1 Counter set condition has occurred.
1 EMC1	Event Counter 1 Flag. Indicates whether or not event counter 1 set condition has occurred. 0 Counter set condition has not occurred. 1 Counter set condition has occurred.
0 EMC0	Event Counter 0 Flag. Indicates whether or not event counter 0 set condition has occurred. 0 Counter set condition has not occurred. 1 Counter set condition has occurred.

15.3.3.3 Counter Configuration Registers (EMCCx)

Each counter register has its own Counter Configuration register (four total). The purpose of each Counter Configuration registers is to do the following:

- Specify the counter event source
- Define the counter set condition (the flag is in the Counter Status register)
- Enable appropriate interrupts

Figure 15-4 shows the register; Table 15-7 provides its field descriptions.

0x43F0_8008 (EMCC0)
0x43F0_800C (EMCC1)
0x43F0_8010 (EMCC2)
0x43F0_8014 (EMCC3)

Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	SOURCE					FLAG	INTEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 15-4. EMCCx Registers

Table 15-7. EMCCx Field Descriptions

Field	Description
31–7	Reserved
6–2 SOURCE	Counter Event Source. Specifies the type of event to be counted by the respective counter. Refer to Table 15-8, Event Sources, for a list of possible events.
1 FLAG	Counter Flag Set Condition. Counter flag set condition configuration bit. 0 Counter flag set on overflow (default) 1 Counter flag set on increment
0 INTEN	Counter Interrupt Generation Enable. Counter interrupt enable bit. 0 No interrupt is generated by the counter (default). 1 An interrupt is generated when the counter flag set condition is met.

Table 15-8. EVTMON Event Descriptions

Encoding	L2CC Pin	Event Description
00000	N/A	Counter disabled
00001	BWABT	Buffered write abort
00010	CO	L2 cache half-line eviction (2 events for one full-line eviction)
00011	DRHIT	Data read hit
00100	DRREQ	Data read request
00101	DWHIT	Data write hit
00110	DWREQ	Data write request
00111	DWTREQ	Data write request with write-through attribute
01000	IRHIT	Instruction read hit
01001	IRREQ	Instruction read request

Table 15-8. EVTMON Event Descriptions (continued)

Encoding	L2CC Pin	Event Description
01010	WA	Write allocate (L2 allocation caused from write transaction)
01011	N/A	reserved
01100	N/A	reserved
01101	N/A	EMC3 overflow
01110	N/A	EMC2 overflow
01111	N/A	EMC1 overflow
10000	N/A	EMC0 overflow
10001	N/A	CLK cycle (counter increments on every CLK cycle)

15.3.3.4 Counter Registers (EMCx)

There are four 32-bit read-only event counter registers. The event source for each counter is configured in the corresponding EMCCx register.

Figure 15-5 shows the register; Table 15-9 provides its field descriptions.

0x43F0_8020 (EMC0)
0x43F0_8024 (EMC1)
0x43F0_8028 (EMC2)
0x43F0_802C (EMC3)

Access: User read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	COUNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COUNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 15-5. EMCx Registers

Table 15-9. EMCx Descriptions

Field	Description
31–0 COUNT	Counter bits. Read only, reset by writing to the EMMC register (bits 11:8 respectively)

15.4 EVTMON Interrupts

The L2 Cache Controller (L2CC) event monitor can be configured to control how interrupts are generated using the following four steps:

- Interrupt signal setup: the interrupt line can be configured to be level-sensitive (interrupt tied to active level until interrupt is cleared) or edge-sensitive (one pulse to active level lasting a programmable number of CLK cycles). The active level is also configurable for active high or low.
- Counter setup: for each counter, the incrementing trigger is selected from all possible events.
- Counter Flag set condition setup: Counter flags can be set by two conditions, increment or overflow.
- Interrupt enable: An interrupt can only be generated by a counter if the associated Counter Interrupt Generation Enable bit is set in the corresponding Counter Configuration register, refer to Section 15.3.3.3, “Counter Configuration Registers (EMCCx)” for details.

15.5 Clock Gating

The EVTMON employs module level clock gating on both the high speed clock in the ap_clk domain and the ipg_clk in the per_clk domain.

The l2cc_clken input is used to determine if the L2CC’s clock is running and, thus, if the EVTMON high speed clock must also be running (as long as the EVTMON enable bit is set in the EMMC register). The l2cc_clken is also used to disable the counters in the event that the L2CC clock is gated off.

NOTE

The l2cc_clken signal can be set to a continuous enabled state by disabling clock gating of the L2CC. This is done by clearing bit 8 of the CLKCTL Control Register, refer to the ARM11 Platform CLKCTL specification for details.

IP-Bus transactions to the registers can always occur, be it to read, reset the counters or clear status flags, even if l2cc_clken is de-asserted.

Book II, Part 4: External Interfaces

Introduction

The i.MX31 and i.MX31L contain the following external interfaces:

External Memory Interface (EMI)

The External Memory Interface (NAND Flash Controller) controls all IC external memory accesses (read/write/erase/program) from all the masters in the system, through two Master Port Gaskets (MPG)—(interface [AHB 32-bit] and MPG64 [AHB 64-bit])—to different external memories. All accesses are arbitrated by the Multi Master Memory Interface (M3IF) module and controlled by the respective memory controller.

The EMI contains different external memory controllers to support several memory devices:

- M3IF—Multi Master Memory Interface
- ESDCTL/MDDRC—Enhanced SDRAM/LPDDR memory controller
- PCMCIA—PCMCIA memory controller
- NFC—NAND Flash memory controller
- WEIM—SRAM/PSRAM/FLASH memory controller

Multi-Master Memory Interface (M3IF)

The Multi-Master Memory Interface (M3IF) controls memory accesses (read/write/erase/program) from one or more masters through different port interfaces to different external memory controllers ESDCTL/MDDRC, PCMCIA, NANDFLASH, and WEIM.

Wireless External Interface Module (WEIM)

The Wireless External Interface Module (WEIM) provides the capability to the system for accessing external Flash and RAM memories connected to either of its six chip selects. It has the ability to provide a single-cycle burst access for external flashes and support for multiple burst devices when not using its smart burst feature. Other features include Big/Endian mode support, Cellular RAM support, and support for multiplexed address/data bus.

Enhanced SDRAM Controller (ESDCTL)

The Enhanced Synchronous Dynamic RAM Controller (ESDCTL) provides interface and control for synchronous DRAM memories for the system. SDRAM memories use a synchronous interface with all signals registered on a clock edge. A command protocol is used for initialization, read, write, and refresh operations to the SDRAM and is generated on the signals by the controller when required due to external or internal requests. It has support for both single data rate RAMs and double data rate SDRAMs. It supports 64, 128, 256, and 512-Mbit, 4 bank synchronous DRAM by two independent chip selects and with up to 64 Mbytes addressable memory per chip select.

NAND Flash Controller (NANDFC)

The NAND Flash Controller (NANDFC) device is a type of flash memory that is optimized for data storage applications with its unique cell structure, providing significant cost advantages over conventional NOR flash memory. The NAND Flash Controller integrates the functionality necessary for access to these devices. NAND Flash has a smaller bit cell but has a fast sequential access as compared to a NOR flash which has a large bit cell but a fast random access. This makes NAND Flash perfect as a data memory for storing audio/video files in NAND Flash because typically these files are stored in sequential manner while access to processor code is quite random.

Personal Computer Memory Card International Association (PCMCIA) Controller

The PCMCIA host adapter module provides the control logic for PCMCIA socket interfaces, and requires some additional external analog power switching logic and buffering. The additional external buffers allow the PCMCIA host adapter module to support one PCMCIA socket.

Chapter 16
External Memory Interface (EMI)

The External Memory Interface (EMI) controls all IC external memory accesses (read, write, erase, program) from all the masters in the system, through two port interfaces—Master Port Gasket (MPG) [AHB 32 bit] and MPG64 [AHB 64-bit])—to different external memories. All accesses are arbitrated by the Multi-Master Memory Interface (M3IF) module, and are controlled by the respective memory controller.

The EMI contains different external memory controllers in order to support several memory devices:

- M3IF—Multi Master Memory Interface
- ESDCTL/MDDRC—Enhanced SDRAM/LPDDR memory controller
- PCMCIA—PCMCIA memory controller
- NANDFC—NAND Flash memory controller
- WEIM—SRAM/PSRAM/FLASH memory controller

Figure 16-1 is the top-level diagram of the EMI that shows the functional organization of the block.

NOTE

The MBX R-S graphics accelerator is not available in the i.MX31L.

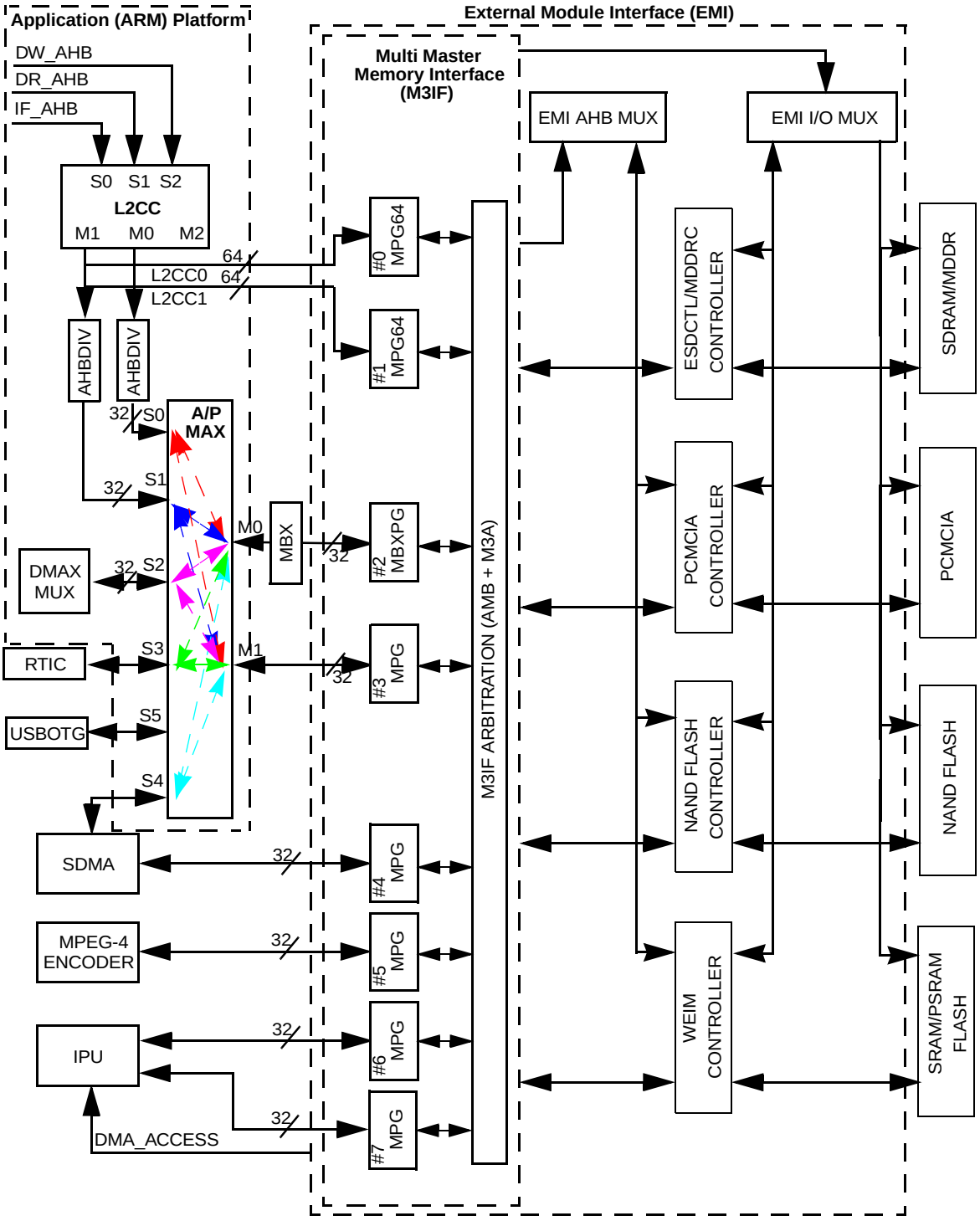


Figure 16-1. EMI System Block Diagram

16.1 Overview

The EMI provides the ability to connect to a wide variety of memory devices. This chapter contains the technical information about the operation and configuration of the EMI modules in the chip to allow the designer to quickly integrate external memory devices into new and existing designs. Several of the modules in the EMI portion of the chip share pins with the PCMCIA, EIM, SDRAMC, and NAND controllers.

The chip contains interfaces for the following types of memory devices:

- PCMCIA
- Flash Memory Devices
- SDRAM/Low Power DDR (LPDDR)

The M3IF-ESDCTL/MDDRC interface is optimized and designed to reduce access latency by generating multiple accesses through the dedicated ESDCTL/MDDRC arbitration (MAB) module, which controls the access towards/from the Enhanced SDRAM/MDDR memory controller.

For the other memory interfaces, the M3IF only arbitrates and forwards the masters requests received through the Master Port Gasket (MPG/MPG64) interface (and M3IF arbitration) to the respective memory controller.

When a master request a memory access, the access will immediately be taken by the M3IF if no other access is in progress. The M3IF will forward the access to the respective memory controller (slave), and depending on the respective memory controller state a command to the memory will be generated. If the access cannot be started due to a previous active access, the master request remains pending (“HREADY” held negated) until it will be executed by the memory controller. When the access execution is completed the HREADY will be asserted and a new request can be processed.

16.2 Features

The EMI includes these distinctive features:

- Multi Master Memory Interface (M3IF)
  - — Supports multiple requests from 8 masters through two different input ports interfaces:
    - – Master Port Gasket (MPG)—ARM11 AMBA AHB Lite bus protocol.
    - – Master Port Gasket (MPG64)—AMBA AHB access with 64 bits data bus width.
  - — Supports memory “snooping,” that is, monitor a region (from 2 Kbytes up to 16 Mbytes) in external memory for write accesses.
- Enables AHB accesses to four different memory controllers (that share some of their I/O pads, through the EMI AHB MUX and EMI I/O MUX)
- Enhanced SDRAM Controller (ESDCTL) or MDDR Controller (MDDRC)
  - — Up to 2 chip selects (due to PADS sharing all 2 chip selects are supported only in case that WEIM CS2 and CS3 are not is use).
  - — Support 32 bit SDR SDRAM (up to 2 Gbit @133 MHz)
  - — Support 32 bit MDDR SDRAM (up to 2 Gbit @ 266 MHz)

- NAND Flash Controller—(NANDFC)
  - 8/16 bit NANDFLASH (up to 2 GB address space)
  - 2K RAM Internal Buffer
- Personal Computer Memory Card International Association Controller—(PCMCIA)
  - Support PCMCIA Rev. 2.1
  - Compact Flash
  - PC Card
  - TrueID Mode
- Wireless External Interface Memory Controller—(WEIM)
  - Up to 6 chip selects (due to PADS sharing all 6 chip selects are supported only in case that both ESDCTL/MDDRC chip selects are not in use).
  - Support 16 bit SRAM memories
  - Support 16 bit PSRAM (up to 133 MHz) memories
  - Support 16 bit (NOR) FLASH memories

16.3 PCMCIA Host Adaptor

The PCMCIA (Personal Computer Memory Card International Association) interface provides a glueless interface to devices that comply with the PCMCIA association standard PCMCIA 2.1, which defines the usage of memory and I/O devices as insertable and exchangeable peripherals for personal computers or PDAs. Examples of these types of devices include Compact Flash and WLAN adapters. Figure 16-2 shows a simplified block diagram of the PCMCIA controller.

The PCMCIA host adapter module provides the all the necessary control logic for a single PCMCIA socket and only requires some additional external analog power switching logic and buffering for PC card operations.The PCMCIA host adapter module can support one PCMCIA socket.

The PCMCIA controller shares its pins with other modules in the EIM area of the chip. The modules that share pins with the PCMCIA are the EIM, SDRAMC, and NAND controllers.

16.3.1 Interrupt Generation

There are 14 sources of interrupts in the PCMCIA controller which generates an interrupt signal for each interrupt source. In addition, the PCMCIA generates a signal which is a locator of all the possible interrupts. It is up to the system’s integrator to decide which signal(s) to connect to the system’s interrupt controller module. The PCMCIA input pins register (PIPR) reports any change of inputs from the PCMCIA card to the host (BVD,CD,RDY,VS). The content of the PCMCIA controller status changed register (PSCR) are logically anded with the PCMCIA controller enable register (PER) to generate a PCMCIA controller interrupt. The interrupt level is user programmable and the PCMCIA controller can generate an additional interrupt for RDY/IREQ that can trigger upon a level (low or high) change or edge (fall or rise) of the input signal.

16.3.2 Card Extraction

When a PC Card is extracted the PCMCIA controller's registers are not reset. The registers settings remain the same as before the card's extraction. This allows the host software to quickly activate the card once the CIS indicates that it is the same card on reinsertion.

16.3.3 TrueIDE Support

The ATA standard specifies the AT attachment interface between host systems and storage devices. The PCMCIA controller can be dynamically configured to support a PCMCIA-compatible ATA disk interface (commonly known as IDE) instead of the standard PCMCIA card interface. Using the TrueIDE interface on the PCMCIA controller changes the function of some card socket signals to support the needs of the ATA disk interface.

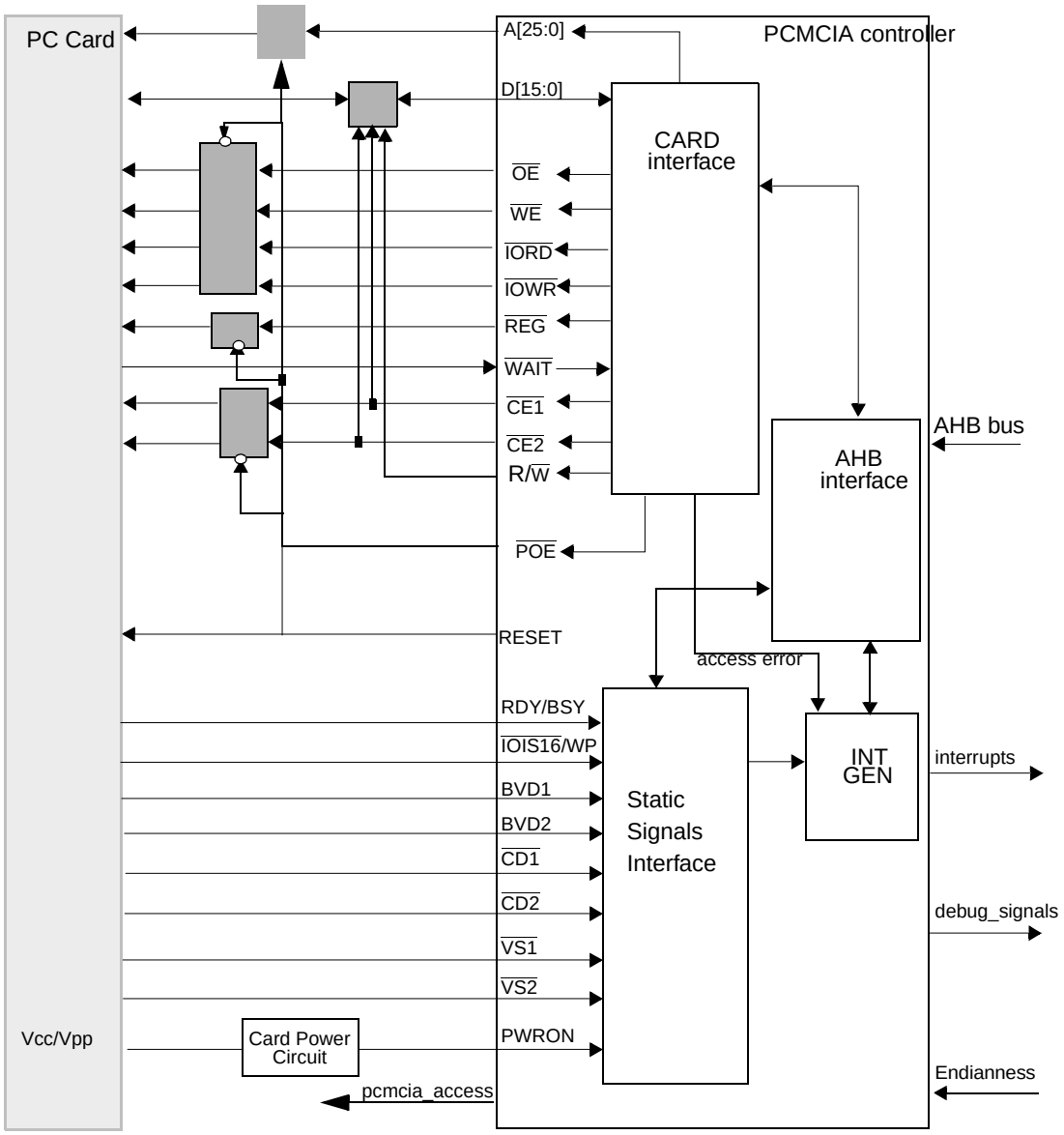


Figure 16-2. PCMCIA Host Adapter Simplified Block Diagram

16.4 NAND Flash Controller

The NAND Flash Controller (NANDFC or NFC) interfaces standard NAND Flash devices to the IC and hides the complexities of accessing a NAND Flash memory device. It provides a glueless interface to both 8-bits and 16-bits NAND Flash parts with page sizes of 512 Bytes or 2 Kilobytes and densities up to 2 Gbits. See Figure 16-3 for a simplified block diagram of the NAND Flash Controller.

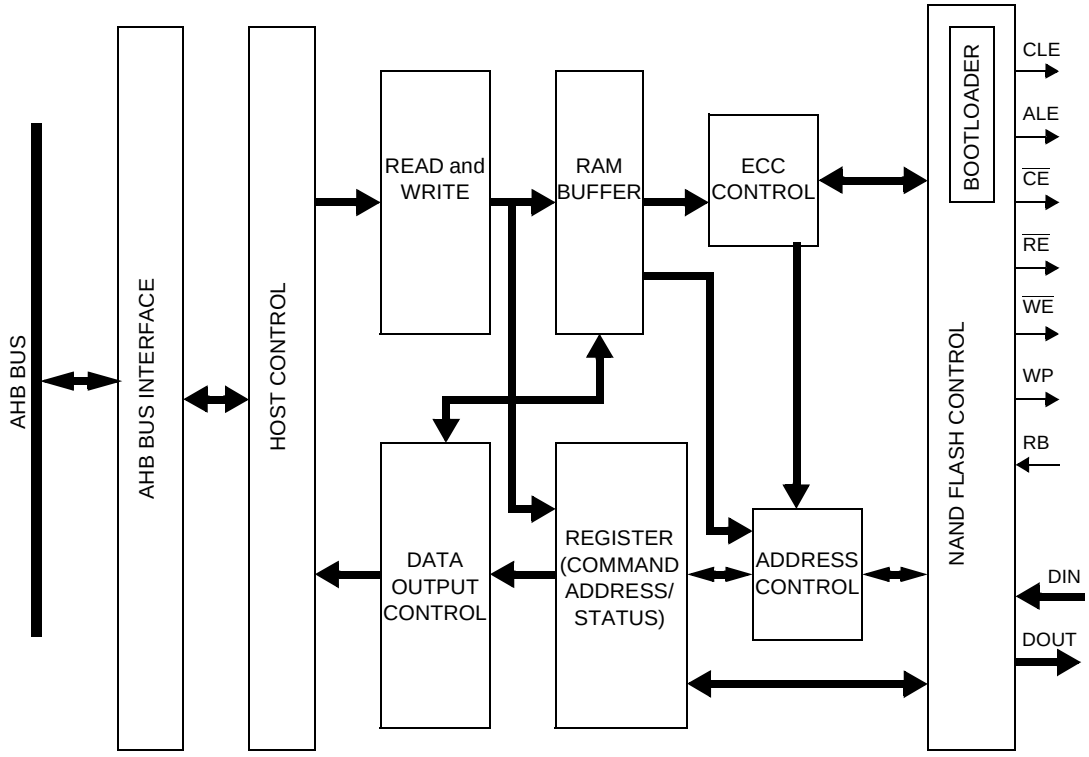


Figure 16-3. NAND Flash Controller Simplified Block Diagram

16.5 Operation

Communicating with a Flash memory device begins by the AHB Host initiating a read from the NAND Flash Controller (NANDFC). This is accomplished by configuring the NANDFC and then waiting for an interrupt from the Flash memory device to be generated. When the NANDFC receives the interrupt, it inputs a page from the Flash memory device, and upon completion, generates an interrupt to the AHB Host.

When the AHB Host receives the NANDFC interrupt, it reads the content from the internal RAM buffer of the NANDFC. To complete the operation the AHB Host checks the status of the operation by reading the NANDFC status registers.

Data that is exchanged with the Flash memory device is temporarily maintained in a 2 kilobyte RAM buffer. This buffer is used as the boot RAM during a cold reset (if the IC is configured for a boot to be carried out from the NAND Flash device). After the boot load completes, the RAM is available as buffer RAM for normal Flash memory operations.

16.5.1 Internal and External Communications

To ensure the greatest degree of flexibility the NANDFC provides an internal X16 bit and X32 bit interface to the AHB bus allowing, 16-bit or 32-bit bus transfers, and a pin selectable X8 or X16 interface to the external NAND Flash memory device.

All communications between the NANDFC and the ARM11 Platform is accomplished through the AHB Host. Configuration and control of the NANDFC by the host is done using 14 16-bit registers. The NANDFC generates all the control signals that control the NAND Flash: $\overline{CE}$ (Flash Chip Enable), and $\overline{RE}$ (Read Enable for read operations), $\overline{WE}$ (Flash Write Enable), CLE (Flash Command Latch Enable), ALE (Flash Address Latch Enable). It also monitors the R/nB (Flash Ready/Busy indication) signal to check if the NAND Flash memory device is currently in the middle of an operation.

Data integrity of the Flash memory data is monitored by the automatic generation of ECC code of data during NANDFCs data loading from NAND Flash memory devices.

16.5.2 Sharing of I/O Pins

The NANDFC provides necessary logic to share I/O pins with pins of another memory controller. The NANDFC state machine halts when a request to free the pins is asserted. The NAND Flash signals when it finishes the existing transfer allowing the other memory controller to be able to control them.

Since the NAND Flash memory accesses are typically long and relatively slow, priority is given to the other memory controller sharing the pins. The NANDFC must wait until the other memory controller is finished with it operation and the pins are free before it can continue with its accesses. One example for this pin muxing is sharing the 16 I/O pins of the NAND Flash Controller with the Data pins of the Wireless External Interface Module (WEIM) when interfacing to a PSRAM.

16.6 The Enhanced SDRAM Controller (ESDCTL)

The ESDCTL provides interface, configuration and control for many different types synchronous SDRAM and Low Power Mobile DDR (LPDDR) memories. Figure 16-4 is the Enhanced SDRAM Controller top-level diagram that shows the functional organization of the block.

The Enhanced SDRAM Controller consists of 9 major blocks, including the SDRAM command state machine controller, bank register (page and bank address comparators), Row/Column Address Multiplexer, configuration registers, refresh request counter, command sequencer, size logic (splitting access), data path (data aligner/multiplexer), LPDDR interface and the Power Down timer.

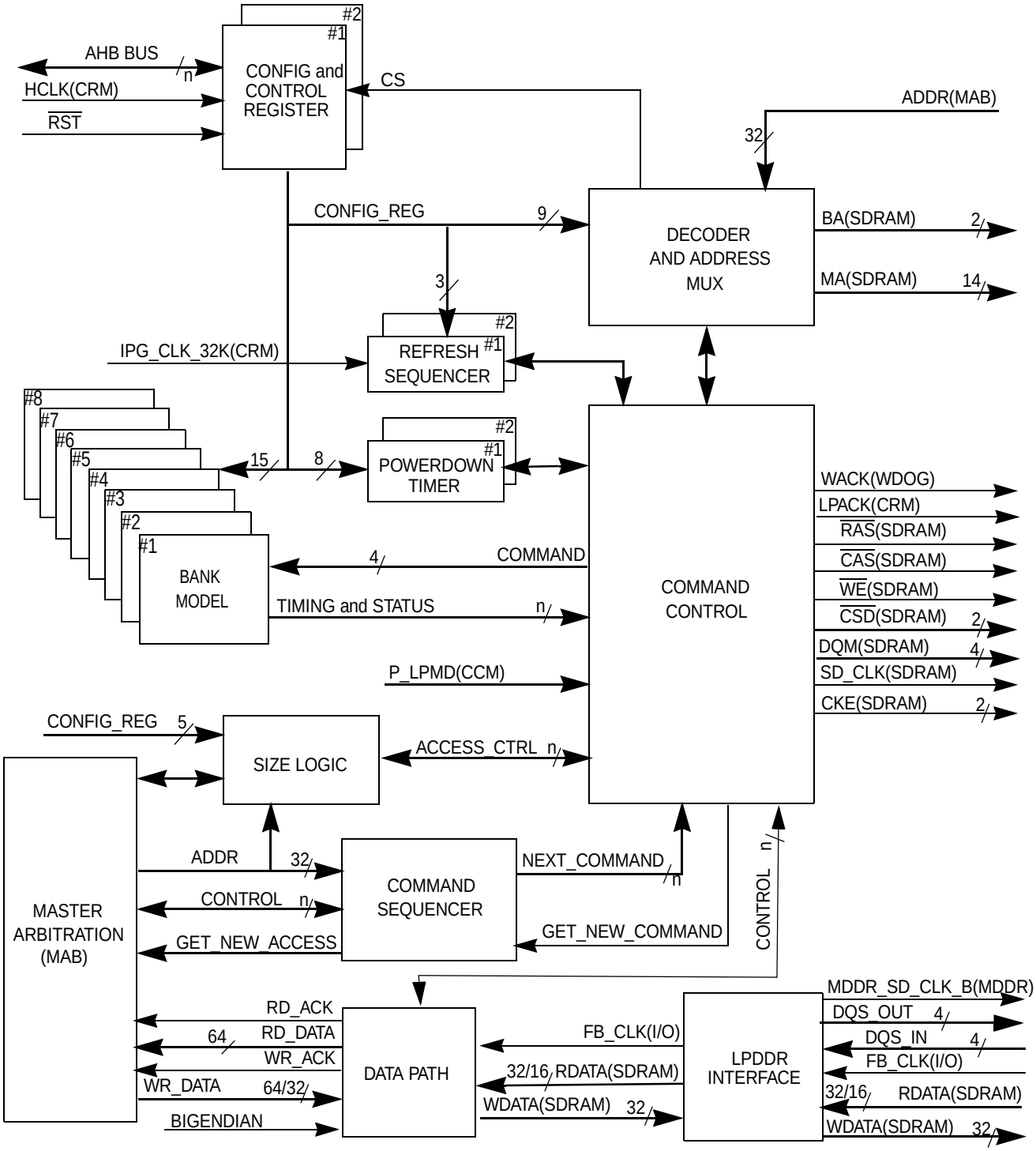


Figure 16-4. Enhanced SDR/LPDDR SDRAM Controller Block Diagram

16.7 EMI AHB MUX

The EMI AHB MUX controls the traffic on the AHB bus (address and controls) between the memory controllers and the IC (via the M3IF), and vice versa, that is, from the IC to the memory controllers. The EMI uses several muxes/glue logic to control the traffic on the AHB bus.

Only several AHB signals/busses are routed through the EMI AHB MUX toward the memory controllers. Most of the AHB busses (data, address and controls) are directly routed from the M3IF to the memory controllers.

16.7.1 Overview of EMI AHB MUX Operation

Figure 16-5 illustrates the EMI AHB MUX block diagram. The interface is ARM's 11 AMBA-AHB Lite compliant (does not support RETRY and SPLIT transfers). All AHB signals that are not shown in Figure 16-5 are directly routed between the M3IF and the relevant memory controllers. For the entire list of AHB signals, refer to Table 16-5 and to the relevant memory controller specification document.

The EMI AHB MUX generates the HSEL signals for all memory controllers, excluding the ESDCTL (generated within the M3IF due to latency hiding logic).

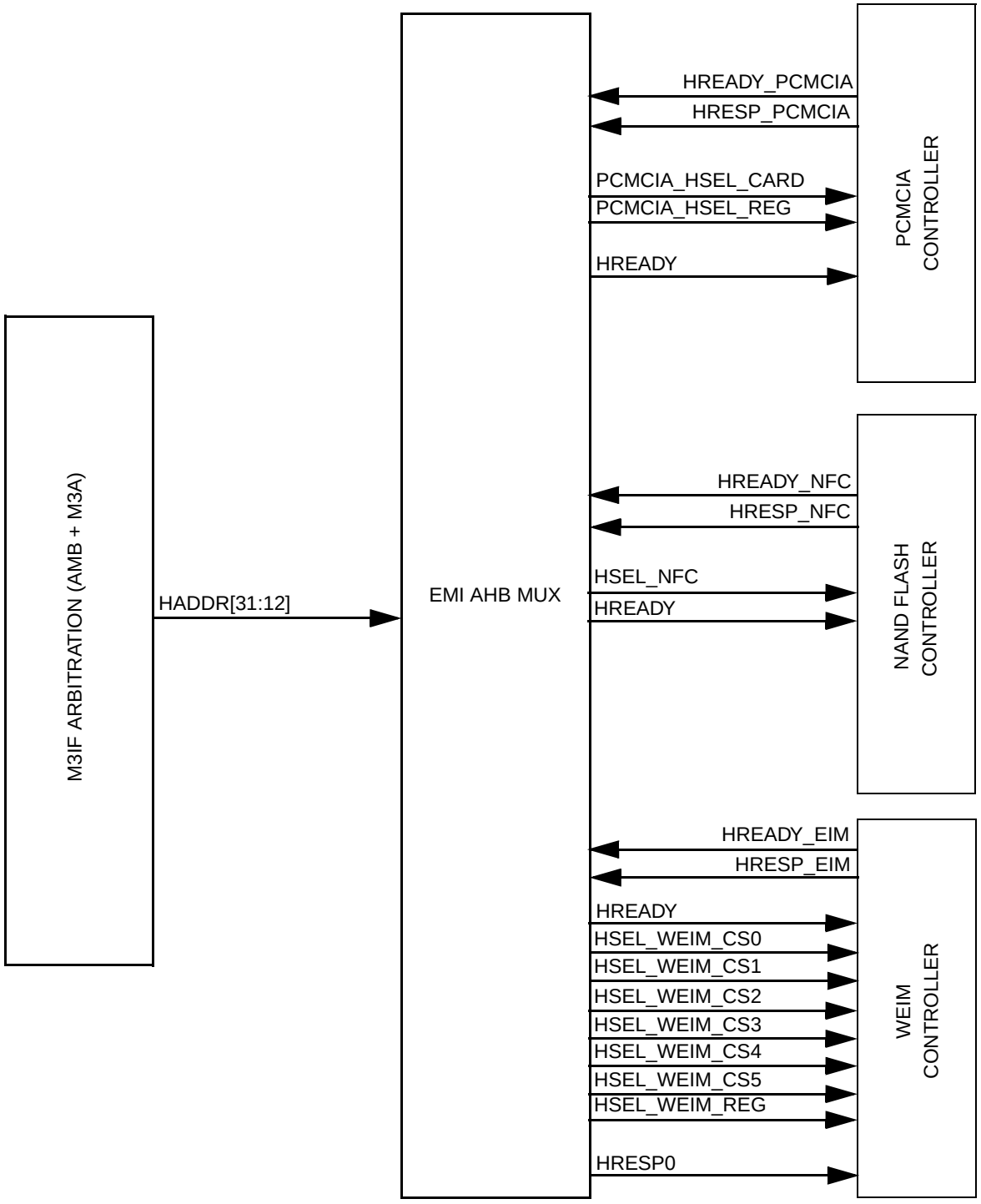


Figure 16-5. EMI AHB MUX Interface Diagram

16.8 EMI I/O MUX

The EMI I/O MUX controls the traffic (data, address and controls) between the memory controllers and the external devices (via the IC I/O MUX/PADS), and vice versa, for example, from the external devices to the memory controllers. The EMI uses several muxes/glue logic to control the traffic. Refer to Figure 16-5 and Figure 16-6 for a top level diagram of the EMIAHB and the EMI I/O MUX.

Only shared IC PADS signals/busses are routed through the EMI I/O MUX toward the external devices. Signals (mainly controls) that have dedicated PADS are directly routed from the memory controllers to the external devices.

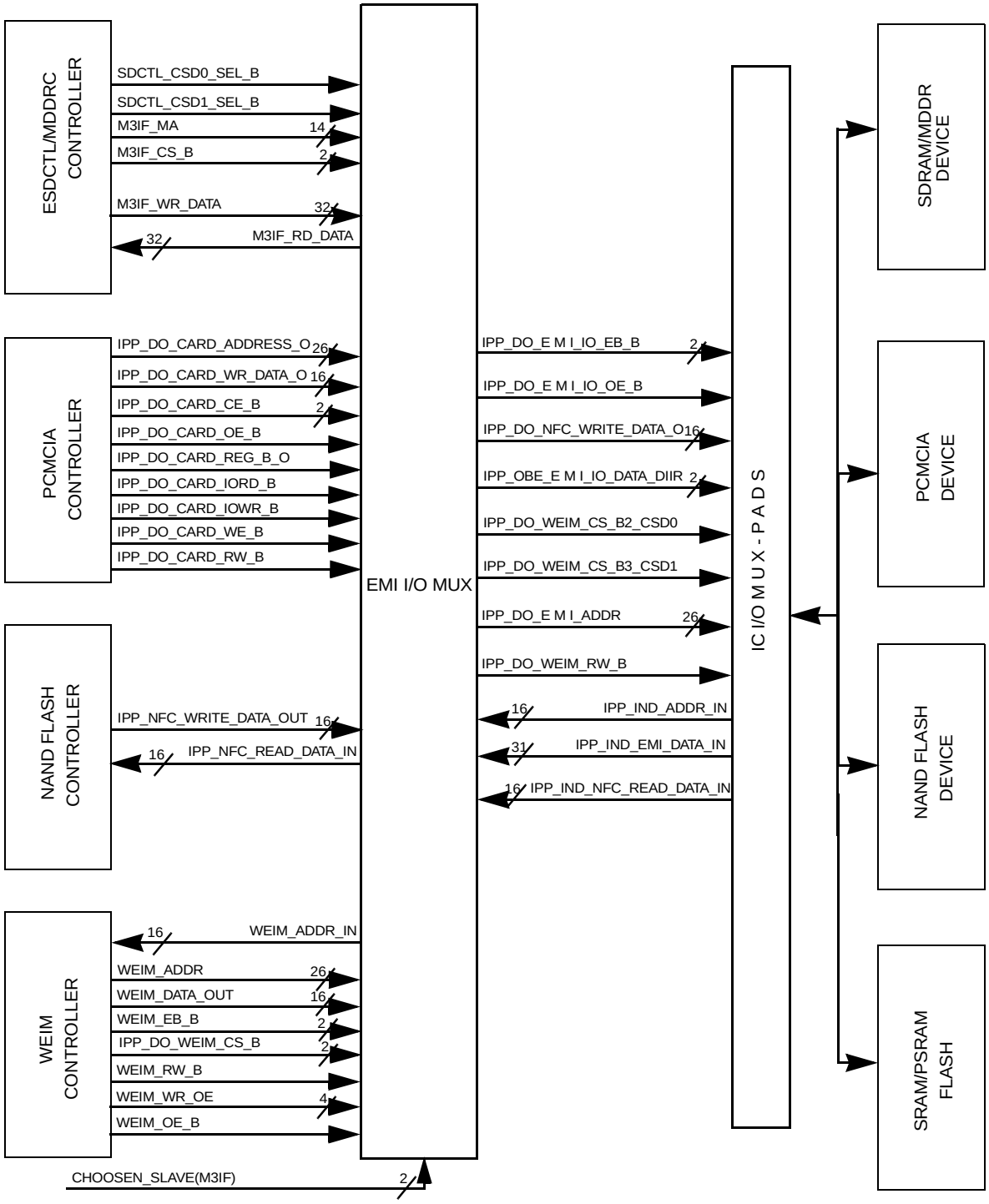


Figure 16-6. EMI I/O MUX Interface Diagram

16.8.1 Overview of EMI I/O MUX Operation

Figure 16-6 shows a block diagram of the EMI I/O MUX interface signals.

NOTE

The signals that are not shown in Figure 16-6 are directly routed between the memory controllers and the respective external device. For the entire list of signals, refer to Table 16-5 and to the respective memory controller sections.

The select for the muxes is the CHOOSEN_SLAVE[1:0] bus driven by the M3IF. The CHOOSEN_SLAVE encoding is listed in Table 16-1. Table 16-2 summarizes the EMI outputs to IC pads (dedicated and shared among all memory controllers).

Table 16-1. CHOOSEN_SLAVE Encoding

CHOOSEN_SLAVE Value	Selected Memory Controller
00	ESDCTL/MDDRC
01	WEIM
10	PCMCIA
11	NANDFC

Table 16-2. EMI Output Summary

MEMORY CONTROLLERS OUTPUTS					EMI OUTPUT	IC PIN NAME
SDRAMC	MDDRC	PCMCIA	WEIM	NANDFC		
WEIM/ESDCTL/PCMCIA ADDRESS MUXING/WEIM MUXED MODE DATA[15:0]						
M3IF_MA[0]		IPP_DO_CARD_ADDRESS_O[0]	WEIM_ADDR_DATA_OUT[0]	—	IPP_DO_EMI _ADDR [0]	A0
—		—	WEIM_ADDR_DATA_IN[0]		IPP_IND_ADDR_IN [0]	
M3IF_MA[1]		IPP_DO_CARD_ADDRESS_O[1]	WEIM_ADDR_DATA_OUT[1]	—	IPP_DO_EMI _ADDR [1]	A1
—		—	WEIM_ADDR_DATA_IN[1]		IPP_IND_ADDR_IN [1]	
M3IF_MA[2]		IPP_DO_CARD_ADDRESS_O[2]	WEIM_ADDR_DATA_OUT[2]	—	IPP_DO_EMI _ADDR [2]	A2
—		—	WEIM_ADDR_DATA_IN[2]		IPP_IND_ADDR_IN [2]	
M3IF_MA[3]		IPP_DO_CARD_ADDRESS_O[3]	WEIM_ADDR_DATA_OUT[3]	—	IPP_DO_EMI _ADDR [3]	A3
—		—	WEIM_ADDR_DATA_IN[3]		IPP_IND_ADDR_IN [3]	
M3IF_MA[4]		IPP_DO_CARD_ADDRESS_O[4]	WEIM_ADDR_DATA_OUT[4]	—	IPP_DO_EMI _ADDR [4]	A4
—		—	WEIM_ADDR_DATA_IN[4]		IPP_IND_ADDR_IN [4]	

Table 16-2. EMI Output Summary (continued)

MEMORY CONTROLLERS OUTPUTS					EMI OUTPUT	IC PIN NAME
SDRAMC	MDDRC	PCMCIA	WEIM	NANDFC		
M3IF_MA[5]		IPP_DO_CARD_ADDRESS_O[5]	WEIM_ADDR_DATA_OUT[5]	—	IPP_DO_EMI_ADDR [5]	A5
—		—	WEIM_ADDR_DATA_IN[5]		IPP_IND_ADDR_IN [5]	
M3IF_MA[6]		IPP_DO_CARD_ADDRESS_O[6]	WEIM_ADDR_DATA_OUT[6]	—	IPP_DO_EMI_ADDR [6]	A6
—		—	WEIM_ADDR_DATA_IN[6]		IPP_IND_ADDR_IN [6]	
M3IF_MA[7]		IPP_DO_CARD_ADDRESS_O[7]	WEIM_ADDR_DATA_OUT[7]	—	IPP_DO_EMI_ADDR [7]	A7
—		—	WEIM_ADDR_DATA_IN[7]		IPP_IND_ADDR_IN [7]	
M3IF_MA[8]		IPP_DO_CARD_ADDRESS_O[8]	WEIM_ADDR_DATA_OUT[8]	—	IPP_DO_EMI_ADDR [8]	A8
—		—	WEIM_ADDR_DATA_IN[8]		IPP_IND_ADDR_IN [8]	
M3IF_MA[9]		IPP_DO_CARD_ADDRESS_O[9]	WEIM_ADDR_DATA_OUT[9]	—	IPP_DO_EMI_ADDR [9]	A9
—		—	WEIM_ADDR_DATA_IN[9]		IPP_IND_ADDR_IN [9]	
—		IPP_DO_CARD_ADDRESS_O[10]	WEIM_ADDR_DATA_OUT[10]	—	IPP_DO_EMI_ADDR [10]	A10
—		—	WEIM_ADDR_DATA_IN[10]		IPP_IND_ADDR_IN [10]	
M3IF_MA[11]		IPP_DO_CARD_ADDRESS_O[11]	WEIM_ADDR_DATA_OUT[11]	—	IPP_DO_EMI_ADDR [11]	A11
—		—	WEIM_ADDR_DATA_IN[11]		IPP_IND_ADDR_IN [11]	
M3IF_MA[12]		IPP_DO_CARD_ADDRESS_O[12]	WEIM_ADDR_DATA_OUT[12]	—	IPP_DO_EMI_ADDR [12]	A12
—		—	WEIM_ADDR_DATA_IN[12]		IPP_IND_ADDR_IN [12]	
M3IF_MA[13]		IPP_DO_CARD_ADDRESS_O[13]	WEIM_ADDR_DATA_OUT[13]	—	IPP_DO_EMI_ADDR [13]	A13
—		—	WEIM_ADDR_DATA_IN[13]		IPP_IND_ADDR_IN [13]	

Table 16-2. EMI Output Summary (continued)

MEMORY CONTROLLERS OUTPUTS					EMI OUTPUT	IC PIN NAME
SDRAMC	MDDRC	PCMCIA	WEIM	NANDFC		
—		IPP_DO_CARD_ADDRESS_O[14]	WEIM_ADDR_DATA_OUT[14]	—	IPP_DO_EMI _ADDR [14]	A14
			WEIM_ADDR_DATA_IN[14]		IPP_IND_ADDR_IN [14]	
—		IPP_DO_CARD_ADDRESS_O[15]	WEIM_ADDR_DATA_OUT[15]	—	IPP_DO_EMI _ADDR [15]	A15
			WEIM_ADDR_DATA_IN[15]		IPP_IND_ADDR_IN [15]	
—		IPP_DO_CARD_ADDRESS_O[16]	WEIM_ADDR_OUT[16]	—	IPP_DO_EMI _ADDR [16]	A16
—		IPP_DO_CARD_ADDRESS_O[17]	WEIM_ADDR_OUT[17]	—	IPP_DO_EMI _ADDR [17]	A17
—		IPP_DO_CARD_ADDRESS_O[18]	WEIM_ADDR_OUT[18]	—	IPP_DO_EMI _ADDR [18]	A18
—		IPP_DO_CARD_ADDRESS_O[19]	WEIM_ADDR_OUT[19]	—	IPP_DO_EMI _ADDR [19]	A19
—		IPP_DO_CARD_ADDRESS_O[20]	WEIM_ADDR_OUT[20]	—	IPP_DO_EMI _ADDR [20]	A20
—		IPP_DO_CARD_ADDRESS_O[21]	WEIM_ADDR_OUT[21]	—	IPP_DO_EMI _ADDR [21]	A21
—		IPP_DO_CARD_ADDRESS_O[22]	WEIM_ADDR_OUT[22]	—	IPP_DO_EMI _ADDR [22]	A22
—		IPP_DO_CARD_ADDRESS_O[23]	WEIM_ADDR_OUT[23]	—	IPP_DO_EMI _ADDR [23]	A23
—		IPP_DO_CARD_ADDRESS_O[24]	WEIM_ADDR_OUT[24]	—	IPP_DO_EMI _ADDR [24]	A24
—		IPP_DO_CARD_ADDRESS_O[25]	WEIM_ADDR_OUT[25]	—	IPP_DO_EMI _ADDR [25]	A25
ESDCTL ADDRESS BIT M3IF_MA[10] HAS A DEDICATED PAD MA10 (REQUIRED DUE TO PRECHARGE ALL DURING AUTO REFRESH COMMANDS) ESDCTL BANK ADDRESS BITS HAVE DEDICATED PADS DUE TO PRECHARGE BANK DURING PRECHARGE TIMER TIMEOUT WEIM CRE SIGNAL IS DRIVEN ON A23 IN MUXED MODE OPERATION.						
M3IF_MA[10]		—	—	—	IPP_DO_EMI _MA10	MA10
M3IF_BA[0]		IPP_DO_CARD_CE_B[2]	—	—	IPP_DO_SDBA[1:0]	SDBA0
M3IF_BA[1]		IPP_DO_CARD_CE_B[1]	—	—		SDBA1

Table 16-2. EMI Output Summary (continued)

MEMORY CONTROLLERS OUTPUTS					EMI OUTPUT	IC PIN NAME
SDRAMC	MDDRC	PCMCIA	WEIM	NANDFC		
SINCE THE SDBA PADS ARE SHARED BETWEEN THE SDR?DDR SDRAM BANK ADDRESS AND PCMCIA CE‘, THE ESDCTL PRECHARGE TIMER CAN”T BE USED. DURING HE PRECHARGE TIMER, AFTER THE SELECTED INACTIVITY PERIOD OF TIME EXPIRES, THE ESDCTL ISSUE A PRECHARGE COMMAND TO A SPECIFIC BANK DURING “OFF LINE” PERIOD, MEANS THAT THE PRECHARGE COMMAND CAN BE ISSUED DURING THE TIME THAT THE EMI BUS IS NOT IN ESDCTL POSSES.						
SDRAM/MDDR DEDICATED DATA PADS						
M3IF_WR_DATA[0]		—	—	—	IPP_DO_EMII_DATA [0]	SD0
M3IF_RD_DATA[0]					IPP_IND_EMII_DATA_IN [0]	
M3IF_WR_DATA[1]		—	—	—	IPP_DO_EMII_DATA [1]	SD1
M3IF_RD_DATA[1]					IPP_IND_EMII_DATA_IN [1]	
M3IF_WR_DATA[2]		—	—	—	IPP_DO_EMII_DATA [2]	SD2
M3IF_RD_DATA[2]					IPP_IND_EMII_DATA_IN [2]	
M3IF_WR_DATA[3]		—	—	—	IPP_DO_EMII_DATA [3]	SD3
M3IF_RD_DATA[3]					IPP_IND_EMII_DATA_IN [3]	
M3IF_WR_DATA[4]		—	—	—	IPP_DO_EMII_DATA [4]	SD4
M3IF_RD_DATA[4]					IPP_IND_EMII_DATA_IN [4]	
M3IF_WR_DATA[5]		—	—	—	IPP_DO_EMII_DATA [5]	SD5
M3IF_RD_DATA[5]					IPP_IND_EMII_DATA_IN [5]	
M3IF_WR_DATA[6]		—	—	—	IPP_DO_EMII_DATA [6]	SD6
M3IF_RD_DATA[6]					IPP_IND_EMII_DATA_IN [6]	

Table 16-2. EMI Output Summary (continued)

MEMORY CONTROLLERS OUTPUTS					EMI OUTPUT	IC PIN NAME
SDRAMC	MDDRC	PCMCIA	WEIM	NANDFC		
M3IF_WR_DATA[7]		—	—	—	IPP_DO_EMII_DATA [7]	SD7
M3IF_RD_DATA[7]					IPP_IND_EMII_DATA_IN [7]	
M3IF_WR_DATA[8]		—	—	—	IPP_DO_EMII_DATA [8]	SD8
M3IF_RD_DATA[8]					IPP_IND_EMII_DATA_IN [8]	
M3IF_WR_DATA[9]		—	—	—	IPP_DO_EMII_DATA [9]	SD9
M3IF_RD_DATA[9]					IPP_IND_EMII_DATA_IN [9]	
M3IF_WR_DATA[10]		—	—	—	IPP_DO_EMII_DATA [10]	SD10
M3IF_RD_DATA[10]					IPP_IND_EMII_DATA_IN [10]	
M3IF_WR_DATA[11]		—	—	—	IPP_DO_EMII_DATA [11]	SD11
M3IF_RD_DATA[11]					IPP_IND_EMII_DATA_IN [11]	
M3IF_WR_DATA[12]		—	—	—	IPP_DO_EMII_DATA [12]	SD12
M3IF_RD_DATA[12]					IPP_IND_EMII_DATA_IN [12]	
M3IF_WR_DATA[13]		—	—	—	IPP_DO_EMII_DATA [13]	SD13
M3IF_RD_DATA[13]					IPP_IND_EMII_DATA_IN [13]	
M3IF_WR_DATA[14]		—	—	—	IPP_DO_EMII_DATA [14]	SD14
M3IF_RD_DATA[14]					IPP_IND_EMII_DATA_IN [14]	

Table 16-2. EMI Output Summary (continued)

MEMORY CONTROLLERS OUTPUTS					EMI OUTPUT	IC PIN NAME
SDRAMC	MDDRC	PCMCIA	WEIM	NANDFC		
M3IF_WR_DATA[15]		—	—	—	IPP_DO_EMII_DATA [15]	SD15
M3IF_RD_DATA[15]					IPP_IND_EMII_DATA_IN [15]	
M3IF_WR_DATA[16]		—	—	—	IPP_DO_EMII_DATA [16]	SD16
M3IF_RD_DATA[16]					IPP_IND_EMII_DATA_IN [16]	
M3IF_WR_DATA[17]		—	—	—	IPP_DO_EMII_DATA [17]	SD17
M3IF_RD_DATA[17]					IPP_IND_EMII_DATA_IN [17]	
M3IF_WR_DATA[18]		—	—	—	IPP_DO_EMII_DATA [18]	SD18
M3IF_RD_DATA[18]					IPP_IND_EMII_DATA_IN [18]	
M3IF_WR_DATA[19]		—	—	—	IPP_DO_EMII_DATA [19]	SD19
M3IF_RD_DATA[19]					IPP_IND_EMII_DATA_IN [19]	
M3IF_WR_DATA[20]		—	—	—	IPP_DO_EMII_DATA [20]	SD20
M3IF_RD_DATA[20]					IPP_IND_EMII_DATA_IN [20]	
M3IF_WR_DATA[21]		—	—	—	IPP_DO_EMII_DATA [21]	SD21
M3IF_RD_DATA[21]					IPP_IND_EMII_DATA_IN [21]	
M3IF_WR_DATA[22]		—	—	—	IPP_DO_EMII_DATA [22]	SD22
M3IF_RD_DATA[22]					IPP_IND_EMII_DATA_IN [22]	

Table 16-2. EMI Output Summary (continued)

MEMORY CONTROLLERS OUTPUTS					EMI OUTPUT	IC PIN NAME
SDRAMC	MDDRC	PCMCIA	WEIM	NANDFC		
M3IF_WR_DATA[23]		—	—	—	IPP_DO_EMII_DATA [23]	SD23
M3IF_RD_DATA[23]					IPP_IND_EMII_DATA_IN [23]	
M3IF_WR_DATA[24]		—	—	—	IPP_DO_EMII_DATA [24]	SD24
M3IF_RD_DATA[24]					IPP_IND_EMII_DATA_IN [24]	
M3IF_WR_DATA[25]		—	—	—	IPP_DO_EMII_DATA [25]	SD25
M3IF_RD_DATA[25]					IPP_IND_EMII_DATA_IN [25]	
M3IF_WR_DATA[26]		—	—	—	IPP_DO_EMII_DATA [26]	SD26
M3IF_RD_DATA[26]					IPP_IND_EMII_DATA_IN [26]	
M3IF_WR_DATA[27]		—	—	—	IPP_DO_EMII_DATA [27]	SD27
M3IF_RD_DATA[27]					IPP_IND_EMII_DATA_IN [27]	
M3IF_WR_DATA[28]		—	—	—	IPP_DO_EMII_DATA [28]	SD28
M3IF_RD_DATA[28]					IPP_IND_EMII_DATA_IN [28]	
M3IF_WR_DATA[29]		—	—	—	IPP_DO_EMII_DATA [29]	SD29
M3IF_RD_DATA[29]					IPP_IND_EMII_DATA_IN [29]	
M3IF_WR_DATA[30]		—	—	—	IPP_DO_EMII_DATA [30]	SD30
M3IF_RD_DATA[30]					IPP_IND_EMII_DATA_IN [30]	

Table 16-2. EMI Output Summary (continued)

MEMORY CONTROLLERS OUTPUTS					EMI OUTPUT	IC PIN NAME
SDRAMC	MDDRC	PCMCIA	WEIM	NANDFC		
M3IF_WR_DATA[31]		—	—	—	IPP_DO_EMII_DATA [31]	SD31
M3IF_RD_DATA[31]					IPP_IND_EMII_DATA_IN [31]	
WEIM/NANDFC/PCMCIA DATA MUXING						
—		IPP_DO_CARD_WR_DATA_O[0]	WEIM_DATA_OUT[0]	IPP_NFC_WRITE_DATA_OUT[0]	IPP_DO_NFC_WRITE_DATA_OUT[0]	D0
		IPP_IND_CARD_RD_DATA_I[0]	WEIM_DATA_IN[0]	IPP_NFC_READ_DATA_IN[0]	IPP_IND_NFC_READ_DATA_IN [0]	
—		IPP_DO_CARD_WR_DATA_O[1]	WEIM_DATA_OUT[1]	IPP_NFC_WRITE_DATA_OUT[1]	IPP_DO_NFC_WRITE_DATA_OUT[1]	D1
		IPP_IND_CARD_RD_DATA_I[1]	WEIM_DATA_IN[1]	IPP_NFC_READ_DATA_IN[1]	IPP_IND_NFC_READ_DATA_IN [1]	
—		IPP_DO_CARD_WR_DATA_O[2]	WEIM_DATA_OUT[2]	IPP_NFC_WRITE_DATA_OUT[2]	IPP_DO_NFC_WRITE_DATA_OUT[2]	D2
		IPP_IND_CARD_RD_DATA_I[2]	WEIM_DATA_IN[2]	IPP_NFC_READ_DATA_IN[2]	IPP_IND_NFC_READ_DATA_IN [2]	
—		IPP_DO_CARD_WR_DATA_O[3]	WEIM_DATA_OUT[3]	IPP_NFC_WRITE_DATA_OUT[3]	IPP_DO_NFC_WRITE_DATA_OUT[3]	D3
		IPP_IND_CARD_RD_DATA_I[3]	WEIM_DATA_IN[3]	IPP_NFC_READ_DATA_IN[3]	IPP_IND_NFC_READ_DATA_IN [3]	
—		IPP_DO_CARD_WR_DATA_O[4]	WEIM_DATA_OUT[4]	IPP_NFC_WRITE_DATA_OUT[4]	IPP_DO_NFC_WRITE_DATA_OUT[4]	D4
		IPP_IND_CARD_RD_DATA_I[4]	WEIM_DATA_IN[4]	IPP_NFC_READ_DATA_IN[4]	IPP_IND_NFC_READ_DATA_IN [4]	

Table 16-2. EMI Output Summary (continued)

MEMORY CONTROLLERS OUTPUTS					EMI OUTPUT	IC PIN NAME
SDRAMC	MDDRC	PCMCIA	WEIM	NANDFC		
—		IPP_DO_CARD_WR_DATA_O[5]	WEIM_DATA_OUT[5]	IPP_NFC_WRITE_DATA_OUT[5]	IPP_DO_NFC_WRITE_DATA_OUT[5]	D5
		IPP_IND_CARD_RD_DATA_I[5]	WEIM_DATA_IN[5]	IPP_NFC_READ_DATA_IN[5]	IPP_IND_NFC_READ_DATA_IN [5]	
—		IPP_DO_CARD_WR_DATA_O[6]	WEIM_DATA_OUT[6]	IPP_NFC_WRITE_DATA_OUT[6]	IPP_DO_NFC_WRITE_DATA_OUT[6]	D6
		IPP_IND_CARD_RD_DATA_I[6]	WEIM_DATA_IN[6]	IPP_NFC_READ_DATA_IN[6]	IPP_IND_NFC_READ_DATA_IN [6]	
—		IPP_DO_CARD_WR_DATA_O[7]	WEIM_DATA_OUT[7]	IPP_NFC_WRITE_DATA_OUT[7]	IPP_DO_NFC_WRITE_DATA_OUT[7]	D7
		IPP_IND_CARD_RD_DATA_I[7]	WEIM_DATA_IN[7]	IPP_NFC_READ_DATA_IN[7]	IPP_IND_NFC_READ_DATA_IN [7]	
—		IPP_DO_CARD_WR_DATA_O[8]	WEIM_DATA_OUT[8]	IPP_NFC_WRITE_DATA_OUT[8]	IPP_DO_NFC_WRITE_DATA_OUT[8]	D8
		IPP_IND_CARD_RD_DATA_I[8]	WEIM_DATA_IN[8]	IPP_NFC_READ_DATA_IN[8]	IPP_IND_NFC_READ_DATA_IN [8]	
—		IPP_DO_CARD_WR_DATA_O[9]	WEIM_DATA_OUT[9]	IPP_NFC_WRITE_DATA_OUT[9]	IPP_DO_NFC_WRITE_DATA_OUT[9]	D9
		IPP_IND_CARD_RD_DATA_I[9]	WEIM_DATA_IN[9]	IPP_NFC_READ_DATA_IN[9]	IPP_IND_NFC_READ_DATA_IN [9]	
—		IPP_DO_CARD_WR_DATA_O[10]	WEIM_DATA_OUT[10]	IPP_NFC_WRITE_DATA_OUT[10]	IPP_DO_NFC_WRITE_DATA_OUT[10]	D10
		IPP_IND_CARD_RD_DATA_I[10]	WEIM_DATA_IN[10]	IPP_NFC_READ_DATA_IN[10]	IPP_IND_NFC_READ_DATA_IN [10]	

Table 16-2. EMI Output Summary (continued)

MEMORY CONTROLLERS OUTPUTS					EMI OUTPUT	IC PIN NAME
SDRAMC	MDDRC	PCMCIA	WEIM	NANDFC		
—		IPP_DO_CARD_WR_DATA_O[11]	WEIM_DATA_OUT[11]	IPP_NFC_WRITE_DATA_OUT[11]	IPP_DO_NFC_WRITE_DATA_OUT[11]	D11
		IPP_IND_CARD_RD_DATA_I[11]	WEIM_DATA_IN[11]	IPP_NFC_READ_DATA_IN[11]	IPP_IND_NFC_READ_DATA_IN [11]	
—		IPP_DO_CARD_WR_DATA_O[12]	WEIM_DATA_OUT[12]	IPP_NFC_WRITE_DATA_OUT[12]	IPP_DO_NFC_WRITE_DATA_OUT[12]	D12
		IPP_IND_CARD_RD_DATA_I[12]	WEIM_DATA_IN[12]	IPP_NFC_READ_DATA_IN[12]	IPP_IND_NFC_READ_DATA_IN [12]	
—		IPP_DO_CARD_WR_DATA_O[13]	WEIM_DATA_OUT[13]	IPP_NFC_WRITE_DATA_OUT[13]	IPP_DO_NFC_WRITE_DATA_OUT[13]	D13
		IPP_IND_CARD_RD_DATA_I[13]	WEIM_DATA_IN[13]	IPP_NFC_READ_DATA_IN[13]	IPP_IND_NFC_READ_DATA_IN [13]	
—		IPP_DO_CARD_WR_DATA_O[14]	WEIM_DATA_OUT[14]	IPP_NFC_WRITE_DATA_OUT[14]	IPP_DO_NFC_WRITE_DATA_OUT[14]	D14
		IPP_IND_CARD_RD_DATA_I[14]	WEIM_DATA_IN[14]	IPP_NFC_READ_DATA_IN[14]	IPP_IND_NFC_READ_DATA_IN [14]	
—		IPP_DO_CARD_WR_DATA_O[15]	WEIM_DATA_OUT[15]	IPP_NFC_WRITE_DATA_OUT[15]	IPP_DO_NFC_WRITE_DATA_OUT[15]	D15
		IPP_IND_CARD_RD_DATA_I[15]	WEIM_DATA_IN[15]	IPP_NFC_READ_DATA_IN[15]	IPP_IND_NFC_READ_DATA_IN [15]	
MASK (BYTE ENABLE) MUXING						
—	—	IPP_DO_CARD_REG_B_O	WEIM_EB_B[0]	—	IPP_DO_EMI_IO_EB_B[1:0]	EB0
—	—	IPP_DO_CARD_IORD_B	WEIM_EB_B[1]	—		EB1
EB_B[2] and EB_B[3] WILL BE DRIVEN ON A24 and A25 RESPECTIVELLY DURING MUXED MODE (IN ORDER TO USE 32 BIT MEMORY DEVICE.						

Table 16-2. EMI Output Summary (continued)

MEMORY CONTROLLERS OUTPUTS					EMI OUTPUT	IC PIN NAME
SDRAMC	MDDRC	PCMCIA	WEIM	NANDFC		
SDRAM/MDDR MASK (BYTE ENABLE)						
M3IF_DQM[0]		—	—	—	IPP_DO_DQM [3:0]	DQM0
M3IF_DQM[1]		—	—	—		DQM1
M3IF_DQM[2]		—	—	—		DQM2
M3IF_DQM[3]		—	—	—		DQM3
OUTPUT ENABLE MUXING						
—		IPP_DO_CARD_IOWR_B	WEIM_WR_OE	—	IPP_DO_EMI_OE_B	OE
CHIP SELECT MUXING						
—	—	—	IPP_DO_WEIM_CS_B[0]	—	IPP_DO_WEIM_ CS_B0	CS0
—	—	—	IPP_DO_WEIM_CS_B[1]	—	IPP_DO_WEIM_ CS_B1	CS1
M3IF_CS_B[0]		—	IPP_DO_WEIM_CS_B[2]	—	IPP_DO_WEIM_ CS_B2_CSD0	CS2
M3IF_CS_B[1]		—	IPP_DO_WEIM_CS_B[3]	—	IPP_DO_WEIM_ CS_B3_CSD1	CS3
—	—	—	IPP_DO_WEIM_CS_B[4]	—	IPP_DO_WEIM_ CS_B4	CS4
—	—	—	IPP_DO_WEIM_CS_B[5]	—	IPP_DO_WEIM_ CS_B5	CS5
THE “CHIP SELECT” SELECTORS ARE THE SYSTEM CONTROL REGISTER BITS SDCTL_CSD0_SEL AND SDCTL_CSD1_SEL RESPECTiVELY. THE DEFAULT SELECT FOR BOTH CHIP SELECTS IS FOR THE ESDCTL/MDDRC.						
WRITE ENABLE MUXING						
—	—	IPP_DO_CARD_WE_B	WEIM_RW_B	—	IPP_DO_WEIM_ RW_B	RW
SDRAM/MDDR COMMAND DEDICATED PADS						
RAS_B		—	—	—	IPP_DO_M3IF_RAS_B	RAS
CAS_B		—	—	—	IPP_DO_M3IF_CAS_B	CAS
WE_B		—	—	—	IPP_DO_SDRC_SDWE	SDWE

Table 16-2. EMI Output Summary (continued)

MEMORY CONTROLLERS OUTPUTS					EMI OUTPUT	IC PIN NAME
SDRAMC	MDDRC	PCMCIA	WEIM	NANDFC		
CKE[1]		—	—	—	IPP_DO_SDRC_SDCKE[1]	SDCKE [1]
CKE[0]		—	—	—	IPP_DO_SDRC_SDCKE[0]	SDCKE [0]
SDCLK_OUT		—	—	—	IPP_DO_SDRC_SDCLK	SDCLK
—	MDDR_SDCLK_B	—	—	—	IPP_DO_MDDR_SDCLK_B	SDCLK_B
—	DQS_OUT [3]	—	—	—	IPP_DO_DQS[3]	DQS[3]
	DQS_IN [3]				IPP_DIN_DQS[3]	
—	DQS_OUT [2]	—	—	—	IPP_DO_DQS[2]	DQS[2]
—	DQS_IN [2]	—	—	—	IPP_DIN_DQS[2]	
—	DQS_OUT [1]	—	—	—	IPP_DO_DQS[1]	DQS[1]
—	DQS_IN [1]	—	—	—	IPP_DIN_DQS[1]	
—	DQS_OUT [0]	—	—	—	IPP_DO_DQS[0]	DQS[0]
—	DQS_IN [0]	—	—	—	IPP_DIN_DQS[0]	
—	—	—	—	—	M_REQUEST	M_REQUEST
—	—	—	—	—	M_GRANT	M_GRANT
NANDFC COMMAND DEDICATED PADS						
—	—	—	—	IPP_NFC_WE_OUT	IPP_NFC_WE_OUT	NFWE_B
—	—	—	—	IPP_NFC_WP_OUT	IPP_NFC_WP_OUT	NFWP_B
—	—	—	—	IPP_NFC_RE_OUT	IPP_NFC_RE_OUT	NFRE_B

Table 16-2. EMI Output Summary (continued)

MEMORY CONTROLLERS OUTPUTS					EMI OUTPUT	IC PIN NAME
SDRAMC	MDDRC	PCMCIA	WEIM	NANDFC		
—	—	—	—	IPP_NFC_ALE_OUT	IPP_NFC_ALE_OUT	NFALE
—	—	—	—	IPP_NFC_CLE_OUT	IPP_NFC_CLE_OUT	NFCLE
—	—	—	—	IPP_NFC_CE_OUT	IPP_NFC_CE_OUT	NFCE_B
—	—	—	—	IPP_NFC_RB_IN	IPP_NFC_RB_IN	NFRB
WEIM COMMAND DEDICATED PADS						
—	—	IPP_DO_CARD_OE_B	IPP_DO_WEIM_LBA_B	—	IPP_DO_WEIM_LBA_B	LBA_B
—	—	—	IPP_DO_WEIM_BCLK	—	IPP_DO_WEIM_BCLK	BCLK
—	—	—	IPP_IND_WEIM_ECB_B	—	IPP_IND_WEIM_ECB_B	ECB
PCMCIA COMMAND DEDICATED PADS						
—	—	IPP_INT_CD1_B	—	—	IPP_INT_CD1_B	PC_CD1_B
—	—	IPP_INT_CD2_B	—	—	IPP_INT_CD2_B	PC_CD2_B
—	—	IPP_IND_WAIT_B_I	—	—	IPP_IND_WAIT_B_I	PC_WAIT_B
—	—	IPP_IND_PWR_ON_I	—	—	IPP_IND_PWR_ON_I	PC_PWRON
—	—	IPP_IND_RDY_IRQ_B_I	—	—	IPP_IND_RDY_IRQ_B_I	PC_READY
—	—	IPP_IND_VS1_B	—	—	IPP_IND_VS1_B	PC_VS1
—	—	IPP_IND_VS2_B	—	—	IPP_IND_VS2_B	PC_VS2
—	—	IPP_IND_BVD1_STSCH_B_I	—	—	IPP_IND_BVD1_STSCH_B_I	PC_BVD1
—	—	IPP_IND_BVD2_SPKR_I	—	—	IPP_IND_BVD2_SPKR_I	PC_BVD2
—	—	IPP_DO_CARD_RESET	—	—	IPP_DO_CARD_RESET	PC_RST

Table 16-2. EMI Output Summary (continued)

MEMORY CONTROLLERS OUTPUTS					EMI OUTPUT	IC PIN NAME
SDRAMC	MDDRC	PCMCIA	WEIM	NANDFC		
—	—	IPP_DO_CARD_POE_O_B	—	—	IPP_DO_CARD_POE_O_B	PC_POE
—	—	IPP_DO_CARD_RW_B	—	—	IPP_DO_CARD_RW_B	PC_RW_B
—	—	IPP_IND_WP_I	—	—	IPP_IND_WP_I	IOIS16

16.8.2 EMI Input/Output Signals

This section lists the input and output signals for the entire EMI module. Table 16-5 summarizes the interface signals, and for the detailed description of each signal function, refer to the relevant module chapter in this document.

Table 16-5. EMI Signal Properties

Name	Port	Function	Reset State
AHB INTERFACE OUTPUTS			
M3IF_HREADY_M0	O	AHB access completion strobe to master #0—L2CC0	1
M3IF_HREADY_M1	O	AHB access completion strobe to master #1—L2CC1	1
M3IF_HREADY_M2	O	AHB access completion strobe to master #2 (MAX)—MAX port 0	1
M3IF_HREADY_M3	O	AHB access completion strobe to master #3 (MAX)—MAX port 1	1
M3IF_HREADY_M4	O	AHB access completion strobe to master #4—SDMA	1
M3IF_HREADY_M5	O	AHB access completion strobe to master #5—MPEG-4 ENCODER	1
M3IF_HREADY_M6	O	AHB access completion strobe to master #6—IPU	1
M3IF_HREADY_M7	O	AHB access completion strobe to master #7—IPU	1
M3IF_HRESP_M0	O	AHB error response to master #0—L2CC0	0
M3IF_HRESP_M1	O	AHB error response to master #1—L2CC1	0
M3IF_HRESP_M2	O	AHB error response to master #2 (MAX)—MAX port 0	0
M3IF_HRESP_M3	O	AHB error response to master #3 (MAX)—MAX port 1	0
M3IF_HRESP_M4	O	AHB error response to master #4—SDMA	0
M3IF_HRESP_M5	O	AHB error response to master #5—MPEG-4 ENCODER	0
M3IF_HRESP_M6	O	AHB error response to master #6—IPU	0
M3IF_HRESP_M7	O	AHB error response to master #7—IPU	0
M3IF_HRDATA_M0[63:0]	O	AHB read data bus to master #0—L2CC0	0

Table 16-5. EMI Signal Properties (continued)

Name	Port	Function	Reset State
M3IF_HRDATA_M1[63:0]	O	AHB read data bus to master #1—L2CC1	0
M3IF_HRDATA_M2[63:0]	O	AHB read data bus to master #2 (MAX)—MAX port 0	0
M3IF_HRDATA_M3[31:0]	O	AHB read data bus to master #3 (MAX)—MAX port 1	0
M3IF_HRDATA_M4[63:0]	O	AHB read data bus to master #4—SDMA	0
M3IF_HRDATA_M5[63:0]	O	AHB read data bus to master #5—MPEG-4 ENCODER	0
M3IF_HRDATA_M6[63:0]	O	AHB read data bus to master #6—IPU	0
M3IF_HRDATA_M7[31:0]	O	AHB read data bus to master #7—IPU	0
M3IF and ESDCTL/MDDRC OUTPUTS			
IPP_DO_SDRC_SDCKE[1:0]	O	SDRAM/MDDR clock enable	0
M3IF_DQM[3:0]	O	SDRAM data mask strobes. DQM0 corresponds to DQ0–DQ7, DQM1 corresponds to DQ8–DQ15, DQM2 corresponds to DQ16–DQ23 and DQM3 corresponds to DQ24–DQ31.	0
DQS_OUT[3:0]	O	MDDR data sample strobes for write accesses. DQS0 corresponds to DQ0–DQ7, DQS1 corresponds to DQ8–DQ15, DQS2 corresponds to DQ16–DQ23 and DQS3 corresponds to DQ24–DQ31.	0
DQS_OUT_EN_X	O	DQS output enable strobe	0
M3IF_MA[13:0]	O	SDRAM/MDDR address bits	0
M3IF_BA[1:0]	O	SDRAM/MDDR bank address bits	0
IPP_DO_M3IF_MA10	O	SDRAM/MDDR address bit A10	0
M3IF_CS_B[1:0]	O	SDRAM/MDDR chip select strobe	3
CAS_B	O	SDRAM/MDDR CAS strobe	1
RAS_B	O	SDRAM/MDDR RAS strobe	1
WE_B	O	SDRAM/MDDR WE strobe	1
M3IF_CHOOSEN_MASTER[2:0]	O	M3IF arbitration chosen master (for debug)	3
IPP_DO_SDRC_SDCLK	O	SDRAM/MDDR clock (up to 133MHz)	0
IPP_DO_MDDR_SD_CLK_B	O	MDDR clock (up to 133MHz)	0
LPACK	O	Low power mode acknowledge—toward CCM	1
SDRC_SF_WACK	O	Memory wakeup acknowledge indication to WDOG	0
NANDFC OUTPUTS			
IPP_DO_NFC_WRITE_DATA_OUT[15:0]	O	NANDFC write data out toward I/O MUX/PADS.	0
IPI_INT_NFC_B	0	NANDFC interrupt (indicating an access completion)	1
IPP_NFC_ALE_OUT	O	NANDFC out NF_ALE	0
IPP_NFC_CE_OUT	O	NANDFC out NF_CE	0

Table 16-5. EMI Signal Properties (continued)

Name	Port	Function	Reset State
IPP_NFC_CLE_OUT	O	NANDFC out NF_CLE	0
IPP_NFC_RE_OUT	O	NANDFC out NF_RE	0
IPP_NFC_WE_OUT	O	NANDFC out NF_WE	0
IPP_NFC_WP_OUT	O	NANDFC out NF_WP	0
WEIM OUTPUTS			
IPP_DO_WEIM_CS_B0	O	WEIM CS0 chip select toward I/O MUX/PADS	1
IPP_DO_WEIM_CS_B1	O	WEIM CS1 chip select toward I/O MUX/PADS	1
IPP_DO_WEIM_CS_B2_CSD0	O	WEIM CS2 or ESDCTL/MDDRC CSD0 chip select toward I/O MUX/PADS	1
IPP_DO_WEIM_CS_B3_CSD1	O	WEIM CS2 or ESDCTL/MDDRC CSD1 chip select toward I/O MUX/PADS	1
IPP_DO_WEIM_CS_B4	O	WEIM CS4 chip select toward I/O MUX/PADS	1
IPP_DO_WEIM_CS_B5	O	WEIM CS5 chip select toward I/O MUX/PADS	1
IPP_DO_WEIM_BCLK	O	WEIM Burst Clock	0
IPP_DO_LBA_B	O	WEIM Load Burst Address (LBA)	1
IPP_DO_WEIM_RW_B	O	WEIM/PCMCIA read/write strobe	1
PCMCIA OUTPUTS			
IPI_INT_BVD1_B	O	BVD1 changed interrupt	1
IPI_INT_BVD2_B	O	BVD2 changed interrupt	1
IPI_INT_CD1_B	O	CD1 changed interrupt	1
IPI_INT_CD2_B	O	CD2 changed interrupt	1
IPI_INT_ERR_B	O	Access error interrupt	1
IPI_INT_IRQ_B	O	or of all the ready interrupts	1
IPI_INT_NFC_B	O	NANDFC interrupt (indicating an action completed)	1
IPI_INT_PCMCIA_B	O	or of all the interrupts (status+access+ready)	1
IPI_INT_POWERON_B	O	POWER_ON changed interrupt	1
IPI_INT_RDY_F_B	O	RDY negedge interrupt	1
IPI_INT_RDY_H_B	O	RDY is high interrupt	1
IPI_INT_RDY_L_B	O	RDY is low level sensitive interrupt	1
IPI_INT_RDY_R_B	O	RDY posedge sensitive interrupt	1
IPI_INT_STS_B	O	or of all the status interrupts	1
IPI_INT_VS1_B	O	VS1 changed interrupt	1

Table 16-5. EMI Signal Properties (continued)

Name	Port	Function	Reset State
IPI_INT_VS2_B	O	VS2 changed interrupt	1
IPI_INT_WP_B	O	WP changed interrupt	1
IPP_DO_CARD_POE_O	O	POE signal to transceivers	0
IPP_DO_CARD_RESET_O	O	RESET signal to card	0
IPP_DO_CARD_RW_B	O	RW_B Signal to data transceiver	1
IPP_DO_SPKR_OUT_O	O	speaker out	0
GLOBAL OUTPUTS			
M3IF_DMA_ACCESS	O	Snooping detection indication toward IPU module	0
IPP_OBE_DDR_EN	O	MDDR active indication to ESDCTL/MDDRC DATA PADS	0
IPP_DO_EMI_ADDR[25:0]	O	EMI address out toward I/O MUX/PADS	0
IPP_DO_NFC_WRITE_DATA_OUT[15:0]	O	EMI data (NANDFC, WEIM, or PCMCIA) out toward I/O MUX/PADS	0
IPP_DO_EMI_DATA[31:0]	O	EMI SDRAM/DDR data out toward I/O MUX/PADS	0
IPP_OBE_EMI_DATA_DIR	O	EMI SDRAM/DDR data direction toward I/O MUX/PADS	0
IPP_OBE_NFC_DIR_HIGH	O	EMI (NANDFC, WEIM, PCMCIA) data direction toward I/O MUX/PADS	0
IPP_OBE_NFC_DIR_LOW	O	EMI (NANDFC, WEIM, PCMCIA) data direction toward I/O MUX/PADS	0
IPP_DO_SDBA[1:0]	O	SDRAM/MDDR bank address toward I/O MUX/PADS	0
IPP_DO_M3IF_MA10	O	SDRAM/MDDR address bit MA10 toward I/O MUX/PADS	0
IPP_DO_EMI_IO_EB_B[1:0]	O	EMI enable byte out toward I/O MUX/PADS	0
IPP_DO_EMI_IO_DQM[3:0]	O	SDRAM/MDDR enable bytes toward I/O MUX/PADS	0
IPP_DO_EMI_OE_B	O	EMI output enable toward I/O MUX/PADS	0
IPP_OBE_IO_ADDR_DIR[1:0]	O	EMI output enable (dir) toward I/O ADDR/WEIM MUXED DATA MUX/PADS	0
AHB INTERFACE INPUTS			
M3IF_HADDR_M0[31:0]	I	AHB address bus from master #0—L2CC0	0
M3IF_HADDR_M1[31:0]	I	AHB address bus from master #1—L2CC1	0
M3IF_HADDR_M2[31:0]	I	AHB address bus from master #2 (MAX)—MAX port 0	0
M3IF_HADDR_M3[31:0]	I	AHB address bus from master #3 (MAX)—MAX port 1	0
M3IF_HADDR_M4[31:0]	I	AHB address bus from master #4—SDMA	0
M3IF_HADDR_M5[31:0]	I	AHB address bus from master #5—MPEG-4 ENCODER	0
M3IF_HADDR_M6[31:0]	I	AHB address bus from master #6—IPU	0
M3IF_HADDR_M7[31:0]	I	AHB address bus from master #7—IPU	0

Table 16-5. EMI Signal Properties (continued)

Name	Port	Function	Reset State
M3IF_HWDATA_M0[63:0]	I	AHB write data bus (64 bit) from master #0—L2CC0	0
M3IF_HWDATA_M1[63:0]	I	AHB write data bus (64 bit) from master #1—L2CC1	0
M3IF_HWDATA_M2[63:0]	I	AHB write data bus (32 bit) from master #2 (MAX)—MAX port 0	0
M3IF_HWDATA_M3[31:0]	I	AHB write data bus (32 bit) from master #3 (MAX)—MAX port 1	0
M3IF_HWDATA_M4[63:0]	I	AHB write data bus (32 bit) from master #4—SDMA	0
M3IF_HWDATA_M5[63:0]	I	AHB write data bus (32 bit) from master #5—MPEG-4 ENCODER	0
M3IF_HWDATA_M6[63:0]	I	AHB write data bus (32 bit) from master #6—IPU	0
M3IF_HWDATA_M7[31:0]	I	AHB write data bus (32 bit) from master #7—IPU	0
M3IF_HBURST_M0[2:0]	I	AHB burst size bus from master #0—L2CC0	0
M3IF_HBURST_M1[2:0]	I	AHB burst size bus from master #1—L2CC1	0
M3IF_HBURST_M2[2:0]	I	AHB burst size bus from master #2(MAX)—MAX port 0	0
M3IF_HBURST_M3[2:0]	I	AHB burst size bus from master #3 (MAX)—MAX port 1	0
M3IF_HBURST_M4[2:0]	I	AHB burst size bus from master #4—SDMA	0
M3IF_HBURST_M5[2:0]	I	AHB burst size bus from master #5—MPEG-4 ENCODER	0
M3IF_HBURST_M6[2:0]	I	AHB burst size bus from master #6—IPU	0
M3IF_HBURST_M7[2:0]	I	AHB burst size bus from master #7—IPU	0
M3IF_HSIZE_M0[1:0]	I	AHB data transfer width bus from master #0—L2CC0	0
M3IF_HSIZE_M1[1:0]	I	AHB data transfer width bus from master #1—L2CC1	0
M3IF_HSIZE_M2[1:0]	I	AHB data transfer width bus from master #2 (MAX)—MAX port 0	0
M3IF_HSIZE_M3[1:0]	I	AHB data transfer width bus from master #3 (MAX)—MAX port 1	0
M3IF_HSIZE_M4[1:0]	I	AHB data transfer width bus from master #4—SDMA	0
M3IF_HSIZE_M5[1:0]	I	AHB data transfer width bus from master #5—MPEG-4 ENCODER	0
M3IF_HSIZE_M6[1:0]	I	AHB data transfer width bus from master #6—IPU	0
M3IF_HSIZE_M7[1:0]	I	AHB data transfer width bus from master #7—IPU	0
M3IF_HBSTRB_M0[7:0]	I	byte lane (8) bus from master #0—L2CC0	0
M3IF_HBSTRB_M1[7:0]	I	byte lane (8) bus from master #1—L2CC1	0
M3IF_HBSTRB_M2[7:0]	I	byte lane (4) bus from master #2 (MAX)—MAX port 0	0
M3IF_HBSTRB_M3[3:0]	I	byte lane (4) bus from master #3 (MAX)—MAX port 1	0
M3IF_HBSTRB_M4[7:0]	I	byte lane (4) bus from master #4—SDMA	0
M3IF_HBSTRB_M5[7:0]	I	byte lane (4) bus from master #5—MPEG-4 ENCODER	0
M3IF_HBSTRB_M6[7:0]	I	byte lane (4) bus from master #6—IPU	0

Table 16-5. EMI Signal Properties (continued)

Name	Port	Function	Reset State
M3IF_HBSTRB_M7[3:0]	I	byte lane (4) bus from master #7—IPU	0
M3IF_HTRANS_M0[1:0]	I	AHB transfer state bus from master #0—L2CC0	0
M3IF_HTRANS_M1[1:0]	I	AHB transfer state bus from master #1—L2CC1	0
M3IF_HTRANS_M2[1:0]	I	AHB transfer state bus from master #2 (MAX)—MAX port 0	0
M3IF_HTRANS_M3[1:0]	I	AHB transfer state bus from master #3 (MAX)—MAX port 1	0
M3IF_HTRANS_M4[1:0]	I	AHB transfer state bus from master #4—SDMA	0
M3IF_HTRANS_M5[1:0]	I	AHB transfer state bus from master #5—MPEG-4 ENCODER	0
M3IF_HTRANS_M6[1:0]	I	AHB transfer state bus from master #6—IPU	0
M3IF_HTRANS_M7[1:0]	I	AHB transfer state bus from master #7—IPU	0
M3IF_HWRITE_M0	I	AHB read/write signal from master #0—L2CC0	0
M3IF_HWRITE_M1	I	AHB read/write signal from master #1—L2CC1	0
M3IF_HWRITE_M2	I	AHB read/write signal from master #2 (MAX)—MAX port 0	0
M3IF_HWRITE_M3	I	AHB read/write signal from master #3 (MAX)—MAX port 1	0
M3IF_HWRITE_M4	I	AHB read/write signal from master #4—SDMA	0
M3IF_HWRITE_M5	I	AHB read/write signal from master #5—MPEG-4 ENCODER	0
M3IF_HWRITE_M6	I	AHB read/write signal from master #6—IPU	0
M3IF_HWRITE_M7	I	AHB read/write signal from master #7—IPU	0
M3IF_HPROT_M0	I	AHB protection mode signal from master #0—L2CC0	0
M3IF_HPROT_M1	I	AHB protection mode signal from master #1—L2CC1	0
M3IF_HPROT_M2	I	AHB protection mode signal from master #2 (MAX)—MAX port 0	0
M3IF_HPROT_M3	I	AHB protection mode signal from master #3 (MAX)—MAX port 1	0
M3IF_HPROT_M4	I	AHB protection mode signal from master #4—SDMA	0
M3IF_HPROT_M5	I	AHB protection mode signal from master #5—MPEG-4 ENCODER	0
M3IF_HPROT_M6	I	AHB protection mode signal from master #6—IPU	0
M3IF_HPROT_M7	I	AHB protection mode signal from master #7—IPU	0
M3IF_HUNALIGN_M0	I	Unalign access signal from master #0—L2CC0	0
M3IF_HUNALIGN_M1	I	Unalign access signal from master #1—L2CC1	0
M3IF_HUNALIGN_M2	I	Unalign access signal from master #2 (MAX)—MAX port 0	0
M3IF_HUNALIGN_M3	I	Unalign access signal from master #3 (MAX)—MAX port 1	0
M3IF_HUNALIGN_M4	I	Unalign access signal from master #4—SDMA	0
M3IF_HUNALIGN_M5	I	Unalign access signal from master #5—MPEG-4 ENCODER	0
M3IF_HUNALIGN_M6	I	Unalign access signal from master #6—IPU	0

Table 16-5. EMI Signal Properties (continued)

Name	Port	Function	Reset State
M3IF_HUNALIGN_M7	I	Unalign access signal from master #7—IPU	0
M3IF and ESDCTL/MDDRC INPUTS			
M3IF_HCLK	I	M3IF AHB system clock—up to 133MHz	0
HCLK32	I	32 KHz clock for ESDCTL refresh counter	0
IPP_IND_SDRC_SDCLK_FB	I	SDRAM/MDDR feedback clock (up to 133MHz)	0
LPMD	I	Low power mode indication signal, “0”=STOP, “1”=RUN.	1
DQS_IN[3:0]	I	MDDR data sample strobes for read accesses. DQS0 corresponds to DQ0–DQ7, DQS1 corresponds to DQ8–DQ15, DQS2 corresponds to DQ16–DQ23 and DQS3 corresponds to DQ24–DQ31.	0
SDCTL_CSD0_SEL_B	I	SDRAM/MDDR CSD0 select multiplexed with CS2 (configurable via the system control register, FMCR)	0
SDCTL_CSD1_SEL_B	I	SDRAM/MDDR CSD1 select multiplexed with CS3 (configurable via the system control register, FMCR)	0
NANDFC INPUTS			
NF16_BOOT_B	I	Boot mode source is 16-bit NAND Flash memory.	application dependent
NF8_BOOT_B	I	Boot mode source is 8-bit NAND Flash memory.	application dependent
NF_16BIT_SEL	I	16-bit NAND Flash memory is use indication.	0
NFC_HCLK	I	NANDFC AHB input clock	0
NFC_RD_OE	I	NANDFC read output enable—controls the direction of data bus	0
NFC_WR_OE	I	NANDFC write output enable—controls the direction of data bus	0
IPP_IND_FLASH_CLK	I	NAND Flash side clock with period of 40nS	0
IPP_IND_NFC_RB_IN	I	NANDFC in NF_RB	0
WEIM INPUTS			
WEIM_BOOT_CFG[3:0]	I	WEIM Boot Mode select (from CCM) For a more detailed WEIM description, see Chapter 18, “Wireless External Interface Module (WEIM).”	application dependent
IPP_IND_WEIM_ECB_B	I	WEIM End Current Burst	1
WEIM_HCLK	I	WEIM AHB input clock	0
IPP_IND_WEIM_DTACK_B	I	External DTACK acknowledge	1
PCMCIA INPUTS			
IPP_IND_CD_B_I[1:0]	I	CD[1:0] signals from card	3
IPP_IND_VS_I[1:0]	I	VS[1:0] signals from card	0

Table 16-5. EMI Signal Properties (continued)

Name	Port	Function	Reset State
IPP_IND_BVD1_STSCH_B_I	I	BVD1/STSCHG signals from card	0
IPP_IND_BVD2_SPKR_I	I	BVD2/SPKR signals from card	0
IPP_IND_PWR_ON_I	I	POWER_ON signals from card/board	0
IPP_IND_RDY_IRQ_B_I	I	READY/IRQ_B signals from card	1
IPP_IND_WP_I	I	WP signals from card	1
GLOBAL INPUTS			
IPP_IND_RESETB	I	reset signal	1
HRESET_HCLK_B	I	reset signal	1
IPP_IND_NFC_READ_DATA_IN[15:0]	I	External memories (non SDRAM) read data in from I/O MUX/PADS	0
IPP_IND_EMI_DATA_IN[31:0]	I	EMI SDRAM/DDR data in from I/O MUX/PADS	0
IPP_IND_ADDR_IN[15:0]	I	EMI WEIM muxed data in from I/O MUX/PADS. WEIM/NANDFC/PCMCIA address out to I/O MUX/PADS.	0
IPP_IND_EMI_DATA_IN[31:0]	I	SDRAM/MDDR read data in from I/O MUX/PADS	0
M3IF_BIGEND_M0	I	Endian mode signal from master #0—L2CC0	master dependent
M3IF_BIGEND_M1	I	Endian mode signal from master #1—L2CC1	master dependent
M3IF_BIGEND_M2	I	Endian mode signal from master #2 (MAX)—MAX port 0	master dependent
M3IF_BIGEND_M3	I	Endian mode signal from master #3 (MAX)—MAX port 1	master dependent
M3IF_BIGEND_M4	I	Endian mode signal from master #4—SDMA	master dependent
M3IF_BIGEND_M5	I	Endian mode signal from master #5—MPEG-4 ENCODER	master dependent
M3IF_BIGEND_M6	I	Endian mode signal from master #6—IPU	master dependent
M3IF_BIGEND_M7	I	Endian mode signal from master #7—IPU	master dependent

16.9 Memory Map/Register Definition

The EMI supports 4 different memory controllers. Table 16-6 and Table 16-7 illustrate the EMI registers and memory map (mapped by all memory controllers). For detailed description regarding both “registers definition” and “memory map,” refer to each relevant chapter.

Table 16-6. EMI Registers Definition

Address	Use	Access
M3IF Registers Space		
0xB800_3000–0xB800_3FFF (ARM)	M3IF registers space (4K)	READ/WRITE
ESDCTL/MDDRC Registers Space		
0xB800_1000–0xB800_1FFF (ARM)	ESDCTL/MDDRC registers space (4K)	READ/WRITE
PCMCIA Registers Space		
0xB800_4000–0xB800_4FFF	PCMCIA registers space (4K)	READ/WRITE
WEIM Registers Space		
0xB800_2000–0xB800_2FFF (ARM)	WEIM registers space (4K)	READ/WRITE
NANDFC Registers Space		
0xB800_0E00–0xB800_0FFF (ARM)	NANDFC registers space	READ/WRITE

Table 16-7. EMI Memory Map

Address	Use	Access
ESDCTL/MDDRC Memory Space		
0x8000_0000–0x8FFF_FFFF (ARM)	CSD0 SDRAM/MDDR memory region (256 Mbytes)	READ/WRITE
0x9000_0000–0x9FFF_FFFF (ARM)	CSD1 SDRAM/MDDR memory region (256 Mbytes)	READ/WRITE
PCMCIA Memory Space		
0xC000_0000–0xC3FF_FFFF	PCMCIA/CF memory region (64 Mbytes)	READ/WRITE
WEIM Memory Space		
0xA000_0000–0xA7FF_FFFF (ARM)	WEIM CS0 memory region[1] (128 Mbytes)	READ/WRITE
0xA800_0000–0xAFFF_FFFF (ARM)	WEIM CS1 memory region (128 Mbytes)	READ/WRITE
0xB000_0000–0xB1FF_FFFF (ARM)	WEIM CS2 memory region (32 Mbytes)7	READ/WRITE
0xB200_0000–0xB3FF_FFFF (ARM)	WEIM CS3 memory region (32 Mbytes)	READ/WRITE
0xB400_0000–0xB5FF_FFFF (ARM)	WEIM CS4 memory region (32 Mbytes)	READ/WRITE

Table 16-7. EMI Memory Map (continued)

Address	Use	Access
0xB600_0000–0xB7FF_FFFF (ARM)	WEIM CS5 memory region (32 Mbytes)	READ/WRITE
NANDFC Memory Space		
0xB800_0000–0xB800_0FFF (ARM)	NANDFC memory region[1] (4K, NAND Flash)	READ/WRITE

1. Can be used as a boot memory region.

Chapter 17
Multi-Master Memory Interface (M3IF)

The Multi-Master Memory Interface (M3IF) controls memory accesses (read/write/erase/program) from one or more masters through different port interfaces to different external memory controllers ESDCTL, PCMCIA, NANDFLASH, and WEIM. Figure 17-1 provides the M3IF top-level diagram that shows the functional organization of the block.

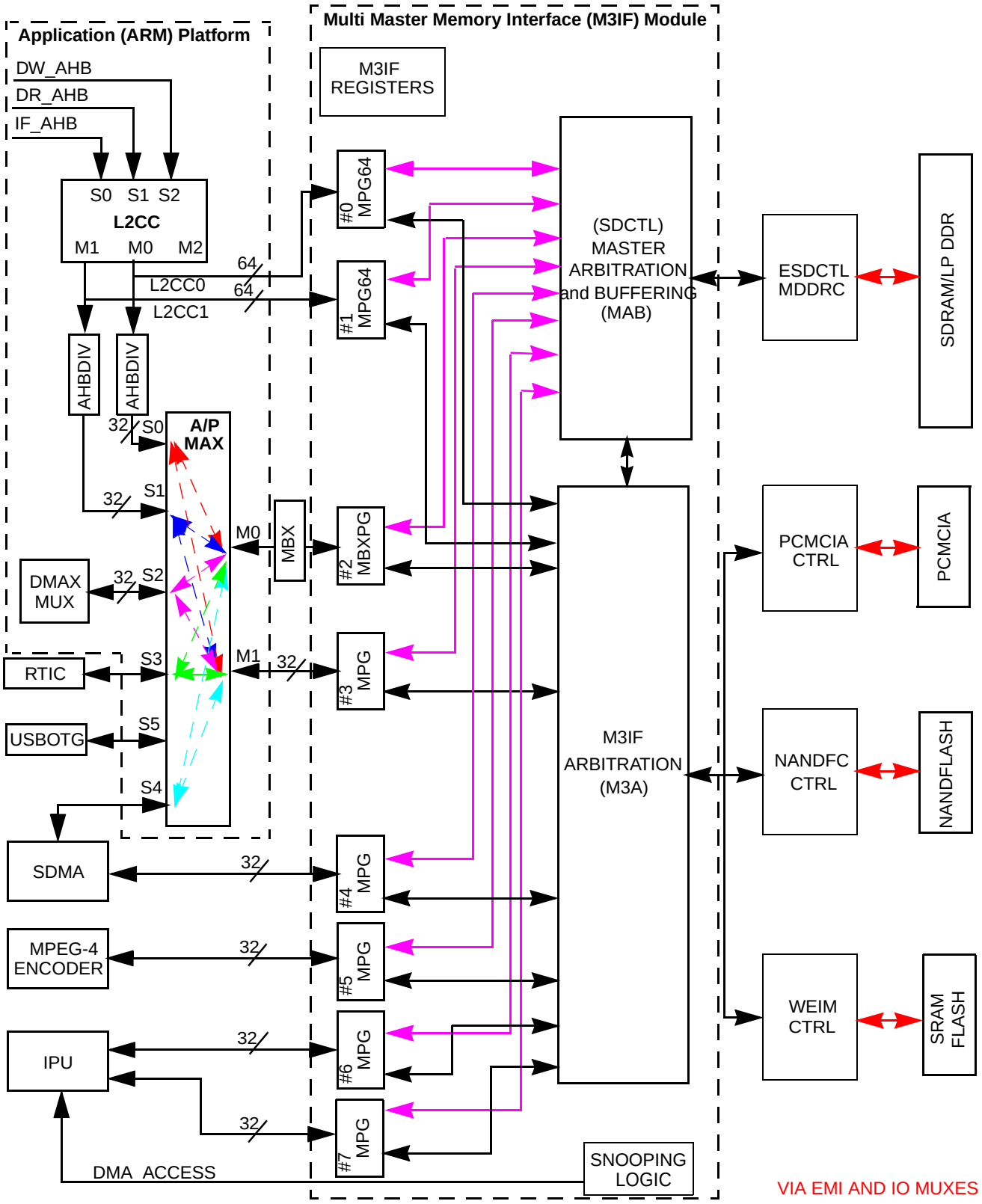


Figure 17-1. M3IF Block Diagram

17.1 Overview

The M3IF-ESDCTL interface is optimized and designed to reduce access latency by generating multiple accesses through the dedicated ESDCTL arbitration (MAB) module, which controls the access to/from the Enhanced SDRAM/MDDR memory controller. For the other port interfaces, the M3IF only arbitrates and forwards the master requests received through the Master Port Gasket (MPG) interface and M3IF Arbitration (M3A) module toward the respective memory controller. The masters that interface with the M3IF include the ARM Platform, SDMA, MPEG-4 encoder, and the IPU. The controllers are the ESDCTL, PCMCIA, NANDFLASH, and WEIM.

When a master requests a memory access, the access is immediately taken by the M3IF if no other access is in progress. The M3IF forwards the access to the respective memory controller (slave), and depending on the state of the respective memory controller, a command to the memory is generated. If the access cannot be started due to a previous active access, the master request remains pending (HREADY held negated) until it is executed by the memory controller. When the access execution is complete, the HREADY is asserted and a new request can be processed.

Accesses to the SDRAM or MDDR external devices are optimized through command anticipation (MIF2 strategy). For example, the next access control phase (memory address and command) is driven during the previous access data phase (data flow to/from the memory), thus an overlap between accesses is created and latency is partially or fully hidden.

17.1.1 M3IF Interfaces

The interface between the M3IF and the controllers can be divided into two different types: M3IF-ESDCTL and M3IF-all others. The M3IF-ESDCTL interface reduces access latency by generating multiple accesses using the dedicated ESDCTL arbitration (MAB) module.

For the other port interfaces, the M3IF arbitrates and forwards the masters' requests received through the Master Port Gasket (MPG) interfaces and the M3IF arbitration (M3A) module toward the respective memory controller. To support multiple accesses to the ESDCTL, the MAB includes a FIFO that controls the access traffic from/to the ESDCTL.

The M3IF can be viewed as a device that has multiple SDRAM/MDDR controllers and one controller per other memory type, therefore, the M3A arbitrates the requests as follows:

- A round robin chooses the next master that is going to grant the bus:
  - If this master is requesting access to non-SDRAM/MDDR memory controller, the M3A waits until the previous access finishes and only then passes the request.
  - If the master is requesting access to ESDCTL, the M3IF arbitration passes the access to the MAB in two cases:
    - After the previous non-ESDCTL access is accomplished.
    - If previous access was to ESDCTL, the M3IF arbitration will pass the request immediately without waiting for the previous access to accomplished.

The M3IF Arbitration (M3A) and the ESDCTL Master Arbitration and Buffering (MAB) supports a round-robin arbitration scheme (which can be programed to non-equal probability). If two masters request access to the memory port on the same cycle the master with the token gains control on the bus to the slave.

To support multiple accesses to the ESDCTL, the MAB includes a FIFO that controls the access traffic from/to the ESDCTL. Once a master grants the bus, the Memory Controller gaining the access converts the access to a command to the specified memory.

17.1.2 Features

The M3IF Master Port Gasket (MPG) converts the master request (data write, data read, address, and controls) to a set of bus/signals that the M3IF arbitration, the SDCTL/MDDRC arbitration, and other memory controllers need. The MPG is also responsible to give the right response to the master after getting the response from the relevant memory controller. The M3IF support 3 port interfaces (the number and types of gasket ports used depends on the system requirements):

- MPG–Master Port Gasket for ARM11 AMBA-AHB lite with 32 bit data bus.
- MPG64—Master Port Gasket for AMBA-AHB lite with 64 bit data bus.
- MBXPG—Dedicated Master Port Gasket for the MBX module (64 bit data bus gasket).

NOTE

The MBX R-S graphics accelerator is not available in the i.MX31L.

The M3IF includes these distinctive features:

- Supports multiple requests from eight masters through three different input port interfaces:
  - Master Port Gasket (MPG)—ARM11 AMBA AHB lite bus protocol.
  - Master Port Gasket (MPG64)—AMBA AHB access with 64 bits data bus width.
  - MBXPG—Dedicated Master Port Gasket for the MBX module (64 bit data bus gasket).
- Arbitrates requests to four different memory controllers (that share some of their I/O pads)
  - Enhanced SDRAM Controller (ESDCTL) or MDDR Controller (MDDRC)
  - NAND Flash Controller—(NANDFC)
  - Personal Computer Memory Card International Association Controller—(PCMCIA)
  - Wireless External Interface Memory Controller—(WEIM)
- Multiple requests capabilities to ESDCTL through a dedicated arbitration mechanism.
- Flexible round robin access arbitration, with equal priority or 50% priority to selective masters.
- Programmable master that controls (lock) accesses to SDRAM/DDR and programmable master that controls (lock) accesses to other memories (= general: NANDFC, WEIM, PCMCIA).
- Multi-Endianness support to all memory controllers.
- Supports memory snooping, an example of which would be monitoring a region in external memory for write accesses:
  - The region’s location is specified by a base address (from 2 Kbytes up to 16 Mbytes), which is divided into 64 equal segments.
  - Each segment has an access status and enable bit in the M3IF register definition.
  - M3IF generates a one cycle DMA_ACCESS for each snooping detection.

17.2 Memory Map and Register Definition

The M3IF programming model consists of two classes of registers, M3IF control and lock registers and snooping configuration and status registers, as shown in Table 17-1. The control and master lock general register defines the M3IF configurable logic functionality. The configuration and status registers set and monitor snooping activity.

All M3IF registers are 32-bits in length with bit fields defined in Figure 17-3 to Figure 17-9. All implemented bits are fully readable and writable in supervisor mode only (an error response will be generated in case of user mode access to M3IF registers). All M3IF (and ESDCTL) registers can be accessed only by a SINGLE word (32-bit) access, through the AHB bus protocol. Accesses of any other size or type will cause an undetermined behavior.

All registers can be accessed by only one master at a time. Multi access to M3IF register causes undetermined behavior. The only exception is the M3IF Master Lock General register that can be accessed by more than one master at a time.

The reset state of each bit is shown underneath the bit field name. An asterisk indicates that the value is dependent on the operating mode selected during reset. Details are provided in the following bit field descriptions.

17.2.1 Memory Map

The M3IF supports four different memory controllers. Each memory controller defines a specific memory address mapped, as shown in Table 17-1.

Table 17-1 shows the M3IF Memory Map, and Table 17-2 shows the M3IF Memory Space Summary.

Table 17-1. M3IF Memory Map

Address	Register	Access	Reset Value	Section/Page
0xB800_3000 (M3IFCTL)	M3IF Control Register	R/W	0x0000_0000	17.2.3.1/17-9
0xB800_3028 (M3IFSCFG0)	M3IF Snooping Configuration Register 0	R/W	0x0000_0000	17.2.3.2/17-11
0xB800_302C (M3IFSCFG1)	M3IF Snooping Configuration Register 1	R/W	0x0000_0000	17.2.3.3/17-13
0xB800_3030 (M3IFSCFG2)	M3IF Snooping Configuration Register 2	R/W	0x0000_0000	17.2.3.4/17-14
0xB800_3034 (M3IFSSR0)	M3IF Snooping Status Register 0	R/W	0x0000_0000	17.2.3.5/17-14
0xB800_3038 (M3IFSSR1)	M3IF Snooping Status Register 1	R/W	0x0000_0000	17.2.3.6/17-15
0xB800_3040 (M3IFMLWE0)	M3IF Master Lock WEIM CS0 Register	R/W	0x0000_0000	17.2.3.7/17-16
0xB800_3044 (M3IFMLWE1)	M3IF Master Lock WEIM CS1 Register	R/W	0x0000_0000	17.2.3.7/17-16

Table 17-1. M3IF Memory Map (continued)

Address	Register	Access	Reset Value	Section/Page
0xB800_3048 (M3IFMLWE2)	M3IF Master Lock WEIM CS2 Register	R/W	0x0000_0000	17.2.3.7/17-16
0xB800_304C (M3IFMLWE3)	M3IF Master Lock WEIM CS3 Register	R/W	0x0000_0000	17.2.3.7/17-16
0xB800_3050 (M3IFMLWE4)	M3IF Master Lock WEIM CS4 Register	R/W	0x0000_0000	17.2.3.7/17-16
0xB800_3054 (M3IFMLWE5)	M3IF Master Lock WEIM CS5 Register	R/W	0x0000_0000	17.2.3.7/17-16

Table 17-2. M3IF Memory Space Summary

Address	Use	Access
ESDCTL Memory Space		
0x8000_0000–0x8FFF_FFFF	CSD0 SDRAM or MDDR memory region (256 Mbyte)	READ/WRITE
0x9000_0000–0x9FFF_FFFF	CSD1 SDRAM or MDDR memory region (256 Mbytes)	READ/WRITE
WEIM Memory Space		
0xA000_0000–0xA7FF_FFFF	WEIM CS0 memory region (128 Mbytes)	READ/WRITE
0xA800_0000–0xAFFF_FFFF	WEIM CS1 memory region (128 Mbytes)	READ/WRITE
0xB000_0000–0xB1FF_FFFF	WEIM CS2 memory region (32 Mbytes)	READ/WRITE
0xB200_0000–0xB3FF_FFFF	WEIM CS3 memory region (32 Mbytes)	READ/WRITE
0xB400_0000–0xB5FF_FFFF	WEIM CS4 memory region (32 Mbytes)	READ/WRITE
0xB600_0000–0xB7FF_FFFF	WEIM CS5 memory region (32 Mbytes)	READ/WRITE
NANDFC (NFC) Memory Space		
0xB800_0000–0xB800_0FFF	NAND Flash memory region[1] (4 Kbytes)	READ/WRITE
PCMCIA Memory Space		
0xC000_0000–0xC3FF_FFFF	PCMCIA memory region (64 Mbytes)	READ/WRITE

1 Can be used as a boot memory region.

17.2.2 Register Summary

Figure 17-2 shows the key to the register fields, and Table 17-3 shows the register figure conventions.

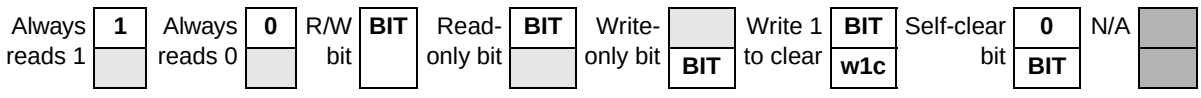


Figure 17-2. Key to Register Fields

Table 17-3. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 17-4 shows the M3IF register summary.

Table 17-4. M3IF Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xB800_3000 (M3IFCTL)	R	SDA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	MLSD_EN	MLSD			MRRP							
	W																
0xB800_3028 (M3IFSCFG0)	R	SWBA															
	W																
	R	SWBA					0	0	0	0	0	0	SWSZ				SE
	W																
0xB800_302C (M3IFSCFG1)	R	SSE0															
	W																
	R	SSE0															
	W																

Table 17-4. M3IF Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xB800_3030 (M3IFSCFG2)	R	SSE1															
	W																
	R	SSE1															
	W																
0xB800_3034 (M3IFSSR0)	R	SSS0															
	W																
	R	SSS0															
	W																
0xB800_3038 (M3IFSSR1)	R	SSS1															
	W																
	R	SSS1															
	W																
0xB800_3040 (M3IFMLWE0)	R	WEMA0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	MLWE0_EN	MLWE0		
	W																
0xB800_3048 (M3IFMLWE2)	R	WEMA2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	MLWE2_EN	MLWE2		
	W																
0xB800_3048 (M3IFMLWE2)	R	WEMA2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	MLWE2_EN	MLWE2		
	W																

Table 17-4. M3IF Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xB800_304C (M3IFMLWE3)	R	WEMA3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	MLWE3_EN	MLWE3		
	W																
0xB800_3050 (M3IFMLWE4)	R	WEMA4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	MLWE4_EN	MLWE4		
	W																
0xB800_3054 (M3IFMLWE5)	R	WEMA5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	MLWE5_EN	MLWE5		
	W																

17.2.3 Register Descriptions

This section contains the detailed register descriptions for the M3IF registers.

17.2.3.1 M3IF Control Register (M3IFCTL)

The M3IFCTL contains the access status of, provides the access control to SDRAM/MDDR memory devices and arbitration priority for M3IF port masters. The field assignments for this register are shown in Figure 17-3, and the field descriptions are listed in Table 17-5.

0xB800_3000 (M3IFCTL) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SDA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	MLSD_EN	MLSD			MRRP							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 17-3. M3IF Control Register

Table 17-5. M3IF Control Register Field Descriptions

Field	Description
31 SDA	SDRAM/MDDR Memory Active. This is a read-only status bit, that if set, indicates that an active/pending access to SDRAM/MDDR memory exists. The SDA bit will be set on one of the following conditions: • MLSD_EN cleared—any active/pending access to SDRAM/MDDR memory space will set the bit (until the access is completed). • MLSD_EN is set—any accesses to SDRAM/MDDR memory space initiated previously to MLSD_EN assertion, will keep the SDA status bit set. The bit will clear after all pending/active accesses execution is completed. Access from master number equal to MLSD field will not assert the status bit. **Note:** When MLSD_EN is set, any new accesses (initiated after MLSD_EN assertion) to SDRAM/MDDR not from MLSD master will be pending without setting SDA to 1. Only the SDRAM/MDDR memory space region will be lock to the MLSD port. Accesses to M3IF/ESDCTL registers are available to all masters in the system and its system/software responsibility not to access those registers during lock period. 0 No active/pending access to SDRAM/MDDR memory exists. 1 Indicates an active/pending access to SDRAM/MDDR memory exists.
30–12	Reserved
11 MLSD_EN	Master Lock SDRAM/MDDR Access. This bit enables the Master Control SDRAM/MDDR access (MLSD). The reset value of this bit is “0”. 0 Master Control SDRAM/MDDR access (MLSD) is disabled. 1 Master Control SDRAM/MDDR access (MLSD) is enabled.

Table 17-5. M3IF Control Register Field Descriptions (continued)

Field	Description
10–8 MLSD	Master Lock SDRAM/MDDR Access. This 3-bit field defines the master port number (MPG) that will be the only master in the system that will be served by the SDRAM/MDDR controller. All accesses toward the SDRAM/MDDR from the other masters will be postponed, until the MLSD master will clear MLSD_EN bit. The reset value of the MLSD is “0”. **Note:** Accesses to ESDCTL registers are not effected by the MLSD field. For example, they can be accessed by any master even if MLSD_EN is set. Prior to lock accesses, the MLSD master should perform the following steps: 1. Set the MLSD_EN bit and the MLSD field (with the desired value) in the M3IFCTL register. 2. Read M3IFCTL register and check: 3. SDA status bit is cleared (no pending/active accesses to SDRAM/MDDR memory space exists). 4. MLSD_EN bit is set. 5. MLSD (value) points to the required port number (master port number that requires lock accesses). 000 Master Port Gasket 0 001 Master Port Gasket 1 010 Master Port Gasket 2 011 Master Port Gasket 3 100 Master Port Gasket 4 101 Master Port Gasket 5 110 Master Port Gasket 6 111 Master Port Gasket 7
7–0 MRRP	Master Round Robin Priority. MRRP field is an 8-bit field with one bit per master (bit #i to master #i). Masters with their MRRP bit set are added to a priority arbitration “list” so that together they will have 50% probability to gain access through both M3A and MAB arbitration processes (50% probability for each one of the arbitration separately). Assertion of MRRP bit for an unused master is forbidden. If all MRRP bits are cleared the masters will have equal probability to pass the arbitration processes. 0 The respective master is not on the priority arbitration “list.” 1 Add the respective master to the priority arbitration “list” with a 50% probability to pass the arbitration processes.

17.2.3.2 M3IF Snooping Configuration Register 0 (M3IFSCFG0)

The M3IFSCFG0 register contains the Snooping window base address, the size of the snooping window and the snooping control bit fields, which are used by the M3IF to monitor the write access. The Snooping feature is described in detail in Section 17.3.1.1, “Snooping Overview.” The field assignments for this register are shown in Figure 17-4, and the field descriptions are listed in Table 17-6.

0xB800_3028 (M3IFSCFG0) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SWBA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SWBA					0	0	0	0	0	0	SWSZ				SE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 17-4. M3IF Snooping Configuration Register 0 (M3IFSCFG0)

Table 17-6. M3IF Snooping Configuration Register 0 Field Descriptions

Field	Description
31–11 SWBA	Snooping Window Base Address. This field defines the snooping window base address to be monitored by the M3IF. M3IF monitors write accesses to the memory region above the base address window. This address should be aligned on 2KB boundary.
10–5	Reserved
4–1 SWSZ	Snooping Window Size. This field define the snooping window size as described in Table 17-7.
0 SE	Snooping Enable. This bit enables snooping detection. The M3IF monitors and detects write accesses to the snooping window. 0 Snooping feature is disabled. 1 Snooping feature is enabled.

Table 17-7. SWSZ Field Descriptions

SWSZ	Snooping Window Size	Window Base Address Bits	Window Address Bits in Use
0000	2 Kbytes	[31:11]	[10:0]
0001	4 Kbytes	[31:12]	[11:0]
0010	8 Kbytes	[31:13]	[12:0]
0011	16 Kbytes	[31:14]	[13:0]
0100	32 Kbytes	[31:15]	[14:0]
0101	64 Kbytes	[31:16]	[15:0]
0110	128 Kbytes	[31:17]	[16:0]
0111	256 Kbytes	[31:18]	[17:0]
1000	512 Kbytes	[31:19]	[18:0]

Table 17-7. SWSZ Field Descriptions (continued)

SWSZ	Snooping Window Size	Window Base Address Bits	Window Address Bits in Use
1001	1 Mbytes	[31:20]	[19:0]
1010	2 Mbytes	[31:21]	[20:0]
1011	4 Mbytes	[31:22]	[21:0]
1100	8 Mbytes	[31:23]	[22:0]
1101	16 Mbytes	[31:24]	[23:0]
1110	Reserved	—	—
1111	Reserved	—	—

17.2.3.3 M3IF Snooping Configuration Register 1 (M3IFSCFG1)

The M3IFSCFG1 register contains the enable bits for the lower 32 segments [31:0] in M3IFSCFG0 register. The Snooping feature is described in detail in Section 17.3.1.1, “Snooping Overview.” The field assignments for this register are shown in Figure 17-5, and the field descriptions are listed in Table 17-8.

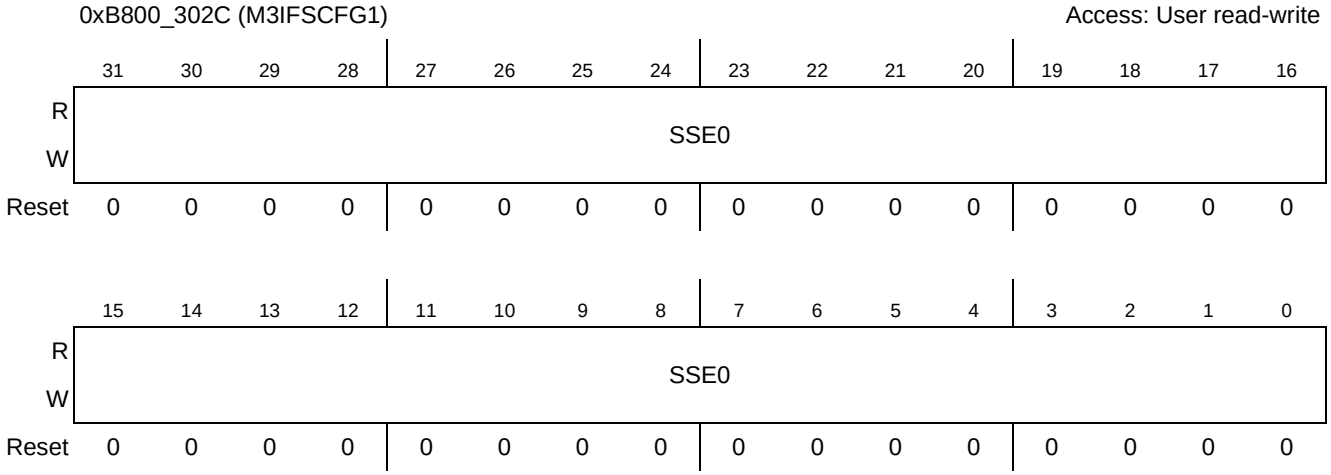


Figure 17-5. M3IF Snooping Configuration Register 1 (M3IFSCFG1)

Table 17-8. M3IF Snooping Configuration Register 1 Field Descriptions

Field	Description
31–0 SSE0	Snooping Segment Enable 0. This register contains the enable bits for the lower 32 segments [31:0] in the snooping window (defined by the M3IFSCFG0 register). If snooping is enabled for segment #x (respective SSE0 bit is high), than any write access detected to that segment will set the DMA_ACCESS for one cycle and the respective snooping status bit will be set. If the SSE0 bit is low, and a write access to the respective segment is detected by the M3IF, only the relevant status bit in the snooping status register will be set but the DMA_ACCESS will not be generated. 0 Snooping segment #x is disabled. 1 Snooping segment #x is enabled.

17.2.3.4 M3IF Snooping Configuration Register 2 (M3IFSCFG2)

M3IFSCFG2 register contains the enable bits for the upper 32 segments [63:32] in M3IFSCFG0 register.The Snooping feature is described in detail in Section 17.3.1.1, “Snooping Overview.” The field assignments for this register are shown in Figure 17-6, and the field descriptions are listed in Table 17-9.

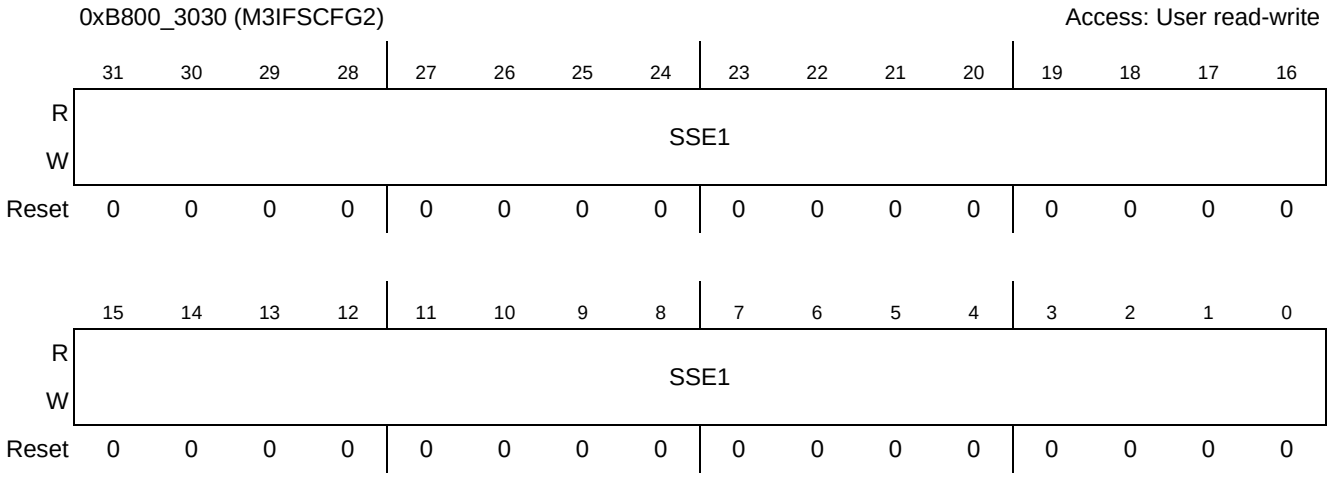


Figure 17-6. M3IF Snooping Configuration Register 2 (M3IFSCFG2)

Table 17-9. M3IF Snooping Configuration Register 2 Field Descriptions

Field	Description
31–0 SSE1	Snooping Segment Enable 1. This register contains the enable bits for the higher 32 segments [63:32] in the snooping window (defined by the M3IFSCFG1 register). If snooping is enabled for segment #x (respective SSE1 bit is high), than any write access detected to that segment will set the DMA_ACCESS for one cycle and the respective snooping status bit will be set. If the SSE1 bit is low, and a write access to the respective segment is detected by the M3IF, only the relevant status bit in the snooping status register will be set but the DMA_ACCESS will not be generated. 0 Snooping segment #x is disabled. 1 Snooping segment #x is enabled.

17.2.3.5 M3IF Snooping Status Register 0 (M3IFSSR0)

The M3IFSSR0 register contains the snooping status bits for the lower 32 segments. The Snooping feature is described in detail in Section 17.3.1.1, “Snooping Overview.” The field assignments for this register are shown in Figure 17-7, and the field descriptions are listed in Table 17-10.

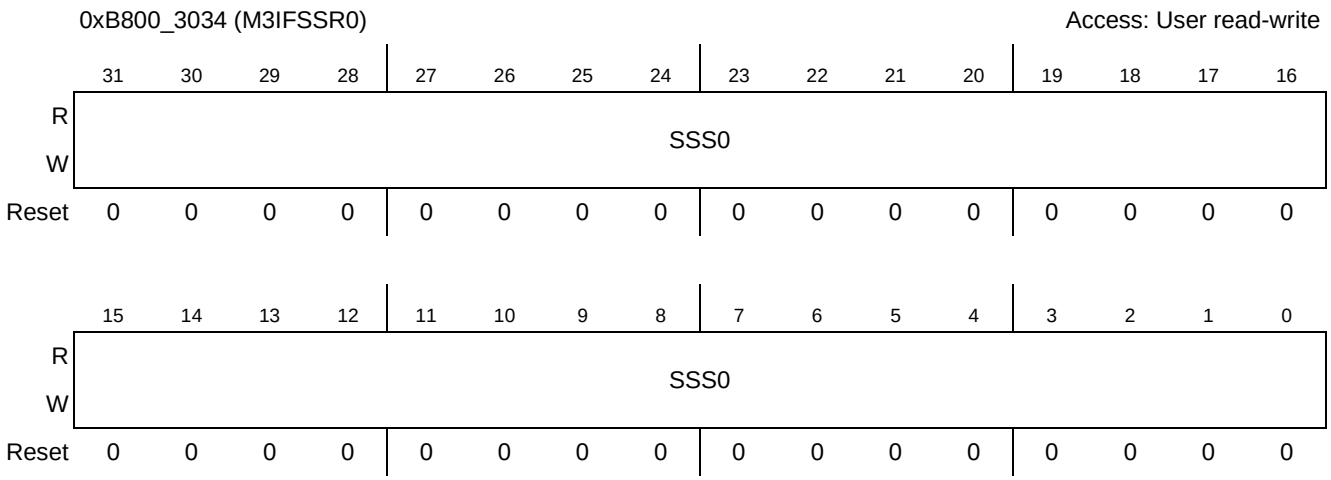


Figure 17-7. M3IF Snooping Status Register 0 (M3IFSSR0)

Table 17-10. M3IF Snooping Status Register 0 Field Descriptions

Field	Description
31–0 SSS0	Snooping Segment Status 0. This register contains the snooping status bits for the lower 32 segments [31:0] in the snooping window (defined by the M3IFSCFG0 register). A bit in the SSS0 register is asserted if snooping to the respective segment occurred. Write 0 to clear status bit. **Note:** If snooping occurred the status bit will be updated regardless of the respective snooping segment enable bit SSE0[x]. The DMA_ACCESS will be asserted only if the respective snooping segment enable bit SSE0[x] is enabled. 0 Snooping for segment #x did not occur. 1 Snooping for segment #x occurred.

17.2.3.6 M3IF Snooping Status Register 1 (M3IFSSR1)

The M3IFSSR1 register contains the snooping status bits for the lower 32 segments. The Snooping feature is described in detail in Section 17.3.1.1, “Snooping Overview.” The field assignments for this register are shown in Figure 17-8, and the field descriptions are listed in Table 17-11.

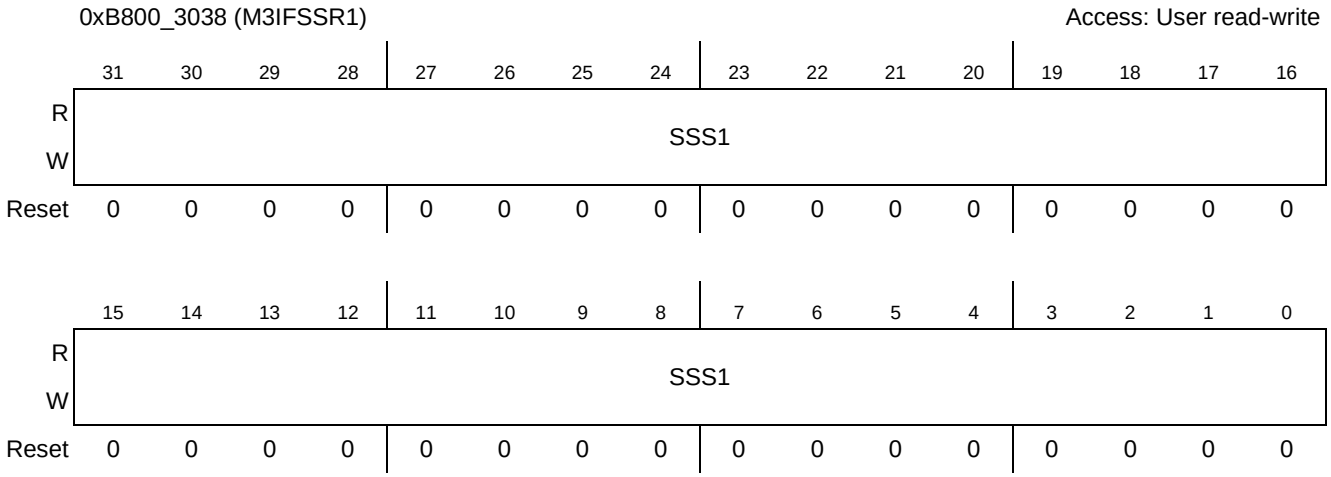


Figure 17-8. M3IF Snooping Status Register 1 (M3IFSSR1)

Table 17-11. M3IF Snooping Status Register 0 Field Descriptions

Field	Description
31–0 SSS1	SSS1—Snooping Segment Status 1. This register contains the snooping status bits for the higher 32 segments [63:32] in the snooping window (defined by the M3IFSCFG1 register). A bit in the SSS1 register is asserted if snooping to the respective segment occurred. Write 0 to clear status bit. **Note:** If snooping occurred the status bit will be updated regardless of the respective snooping segment enable bit SSE0[x]. The DMA_ACCESS will be asserted only if the respective snooping segment enable bit SSE1[x] is enabled. 0 Snooping for segment #x did not occur. 1 Snooping for segment #x has occurred.

17.2.3.7 M3IF Master Lock WEIM CSx Register (M3IFMLWEx)

The field assignments for this register are shown in Figure 17-9, and the field descriptions are listed in Table 17-12.

0xB800_3040 (M3IFMLWE0)
0xB800_3044 (M3IFMLWE1)
0xB800_3048 (M3IFMLWE2)
0xB800_304C (M3IFMLWE3)
0xB800_3050 (M3IFMLWE4)
0xB800_3054 (M3IFMLWE5)

Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	WEMAx	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0									MLGE_EN	MLGE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 17-9. M3IF Lock General Register (M3IFMLGE)

Table 17-12. M3IF Lock General Register Field Descriptions

Field	Description
31 WEMAx	WEIM CSx (0-5) Memory Active. This is a read-only status bit, that if set indicates that an active/pending access to a WEIM CSx memory exists. The WEIMx bit is set on one of the following conditions: • MLWEx_EN cleared—any active/pending access to WEIM CSx memory space will set the bit (until the access is completed). • MLWEx_EN is set—any accesses to WEIM CSx memory space initiated previously to MLWEx_EN assertion, will keep the WEMAx status bit set. The bit clears after all pending/active accesses execution is completed. Access from master number equal to MLWEx field does not assert the status bit. **Note:** When MLWEx_EN is set, any new accesses (initiated after MLWEx_EN assertion) to WEIM CSx memories (or to M3IFMLWEx register**)** not from MLWEx master will be pending without setting WEMAx to 1. Both the M3IFMLWEx register and the WEIM CSx space region will be lock to the MLWEx port. 0 No active/pending access to WEIM CSx memory exists. 1 Indicates an active/pending access to WEIM CSx memory exists.
30–4	Reserved

Table 17-12. M3IF Lock General Register Field Descriptions (continued)

Field	Description
3 MLWEx_EN	Master Lock WEIM CSx Access Enable. This bit enables the Master Lock WEIM CSx access (MLWEx). The reset value of this bit is 0. **Note:** After MLWEx master does not need the lock any more, the master should clear MLWEx_EN bit, so WEIM CSx memory region is open to all masters. 0 Master Lock WEIM CSx access (MLWEx) disabled. 1 Master Lock WEIM CSx access (MLWEx) enabled
2–0 MLWEx	MLWEx—Master Lock WEIM CSxl Access. This 3 bits field defines the master port number (MPG) that will be the only one in the system served by the WEIM controller. All accesses to the WEiM CSx memory space from the other masters will be postponed. The reset value of the MLWEx is 0. 1. Prior to lock accesses, the MLGE master should perform the following steps: 2. Set the MLWEx_EN bit and the MLWEx field (with the desired value) in the M3IFMLWEx register. 3. Read M3IFMLWEx register and check: WEMAx status bit is cleared (no pending/active accesses to WEIM CSx memory space exists). MLWEx_EN bit is set. MLWEx (value) points to the required port number (master port number that requires lock accesses). 000 Master Port Gasket 0 001 Master Port Gasket 1 010 Master Port Gasket 2 011 Master Port Gasket 3 100 Master Port Gasket 4 101 Master Port Gasket 5 110 Master Port Gasket 6 111 Master Port Gasket 7

17.3 Functional Description

This section provides the functional description for the M3IF module.

17.3.1 Snooping Logic

17.3.1.1 Snooping Overview

The M3IF snooping feature (only used by the Image Processor Unit, IPU), monitors and detects write accesses to a configurable window (snooping window) in one of the memory regions mapped by the M3IF. The snooping window base address, memory region, and window size are configurable parameters through the M3IFSCFG0 register (see register details at Section 17.2.3.2, “M3IF Snooping Configuration Register 0 (M3IFSCFG0)”). The snooping window is further divided into 64 equally sized segments.

A detected write access to the snooping window results in:

- The respective segment status bit in M3IFSSR0 and/or M3IFSSR1 registers is set.
- DMA_ACCESS strobe is asserted for 1 clock cycle if the snooping segment enable bit is set for the snooped segment. The snooping segment enable bit is configured via 2 snooping configuration registers, M3IFSCFG1 and M3IFSCFG2.
- IPU DMA access remains asserted as long as the write access that has triggered the snooping logic.

It is the software's responsibility to clear the snooped segment status bits, but the snooped segment status bit will be set for each snooping detection regardless of its value.

17.4 Initialization/Application Information

17.4.1 M3IF in a System

This section provides an example of M3IF initialization, integration and configuration in a given system.The system requirements are listed below:

- Several masters with external memories access capabilities.
  - — Several masters access the M3IF via AP MAX crossbar switch.
  - — ARM I-Cache—32-bit data bus
  - — ARM D-Cache—32-bit data bus
  - — Layer 2 Cache—64 bit data bus
  - — IPU—32-bit data bus with the following snooping requirements
    - – 512Kbyte snooping window in SDRAM CSD1 memory region
    - – Snooping window base address 0x9000_F000
    - – Enable snooping for segments: 3, 7, 12-27, 32, 36-56
- The system uses 2 SDRAM memory devices (32 bits), a 16-bit Flash (via WEIM CS0) and one SRAM (via WEIM CS1).

17.4.1.1 Snooping Window Settings

The following M3IF settings are needed to implement the system snooping requirements.

- Write 0x9000_F011 to M3IFSCFG0 register to set the snooping window base address and size.
- Write 0x0FFF_F088 to M3IFSCFG1 register to enable snooping for the specified lower segments.
- Write 0x01FF_FFF1 to M3IFSCFG2 register to enable snooping for the specified higher segments.

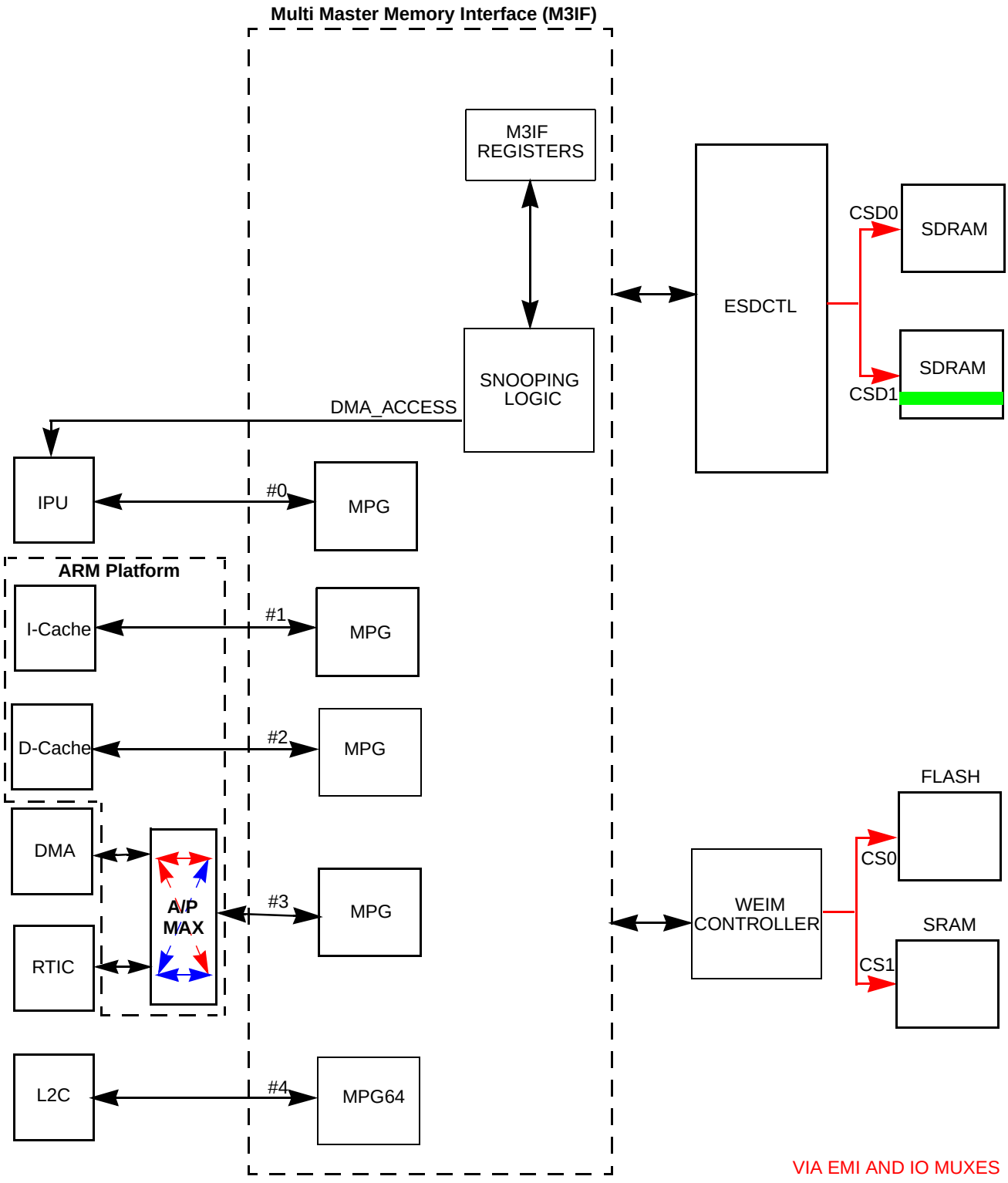


Figure 17-10. M3IF Integration

17.4.1.2 Snooping Window Settings

The following M3IF settings are needed to implement the system snooping requirements.

- Write 0x9000_F011 to M3IFSCFG0 register to set the snooping window base address and size.
- Write 0x0FFF_F088 to M3IFSCFG1 register to enable snooping for the specified lower segments.
- Write 0x01FF_FFF1 to M3IFSCFG2 register to enable snooping for the specified higher segments.

Chapter 18
Wireless External Interface Module (WEIM)

The Wireless External Interface Module (WEIM) provides the capability to the system for accessing external Flash and RAM memories connected to either of its six chip selects. It has the ability to provide a single-cycle burst access for external flashes and support for multiple burst devices when not using its smart burst feature. Other features include Big/Endian mode support, CellularRAM support, and support for multiplexed address/data bus. Figure 18-1 shows a top-level WEIM block diagram. All block signals are shown in Table 18-1 and are described in Table 18-2.

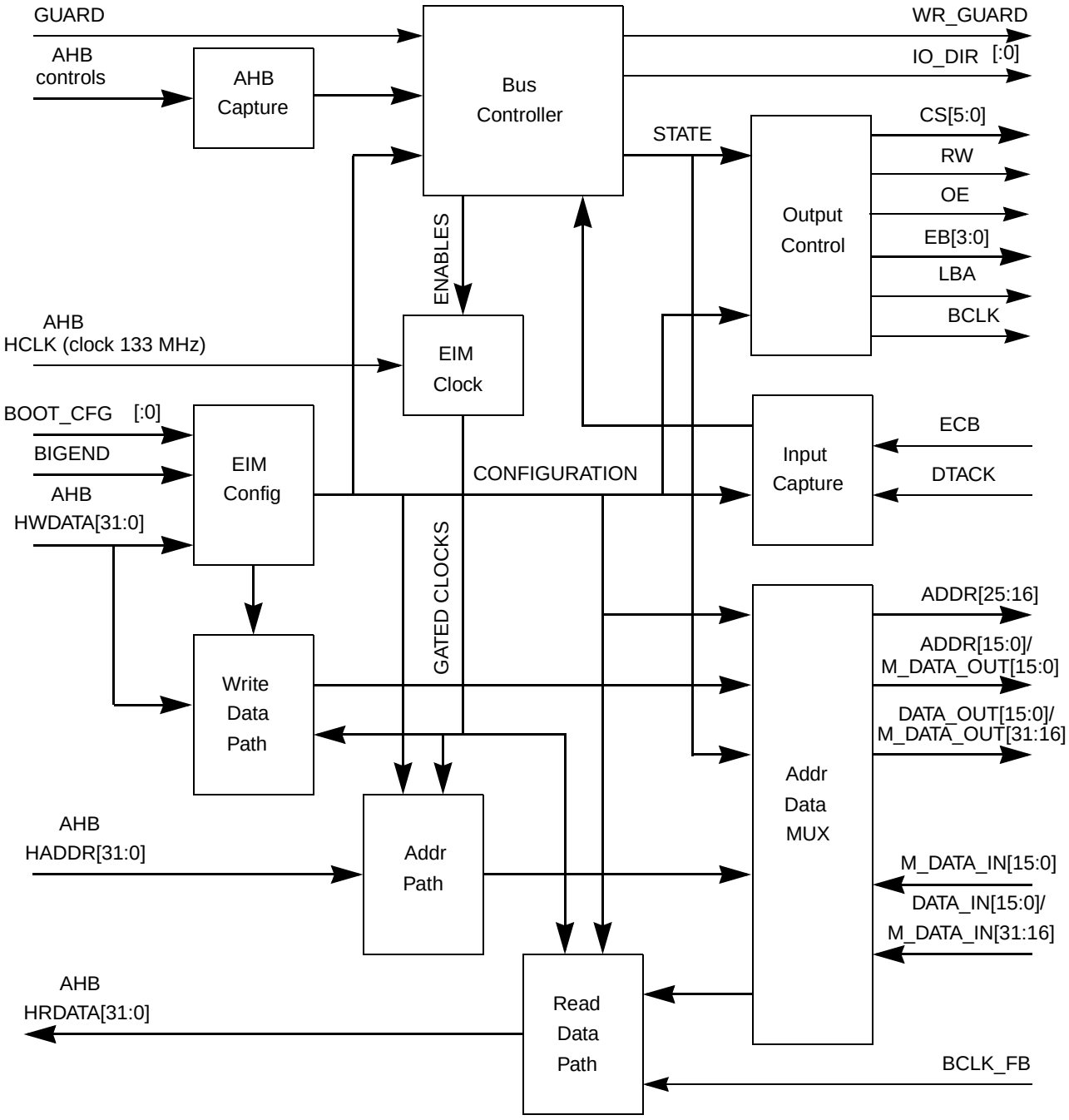


Figure 18-1. WEIM Block Diagram

18.1 Overview

The WEIM handles the interface to devices external to the chip, including generation of chip selects, clock and control for external peripherals and memory. It provides asynchronous and synchronous access to devices with SRAM-like interface.

18.1.1 Features

The WEIM includes the following features:

- Six Chip Selects for external devices, with $\overline{CS}0$ and $\overline{CS}1$ each covering a range of 128 Mbytes, and $\overline{CS}2$–$\overline{CS}5$, each covering a range of 32 Mbytes
- $\overline{CS}0$ range can be increased to 256 Mbytes when collapsed with $\overline{CS}1$
- Selectable Protection for each Chip Select
- Programmable Data Port Size for each Chip Select
- Asynchronous accesses with programmable setup and hold times for control signals
- Synchronous Memory Burst Read Mode support for AMD, Intel, and Micron burst flash memory
- Synchronous Memory Burst Write Mode support for PSRAM (CellularRAM™ from Micron, Infineon, and Cypress)
- Support for multiplexed address/data bus operation
- External cycle termination/postpone with $\overline{DTACK}$ signal
- Programmable Wait-State generator for each Chip Select
- Support for Big Endian and Little Endian modes of operation per access
- ARM AHB slave interface

18.1.2 Modes of Operation

The WEIM has six modes of operation. The WEIM does not require a dedicated low-power mode because most of the clocks are gated anytime there are no accesses to the WEIM providing maximum energy conservation.

- Asynchronous mode. This is memory non-burst mode is used for SRAM access. In this mode a single data is read/written with each access (asserted address). All output signal timing is controlled by selecting preset values in Chip Select control registers.
- Synchronous read mode. This is memory burst mode is used for reading Flash memory devices. In this mode after address assertion a burst of sequential data can be read. Data exchange is carried out according to BCLK clock been generated by WEIM. After first word of data access may be delayed by external $\overline{ECB}$ signal assertion.
- Page mode.This mode is used for memory burst accesses, but the address is asserted for each data in the burst as $\overline{LBA}$ and BCLK operate Asynchronously. In this mode the setup time is greater for the first data burst than for the remaining data in the burst. (in the same page).
- Synchronous write mode. In this mode synchronous write is permitted and access may be additionally delayed according to $\overline{ECB}$ state before first piece of data arrived (refresh wait enable).
- DTACK mode. This is memory non-burst mode is used for PCMCIA accesses. In this mode WEIM waits for $\overline{DTACK}$ acknowledge until 1024 counts of the AHB clock have passed. In this mode $\overline{DTACK}$ can be used as posedge or level sensitive according to WSC field or EW bit settings.
- Multiplexed Address/Data mode. In this mode multiplexing addresses and data bits on the same pins is supported for synchronous/asynchronous accesses to 32-bit data width memory devices.

18.2 External Signal Description

18.2.1 Overview

This section discusses input and output signals (see Table 18-1) between the WEIM and the external devices.

Table 18-1. WEIM Signal Properties

Name	Port	Function	Direction	Reset State
ADDR[25:16]	—	Address Bus MSB/$\overline{EB}$[3:2], CRE	Out	Low
ADDR[15:0]/ M_DATA_OUT[15:0]	—	Address Bus LSB/Output Data Multiplexed Bus LSB	Out	Low
BCLK	—	Burst Clock	Out	Low
BCLK_FB	—	Feedback Burst Clock	In	—
BIGEND	—	Data Endian	In	—
BOOT_CFG[:0]	—	Boot Configuration	In	—
$\overline{CS}$[5:0]	—	Chip Selects	Out	High
DATA_IN[15:0]/ M_DATA_IN[31:16]	—	Input Data Bus LSB/Input Data Multiplexed Bus MSB	In	—
DATA_OUT[15:0]/ M_DATA_OUT[31:16]	—	Output Data Bus LSB/Output Data Multiplexed Bus MSB	Out	Low
$\overline{DTACK}$	—	Data Transfer Acknowledge/Wait	In	—
$\overline{EB}$[3:0]	—	Enable Byte	Out	High
$\overline{ECB}$	—	End Current Burst/Wait	In	—
GUARD	—	Input Guard	In	—
IO_DIR[:0]	—	IO Direction	Out	Low
$\overline{LBA}$	—	Load Burst Address	Out	High
M_DATA_IN[15:0]	—	Input Data Multiplexed Bus LSB	In	—
$\overline{OE}$	—	Output Enable	Out	High
$\overline{RW}$	—	Read/Write	Out	High
WR_GUARD	—	Write Guard	Out	High

18.3 Detailed Signal Descriptions

Table 18-2 gives a detailed description of the WEIM signals.

Table 18-2. WEIM Detailed Signal Descriptions

Signal	I/O	Description
$\overline{\text{ADDR[25:16]}}$	O	Address Bus MSB. These pins are used as address bits [25:16]. In the multiplexed mode these pins do not change their state. These pins represent [27:18] AHB address bits for word-width memory, [26:17] bits for halfword width memory and [25:16] bits for byte-width memory. ADDR[25:24] also are used as $\overline{\text{EB}}$[3:2] in the muxed mode (MUM=1). ADDR[23] also is used as CRE in PSRAM mode (PSR=1).
$\overline{\text{ADDR[15:0]}}$/ $\overline{\text{M_DATA_OUT[15:0]}}$	O	Multiplexed Address Bus LSB/Output Data Bus LSB. In non-multiplexed mode those pins are used as address bits [15:0]. hose pins reflect [17:2] AHB address bits for word width memory, [16:1] bits for halfword width memory and [15:0] bits for byte width memory. In multiplexed Address/Data mode those bits are multiplexed between address and output data [15:0] bits. In the multiplexed Address/Data mode its behavior is affected by the LBA, LBN and LAH bits in the WEIM control registers. In synchronous multiplexed mode its behavior is affected by the BCS, BCD and LAH bits in the WEIM control registers.
$\overline{\text{BCLK}}$	O	Burst Clock. This active-high output signal BCLK is used to clock external, burst-capable devices to synchronize the loading and incrementing of addresses and delivery of burst read and write data to/from the WEIM. Its behavior is affected by the BCM bit in the WEIM configuration register and the SYNC, BCD and BCS bits in the WEIM control registers. BCLK can start on both rising and falling edge of HCLK.
BCLK_FB	I	Burst Clock Feedback. This input is used to provide input data sampling clock in high data speed. It is a feedback from the IO PAD of the BCLK output pin that intend to align the clock used by the memory, and the one that is used to sample the read data.
BIGEND	I	Big/Little Endian. This input is used to provide Big/Little Endian support in the WEIM. WEIM supports Big and Little Endian accesses according Table 18-1. WEIM supports mixing Big and Little Endian accesses.
BOOT_CFG[:0]	O	Boot Configuration. These input pins determine the state of WEIM configuration bits after hardware reset. See Table 18-3 for settings.
$\overline{\text{CS}}$[5:0]	O	Chip Selects. The CS[5:0] signals are chip selects active-low output pins. Its behavior is affected by the CSA and CSN bits in the WEIM control registers. CS[0] address space range can be increased to 256 Mbytes by merging the CS[0] and CS[1] ranges. In this case the CS[1] pin is used as address line A26, and the Merged Address Space (MAS) status will be set in the WEIM Configuration register. As shown in Table 18-4, the chip select signals are asserted based on a decode of address lines [31:24] (when enabled).
DATA_IN[15:0]/M_DATA_IN[31:16]	*IO*	Input Data Bus LSB/Input Data Multiplexed Bus MSB. These signals are the input data bus used to transfer data from an external devices to the processor. In a non multiplexed mode it is LSB, in a multiplexed mode it is MSB.

Table 18-2. WEIM Detailed Signal Descriptions (continued)

Signal	I/O	Description
DATA_OUT[15:0]/M_DATA_OUT[31:16]	O	Output Data Bus LSB/Output Data Multiplexed Bus MSB. These signals are the output data bus used to transfer data from the processor to an external devices. In a non-multiplexed mode it is LSB, in a multiplexed mode it is MSB.
$\overline{\text{DTACK}}$	I	Data Transfer Acknowledge. This input signal is used to externally terminate a data transfer when enabled. For $\overline{\text{DTACK}}$ enabled cycles, the bus time-out monitor generates a bus error if $\overline{\text{DTACK}}$ after has been asserted is not deasserted before 1024 clocks have elapsed. This signal is used in two modes: with rising edge detection or with level detection with an insensitiveness time. Edge detection mode is used for devices like PCMCI card. Level detection mode is used for asynchronous devices like ATI graphic controller. $\overline{\text{DTACK}}$ control keeps backward compatibility with previous architectures that used it in the applications processors.
$\overline{\text{EB[3:0]}}$	O	Enable Byte. Those active-low output pins indicate active data bytes for the current access. They may be configured to assert for read and write cycles or for write cycles only as programmed in the CS control registers. $\overline{\text{EB[0]}}$ corresponds to DATA_OUT[7:0] and M_DATA_OUT[7:0]. $\overline{\text{EB[1]}}$ corresponds to DATA_OUT[15:8] and M_DATA_OUT[15:8]. $\overline{\text{EB[2]}}$ corresponds to M_DATA_OUT[23:16]. $\overline{\text{EB[3]}}$ corresponds to M_DATA_OUT[31:24]. $\overline{\text{EB[3:2]}}$ also are muxed to the ADDR[25:24] bits in the muxed mode (MUM = 1). In the write accesses its behavior is affected by the EBWA, EBWN and EBC bits in the WEIM control registers. In the read accesses its behavior is affected by the EBRA and EBRN bits in the WEIM control registers.
$\overline{\text{ECB}}$	I	End Current Burst (WAIT). This active-low input signal $\overline{\text{ECB}}$ is asserted by external burst capable devices. It is serviced in synchronous mode only (SYNC=1). This signal can be used in two different modes depending on the EW bit in the Chip Select Control Register. In the ECB mode (EW=0) $\overline{\text{ECB}}$ indicates the end of the current (continuous) burst sequence. Following assertion, the WEIM terminates the current burst sequence and initiate a new one. In the WAIT mode (EW=1) the memory device asserts this signal to insert wait states during refresh collisions or during a row boundary crossing. Following assertion, the WEIM does not terminate the current burst sequence and continues it once WAIT is negated. $\overline{\text{ECB}}$ will have a pull up resistor in IO. For burst devices ECB/WAIT output should be configured to change one cycle before data is ready (before delay).
GUARD	I	GUARD. This active-high input signal indicates that IO is locked by another module and current access should be postponed till IO will be unlocked. GUARD and WR_GUARD together are used in back to back accesses between memory controllers to avoid contention on the shared pins.
IO_DIR[:0]	O	IO_DIR[:0]. These active-high output signal indicates IO direction(0 for input and “1” for output). Bidirectional buses are made in IO module from the ADDR[15:0]/M_DATA_OUT[15:0] and M_DATA_IN[15:0] from the DATA_OUT[15:0]/M_DATA_OUT[31:16] and DATA_IN[15:0] / M_DATA_IN[31:16]. Bit IO_DIR[0] corresponds to ADDR[7:0] / M_DATA_OUT[7:0]/M_DATA_IN[7:0], bit IO_DIR[1] corresponds to ADDR[15:8] / M_DATA_OUT[15:8]/M_DATA_IN[15:8], bit IO_DIR[2] corresponds to DATA_OUT[7:0]/M_DATA_OUT[23:16] / DATA_IN[7:0]/M_DATA_IN[23:16], bit IO_DIR[3] corresponds to DATA_OUT[15:8]/M_DATA_OUT[31:24] / DATA_IN[15:8]/M_DATA_IN[31:24]
$\overline{\text{LBA}}$	O	Load Burst Address. This active-low output signal is asserted during burst mode accesses to cause the external burst capable device to load a new starting burst address. Assertion of $\overline{\text{LBA}}$ indicates that a valid address is present on the address bus. Its behavior is affected by the SYNC, BCD, BCS, LBA and LBN bits in the WEIM control registers.

Table 18-2. WEIM Detailed Signal Descriptions (continued)

Signal	I/O	Description
M_DATA_IN[15:0]	I	MUX Data Input Bus. These signals are LSB input data bus used to transfer data from an external devices to the processor in multiplexed mode.
$\overline{OE}$	O	Output Enable. This active-low output signal $\overline{OE}$ indicates the bus access is a read and enables slave devices to drive the data bus with read data. Its behavior is affected by the OEA and OEN bits in the WEIM control registers.
$\overline{RW}$	O	Read/Write. The $\overline{RW}$ output signal indicates if the current bus access is a read or write cycle. A high (logic one) level indicates a read cycle and a low (logic zero) level indicates a write cycle. Its behavior is affected by the RWA and RWN bits in the WEIM control registers.
WR_GUARD	I	WR_GUARD. This active-high output signal indicates that IO is locked by WEIM. (It is asserted also in Extra Dead Cycles). WR_GUARD and GUARD together are used in back to back accesses between memory controllers to avoid contention on the shared pins.

Table 18-3. Boot Configuration Settings

BOOT_CFG Bits	Configured Bits	Place
	MAS	WCR
:0	DSZ[:0]	CSCR0U

Table 18-4. WEIM Out/in Data in Case AHB Out/in Data is 0xB3B2B1B0

Endian Mode	AHB Access	AHB Address [1:0]	Port Size and Used Bits								
			Word Port				Halfword Port			Byte Port	
			[31:24]	[23:16]	[15:8]	[7:0]	External Address [0]	[31:24] ([15:8])	[23:16] ([7:0])	External address [1:0]	[31:24] ([15:8], [23:16], [7:0])
Big	Word	0	0xB3	0xB2	0xB1	0xB0	0	0xB3	0xB2	0	0xB3
										1	0xB2
							1	0xB1	0xB0	2	0xB1
										3	0xB0
	Halfword	0	0xB3	0xB2			0	0xB3	0xB2	0	0xB3
										1	0xB2
		2			0xB1	0xB0	1	0xB1	0xB0	2	0xB1
										3	0xB0
	Byte	0	0xB3				0	0xB3		0	0xB3
		1		0xB2					0xB2	1	0xB2
		2			0xB1		1	0xB1		2	0xB1
		3				0xB0			0xB0	3	0xB0

Table 18-4. WEIM Out/in Data in Case AHB Out/in Data is 0xB3B2B1B0 (continued)

Endian Mode	AHB Access	AHB Address [1:0]	Port Size and Used Bits								
			Word Port				Halfword Port			Byte Port	
			[31:24]	[23:16]	[15:8]	[7:0]	External Address [0]	[31:24] ([15:8])	[23:16] ([7:0])	External address [1:0]	[31:24] ([15:8], [23:16], [7:0])
Little	Word	0	0xB3	0xB2	0xB1	0xB0	0	0xB1	0xB0	0	0xB0
										1	0xB1
							1	0xB3	0xB2	2	0xB2
										3	0xB3
	Halfword	0			0xB1	0xB0	0	0xB1	0xB0	0	0xB0
										1	0xB1
		2	0xB3	0xB2			1	0xB3	0xB2	2	0xB2
										3	0xB3
	Byte	0				0xB0	0		0xB0	0	0xB0
		1			0xB1			0xB1		1	0xB1
		2		0xB2			1		0xB2	2	0xB2
		3	0xB3					0xB3		3	0xB3

18.4 Memory Map and Register Definition

The WEIM module includes 19 user-accessible 32-bit registers. There is a common register called the WEIM Configuration Register (WCR) that contains control bits that configure the WEIM for certain operation modes. The other 18 registers named Chip Select Control Register 0–5 Upper, Lower and Additional correspondingly (CSCR0U, CRCR0L, CRCR0A, . . . , CSCR5U, CSCR5L, CSCR5A) and compose 6 groups Chip Select Control Register 0–5 (CSCR0–CSCR5) of control registers for each chip select. The layout of the control register is slightly different for the CSCR0 register output because CSCR0 reset state depends on BOOT_CFG input. These registers are accessible only in supervisor mode with word (32-bit) reads and writes.

Complete decoding is not performed, so shadowing can occur with these registers. The user should not attempt to address these registers at any other address location other than those listed in Table 18-5.

18.4.1 Memory Map

The memory map for the WEIM is shown in Table 18-5, and the memory map for the Chip Select memory ranges is shown in Table 18-6.

Table 18-5. WEIM Memory Map

Address	Use	Access	Reset	Section/Page
0xB800_2000 (CSCR0U)	Chip Select 0 Upper Control Register	R/W	0x0000_0100	18.4.3.1/18-13
0xB800_2004 (CSCR0L)	Chip Select 0 Lower Control Register	R/W	0x0000_0801	18.4.3.2/18-17
0xB800_2008 (CSCR0A)	Chip Select 0 Addition Control Register	R/W	0x0000_8000	18.4.3.3/18-21
0xB800_2010 (CSCR1U)	Chip Select 1 Upper Control Register	R/W	0x0000_0000	18.4.3.1/18-13
0xB800_2014 (CSCR1L)	Chip Select 1 Lower Control Register	R/W	0x0000_0000	18.4.3.2/18-17
0xB800_2018 (CSCR1A)	Chip Select 1 Addition Control Register	R/W	0x0000_0000	18.4.3.3/18-21
0xB800_2020 (CSCR2U)	Chip Select 2 Upper Control Register	R/W	0x0000_0000	18.4.3.1/18-13
0xB800_2024 (CSCR2L)	Chip Select 2 Lower Control Register	R/W	0x0000_0000	18.4.3.2/18-17
0xB800_2028 (CSCR2A)	Chip Select 2 Addition Control Register	R/W	0x0000_0000	18.4.3.3/18-21
0xB800_2030 (CSCR3U)	Chip Select 3 Upper Control Register	R/W	0x0000_0000	18.4.3.1/18-13
0xB800_2034 (CSCR3L)	Chip Select 3 Lower Control Register	R/W	0x0000_0000	18.4.3.2/18-17
0xB800_2038 (CSCR3A)	Chip Select 3 Addition Control Register	R/W	0x0000_0000	18.4.3.3/18-21
0xB800_2040 (CSCR4U)	Chip Select 4 Upper Control Register	R/W	0x0000_0000	18.4.3.1/18-13
0xB800_2044 (CSCR4L)	Chip Select 4 Lower Control Register	R/W	0x0000_0000	18.4.3.2/18-17
0xB800_2048 (CSCR4A)	Chip Select 4 Addition Control Register	R/W	0x0000_0000	18.4.3.3/18-21
0xB800_2050 (CSCR5U)	Chip Select 5 Upper Control Register	R/W	0x0000_0000	18.4.3.1/18-13
0xB800_2054 (CSCR5L)	Chip Select 5 Lower Control Register	R/W	0x0000_0000	18.4.3.2/18-17
0xB800_2058 (CSCR5A)	Chip Select 5 Addition Control Register	R/W	0x0000_0000	18.4.3.3/18-21
0xB800_2060 (WCR)	WEIM Configuration Register (WCR)	R/W	0x0000_0100	18.4.3.4/18-24

Table 18-6. WEIM Chip Selection Memory Map

Address	Use	Access
0xA0000000 ... 0xA7FF_FFFF	$\overline{CS}$0 memory region	R/W
0xA8000000 ... 0xAFFF_FFFF	$\overline{CS}$1 memory region	R/W
0xB0000000 ... 0xB1FF_FFFF	$\overline{CS}$2 memory region	R/W
0xB2000000 ... 0xB3FF_FFFF	$\overline{CS}$3 memory region	R/W
0xB4000000 ... 0xB5FF_FFFF	$\overline{CS}$4 memory region	R/W
0xB6000000 ... 0xB7FF_FFFF	$\overline{CS}$5 memory region	R/W

18.4.2 Register Summary

Figure 18-2 shows the key to the register fields, and Table 18-7 shows the register figure conventions.

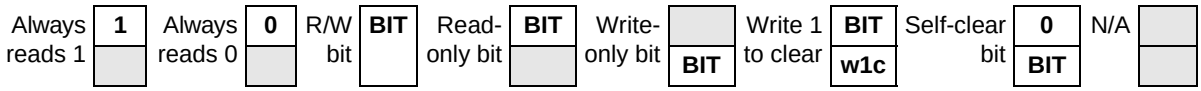


Figure 18-2. Key to Register Fields

Table 18-7. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
R	Read-only. Writing this bit has no effect.
W	Write-only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

18.4.3 Register Descriptions

This section consists of register descriptions in address order. Each description includes a standard register diagram with an associated figure number. Details of register bits and field function follow the register diagrams, in bit order. Table 18-8 provides a summary of the registers in the WEIM module.

Table 18-8. WEIM Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xB800_2000 (CSCR0U)	R	SP	WP	BCD		BCS				PSZ		PME	SYNC	DOL			
	W																
	R	CNC		WSC						EW	WWS			EDC			
	W																
0xB800_2004 (CSCR0L)	R	OEA				OEN				EBWA				EBWN			
	W																
	R	CSA				EBC	DSZ			CSN				PSR	CRE	WRAP	CSEN
	W																
0xB800_2008 (CSCR0A)	R	EBRA				EBRN				RWA				RWN			
	W																
	R	MUM	LAH			LBN		LBA		DWW		DCT		WWU	AGE	CNC2	FCE
	W																
0xB800_2010 (CSCR1U)	R	SP	WP	BCD		BCS				PSZ		PME	SYNC	DOL			
	W																
	R	CNC		WSC						EW	WWS			EDC			
	W																
0xB800_2014 (CSCR1L)	R	OEA				OEN				EBWA				EBWN			
	W																
	R	CSA				EBC	DSZ			CSN				PSR	CRE	WRAP	CSEN
	W																
0xB800_2018 (CSCR1A)	R	EBRA				EBRN				RWA				RWN			
	W																
	R	MUM	LAH			LBN		LBA		DWW		DCT		WWU	AGE	CNC2	FCE
	W																
0xB800_2020 (CSCR2U)	R	SP	WP	BCD		BCS				PSZ		PME	SYNC	DOL			
	W																
	R	CNC		WSC						EW	WWS			EDC			
	W																

Table 18-8. WEIM Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xB800_2024 (CSCR2L)	R	OEA				OEN				EBWA				EBWN			
	W																
	R	CSA				EBC	DSZ			CSN				PSR	CRE	WRAP	CSEN
	W																
0xB800_2028 (CSCR2A)	R	EBRA				EBRN				RWA				RWN			
	W																
	R	MUM	LAH			LBN			LBA	DWW		DCT		WWU	AGE	CNC2	FCE
	W																
0xB800_2030 (CSCR3U)	R	SP	WP	BCD		BCS				PSZ		PME	SYNC	DOL			
	W																
	R	CNC		WSC						EW	WWS			EDC			
	W																
0xB800_2034 (CSCR3L)	R	OEA				OEN				EBWA				EBWN			
	W																
	R	CSA				EBC	DSZ			CSN				PSR	CRE	WRAP	CSEN
	W																
0xB800_2038 (CSCR3A)	R	EBRA				EBRN				RWA				RWN			
	W																
	R	MUM	LAH			LBN			LBA	DWW		DCT		WWU	AGE	CNC2	FCE
	W																
0xB800_2040 (CSCR4U)	R	SP	WP	BCD		BCS				PSZ		PME	SYNC	DOL			
	W																
	R	CNC		WSC						EW	WWS			EDC			
	W																
0xB800_2044 (CSCR4L)	R	OEA				OEN				EBWA				EBWN			
	W																
	R	CSA				EBC	DSZ			CSN				PSR	CRE	WRAP	CSEN
	W																
0xB800_2048 (CSCR4A)	R	EBRA				EBRN				RWA				RWN			
	W																
	R	MUM	LAH			LBN			LBA	DWW		DCT		WWU	AGE	CNC2	FCE
	W																

Table 18-8. WEIM Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xB800_2050 (CSCR5U)	R	SP	WP	BCD		BCS				PSZ		PME	SYNC	DOL			
	W																
	R	CNC		WSC						EW	WWS			EDC			
	W																
0xB800_2054 (CSCR5L)	R	OEA				OEN				EBWA				EBWN			
	W																
	R	CSA				EBC	DSZ			CSN				PSR	CRE	WRAP	CSEN
	W																
0xB800_2058 (CSCR5A)	R	EBRA				EBRN				RWA				RWN			
	W																
	R	MUM	LAH			LBN			LBA	DWW		DCT		WWU	AGE	CNC2	FCE
	W																

18.4.3.1 Chip Select x Upper Control Register (CSCRxU)

The 96 bits used to control Chip Select are divided into three registers: Chip Select x Upper Control Register (CSCRxU), Chip Select x Lower Control Register (CSCRxL) and Chip Select x Additional Control Register (CSCRxA). Table 18-9 shows the register's field descriptions.

- Bits [95:64] are located in Chip Select x Upper Control Register (see Figure 18-3).
- Bits [63:32] are located in Chip Select x Lower Control Register (see Figure 18-4).
- Bits [31:0] are located in Chip Select x Additional Control Register (see Figure 18-5).

0xB800_2000 (CSCR0U)
0xB800_2010 (CSCR1U)
0xB800_2020 (CSCR2U)
0xB800_2030 (CSCR3U)
0xB800_2040 (CSCR4U)
0xB800_2050 (CSCR5U)

Access: User Read/Write

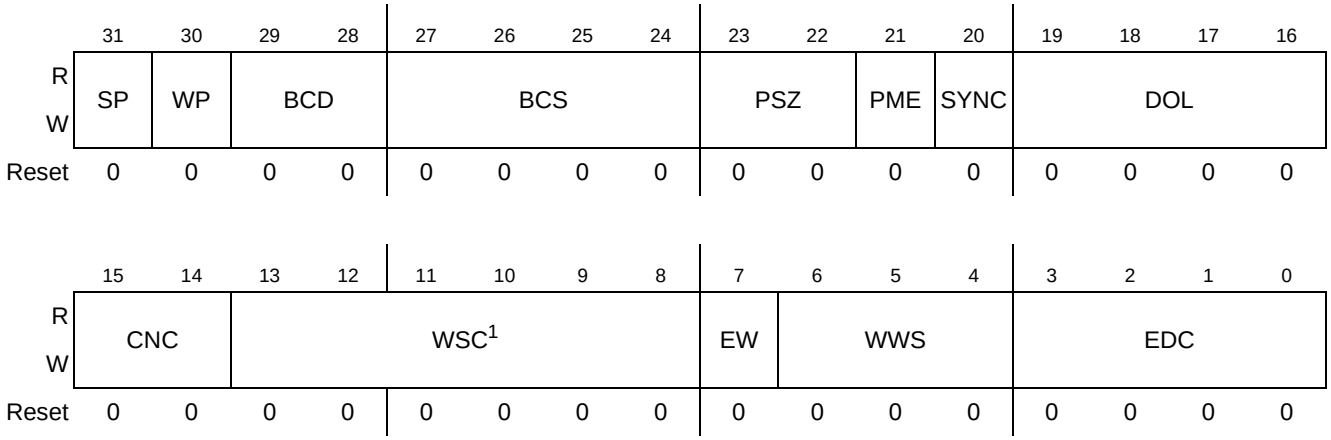


Figure 18-3. Chip Select x Upper Control Register

1 The WSC field (bits 8–13) reset value is 1 for CS0U register and is 0 for all others

Table 18-9. Chip Select x Upper Control Register Field Descriptions

Field	Description
31 SP	Supervisor Protect. This bit prevents accesses to the address range defined by the corresponding chip select when the access is attempted in the User mode of core operation. SP is cleared by a hardware reset. 0 User mode accesses are allowed in the memory range defined by chip select. 1 User mode accesses are prohibited. All attempts to access an address mapped by this chip select in User mode results in a error response on the AHB and no assertion of the chip select output.
30 WP	Write Protect. This bit prevents writes to the address range defined by the corresponding chip select. WP is cleared by a hardware reset. 0 Writes are allowed in the memory range defined by chip. 1 Writes are prohibited. All attempts to write to an address mapped by this chip select result in a error response on the AHB and no assertion of the chip select output.
29–28 BCD	Burst Clock Divisor. This bit field contains the value used to program the burst clock divisor for BCLK generation. It is used to divide the internal AHB bus frequency (133 MHz). When the BCM bit is set in the WEIM configuration register, BCD is ignored. BCD is cleared by a hardware reset. 00 Divide AHB by 1 01 Divide AHB by 2 10 Divide AHB by 3 11 Divide AHB by 4
27–24 BCS	Burst Clock Start. When the BCM bit is set in the WEIM configuration register, this overrides the BCS bits. BCS is cleared by a hardware reset. If SYNC = 1 this bit field determines the number of half cycles after address assertion before the first rising edge of BCLK is seen. See example on Figure 18-41. A value of 0 results in a half clock delay, not an immediate assertion (see Table 18-8).

Table 18-9. Chip Select x Upper Control Register Field Descriptions (continued)

Field	Description
23–22 PSZ	Page Size. If PME is clear (set to 0) the PSZ bit field indicates memory burst length in words (where *word* is defined by the port size or DSZ bits) and should be properly initialized for mixed wrap/increment AHB accesses support. Continuous PSZ value corresponds to continuous burst length setting of the external memory device. If PME is set (set to 1) the PSZ bit field indicates number of words (where “word” is defined by the port size or DSZ bits) in a page in memory. This ensures that the WEIM does not burst past a page boundary at increment access when the PME bit is set. PSZ is cleared by a hardware reset. See Table 18-10 for PSZ Bit Field Combinations.
21 PME	Page Mode Emulation. This bit enables page mode emulation in burst mode. When PME is set (and SYNC equals 1), the external address is asserted for each piece of data requested. Additionally, the $\overline{\text{LBA}}$ and BCLK signals behave in the same way when an asynchronous access is performed (see Figure 18-25). PME is cleared by a hardware reset. 0 Disables page mode emulation 1 Enables page mode emulation
20 SYNC	Synchronous Burst Mode Enable. This bit enables synchronous burst mode if PME is clear (equals 0), if PME is set see PME description). When enabled, the WEIM is capable of interfacing to burst flash devices through additional burst control signals: BCLK, $\overline{\text{LBA}}$, and $\overline{\text{ECB}}$ (see example on the Figure 18-29). The sequencing of these additional I/Os is controlled by other WEIM configuration register bit settings as described in Section 18.5.3, “Burst Mode Memory Operation.” SYNC is cleared by a hardware reset. 0 Disables synchronous burst mode 1 Enables synchronous burst mode
19–16 DOL	Data Output Length. If the SYNC is set (equals 1) the DOL bit field specifies number of wait states during the burst access after the delay of the first data according to the settings shown below (see examples on the Figure 18-25 and Figure 18-40). The reset value 0 specifies that burst data is held for a single AHB clock period. As AHB clock frequencies increase, it may become necessary to delay sampling the data for multiple AHB clock periods to meet burst memory setup and/or frequency specifications and/or WEIM data setup time requirements. DOL has no effect on burst data length when SYNC = 0. DOL is cleared by a hardware reset. 0000 00 0001 01 o 1111 15
15–14 CNC	Chip Select Negation Clock Cycles. This bit field specifies the minimum number of clock cycles a chip select must remain negated after it is negated (but does not guarantee negation for back-to-back accesses, it requires EDC using) according to the settings shown below. See examples on Figure 18-12 and Figure 18-11. CNC has no effect on write accesses when any CSA bit is set. CNC is cleared by a hardware reset. The number of clock cycles of this field can be increased using the CNC2 bit in the appropriate Chip Select Addition Control Register. The number of AHB clock cycles produced by both bit fields is shown in. 00 0 Minimum number of AHB clock cycles $\overline{\text{CS}}$ must remain negated. 01 1 Minimum number of AHB clock cycles $\overline{\text{CS}}$ must remain negated. 10 2 Minimum number of AHB clock cycles $\overline{\text{CS}}$ must remain negated. 11 3 Minimum number of AHB clock cycles $\overline{\text{CS}}$ must remain negated.

Table 18-9. Chip Select x Upper Control Register Field Descriptions (continued)

Field	Description
13–8 WSC	Wait State Control. This bit field programs the number of wait-states for an access to the external device connected to the chip select (see Figure 18-7). For SYNC = 1 WSC programs the number of AHB clock cycles required for the initial access (see Figure 18-25, and Figure 18-32) of a memory burst sequence initiated by the WEIM to an external burst device. For EW = 1 after the wait cycle count expires $\overline{ECB}$ is sampled and the cycle terminates when the $\overline{ECB}$ is negated. On other case WEIM special watch dog counter check that $\overline{ECB}$ will not be asserted for more than 1024 AHB clocks. Additionally in this case WEIM suppresses $\overline{LBA}$ generation after next $\overline{ECB}$ assertion (during increment burst at page boundary crossing). For write accesses the number of wait-states is increased according WWS value or decreased by DWW (see Table 18-11). WSC = 11_1111 indicates operation in a positive edge-sensitive DTACK mode. It selects $\overline{DTACK}$ input as access length control sign (instead of default WSC counter). It means that access length is determined by $\overline{DTACK}$ length. WSC is set to 01_1110 by a hardware reset for CSCR0. WSC is cleared by a hardware reset for CSCR1–CSCR5. **Note:** For SYNC=1 and DOL=0 the WSC value should be at least 4. For SYNC=1, WSC≥ 2(BCD+1)+BCS. For PSR=1, the number of wait-states is increased by one for read access.
7 EW	ECB/WAIT. This bit determines how WEIM supports the $\overline{ECB}$ input in the synchronous mode. In asynchronous mode this bit determines the operation of level-sensitive DTACK mode. (See Table 18-18 for EW effect on WEIM operation modes.) 0 For SYNC = 1, if $\overline{ECB}$ goes to low state in the middle of memory burst access then the WEIM starts a new access by asserting the current AHB address to the ADDR pins and LBA assertion (ECB mode). If SYNC = 0 and WSC=111111, the WEIM waits for $\overline{DTACK}$ posedge for access termination. 1 If $\overline{ECB}$ goes to low state in the middle of a memory burst access then the WEIM waits until $\overline{ECB}$ (goes high (WAIT mode) to continue current access; at the end of first access in burst it allows to wait ECB negation till 1024 clock. For SYNC = 0 WEIM begins access and after (2+DCT) clocks tests $\overline{DTACK}$ input. If $\overline{DTACK}$ is low WEIM waits $\overline{DTACK}$ high state then loads WSC value (see Figure 18-27 and Figure 18-28).
6–4 WWS	Write Wait State. This bit field determines whether additional wait-states are required for write cycles (see Table 18-11). This is useful for writing to memories that require additional data setup time (see example on Figure 18-9). WWS is cleared by a hardware reset. **Note:** To decrease write wait states use the DWW bit field.
3–0 EDC	Extra Dead Cycles. This bit field determines whether idle cycles are inserted before a new access (see example in Figure 18-10). If the currently accessed CS EDC field is not empty then idle cycles are inserted before next access except for two conditions: current access is an asynchronous write (its SYNC = 0) or the next access is an asynchronous read from the same chip select. This field is used in two cases: • Slow memory or peripherals that use long CS or $\overline{OE}$ to output data hold times to prevent data bus contention on back-to-back external transfers. • Synchronous accesses (SYNC = 1) to provide $\overline{CS}$ high minimum pulse width (even for back-to-back accesses). The EDC field is cleared by a hardware reset. 0000 00 No idle AHB clock cycles inserted. 0001 01 1 Idle AHB clock cycle inserted. 1111 15 AHB clock cycles inserted.

Table 18-10. PSZ Bit Field Values

PSZ	PME=0 Memory Burst Length	PME=1 Number of Words in Page
00	4	4
01	8	8

Table 18-10. PSZ Bit Field Values (continued)

PSZ	PME=0 Memory Burst Length	PME=1 Number of Words in Page
10	16	16
11	continuous	32

Table 18-11. WSC Bit Field Values

WSC	Number of Wait-States					
	Read Access	Write Access				
		WWS = 0, DWW = 0	WWS = 1, DWW = 0	WWS = 7, DWW = 0	WWS = 0, DWW = 1	WWS = 0, DWW = 2
000000	1	1	1	7	1	1
000001	1	1	2	8	1	1
000010	2	2	3	9	1	1
000011	3	3	4	10	2	1
000100	4	4	5	11	3	2
o	–	–	–	–	–	–
110111	119	119	120	126	118	117
111000	120	120	121	127	119	118
111001	121	121	122	127	120	119
111010	122	122	123	127	121	120
111011	123	123	124	127	122	121
111100	124	124	125	127	123	122
111101	125	125	126	127	124	123
111110	126	126	127	127	125	124
111111	Posedge sensitive DTACK mode					

18.4.3.2 Chip Select x Lower Control Register (CSCRxL)

The 96 bits used to control Chip Select are divided into three registers: Chip Select x Upper Control Register (CSCRxU), Chip Select x Lower Control Register (CSCRxL) and Chip Select x Additional Control Register (CSCRxA). Table 18-12 shows the register's field descriptions.

- Bits [95:64] are located in Chip Select x Upper Control Register (see Figure 18-3).
- Bits [63:32] are located in Chip Select x Lower Control Register (see Figure 18-4).
- Bits [31:0] are located in Chip Select x Additional Control Register (see Figure 18-5).

0xB800_2004 (CSCR0L)
0xB800_2014 (CSCR1L)
0xB800_2024 (CSCR2L)
0xB800_2034 (CSCR3L)
0xB800_2044 (CSCR4L)
0xB800_2054 (CSCR5L)

Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	OEA				OEN				EBWA				EBWN			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CSA				EBC[1]	DSZ[2]			CSN				PSR	CRE	WRAP	CSEN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0[3]

Figure 18-4. Chip Select x Lower Control Register

1 EBC (bit 11) reset value is 1 for CSCR0L register and 0 for other registers

2 DSZ (bits 8–10) reset value is configurable for the CSCR0L register and 0 for others

3 Bit 0 reset value is 1 for CSCR0L register and 0 for others

Table 18-12. Chip Select x Lower Control Register Field Descriptions

Field	Description
31–28 OEA	$\overline{OE}$ Assert. This bit field determines when $\overline{OE}$ is asserted during read cycles. For SYNC = 0, OEA determines the number of half clocks before $\overline{OE}$ asserts during a read cycle. For SYNC = 1, after the initial memory burst access, $\overline{OE}$ is asserted continuously for subsequent memory burst accesses, and is not affected by OEA (see memory burst read timing diagrams for more details); the behavior of $\overline{OE}$ on the initial memory burst access is the same as when SYNC = 0 (see example on the Figure 18-25). The OEA bits do not affect the cycle length. OEA is set to 0010 by a hardware reset. **Note:** Minimum time $\overline{OE}$ is asserted is one clock cycle. 0000 0 Half AHB clock cycles between $\overline{OE}$ assertion and end of access 0001 1 Half AHB clock cycle between $\overline{OE}$ assertion and end of access o 1111 15 Half AHB clock cycles between $\overline{OE}$ assertion and end of access
27–24 OEN	$\overline{OE}$ Negate. This bit field determines when $\overline{OE}$ is negated during a read cycle (see example on the Figure 18-20). Setting the SYNC bit (SYNC = 1) overrides OEN and $\overline{OE}$ negates at the end of a read access and no sooner. OEN does not affect the cycle length, except in posedge sensitive DTACK mode. OEN is cleared by a hardware reset. **Note:** The term “end of a read access” is the nearest address, data or control signal change by another access (an own next access or an access from others pin shared controller). Minimum time $\overline{OE}$ is asserted is one clock cycle. 0000 0 Half AHB clock cycles between $\overline{OE}$ negation and end of access 0001 1 Half AHB clock cycle between $\overline{OE}$ negation and end of access o 1111 15 Half AHB clock cycles between $\overline{OE}$ negation and end of access

Table 18-12. Chip Select x Lower Control Register Field Descriptions (continued)

Field	Description
23–20 EBWA	Enable Byte Write Assert. This bit field determines when $\overline{EB}$ [3:0] is asserted during write cycles (see example on the Figure 18-8). This is useful to meet data setup time requirements for slow memories. EBWA does not affect the cycle length. EBWA is cleared by a hardware reset. **Note:** Minimum time $\overline{OE}$ is asserted is one clock cycle. 0000 0 Half AHB clock cycles before $\overline{EB}$ is asserted. 0001 1 Half AHB clock cycle before $\overline{EB}$ is asserted. o 1111 15 Half AHB clock cycles before $\overline{EB}$ is asserted.
19–16 EBWN	Enable Byte Write Negate. This bit field determines when $\overline{EB}$ [3:0] outputs are negated during a write cycle (see example on the Figure 18-8). This is useful to meet data hold time requirements for slow memories. EBWN does not affect the cycle length, except in posedge sensitive DTACK mode. Setting the SYNC bit (SYNC = 1) overrides EBWN and $\overline{EB}$ negates at the end of a write access and according AHB hbstrb[3:0]. EBWN is cleared by a hardware reset. **Note:** Minimum time $\overline{OE}$ is asserted is one clock cycle. 0000 0 Half AHB clock cycles between $\overline{EB}$ negation and end of access. 0001 1 Half AHB clock cycle between $\overline{EB}$ negation and end of access. o 111115 Half AHB clock cycles between $\overline{EB}$ negation and end of access.
15–12 CSA	Chip Select Assert. This bit field determines when chip select is asserted for devices that require additional address setup time (see example on the Figure 18-11). It does not affect the cycle length. CSA is cleared by a hardware reset. **Note:** The CSA bit setting affects both reads and writes for all WEIM modes. 0000 0 Half AHB clock cycles before $\overline{CS}$ is asserted. 0001 1 Half AHB clock cycle before $\overline{CS}$ is asserted. o 111115 Half AHB clock cycles before $\overline{CS}$ is asserted.
11 EBC	Enable Byte Control. This bit indicates the types of access that assert Enable Byte outputs $\overline{EB}$[3:0] (see example on Figure 18-7). The $\overline{EB}$[3:0] outputs can be configured as byte write enables. EBC is set by a hardware reset to CSCR0. EBC is cleared by a hardware reset of CSCR1–CSCR5. 0 Both read and write accesses assert the $\overline{EB}$[3:0] 1 Only write accesses assert the $\overline{EB}$[3:0], thus configuring as byte write enables
10–8 DSZ	Data Port Size. This bit field defines the width of an external device's data port as shown in the Table 18-13. DSZ is mapped by a hardware reset for CSCR0 by the value of the BOOT_CFG [:0] bits. DSZ [2], BOOT_CFG [1] maps to DSZ [1] and BOOT_CFG [0] maps to DSZ [0]. DSZ and MUM (multiplexed mode) affected on data port location as it shown in the Table 18-13. DSZ is cleared by a hardware reset of CSCR1–CSCR5.
7–4 CSN	Chip Select Negate. This bit field determines when chip select is negated for devices that require additional address/data hold times (see example on the Figure 18-11). CSN affects only asynchronous (read and write) accesses (SYNC=0), and is ignored on synchronous (SYNC=1). CSN does not affect the cycle length, except in positive edge sensitive DTACK mode. CSN is cleared by a hardware reset. 0000 0 Half AHB clock cycles between $\overline{CS}$ negation and end of access. 0001 1 Half AHB clock cycle between $\overline{CS}$ negation and end of access. o 111115 Half AHB clock cycles between $\overline{CS}$ negation and end of access.

Table 18-12. Chip Select x Lower Control Register Field Descriptions (continued)

Field	Description
3 PSR	Pseudo SRAM Enable (Burst Write Enable). This bit enables four function for Pseudo SRAM (for example CellularRAM™) or any other device that support these modes: a burst write, a write wrap disable, a wait state increase and a memory control register accessibility. A memory burst write is enable (with SYNC = 1). In this mode WRAP bit on write time is masked, unless WWU bit is set in the CSCRxA register, and wait state on read is automatically increased to WSC +1 (see Figure 18-40 and Figure 18-41). An asynchronous access (SYNC=0) should be used with PSR=1 and CRE=1 to write to the memory control register. PSR is cleared by a hardware reset. 0 PSRAM mode is disabled. 1 PSRAM mode is enabled.
2 CRE	Control Register Enable. This bit indicates CRE memory pin state while writing to CS address space, for PSRAM control register write. For PSR=1 the CRE bit will be driven on pin ADDR[23] in a write access time. CRE is cleared by a hardware reset. **Note:** SYNC = 0 should be used to access to PSRAM control register. 0 CRE pin 0 1 CRE pin 1
1 WRAP	Wrap Memory Mode. This bit indicates that memory is in wrap mode. The size of wrap is set using the PSZ bits. In case not matching wrap boundaries in both the memory (PSZ bits) and AHB access on the current address WEIM puts address on address bus and generates $\overline{\text{LBA}}$ signal (see example on the Figure 18-37). WRAP is cleared by a hardware reset. 0 Memory is in not in wrap mode. 1 Memory is in wrap mode.
0 CSEN	Chip Select Enable. This bit controls the operation of the chip select pin. CSEN is set by a hardware reset to CSCR0 to allow CSCR0 to select from an external boot ROM. CSEN is cleared by a hardware reset to CSCR1–CSCR5. 0 Chip select function is disabled; attempts to access an address mapped by this chip select results in a error respond on the AHB and no assertion of the chip select output 1 Chip select is enabled, and is asserted when presented with a valid AHB access.

Table 18-13. DSZ Bit Field Values

DSZ	Data Port Size	
	MUM=0	MUM=1
000		Reserved
001		Reserved
010	8-bit port, resides on DATA_IN/OUT [15:8] pins	Reserved
011	8-bit port, resides on DATA_IN/OUT [7:0] pins	Reserved
100		16-bit port, resides on ADDR/M_DATA_IN/OUT [15:0] pins
101	16-bit port, resides on DATA_IN/OUT [15:0] pins	Reserved
110		32-bit port, resides on ADDR/M_DATA_IN/OUT [15:0] and M_DATA_IN/OUT [31:16] pins
111	The same	The same

18.4.3.3 Chip Select x Additional Control Register (CSCRxA)

The 96 bits used to control Chip Select are divided into three registers: Chip Select x Upper Control Register (CSCRxU), Chip Select x Lower Control Register (CSCRxL) and Chip Select x Additional Control Register (CSCRxA). Table 18-14 shows the register's field descriptions.

- Bits [95:64] are located in Chip Select x Upper Control Register (see Figure 18-3).
- Bits [63:32] are located in Chip Select x Lower Control Register (see Figure 18-4).
- Bits [31:0] are located in Chip Select x Additional Control Register (see Figure 18-5).

0xB800_2008 (CSCR0A)
0xB800_2018 (CSCR1A)
0xB800_2028 (CSCR2A)
0xB800_2038 (CSCR3A)
0xB800_2048 (CSCR4A)
0xB800_2058 (CSCR5A)

Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	EBRA				EBRN				RWA				RWN			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MUM	LAH			LBN			LBA	DWW		DCT		WWU	AGE	CNC2	FCE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-5. Chip Select x Addition Control Register

Table 18-14. Chip Select x Addition Control Register Field Descriptions

Field	Description
31–28 EBRA	Enable Byte Read Assert. This bit field determines when $\overline{EB}$ [3:0] is asserted during read cycles (see example on the Figure 18-25). EBRA does not affect the cycle length. EBRA is cleared by a hardware reset. Minimum time $\overline{EB}$ is asserted is one clock cycle. 0000 0 Half AHB clock cycles before $\overline{EB}$ asserted. 0001 1 Half AHB clock cycle before $\overline{EB}$ asserted. o 1111 15 Half AHB clock cycles before $\overline{EB}$ asserted.
27–24 EBRN	Enable Byte Read Negate. This bit field determines when $\overline{EB}$ [3:0] outputs are negated during a read cycle (see example on the Figure 18-20). EBRN does not affect the cycle length, except when in positive edge sensitive DTACK mode. Setting the SYNC bit (SYNC = 1) overrides EBRN and $\overline{EB}$ negates at the end of a read access but not sooner. EBRN is cleared by a hardware reset. **Note:** Minimum time $\overline{EB}$ is asserted is one clock cycle. 0000 0 Half AHB clock cycles between $\overline{EB}$ negation and end of access. 0001 1 Half AHB clock cycle between $\overline{EB}$ negation and end of access. o 1111 15 Half AHB clock cycles between $\overline{EB}$ negation and end of access.
23–20 RWA	Read/Write Assertion. This bit field determines when $\overline{RW}$ is asserted during write cycles. (see example on the Figure 18-28). RWA is cleared by a hardware reset. **Note:** Minimum time $\overline{EB}$ is asserted is one clock cycle. 0000 0 Half AHB clock cycles $\overline{RW}$ delay 0001 1 Half AHB clock cycle $\overline{RW}$ delay. o 1111 15 Half AHB clock cycles $\overline{RW}$ delay.
19–16 RWN	Read/Write Negation. This bit field determines when $\overline{RW}$ is negated during a write cycle (see example on the Figure 18-28). RWN does not affect the cycle length, except in posedge sensitive DTACK mode. Setting the SYNC bit (SYNC = 1) overrides RWN and $\overline{RW}$ negates at the end of a write access and no sooner. RWN is cleared by a hardware reset. **Note:** Minimum time $\overline{EB}$ is asserted is one clock cycle. 0000 0 Half AHB clock cycles between $\overline{EB}$ negation and end of access. 0001 1 Half AHB clock cycle between $\overline{EB}$ negation and end of access. 1111 15 Half AHB clock cycles between $\overline{EB}$ negation and end of access.
15 MUM	Muxed Mode. This bit determines the address/data muxed mode for asynchronous and synchronous accesses (see examples in Figure 18-43–Figure 18-46). Port mapping is defined in the Table 18-13. MUM is cleared by a hardware reset for CS5–CS1. 0 non-muxed mode 1 muxed mode
14–13 LAH	$\overline{LBA}$ to Address Hold. This bit field determines address hold time after LBA de-assertion for MUM = 1 only. See example on the Figure 18-43 and Figure 18-44. LAH is cleared by a hardware reset. 00 0 AHB half clock cycles between $\overline{LBA}$ negation and address invalid. 01 1 AHB half clock cycle between $\overline{LBA}$ negation and address invalid. 10 2 AHB half clock cycles between $\overline{LBA}$ negation and address invalid. 11 3 AHB half clock cycles between $\overline{LBA}$ negation and address invalid.
12–10 LBN	$\overline{LBA}$ Negation. This bit field determines when $\overline{LBA}$ is negated. For SYNC=0 and MUM =0 LBN determines how many half AHB clock cycle will be between $\overline{LBA}$ negation and end of access (see example on the Figure 18-8). For SYNC=0 and MUM =1 this field determines $\overline{LBA}$ length (see example on the Figure 18-46). Negation time and $\overline{LBA}$ lengths are listed in Table 18-15. For SYNC=1 (MUM=0 and MUM=1) $\overline{LBA}$ negation occurs (LBN+BCD+1) half AHB clock cycles after first BCLK posedge detection. LBN does not affect the cycle length, except in positive edge sensitive DTACK mode. LBN is cleared by a hardware reset. **Note:** Minimum of time $\overline{LBA}$ to be asserted is a one clock for MUM = 0 and two clocks for MUM = 1.

Table 18-14. Chip Select x Addition Control Register Field Descriptions (continued)

Field	Description
9–8 LBA	$\overline{\text{LBA}}$ Assertion. This bit field determines when $\overline{\text{LBA}}$ is asserted according the settings shown below (see example on the Figure 18-11). LBA is cleared by a hardware reset. **Note:** LBA bit affects all modes. Minimum of time $\overline{\text{LBA}}$ to be asserted is a one clock for MUM = 0 and two clocks for MUM = 1. 00 0 AHB half clock cycles between beginning of access and $\overline{\text{LBA}}$ assertion. 01 1 AHB half clock cycle between beginning of access and $\overline{\text{LBA}}$ assertion. 10 2 AHB half clock cycles between beginning of access and $\overline{\text{LBA}}$ assertion. 11 3 AHB half clock cycles between beginning of access and $\overline{\text{LBA}}$ assertion.
7–6 DWW	Decrease Write Wait State. This bit field in combination with WWS determines whether write cycles are shorter than the read cycles (see Table 18-11). DWW is cleared by a hardware reset.
5–4 DCT	$\overline{\text{DTACK}}$ Check Time. This bit field determines time of insensitivity at the beginning of access for SYNC=0 and EW=1 according to the settings shown below (see example on the Figure 18-27). DCT is a number of clock cycles between $\overline{\text{CS}}$ assertion and first $\overline{\text{DTACK}}$ check. DCT is cleared by a hardware reset. 00 AHB clock cycles between $\overline{\text{CS}}$ assertion and first $\overline{\text{DTACK}}$ check. 01 AHB clock cycles between $\overline{\text{CS}}$ assertion and first $\overline{\text{DTACK}}$ check. 10 AHB clock cycles between $\overline{\text{CS}}$ assertion and first $\overline{\text{DTACK}}$ check 11 AHB clock cycles between $\overline{\text{CS}}$ assertion and first $\overline{\text{DTACK}}$ check.
3 WWU	Write Wrap Unmask. This bit allow unmask WRAP bit in case PSR = 1 and write access. WWU is cleared by a hardware reset. 0 Prevents Wrap during write access 1 Allow wrap on write
2 AGE	Acknowledge Glue Enable. This bit is used to enable/disable glue logic between external $\overline{\text{DTACK}}$ and internal control logic. The glue logic is a flip-flop that is reset by CS assertion, its data input is a constant 1, and $\overline{\text{DTACK}}$ goes to its clock input. This glue logic is used to synchronize posedge of the external $\overline{\text{DTACK}}$ in a worst noise or a slowly edge grown conditions. AGE is cleared by a hardware reset. 0 Disable glue logic 1 Enable glue logic
1 CNC2	Chip Select Negation Clock Cycles, Bit [2]. This bit is used to increase the CNC field to a 3-bit field. See description CNC field in the Chip Select x Upper Control Register. CNC2 is cleared by a hardware reset. The number of AHB clock cycles produced by both bit fields is shown in Table 18-16
0 FCE	Feedback Clock Enable. This bit is used to enable/disable data capture by BCLK_FB. If FCE=1 WEIM used addition one clock to synchronize feedback clock captured data to AHB clock, so read access is slow then FCE=0. FCE is cleared by a hardware reset. 0 Data captured using AHB clock 1 Data captured using BCLK_FB

Table 18-15. LBN Bit Field Values

LBN	Half AHB clock cycle between $\overline{\text{LBA}}$ negation and end of access	$\overline{\text{LBA}}$ length, half AHB clock cycle
	MUM = 0	MUM = 1
000	0	2
001	1	3

Table 18-15. LBN Bit Field Values (continued)

LBN	Half AHB clock cycle between $\overline{LBA}$ negation and end of access	$\overline{LBA}$ length, half AHB clock cycle
	MUM = 0	MUM = 1
o	-	-
111	7	9

Table 18-16. CNC/CNC2 Bit Values

CNC2	CNC	Minimum $\overline{CS}$ Negation, in AHB clock cycle
0	00	0
0	01	2
0	10	3
0	11	4
1	00	5
1	01	6
1	10	7
1	11	8

18.4.3.4 WEIM Configuration Register (WCR)

The WCR contains control bits for the configuration and operation of the WEIM. Figure 18-6 shows the register, and Table 18-17 shows the register's field descriptions.

0xB800_2060 (WCR) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	BCM	0	MAS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1/0[1]

Figure 18-6. WCR Register

Table 18-17. WEIM Control Register Field Descriptions

Field	Description
31–3	Reserved
2 BCM	Burst Clock Mode. This bit selects the burst clock mode of operation. BCM is cleared by a hardware reset. 0 The burst clock runs only when accessing a chip select range with the SYNC bit set; when the burst clock is not running it remains in a logic 0 state; when the burst clock is running it is configured by the BCD and BCS bits in the chip select control register. 1 The burst clock runs all the time (independent of chip select accesses)
1	Reserved
0 MAS	Merged Address Space. This bit indicates merged address space mode. If MAS is set the $\overline{CS1}$ address space is merged with $\overline{CS0}$ for a total of 256 (for halfword width port and 512 for word width port) Mbytes. $\overline{CS1}$ output is used as A26. This bit is configured at reset time with the BOOT_CFG[] pin. 0 Standard address space. 1 Merged address space.

18.5 Functional Description

This section provides the functional description for the WEIM module.

18.5.1 Configurable Bus Sizing

The WEIM supports byte, halfword, and word operands allowing access to 8-bit, 16-bit, and 32-bit ports. The port size is programmable via the DSZ[2:0] bits in the corresponding Chip Select control register. In addition, the portion of the data bus used for transfer to or from an 8-bit port is programmable via the same DSZ bits. An 8-bit port can reside on DATA_IN/OUT bus bits [15:8] or [7:0]. A 16-bit port reside on DATA_IN/OUT bus bits [15:0]. A 32-bit multiplexed port resides on M_DATA_IN/OUT bus bits [31:0].

NOTE

Misaligned transfers are not supported.

A word access to or from an 8-bit port requires four external bus cycles to complete the transfer. A word access to or from a 16-bit port requires two external bus cycles to complete the transfer. A halfword access to or from an 8-bit port requires two external bus cycles to complete the transfer.

In the case of a multi-cycle transfer, the lower two address bits (ADDR[1:0]) are incremented appropriately. The WEIM address bus is configured according to DSZ bits, too. There is either one or two bits right shift of AHB address bits for halfword or word width port accordingly.

The WEIM has a data multiplexer which takes the four bytes of the AHB interface data bus and routes them to their required positions to properly interface to memory and peripherals.

18.5.2 WEIM Operational Modes

WEIM has 9 main operational modes selected by control fields settings as described in the Figure 18-18. For details see corresponding bit fields descriptions.

Table 18-18. WEIM Operation Modes Field Settings

Control fields					Brief mode description
SYNC	PME	MUM	EW	WSC	
0	0	0	0	< 11_1111	Asynchronous
		1	0		Asynchronous multiplexed
		0	1		Asynchronous level sensitive DTACK mode
		0	0	11_1111	Asynchronous posedge sensitive DTACK mode
1	1	0	0	< 11_1111	Asynchronous page mode emulation
	0	0	0		Synchronous burst with restart on $\overline{\text{ECB}}$ negation
		1	0		$\overline{\text{Synchronous multiplexed burst with restart on ECB negation}}$
		0	1		Synchronous burst with wait on $\overline{\text{ECB}}$ negation
		1	1		Synchronous multiplexed burst with wait on $\overline{\text{ECB}}$ negation

18.5.3 Burst Mode Memory Operation

With memory burst mode enabled (SYNC = 1), the WEIM attempts to translate AHB burst accesses to memory burst accesses, being limited by the memory burst length, predefined PSZ value, or memory and AHB WRAP/INCR boundary crossing non-matching. The WEIM only displays the first address accessed in a memory burst sequence unless the page mode emulation (PME) bit is set.

WEIM may translate from some AHB sequential accesses to one or few memory bursts, but not from two AHB nonsequential accesses to one memory burst.

For the first access in a memory burst sequence, the WEIM asserts $\overline{\text{LBA}}$-causing the external burst device to latch the starting burst address and then toggle the burst clock (BCLK) a predefined number of cycles to latch the first unit of data. Subsequent accessed data units can then be burst in fewer clock cycles, realizing an overall increase in bus bandwidth.

Memory burst accesses are terminated by the WEIM whenever it detects that:

- the next AHB access is not sequential,
- next sequential access crosses boundary with unequal condition (wrap/increment, burst length) on the AHB and memory,
- current memory burst length reached,
- by the external burst device request if it needs additional cycles to retrieve the next requested memory location.

In last case, the burst memory device provides an $\overline{\text{ECB}}$ (or WAIT) feedback signal to the WEIM whenever it is necessary to terminate/postponed the on-going burst sequence. If EW = 0 WEIM initiate a new (with long first access) memory burst sequence, if EW = 1 WEIM only waits $\overline{\text{ECB}}$ negation to continue current memory burst sequence. Additionally EW = 1 allows wait states insertion after wait state counter expired, but $\overline{\text{ECB}}$ still asserted. Over this a new memory burst sequence should be generated.

The synchronous mode is also used for burst Cellular RAM, which supports memory burst writes, that is enabled by PSR = 1.

18.5.4 Burst Clock Divisor

In some cases it may be necessary to slow the external bus in relation to the internal bus to allow accesses to burst devices that have a maximum operating frequency which is less than the operating frequency of the internal bus. The internal AHB bus frequency (133 MHz) can be divided by two, three, or four for presentation on the external bus in burst mode operation.

By programming the BCD bits to various values, two signals on the external bus are affected; $\overline{\text{LBA}}$ and BCLK. The $\overline{\text{LBA}}$ signal is asserted according to LBA field programming and remain asserted until the first falling edge of the BCLK signal. The BCLK signal runs with a 50% duty cycle until a non-sequential internal request is received or an external $\overline{\text{ECB}}$ signal is recognized.

Caution should be exercised when programming these bits to ensure the WSC and DOL bit fields are coordinated to provide the desired external bus waveforms. As an example, if the BCD bits are programmed to 01, the DOL bits should be programmed to 0001, 0011, 0101, 0111…. If the BCD bits are programmed to 10, the DOL should be programmed to 0010, 0101, 1000, 1011 ….

The BCM bit in the WEIM configuration register has priority over the BCD bits. If BCM = 1, the BCLK runs at full frequency.

18.5.5 Burst Clock Start

In an effort to allow greater flexibility in achieving the minimum number of wait states on bursted accesses, the user can determine when they want the BCLK to start toggling. This allows the BCLK to be skewed from point of data capture on the AHB clock by any number of AHB clock phases. Care must be exercised when setting these bits in conjunction with the BCD, WSC, and DOL bits. See the external timing diagrams from Section 18.7.4, “Burst Memory Accesses Timing Diagrams” for some examples of how to use the BCS, BCD, WSC, and DOL bits together.

18.5.6 Page Mode Emulation

Setting the PME and SYNC bits causes the WEIM to perform memory bursted accesses by emulating page mode operation. The $\overline{\text{LBA}}$ signal remains asserted for the entire access, the burst clock does not send a signal, and the external address asserts for each access are made. The initial access timing is dictated by the WSC bits, and the page mode access timing is dictated by the DOL bits.

See the external timing diagrams from the Section 18.7.2, “Page Mode Timing Diagrams” for some examples.

The WEIM can take advantage of improved page timing for sequential accesses only. Accesses that are on the page but are not sequential in nature have their timing dictated by the WSC bits. The page size can be set via the PSZ bits to 4, 8, 16, or 32 words (the word size is determined by the data width of the external memory, such as the DSZ bits).

18.5.7 PSRAM Mode Operation

A control bit PSR is provided to enable PSRAM operation. For SYNC = 1 this bit enable memory burst write. In this mode WRAP bit on write time is automatic masked (according the CellularRAM™ specification, wrap supports only for read accesses) unless WWU is set. Initial wait state value is automatic incremented on read access (see Figure 18-40 and Figure 18-41).

Bit EW determines how WEIM support $\overline{\text{ECB}}$ input. For SYNC = 1 if $\overline{\text{ECB}}$ goes to low state in the middle of memory burst access then WEIM only waits it'll go high (WAIT mode) to continue current access; at the end of first access in memory burst it allows to wait $\overline{\text{ECB}}$ negation during PSRAM refresh insertion.

Bit CRE and an unused address line can be used to drive the control register enable (CRE) memory input to load the PSRAM configuration registers. For PSR = 1 the CRE bit will be driven on pin ADDR[23] in a write access time.

NOTE

SYNC = 0 should be used to access to PSRAM control register.

Bit MUM allows support memory with multiplexed address/data bus. The LBN and LBH bit fields should be used for properly bus timing setup (see Figure 18-43–Figure 18-46).

18.5.8 Mixed AHB/Memory Burst Modes Support

To provide mixed sequential/wrap accesses with different length WEIM interprets burst signal and generate additional $\overline{\text{LBA}}$ signals whenever there appear unequal address or burst boundary crossing condition (see Section 18.5.3, "Burst Mode Memory Operation"). Bits PSZ and WRAP should be used to notify WEIM about current memory burst and wrap condition for properly external address generation. In case of non matching boundaries in both the memory and AHB access WEIM starts a new memory burst access by putting address from AHB on address bus and generating $\overline{\text{LBA}}$ signal. For example, Table 18-20 shows how WEIM interprets various types of AHB access in case memory configured to 8 beat burst with wrap.

18.5.9 AHB Bus Cycles Support

The WEIM uses an ARM AHB slave interface. It has a 32-bit bus and supports the four transfer types defined in the AHB specification (IDLE, BUSY, NONSEQ, and SEQ).

NOTE

Only 32-bit accesses are supported for SEQ mode.

It also supports the AHB transfers shown on Table 18-19. These AHB cycles will be translated into the necessary cycles on the memory side. For example for optimal operation in case ARM cache is configured to 8 beat burst with wrap, synchronous flash and cellular RAM memory should be configured in 16 word wrap burst mode when using a 16-bit data port, and in 8 word wrap burst mode when using a 32-bit data port. WEIM uses WRAP bit and PSZ bits for support different memory configuration. The controller splits the transaction when needed in some cases (see Section 18.5.3, "Burst Mode Memory Operation").

Table 18-19. AHB Burst Cycles Supported

HBURST	TYPE	Supported	Description
000	SINGLE	Yes	Single transfer
001	INCR	Yes	Incrementing burst
010	WRAP4	Yes	4-beat wrapping burst
011	INCR4	Yes	4-beat incrementing burst
100	WRAP8	Yes	8-beat wrapping burst
101	INCR8	Yes	8-beat incrementing burst
110	WRAP16	Yes	16-beat wrapping burst
111	INCR16	Yes	16-beat incrementing burst

For example, Table 18-20 shows AHB bus sequential accesses breaking in to external memory bursts for memory configured to 8 beat burst with wrap and for some different AHB burst types and start addresses. $\overline{\text{LBA}}$(X) means start memory burst access ($\overline{\text{LBA}}$ generation) from address X (all addresses in hex form).

Table 18-20. External Memory Bursts Start Addresses for Some AHB Burst Accesses

AHB burst type	Memory data port width	AHB burst start address							
		0	4	8	C	10	14	18	1C
WRAP8	16-bit	$\overline{\text{LBA}}$(0) $\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(4) $\overline{\text{LBA}}$(10) $\overline{\text{LBA}}$(0)	$\overline{\text{LBA}}$(8) $\overline{\text{LBA}}$(10) $\overline{\text{LBA}}$(0)	$\overline{\text{LBA}}$(C) $\overline{\text{LBA}}$(10) $\overline{\text{LBA}}$(0)	$\overline{\text{LBA}}$(10) $\overline{\text{LBA}}$(0)	$\overline{\text{LBA}}$(14) $\overline{\text{LBA}}$(0) $\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(18) $\overline{\text{LBA}}$(0) $\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(1C) $\overline{\text{LBA}}$(0) $\overline{\text{LBA}}$(10)
	32-bit	$\overline{\text{LBA}}$(0)	$\overline{\text{LBA}}$(4)	$\overline{\text{LBA}}$(8)	$\overline{\text{LBA}}$(C)	$\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(14)	$\overline{\text{LBA}}$(18)	$\overline{\text{LBA}}$(1C)
INCR8	16-bit	$\overline{\text{LBA}}$(0) $\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(4) $\overline{\text{LBA}}$(10) $\overline{\text{LBA}}$(20)	$\overline{\text{LBA}}$(8) $\overline{\text{LBA}}$(10) $\overline{\text{LBA}}$(20)	$\overline{\text{LBA}}$(C) $\overline{\text{LBA}}$(10) $\overline{\text{LBA}}$(20)	$\overline{\text{LBA}}$(10) $\overline{\text{LBA}}$(20)	$\overline{\text{LBA}}$(14) $\overline{\text{LBA}}$(20) $\overline{\text{LBA}}$(30)	$\overline{\text{LBA}}$(18) $\overline{\text{LBA}}$(20) $\overline{\text{LBA}}$(30)	$\overline{\text{LBA}}$(1C) $\overline{\text{LBA}}$(20) $\overline{\text{LBA}}$(30)
	32-bit	$\overline{\text{LBA}}$(0)	$\overline{\text{LBA}}$(4) $\overline{\text{LBA}}$(20)	$\overline{\text{LBA}}$(8) $\overline{\text{LBA}}$(20)	$\overline{\text{LBA}}$(C) $\overline{\text{LBA}}$(20)	$\overline{\text{LBA}}$(10) $\overline{\text{LBA}}$(20)	$\overline{\text{LBA}}$(14) $\overline{\text{LBA}}$(20)	$\overline{\text{LBA}}$(18) $\overline{\text{LBA}}$(20)	$\overline{\text{LBA}}$(1C) $\overline{\text{LBA}}$(20)
WRAP4	16-bit	$\overline{\text{LBA}}$(0)	$\overline{\text{LBA}}$(4)	$\overline{\text{LBA}}$(8)	$\overline{\text{LBA}}$(C)	$\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(14)	$\overline{\text{LBA}}$(18)	$\overline{\text{LBA}}$(1C)
	32-bit	$\overline{\text{LBA}}$(0)	$\overline{\text{LBA}}$(4) $\overline{\text{LBA}}$(0)	$\overline{\text{LBA}}$(8) $\overline{\text{LBA}}$(0)	$\overline{\text{LBA}}$(C) $\overline{\text{LBA}}$(0)	$\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(14) $\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(18) $\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(1C) $\overline{\text{LBA}}$(10)
INCR4	16-bit	$\overline{\text{LBA}}$(0)	$\overline{\text{LBA}}$(4) $\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(8) $\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(C) $\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(14) $\overline{\text{LBA}}$(20)	$\overline{\text{LBA}}$(18) $\overline{\text{LBA}}$(20)	$\overline{\text{LBA}}$(1C) $\overline{\text{LBA}}$(20)
	32-bit	$\overline{\text{LBA}}$(0)	$\overline{\text{LBA}}$(4)	$\overline{\text{LBA}}$(8)	$\overline{\text{LBA}}$(C)	$\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(14) $\overline{\text{LBA}}$(20)	$\overline{\text{LBA}}$(18) $\overline{\text{LBA}}$(20)	$\overline{\text{LBA}}$(1C) $\overline{\text{LBA}}$(20)

Some examples are shown in Figure 18-32 through Figure 18-39.

18.5.10 DTACK Mode

It is a mode, there WEIM timing depends on $\overline{\text{DTACK}}$ input signal. This signal may be used in two ways: by posedge sensitive or by level sensitive (with an initial insensitiveness time).

Posedge sensitive mode is set by WSC=111111 (imply EW=0) and selects $\overline{\text{DTACK}}$ input as access length control sign (instead of default WSC counter). It means that access length is determined by $\overline{\text{DTACK}}$ length. WEIM begins deasserting control signals after approximately 1.5 clock (synchronization delay) in the sequence according negation control fields.

NOTE

It requires CSA and/or CSN bit fields be programmed for a correct word access to 16 or 8-bit port in this mode if corresponding module is a CS sensitive.

NOTE

CSN maximum value is 6 for this case.

Level sensitive mode is set by EW=1 (imply WSC < 111111). There access length is controlled by WSC. In this case WEIM begins access (by CS assertion) and after (2 +DCT) clocks checks $\overline{\text{DTACK}}$ input. If $\overline{\text{DTACK}}$ is low WEIM waits $\overline{\text{DTACK}}$ high state and reload wait state counter (see Figure 18-27 and Figure 18-28). For sequential AHB accesses, where CS does not negate during the burst, $\overline{\text{DTACK}}$ is being checked on the first access only.

Glue logic Enabled by AGE bit of the CSCRxA register can be used for Noisy or slowly rising $\overline{\text{DTACK}}$. Refer to the AGE bit description for more details.

18.5.11 Internal Input Data Capture

In typical case the input data is not sampled by the WEIM but it is sampled by the AHB master on the rising edge of HCLK when HREADY is high. WEIM assert the HREADY signal to the AHB master (according WSC or DOL counting correspondingly). This allow better performance on the data path.

There are 2 cases where input data do get sampled inside the WEIM.

First one is when the access size is larger then the port size. In this case WEIM sample all the Data coming from the memory device except the last one of that transaction. For example if there is a word access to the byte wide memory the WEIM captures first three input bytes internally and drive them together with the last byte to the AHB master (last byte is not sampled in the WEIM). WEIM captured data by rising edge of HCLK when WSC (or DOL if it is a part of burst) time expired and (if it depends) suitable $\overline{\text{ECB}}$ or $\overline{\text{DTACK}}$ input condition.

Second case is when a feedback clock is used (FCE=1) for synchronous burst. Data will be sampled on the rising edge of the feedback clock (when WSC or DOL time expired and input condition kept) and then those captured data again is sampled by HCLK before it will be driven to the AHB master.

18.5.12 Error Conditions

The following conditions cause an error signal:

- Access to a disabled chip select (access to a mapped chip select address space where the CSEN bit in the corresponding chip select control register is clear)
- Write access to a write-protected chip select address space (the WP bit in the corresponding chip select control register is set)
- User access to a supervisor-protected chip select address space (the SP bit in the corresponding chip select control register is set)
- User read or write access to a chip select control register or the WEIM configuration register
- Byte or halfword access to a chip select control register or the WEIM configuration register
- $\overline{\text{DTACK}}$ acknowledge is absent more then 1024 clock
- WAIT deassertion more then 1024 clock.

18.6 Initialization/Application Information

WEIM is ready to work with $\overline{\text{CS0}}$ after hardware reset, but it has been configured for very slowly access (for boot purpose) without additional setup and hold time. Other $\overline{\text{CS}}$ are disabled by hardware reset. So any CS has to be properly initialized before using it by writing values to high and low configuration register.

Example 18-1 shows how to prepare WEIM and 16-bit flash memory to work in the synchronous mode.

Example 18-1. WEIM and Flash Memory Initialization for Work in Synchronous Mode

```
@; config WEIM to Async access with RWA and RWN
         WRITE WEIM_CS2U, 0x12020800
         WRITE WEIM_CS2L, 0x80330d03

@; config flash to WRAP 8 mode
         WRITEH (CS2_BASE_ADDR+0x2384), 0x60     @; offset = 0x11c2 << 1
         WRITEH (CS2_BASE_ADDR+0x2384), 0x03

         WRITEH (CS2_BASE_ADDR+0x0), 0xff             @; flash to read mode

@; config to WEIM Sync access with WRAP8, 16 bit port
         WRITE WEIM_CS2U, 0x13510802
         WRITE WEIM_CS2L, 0x80330d03
```

18.7 External Bus Timing Diagrams

The following timing diagrams show the timing of accesses to memory or a peripheral with different timing parameters. All examples done for $\overline{\text{CS0}}$, but are valid for any others chip select. $\overline{\text{EB}}$ means one from current used $\overline{\text{EB}}$[3:0]

For asynchronous mode:

- Figure 18-7 through Figure 18-13 show halfword read and write accesses to halfword-width memory.
- Figure 18-14 through Figure 18-24 shows word read and write accesses to halfword-width memory.
- Figure 18-25 shows page mode timing diagrams, Figure 18-26 through Figure 18-28 shows two kinds of $\overline{\text{DTACK}}$ accesses.
- Figure 18-43 and Figure 18-44 shows asynchronous data exchange in muxed A/D mode with word-width memory.

For synchronous (burst) mode:

- Figure 18-29 shows synchronous word read accesses to halfword-width memory.
- Figure 18-30 shows recommended parameter setting using for synchronous accesses: BCLK clock has a short pulse at the end of access.
- Different cases of wrap/increment states on AHB and memory are shown in the Figure 18-32 throughFigure 18-39 for burs size 4. AHB increment/wrap accesses with another length are made like this.
- PSRAM read and write accesses are shown in the Figure 18-40 and Figure 18-41.
- Figure 18-45 and Figure 18-46 show synchronous data exchange in muxed A/D mode with word-width memory.

18.7.1 Asynchronous Memory Accesses Timing Diagrams

18.7.1.1 AHB Halfword Access to Halfword Width Memory

Figure 18-7 through Figure 18-13 show the AHB halfword access to halfword width memory timing diagrams.

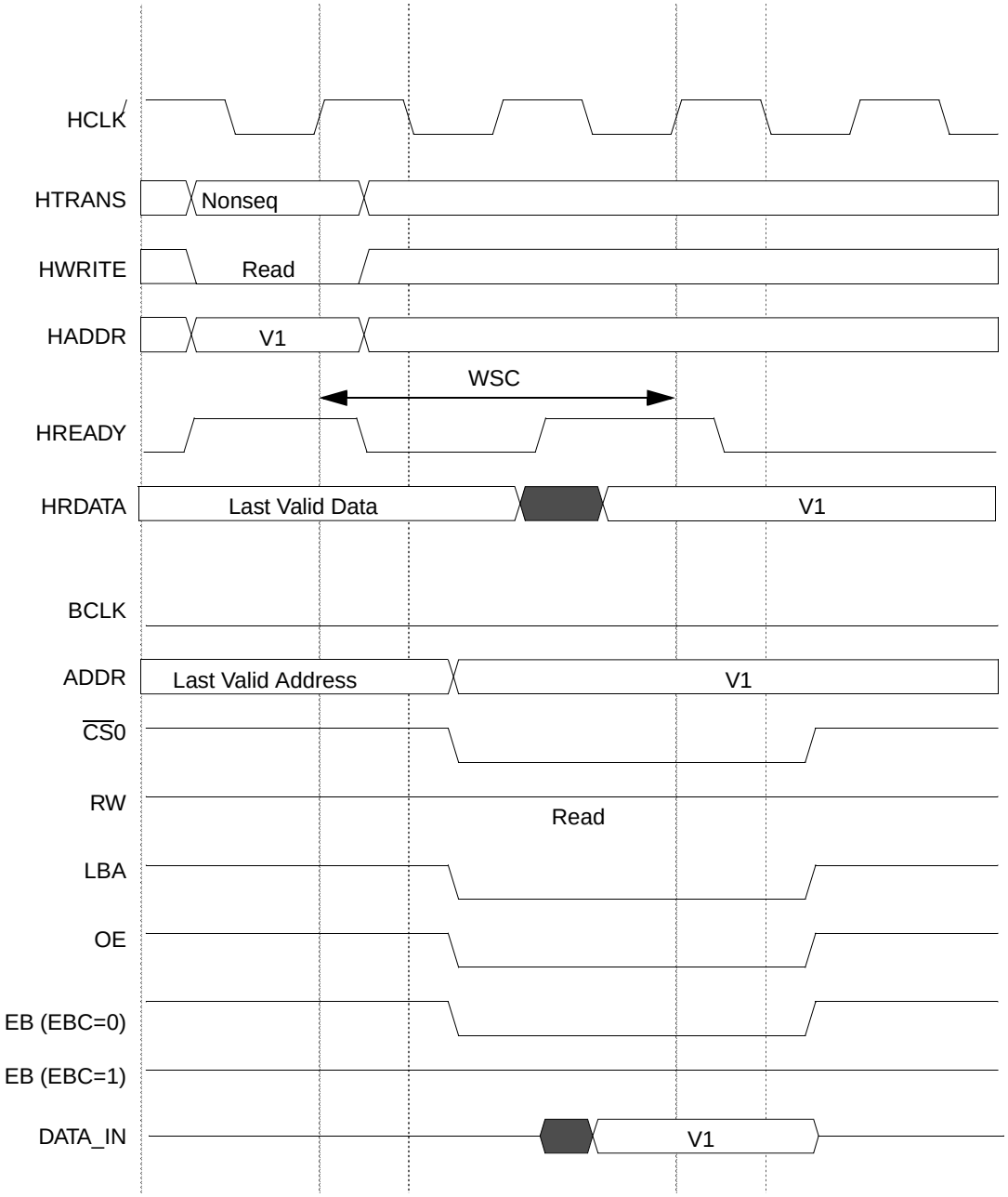


Figure 18-7. Read Access, WSC=1

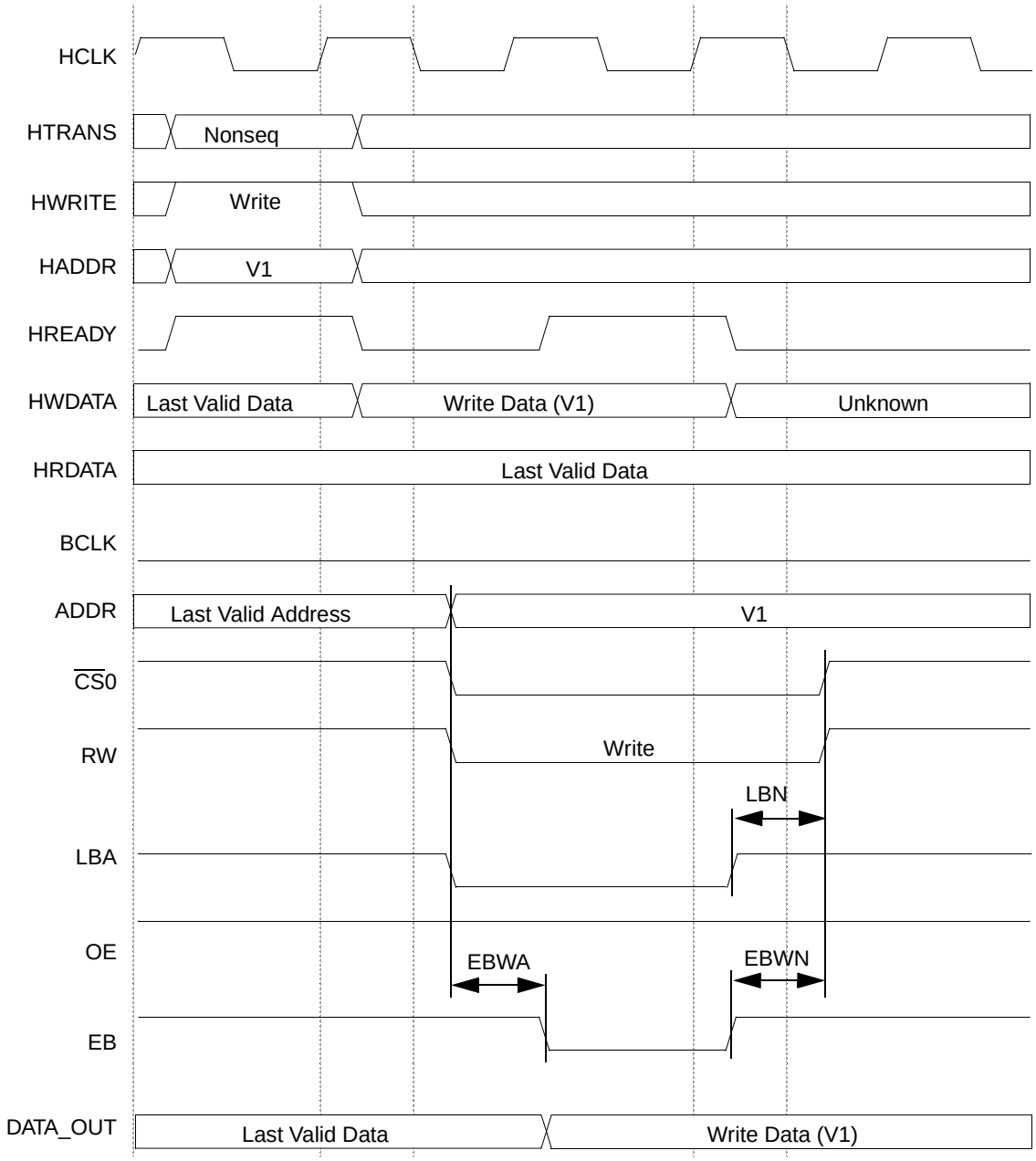


Figure 18-8. Write Access, WSC=1, EBWA=1, EBWN=1, LBN=1

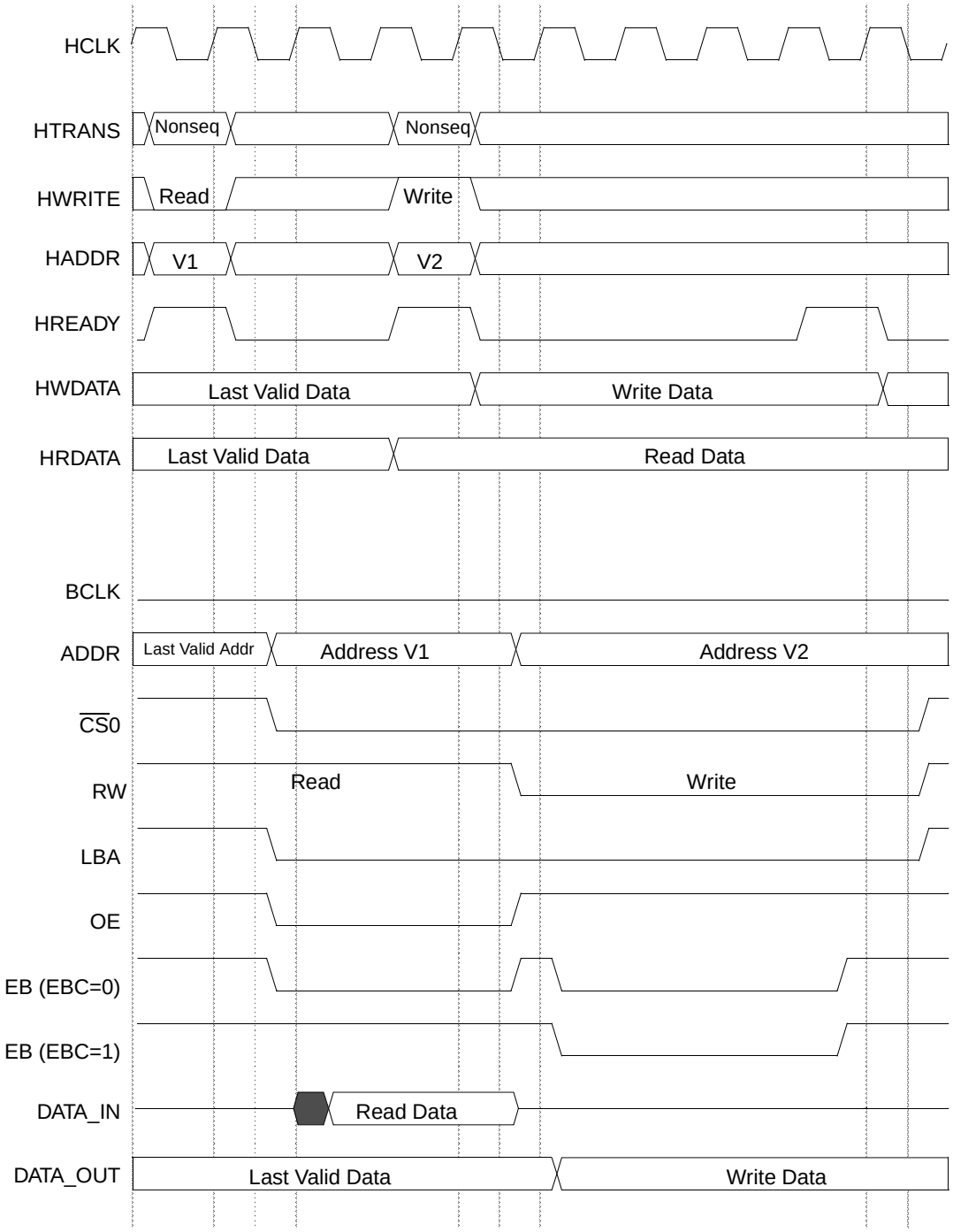


Figure 18-9. Read and Write Accesses, WSC=2, WWS=2, EBWA=1, EBWN=2

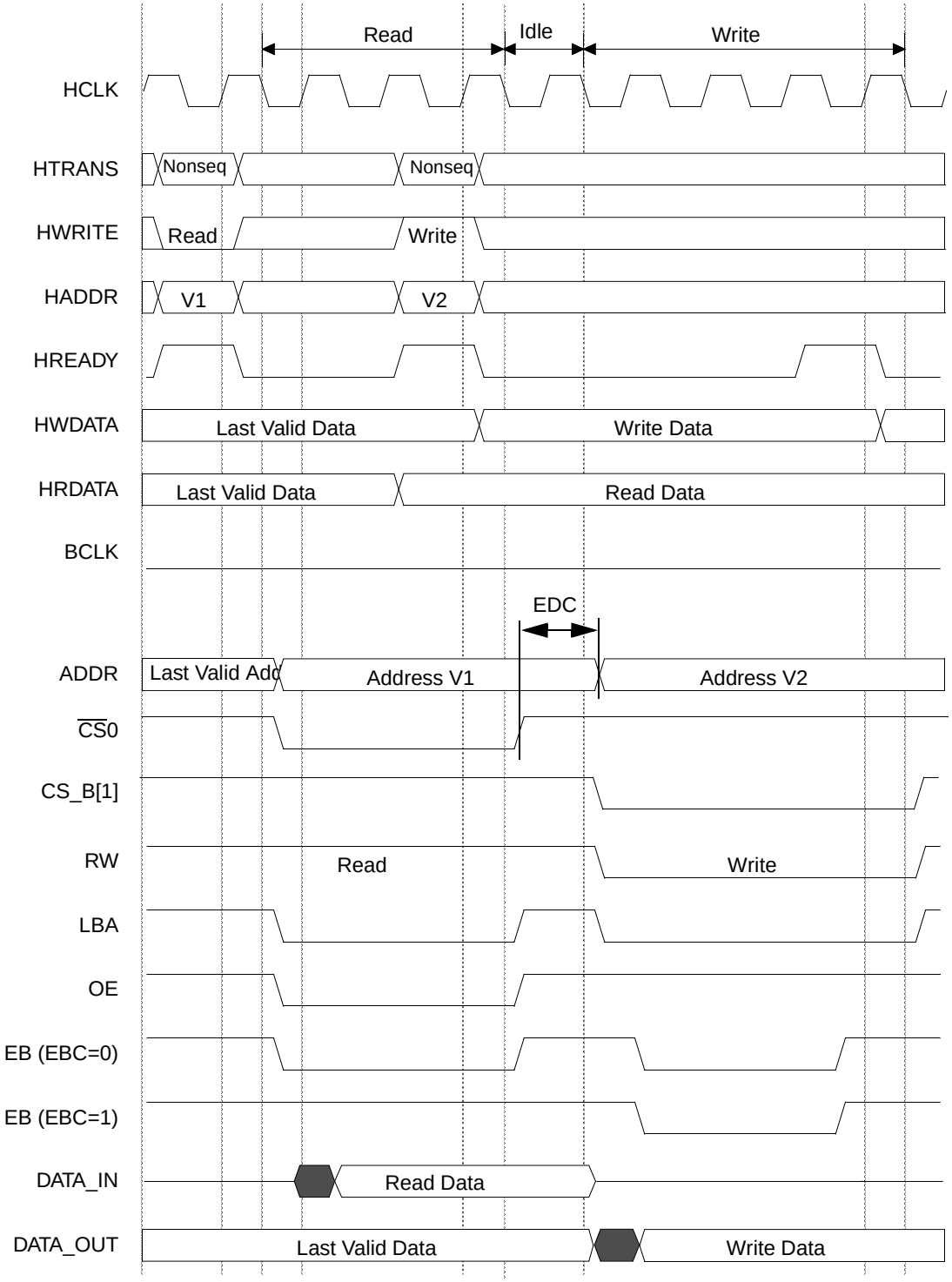


Figure 18-10. Read and Write Accesses, WSC=2, WWS=1, EBWA=1, EBWN=2, EDC=1

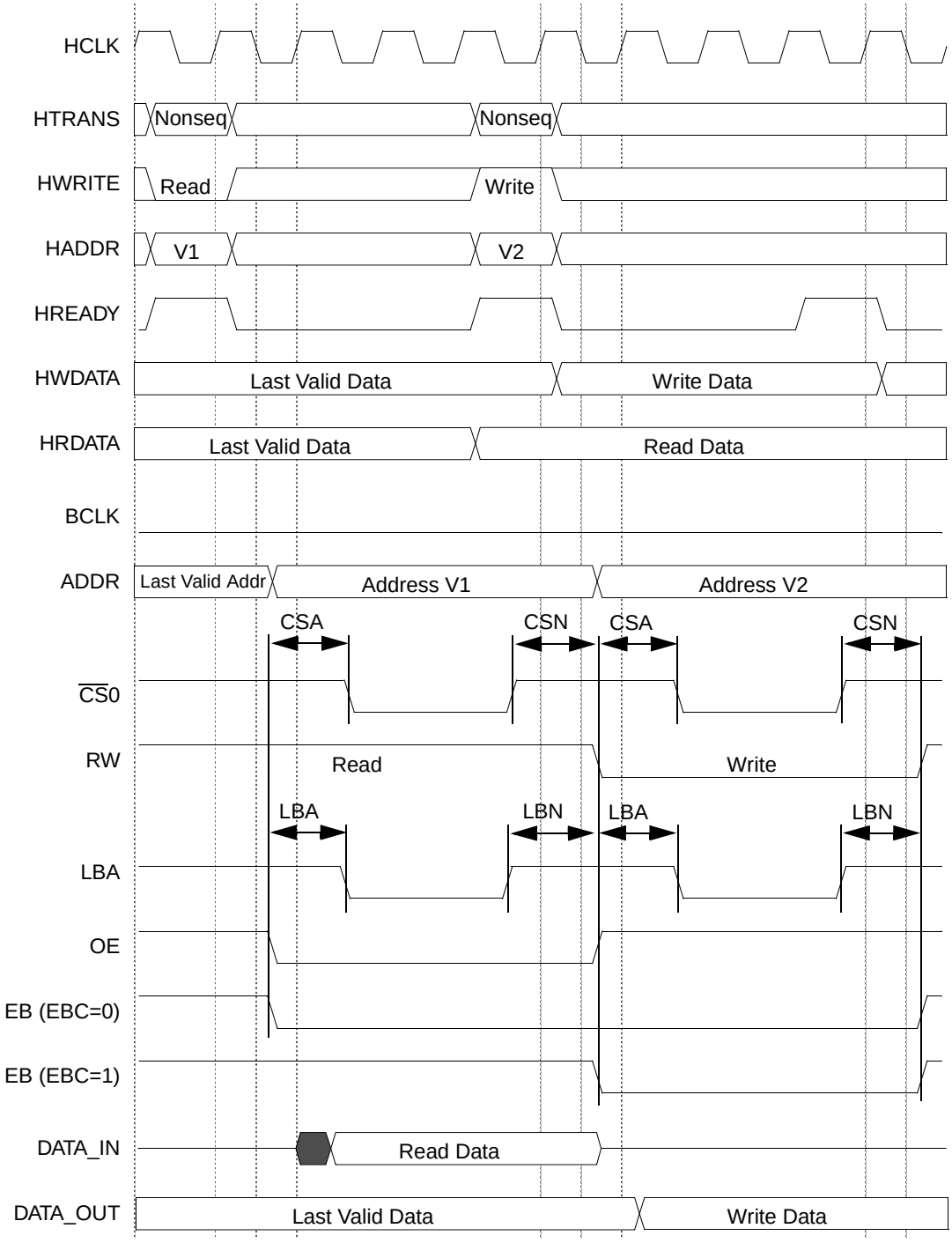


Figure 18-11. Read and Write Accesses, WSC=3, CSA=1, CSN=1, LBA=1, LBN=1

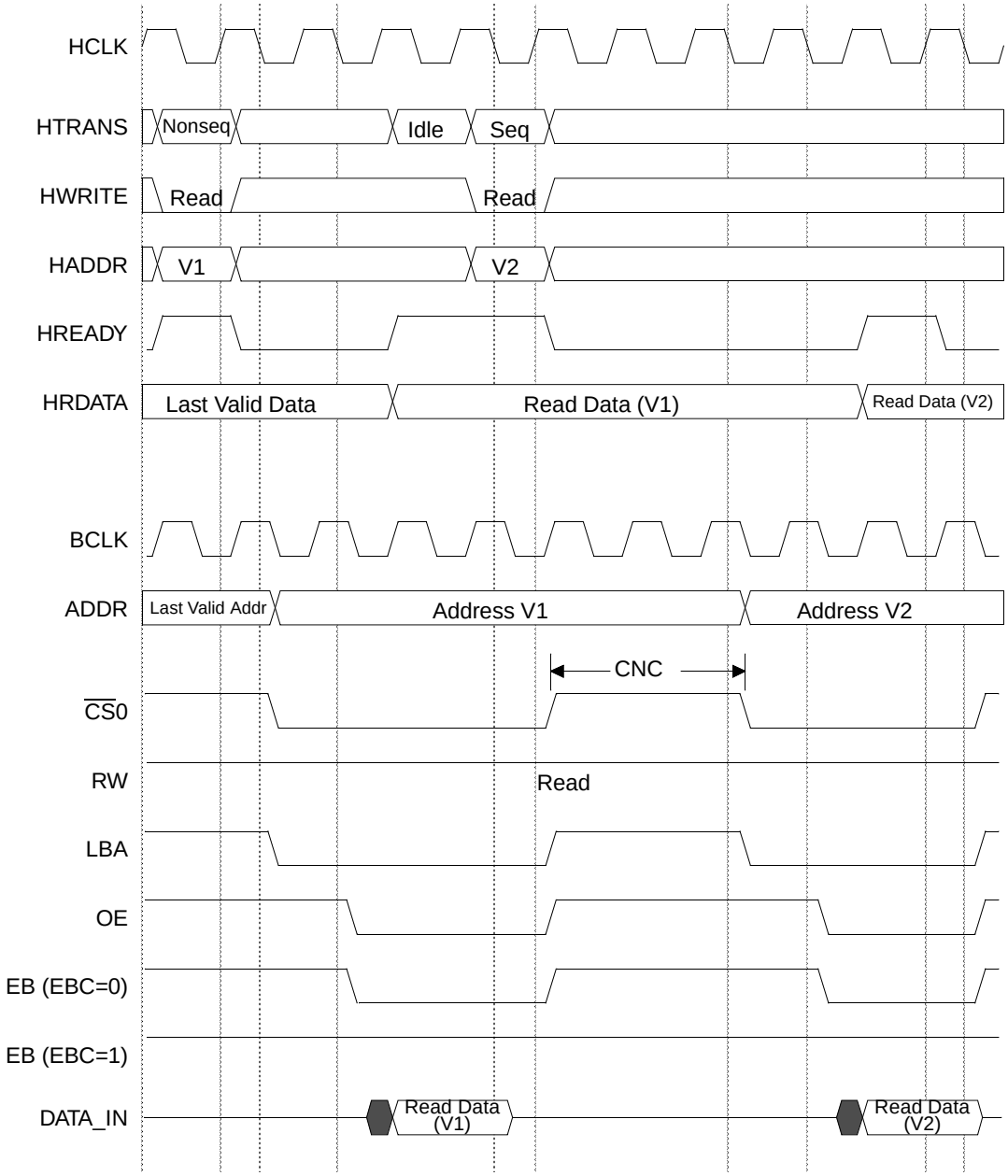


Figure 18-12. Read Accesses, WSC=2, OEA=2, CNC=2, BCM=1, EBRA=2

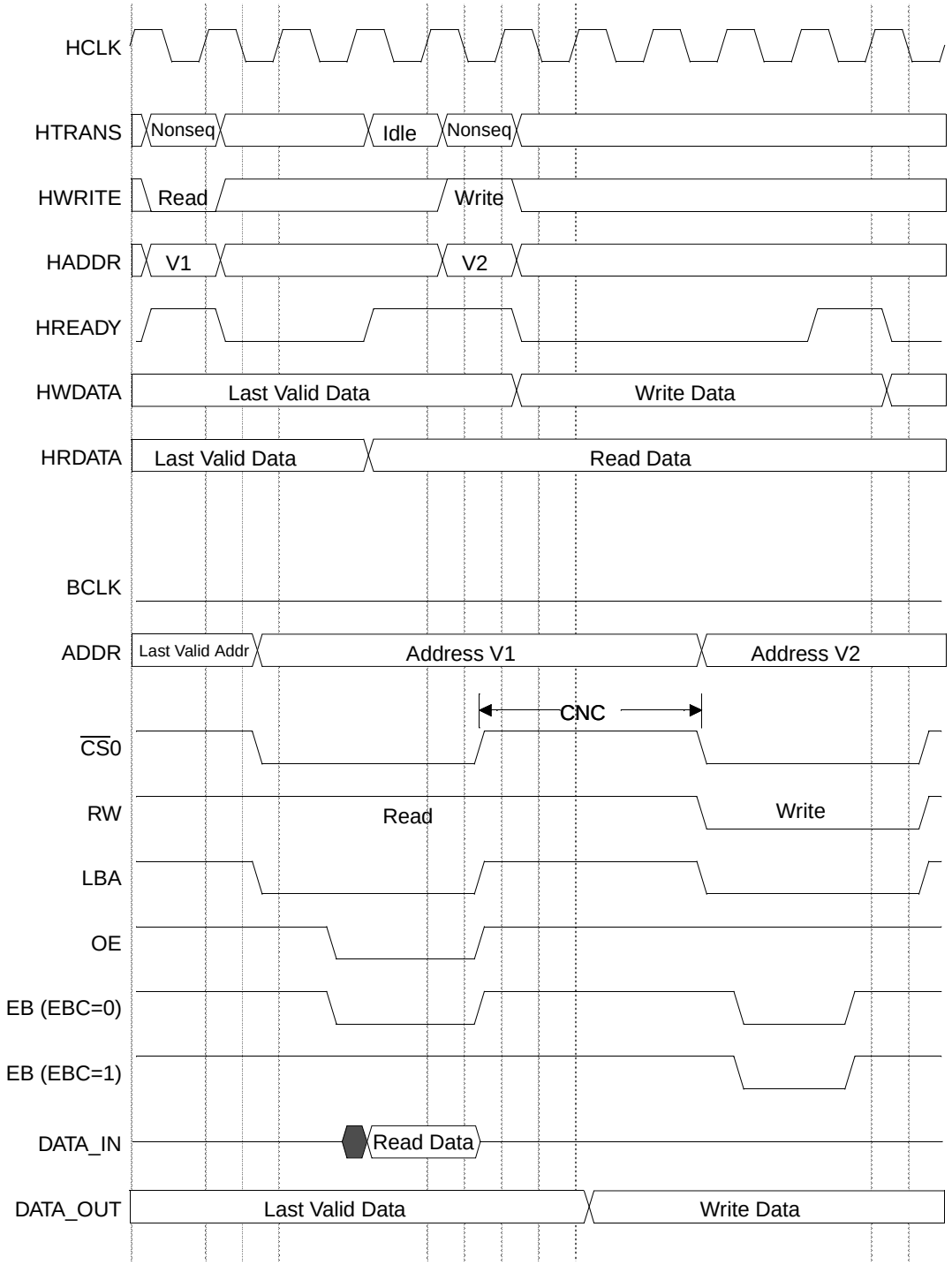


Figure 18-13. Read and Write Accesses, WSC=2, OEA=2, EBWA=1, EBWN=2, CNC=2, EBRA=2

18.7.1.2 AHB Word Access to Halfword Width Memory

Figure 18-13 through Figure 18-24 show the AHB word access to halfword width memory timing diagrams.

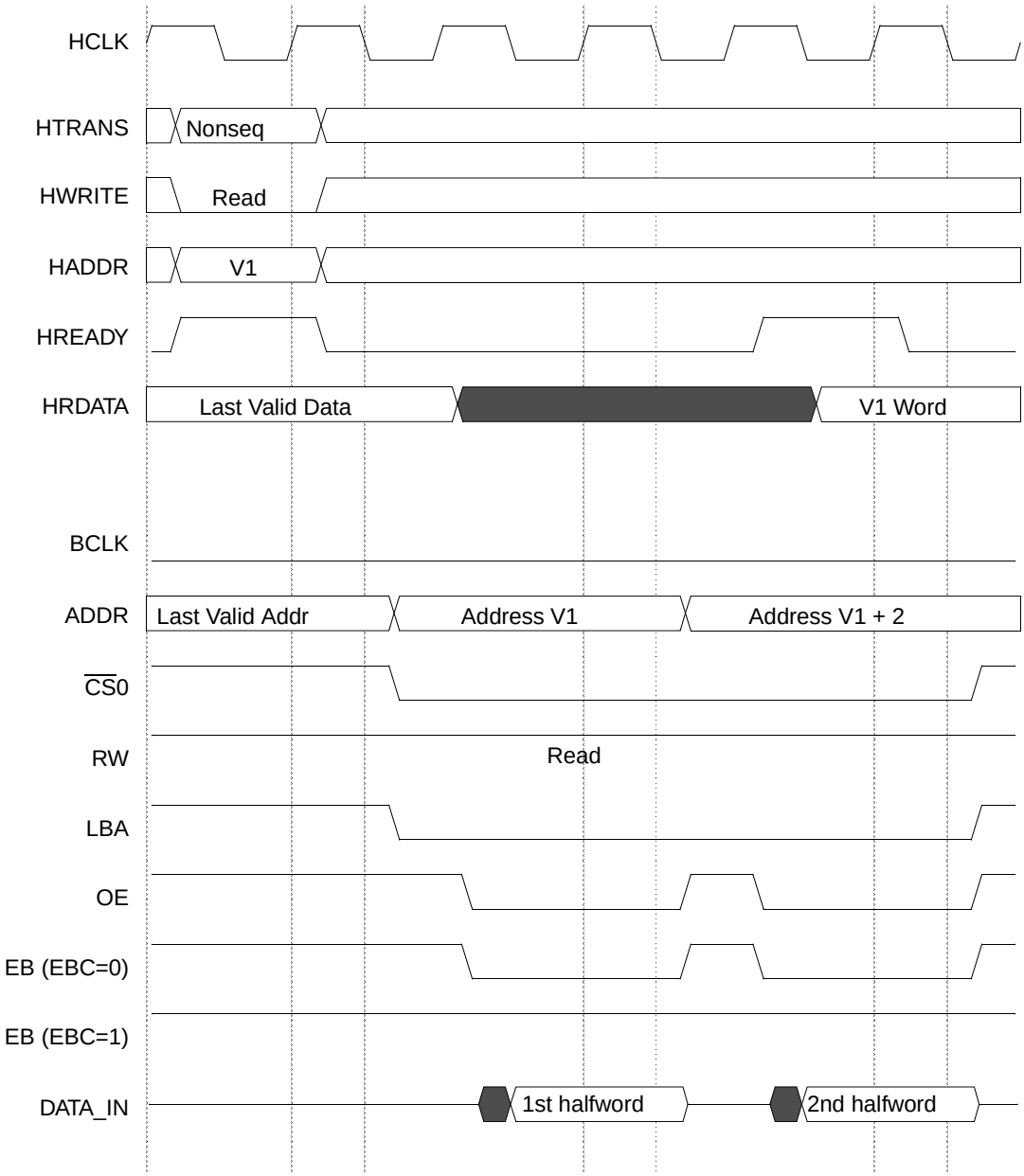


Figure 18-14. Read Access, WSC=1, OEA=1, EBRA=1

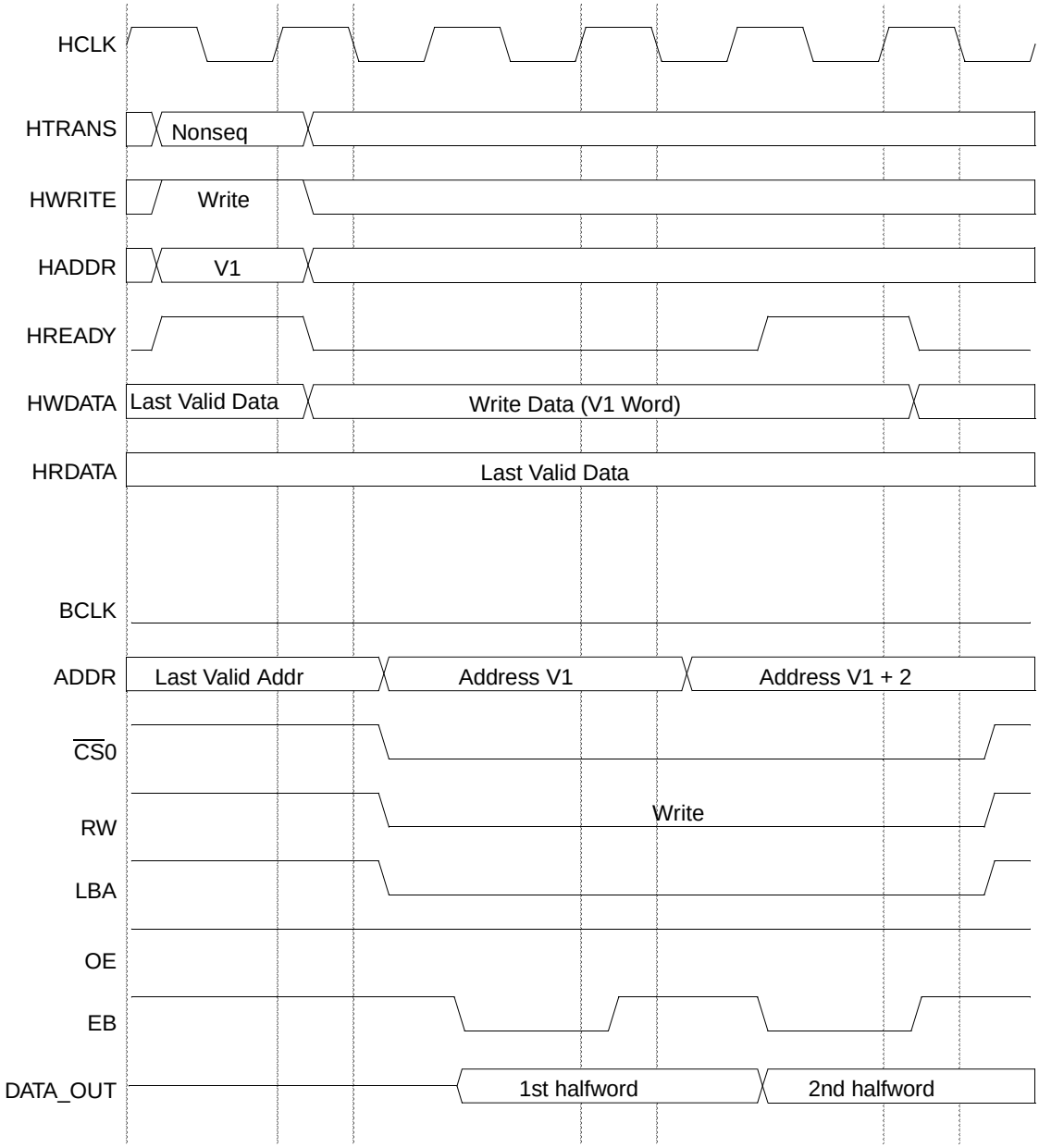


Figure 18-15. Write Access, WSC=1, EBWA=1, EBWN=1

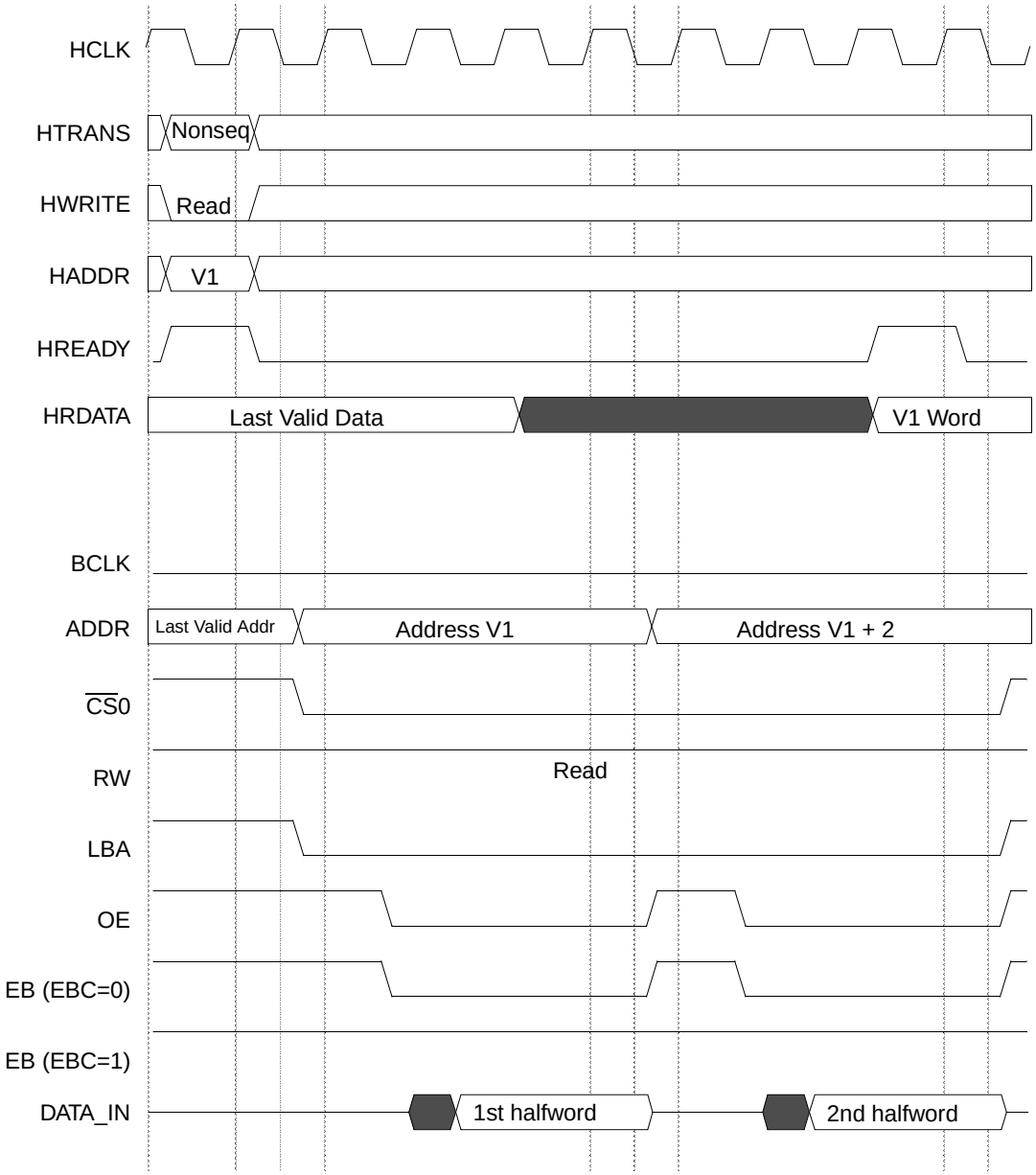


Figure 18-16. Read Access, WSC=3, OEA=2, EBRA=2

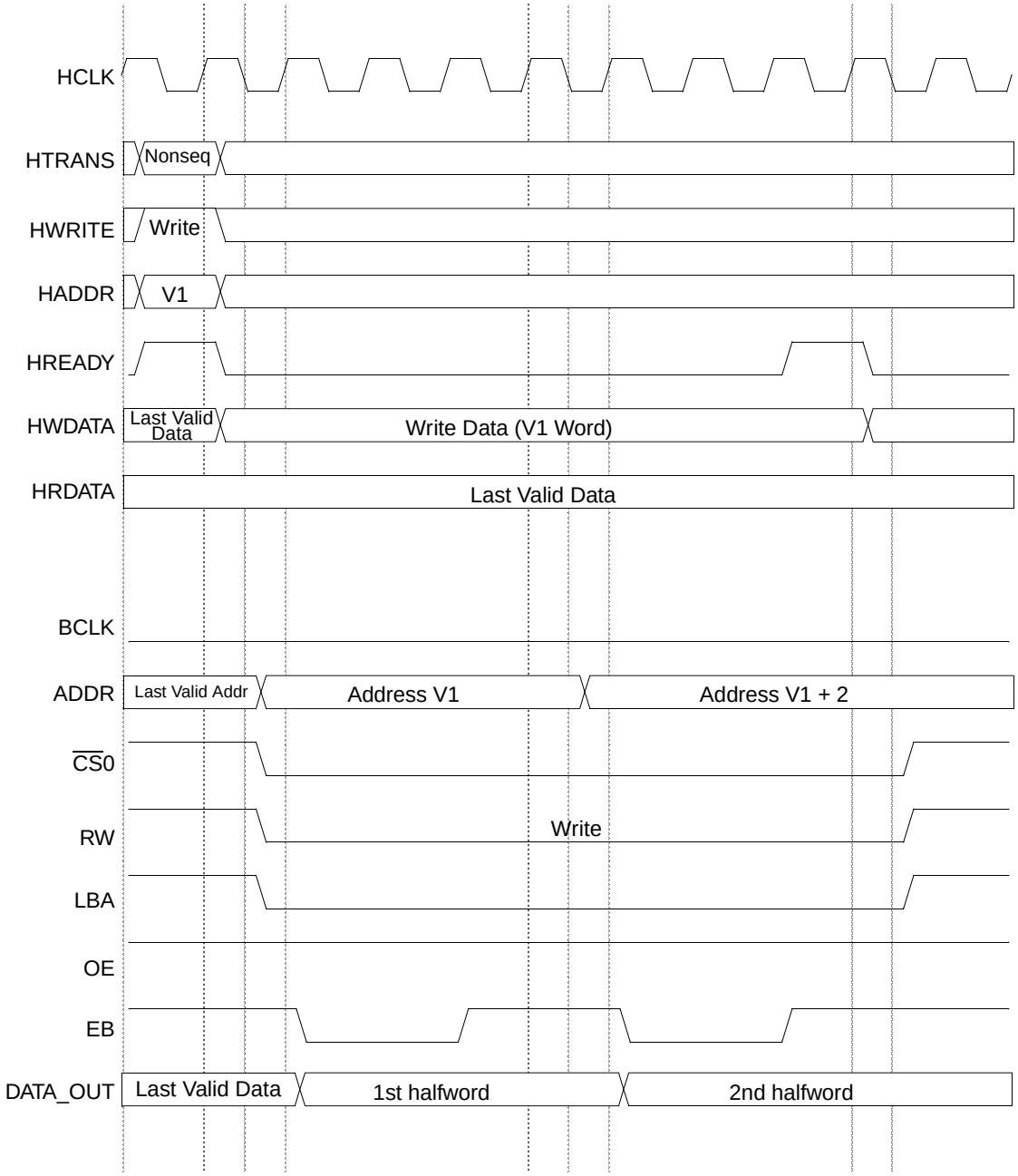


Figure 18-17. Write Access, WSC=3, EBWA=1, EBWN=3

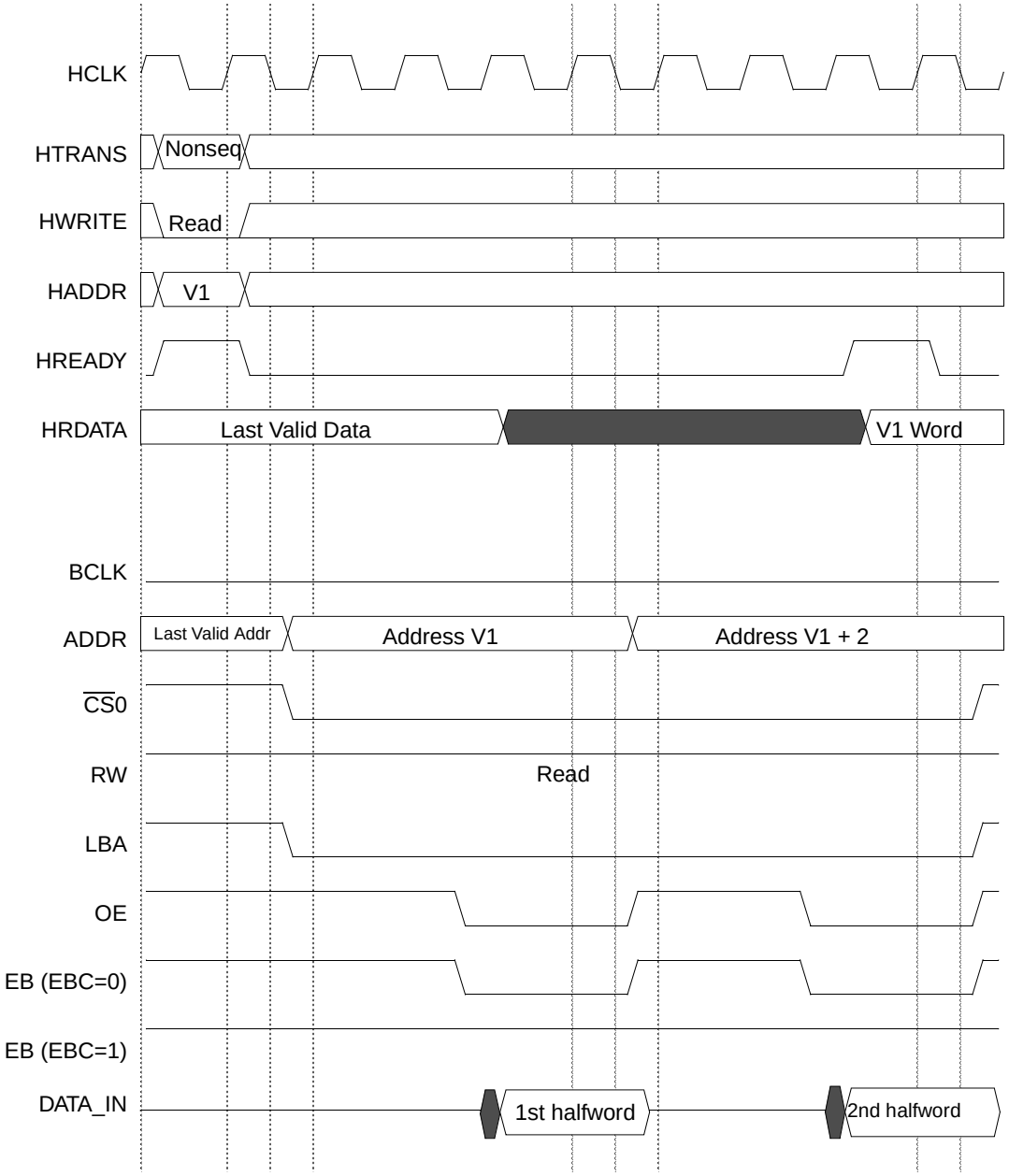


Figure 18-18. Read Access, WSC=3, OEA=4, EBRA=4

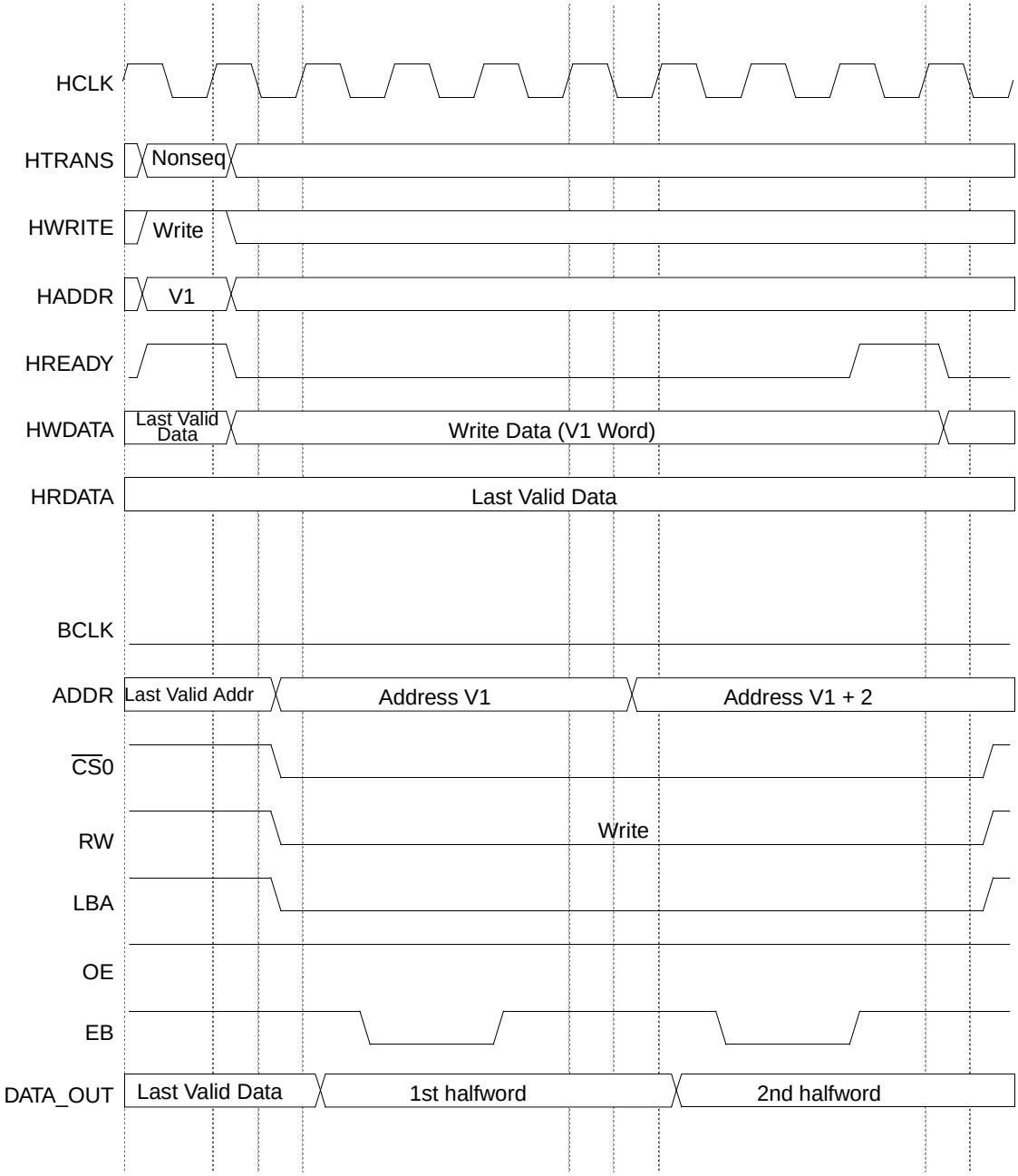


Figure 18-19. Write Access, WSC=3, EBWA2, EBWN=3

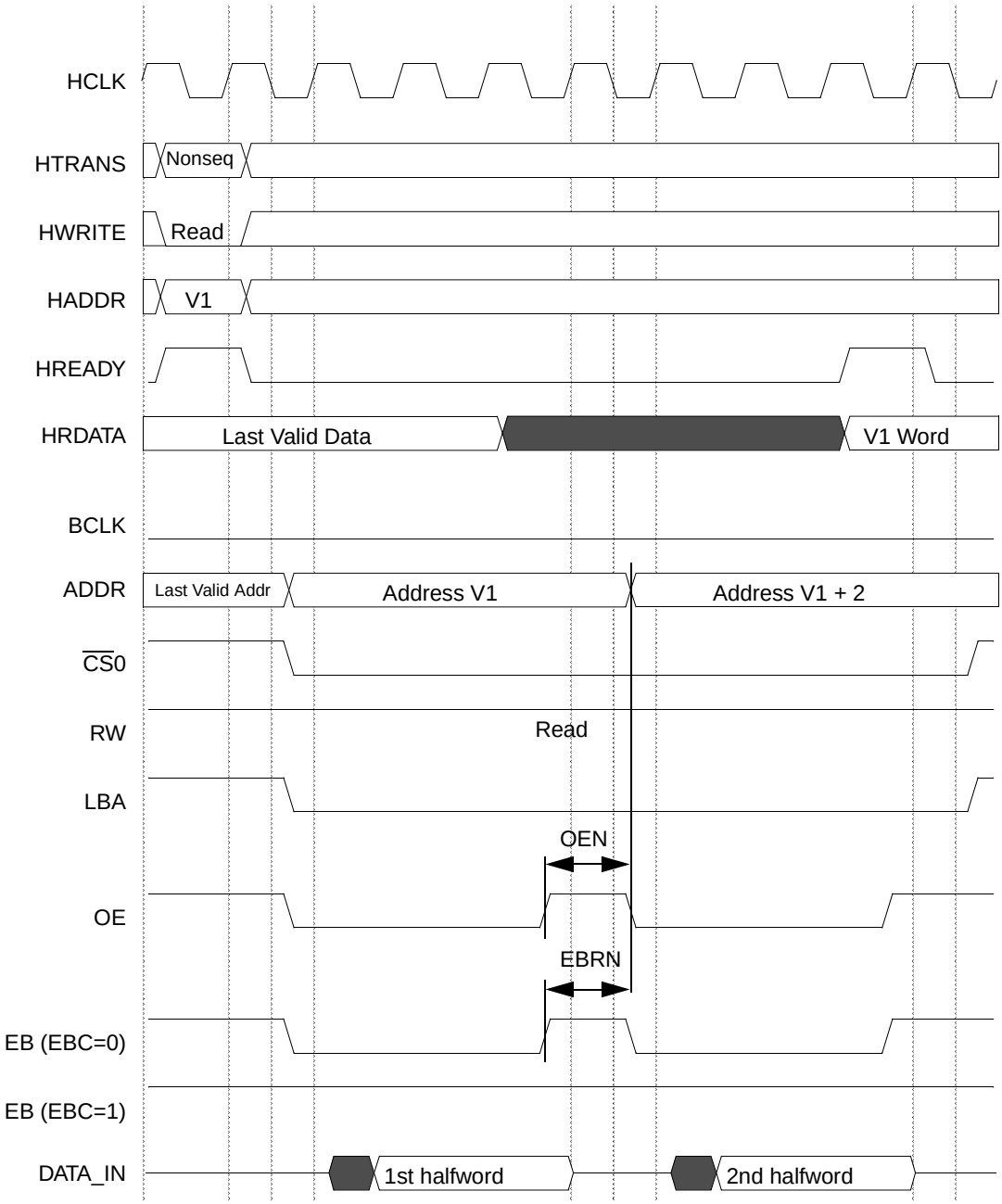


Figure 18-20. Read Access, WSC=3, OEN=2, EBRN=2

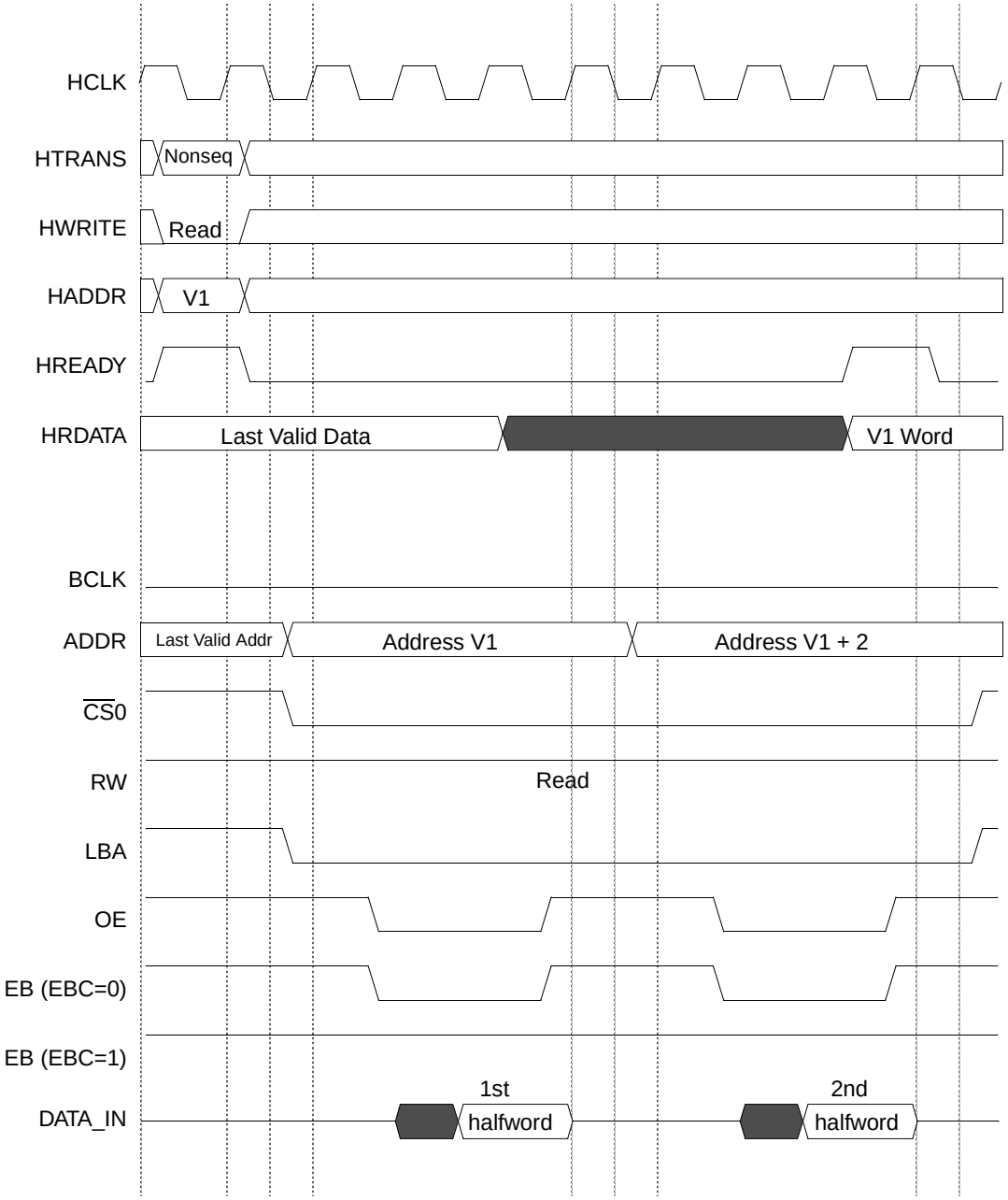


Figure 18-21. Read Access, WSC=3, OEA=2, OEN=2, EBRA=2, EBRN=2

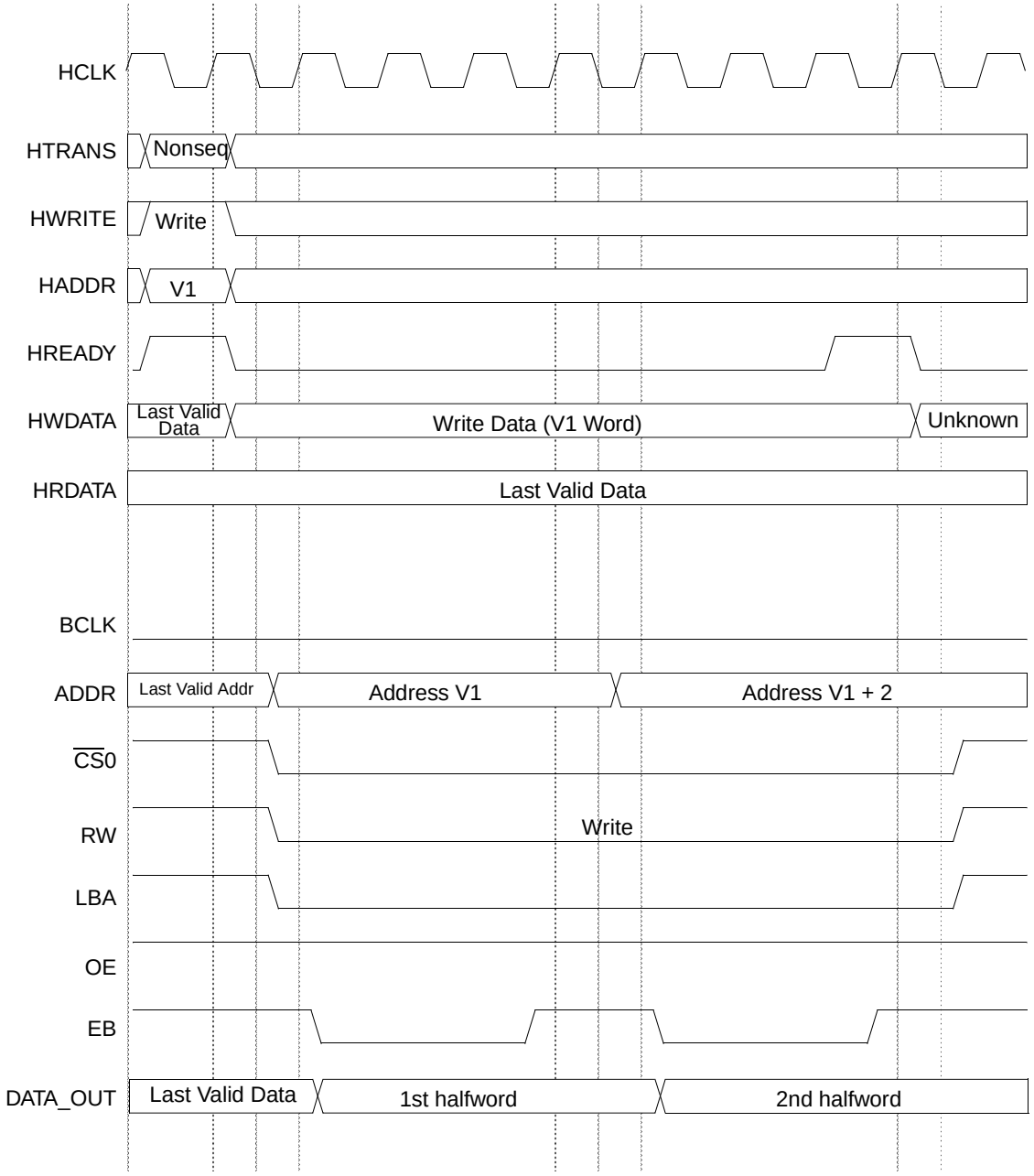


Figure 18-22. Write Access, WSC=2, WWS=1, EBWA=1, EBWN=2

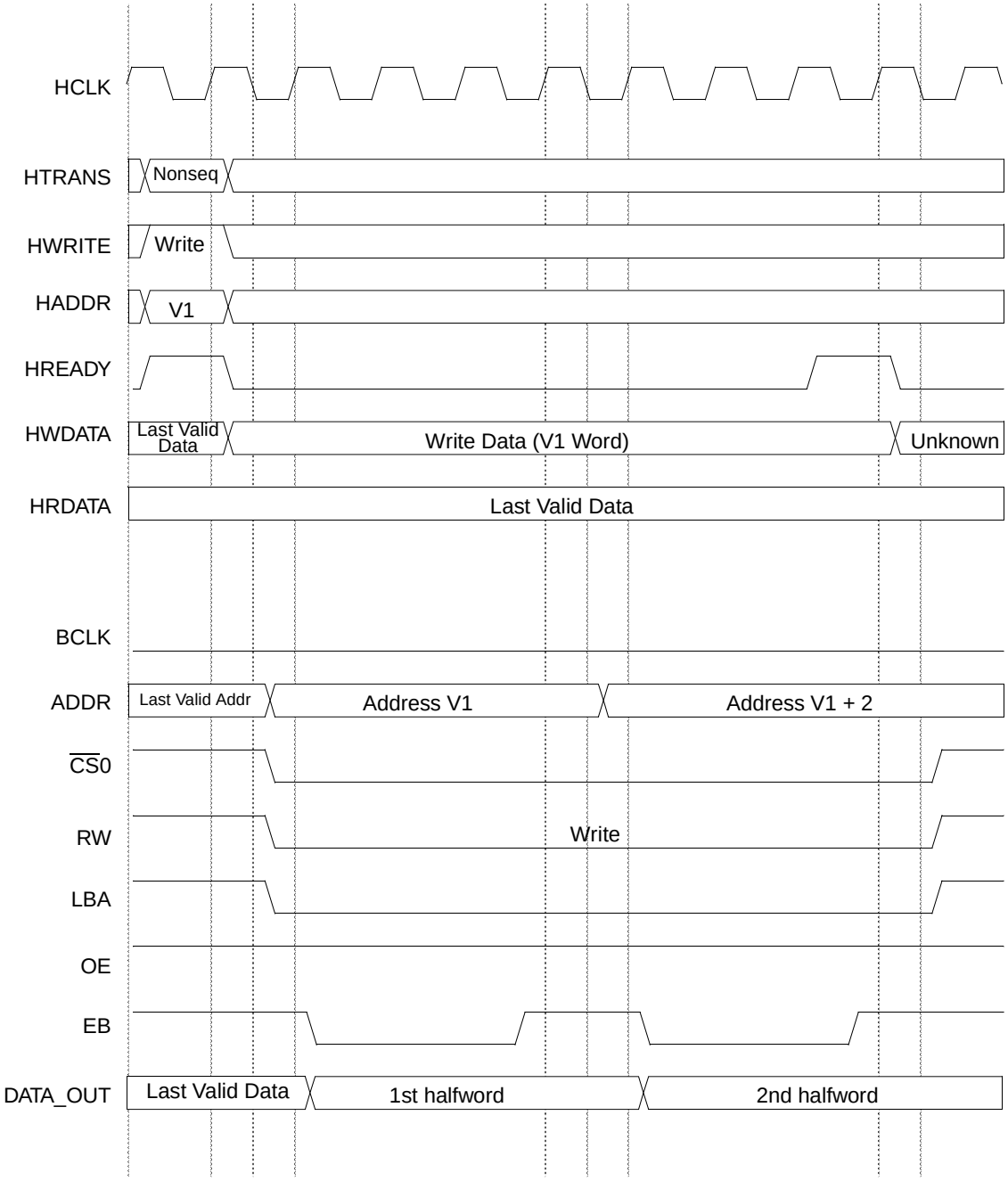


Figure 18-23. Write Access, WSC=1, WWS=2, EBWA=1, EBWN=2

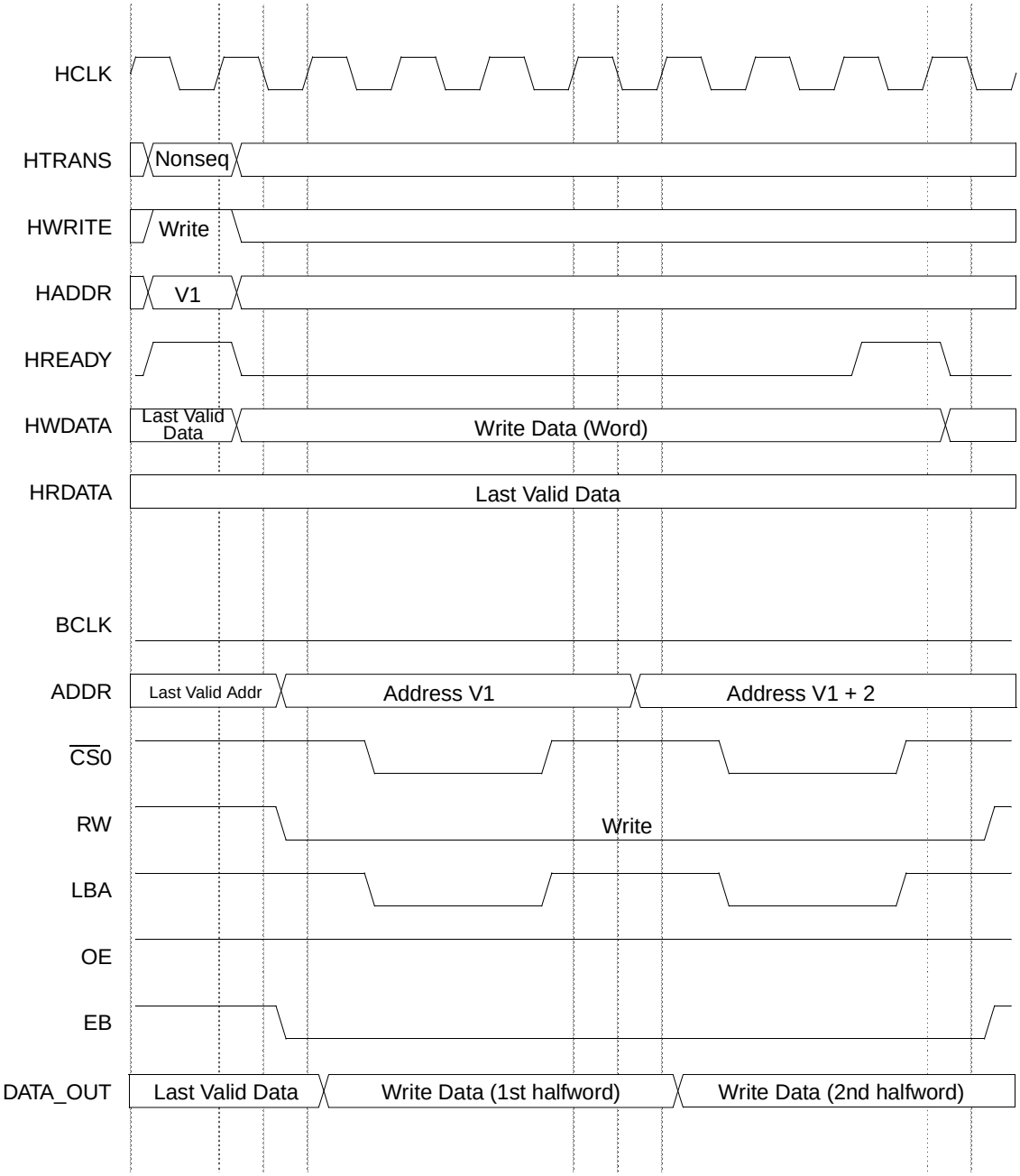


Figure 18-24. Write Access, WSC=2, CSA=1, WWS=1, CSN=1

18.7.2 Page Mode Timing Diagrams

18.7.2.1 AHB Word Accesses to Halfword Width Memory

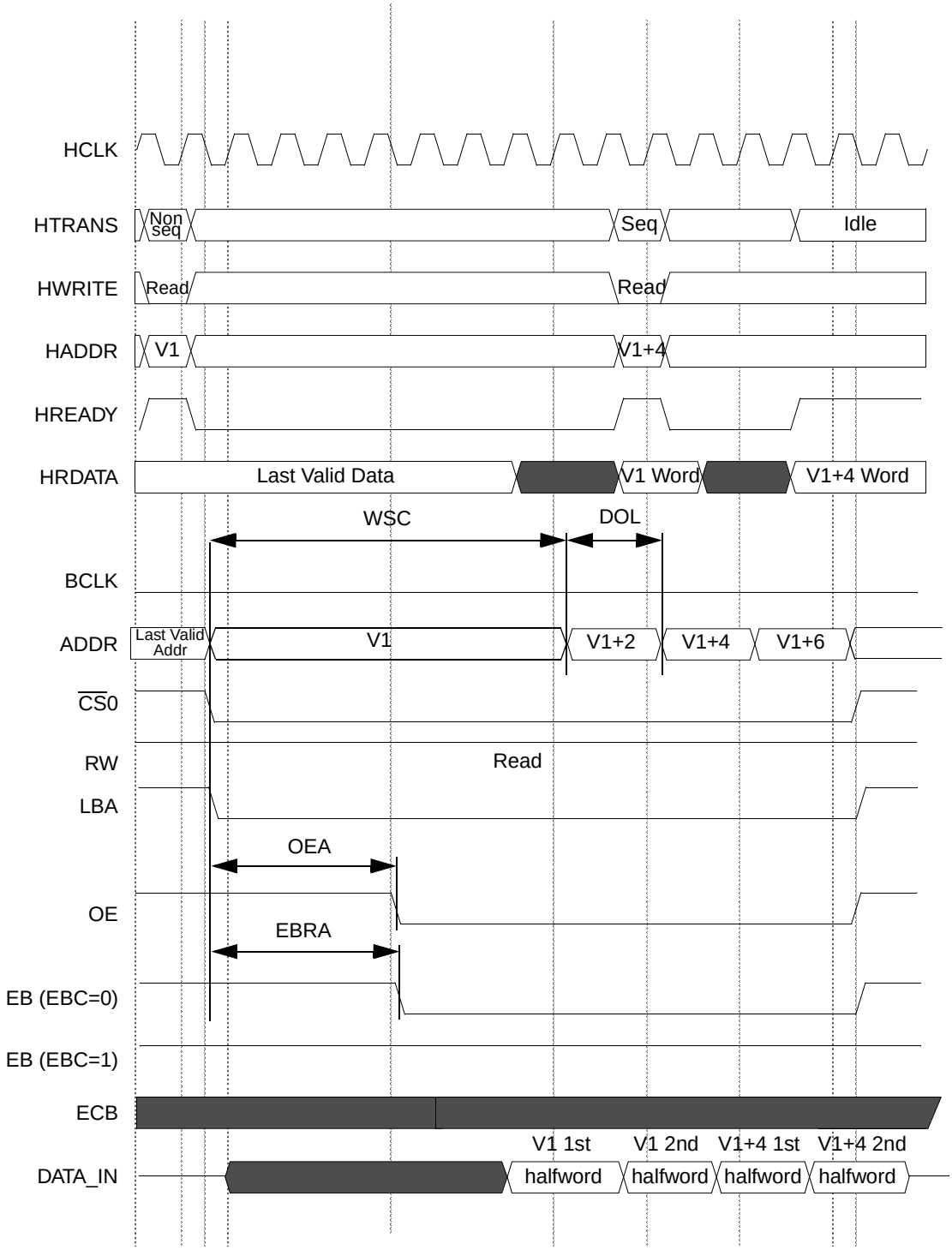


Figure 18-25. Sequential Read Access, WSC=7, OEA=8, PME=1, SYNC=1, DOL=1, EBRA=8

18.7.3 DTACK Mode Memory Accesses Timing Diagrams

18.7.3.1 AHB Word Accesses to Word-Width Memory

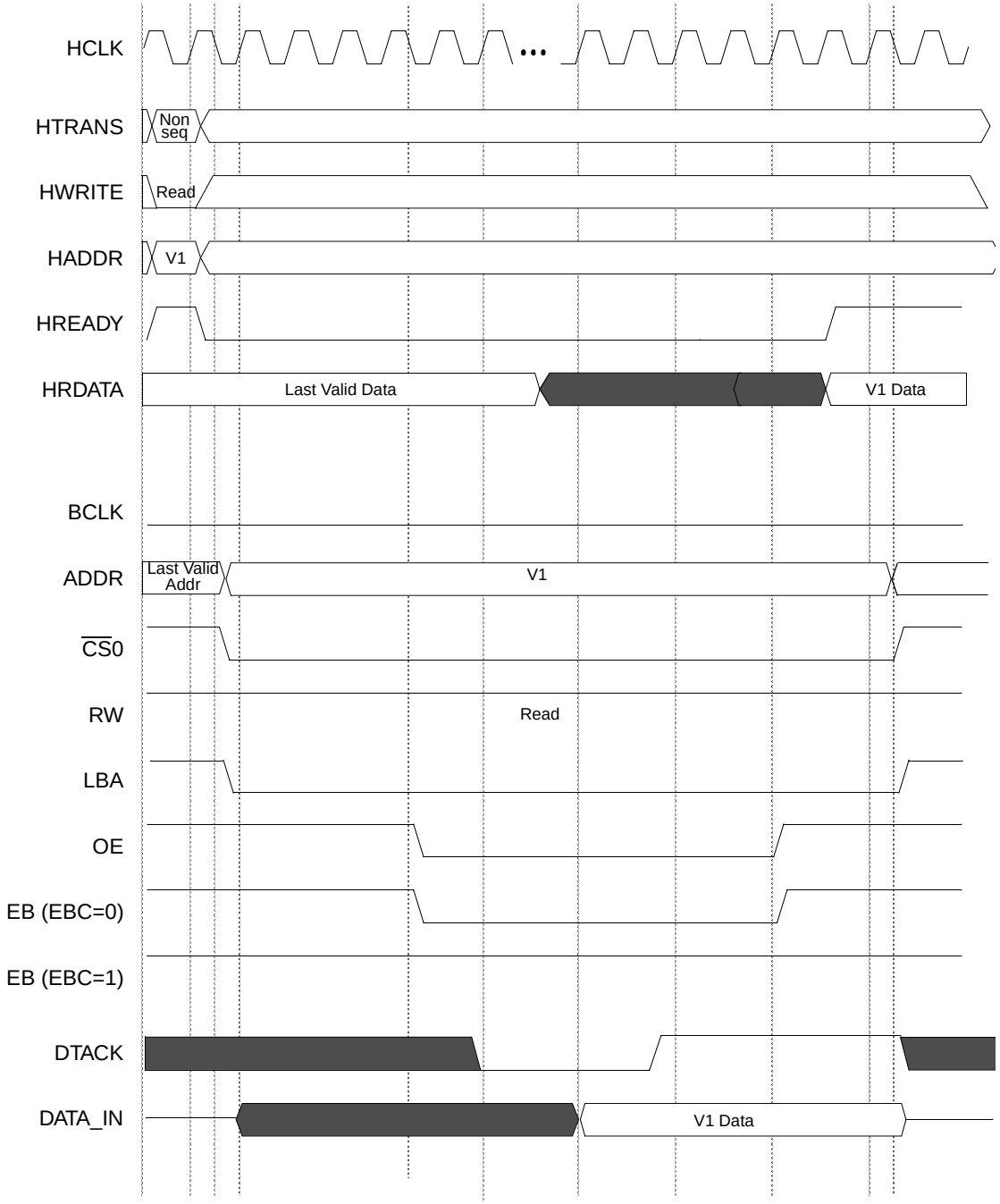


Figure 18-26. Read Access, WSC=3F, OEA=8, OEN=5, EBRA=8, EBRN=5

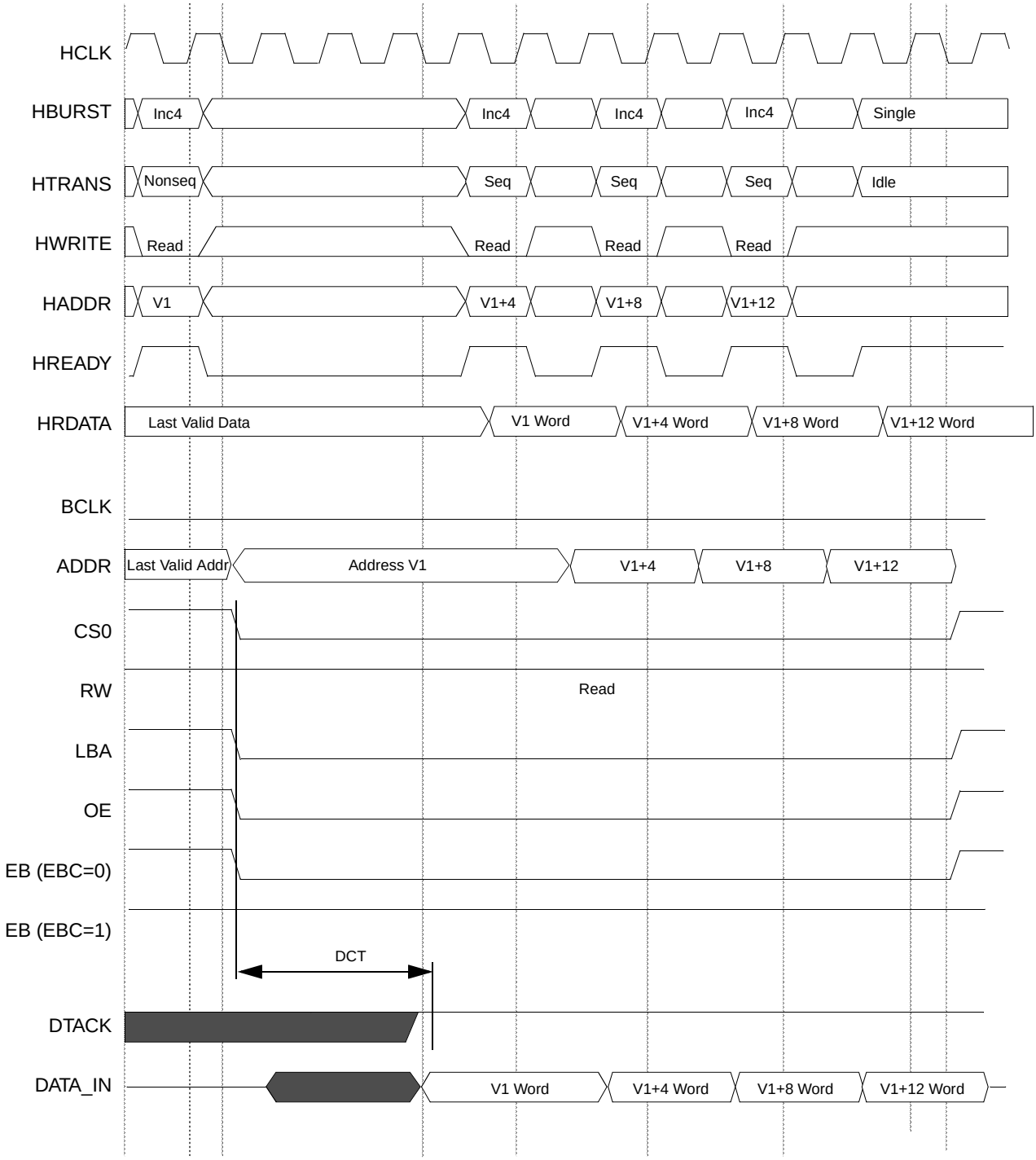


Figure 18-27. Sequential Read Accesses, WSC=1, EW=1, DCT=1

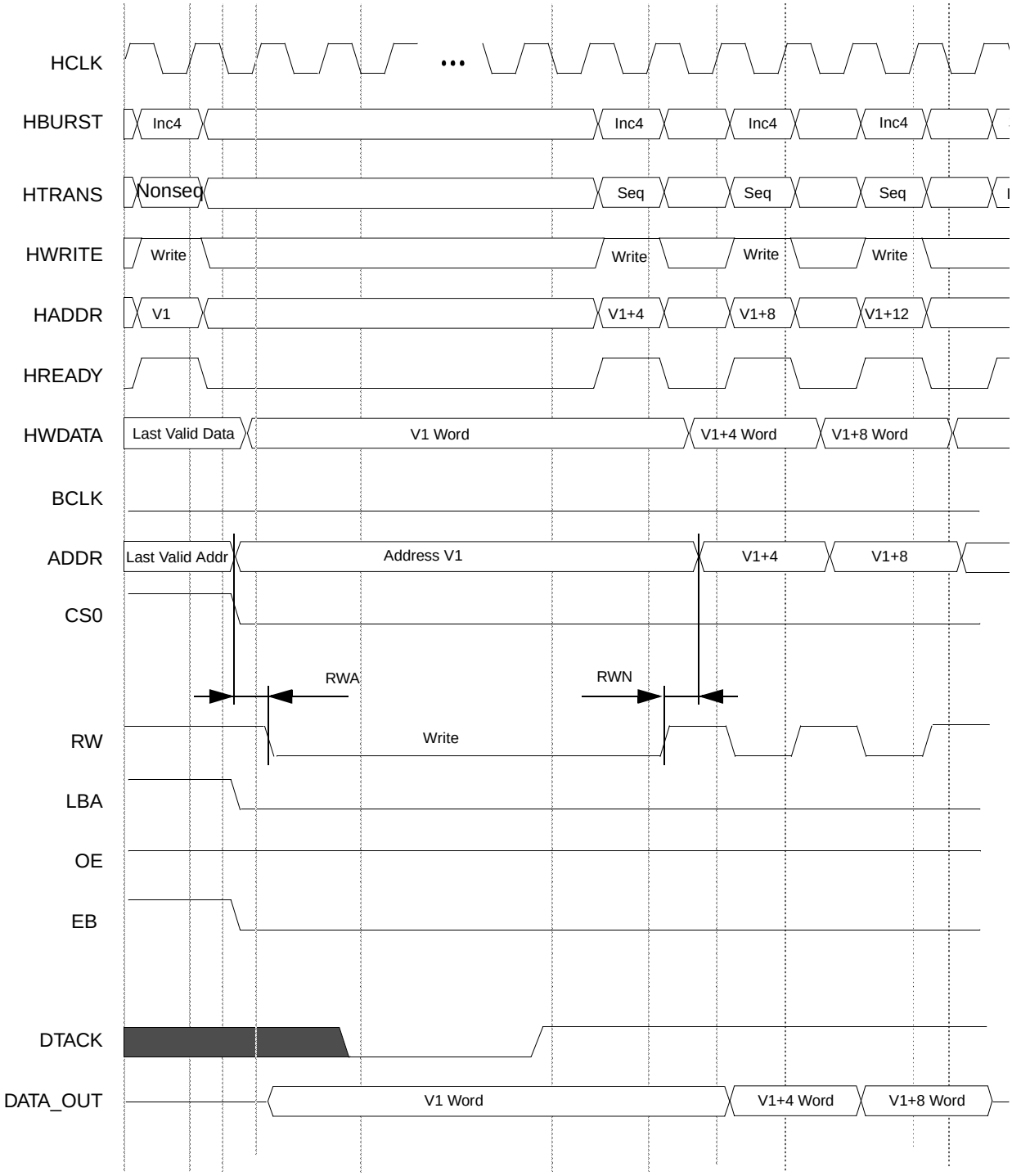


Figure 18-28. Sequential Write Accesses, WSC=1, EW=1, RWA=1, RWN=1

18.7.4 Burst Memory Accesses Timing Diagrams

18.7.4.1 AHB Word Accesses to Halfword Width Memory

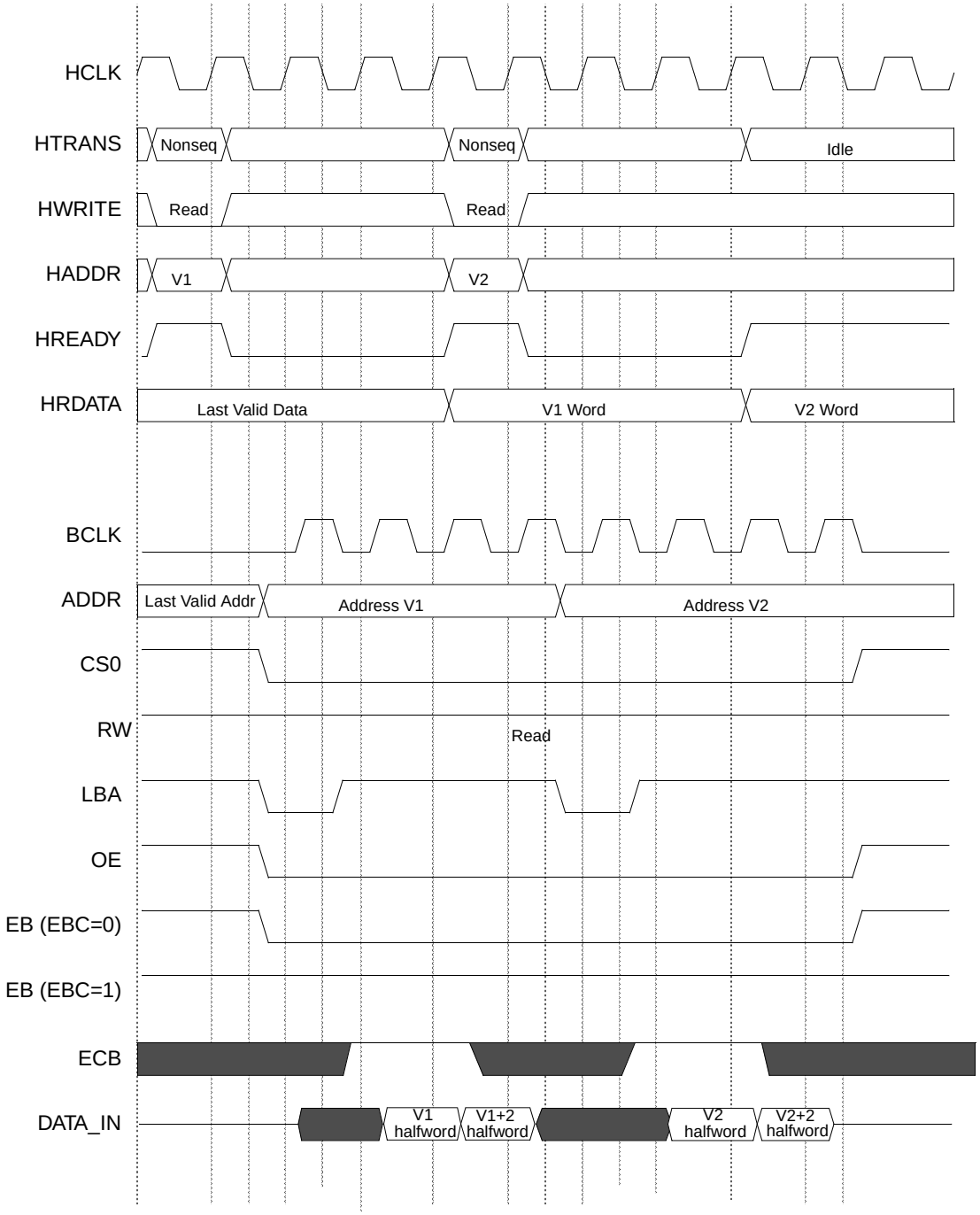


Figure 18-29. Non-Sequential Read Accesses, WSC=2, SYNC=1, DOL=0

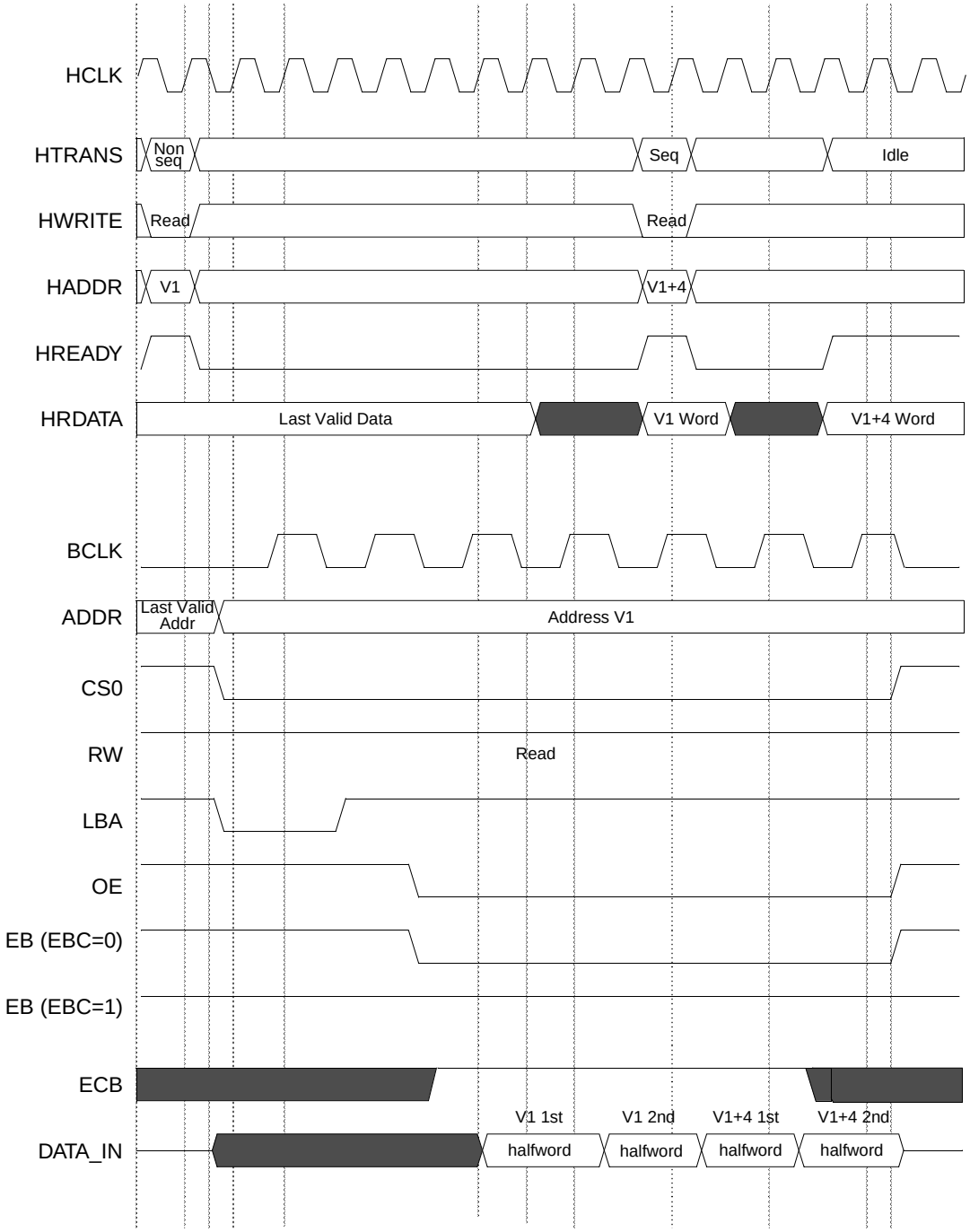


Figure 18-30. Sequential Read Access, WSC=7, OEA=8, SYNC=1, DOL=1, BCD=1, BCS=1, EBRA=8

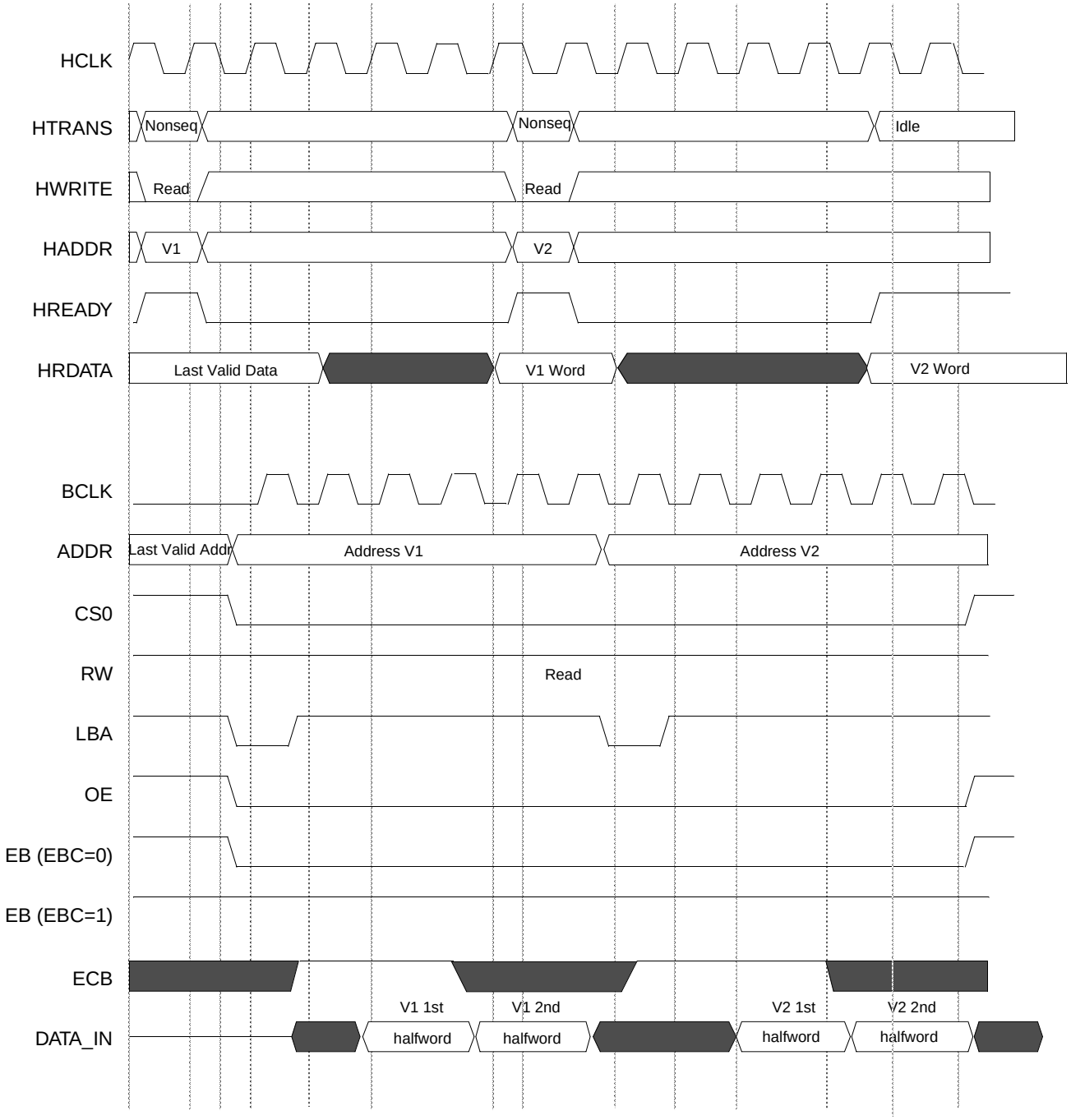


Figure 18-31. Non-Sequential Read Accesses, WSC=3, SYNC=1, DOL=1

18.7.4.2 AHB Accesses to Word-Width Burst Memory

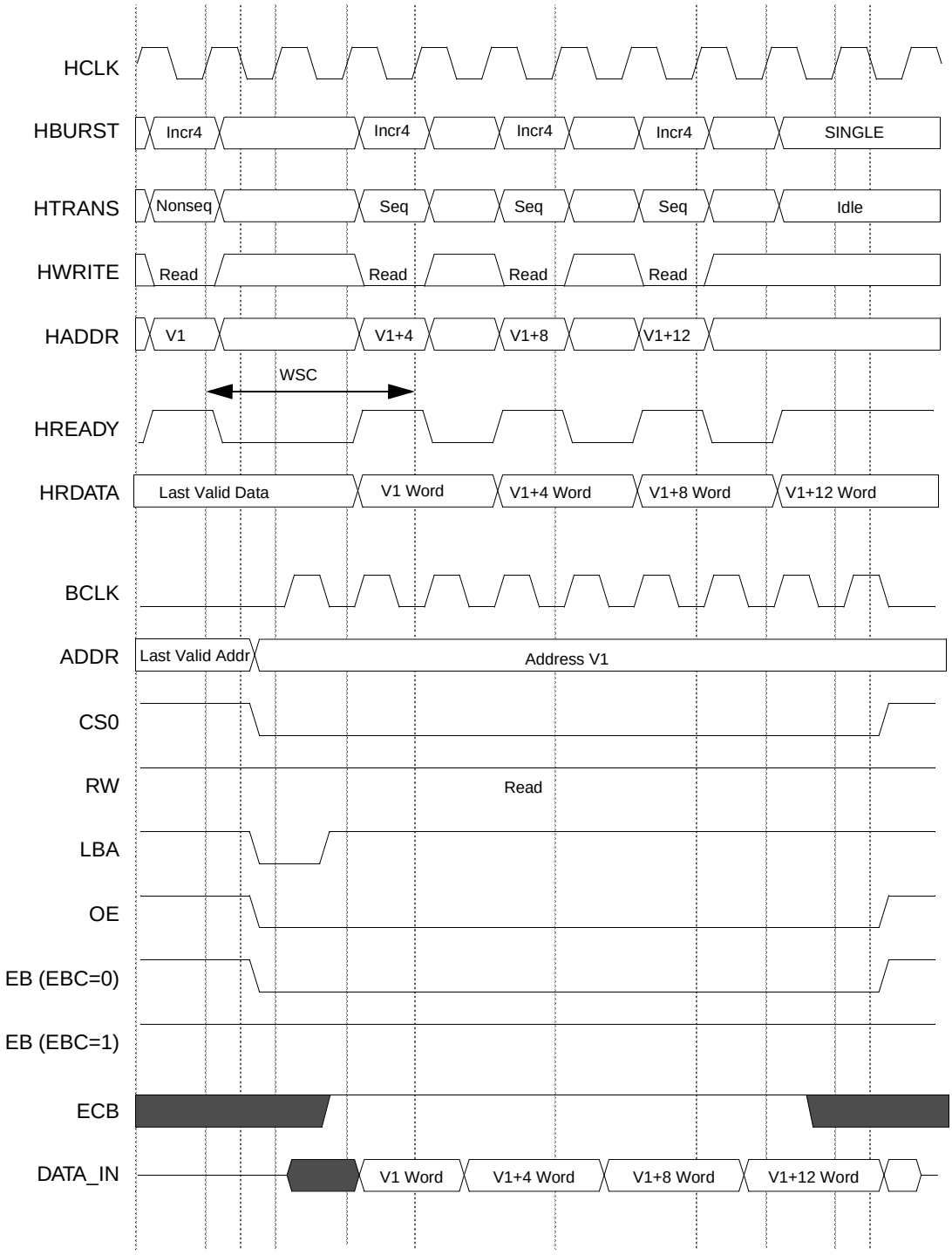


Figure 18-32. Increment 4 AHB Read Access, WSC=2, SYNC=1, DOL=1, WRAP=0

In the accesses on Figure 18-32 any address may be a four word boundary address, but not a memory boundary address.

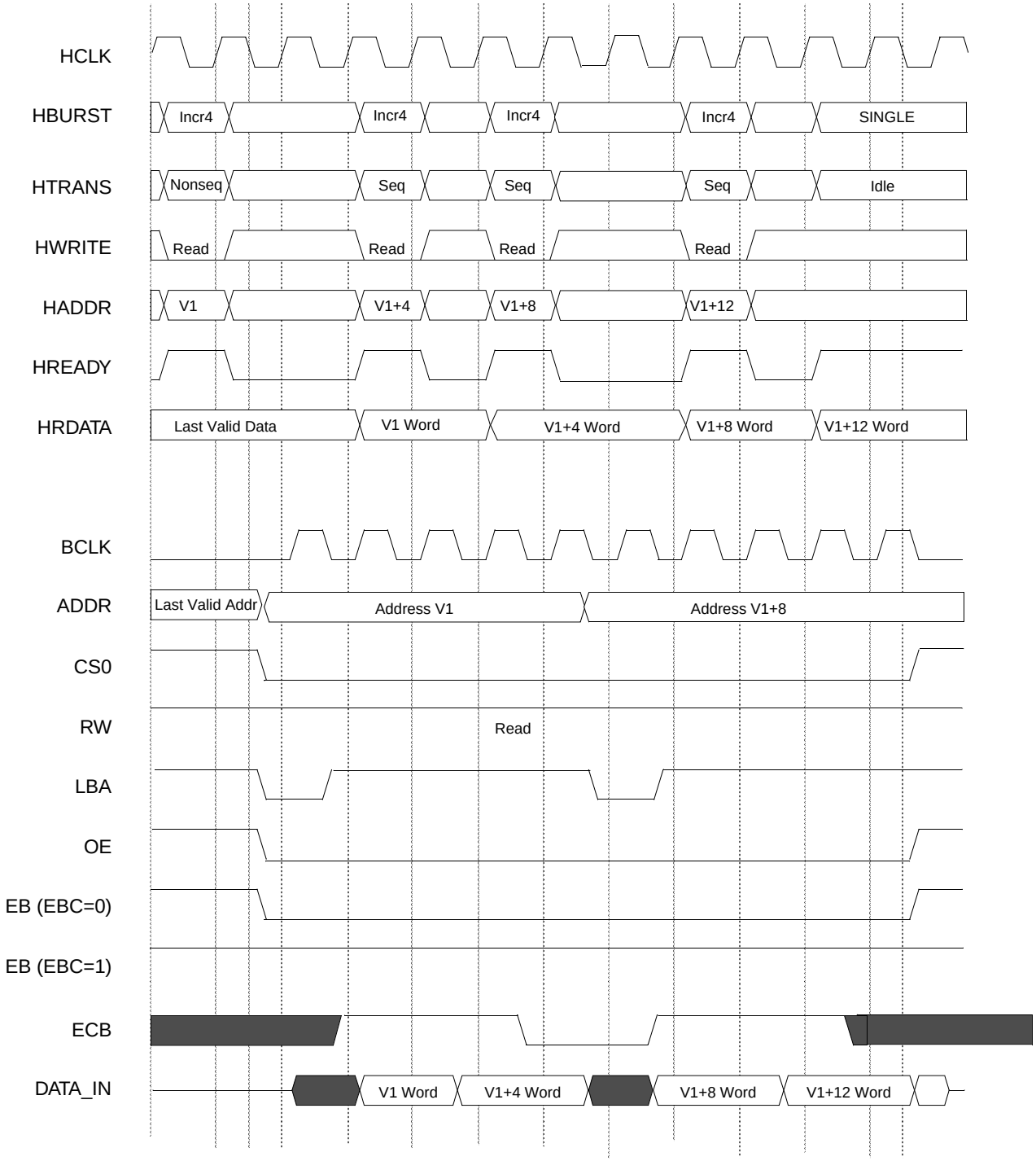


Figure 18-33. Increment 4 AHB Read Access, WSC=2, SYNC=1, DOL=1, WRAP=0

In the accesses on Figure 18-33 (V1+8) is a memory boundary address and may be a four word boundary address.

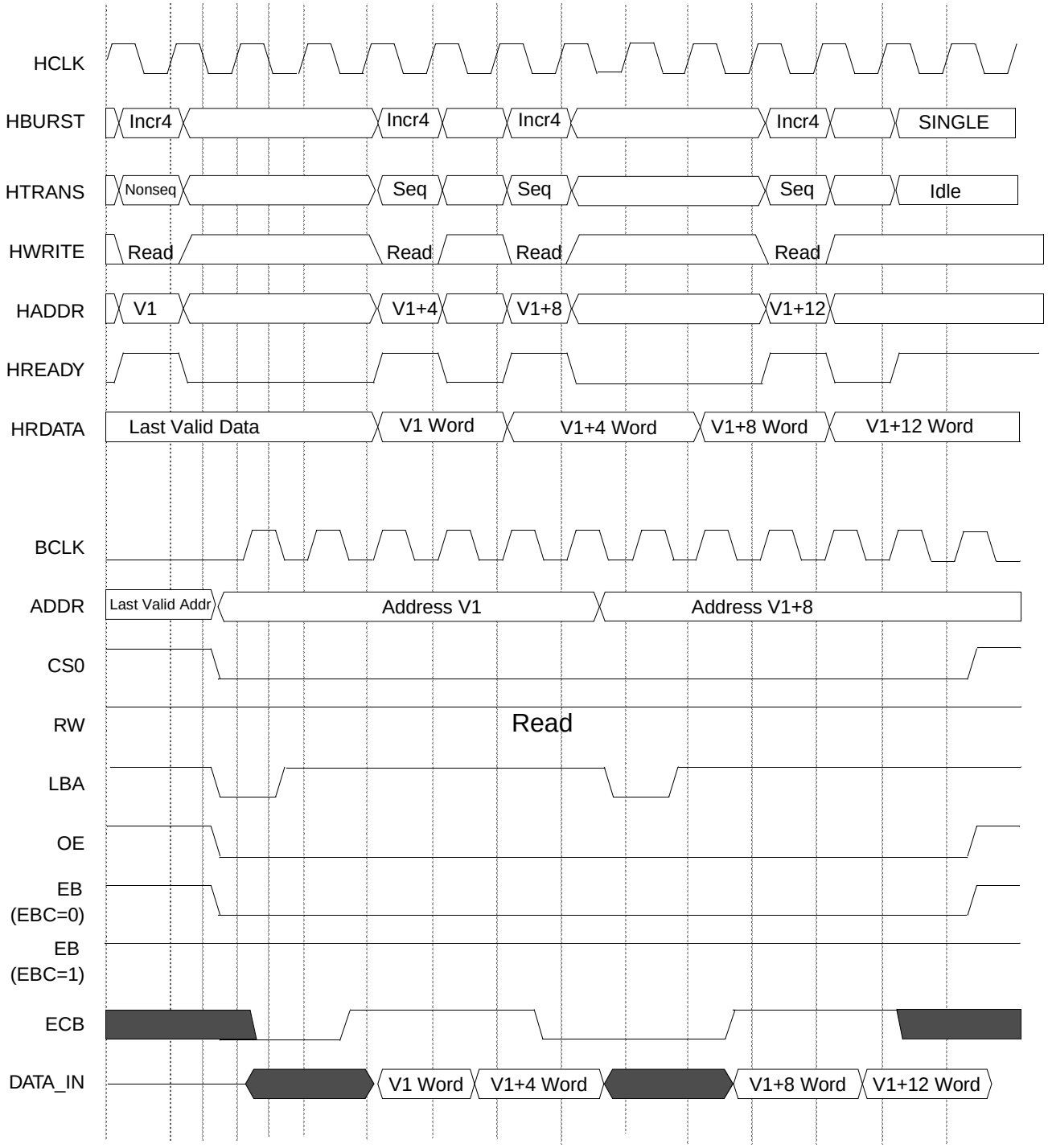


Figure 18-34. Increment 4 AHB Read Access, WSC=3, SYNC=1, DOL=1, WRAP=0, EW=0

In the accesses on Figure 18-34 (V1+8) is a memory boundary address and may be a four word boundary address.

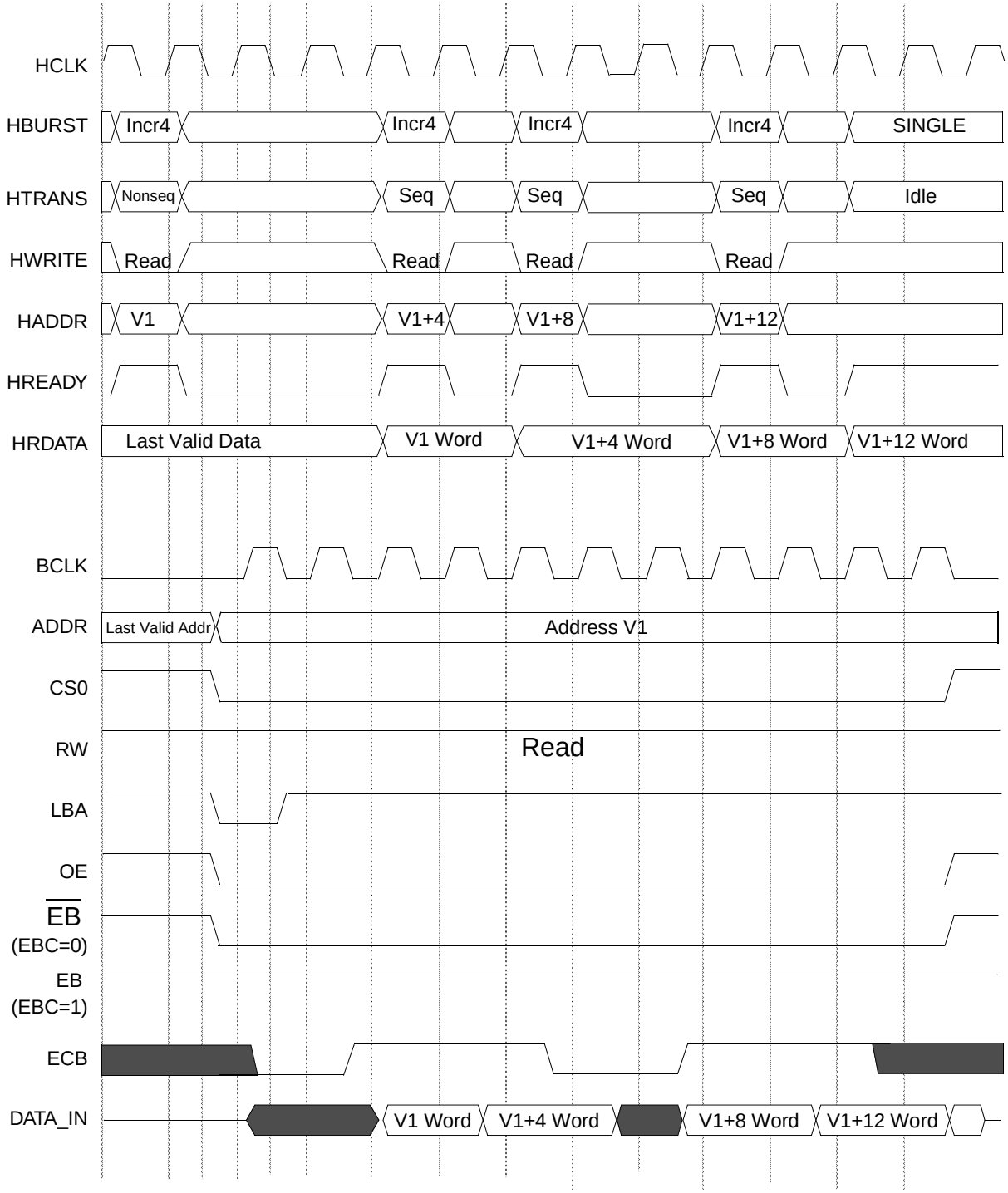


Figure 18-35. Increment 4 AHB Read Access, WSC=3, SYNC=1, DOL=1, WRAP=0, EW=1

In the accesses on Figure 18-35 and Figure 18-36 (V1+8) is a memory boundary address and may be a four word boundary address.

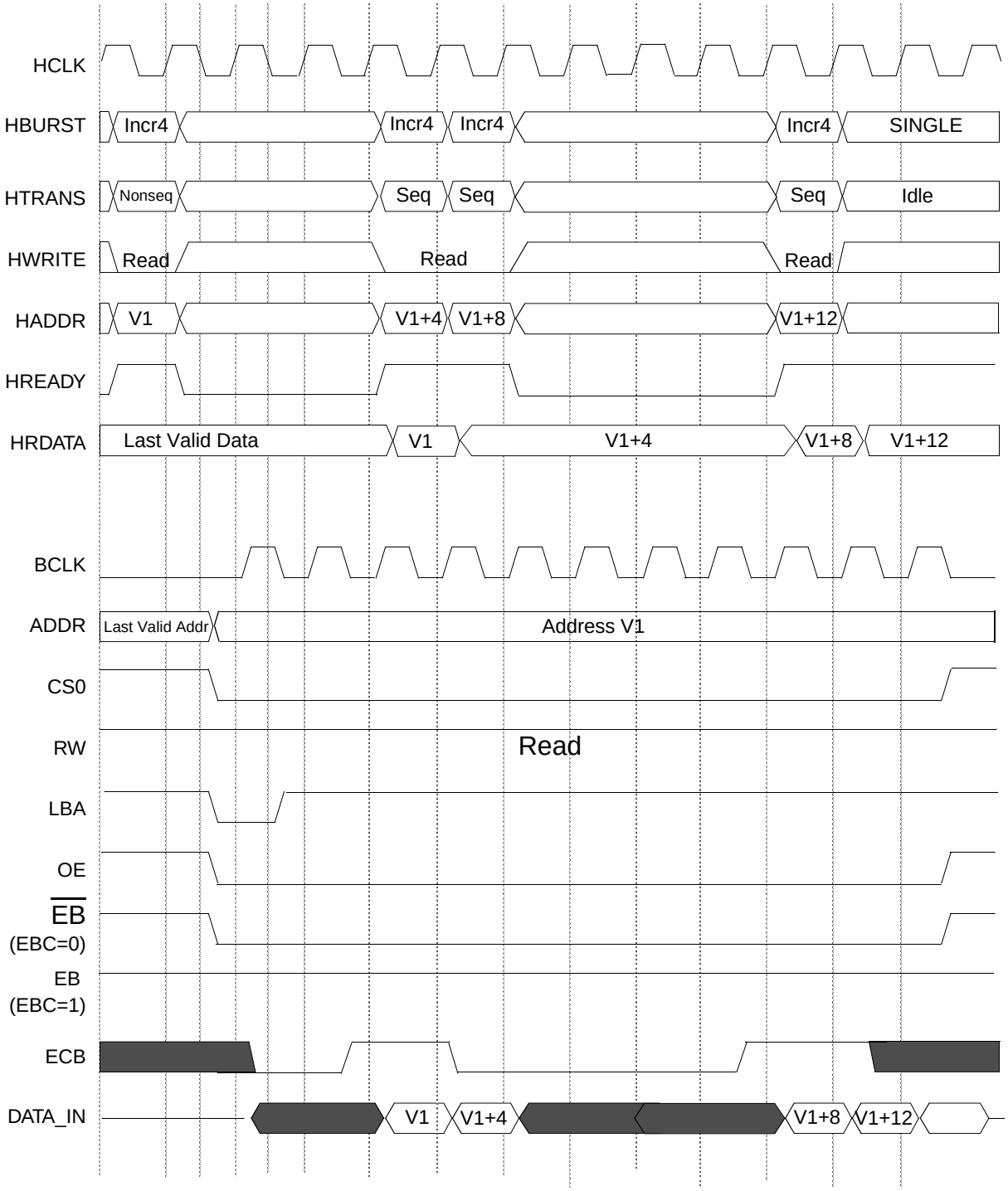


Figure 18-36. Increment 4 AHB Read Access, WSC=3, SYNC=1, WRAP=0, EW=1

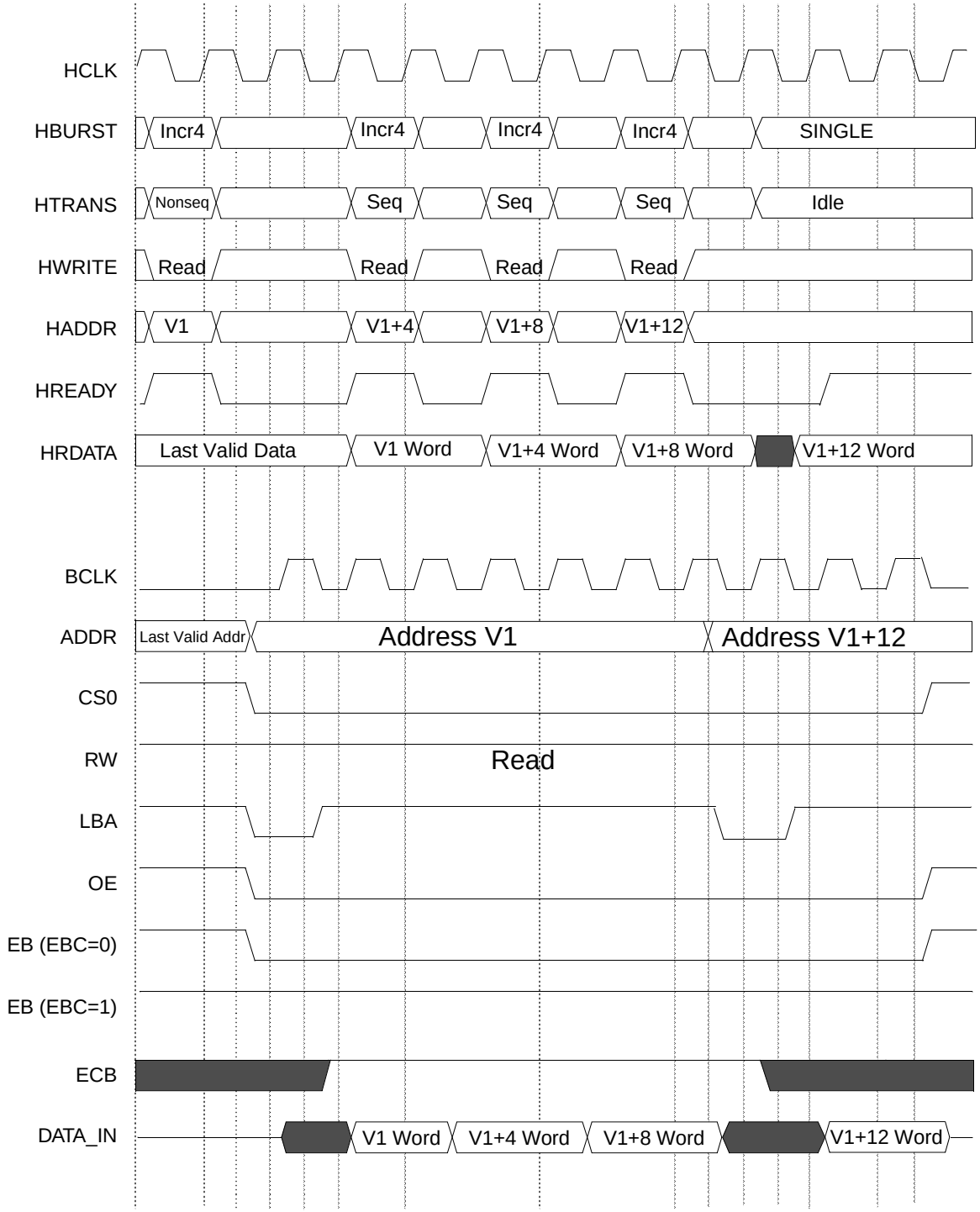


Figure 18-37. Increment 4 AHB Read Access, WSC=2, SYNC=1, DOL=1, WRAP=1, PSZ=0

In the accesses on Figure 18-37 (V1+12) is a four-word boundary address.

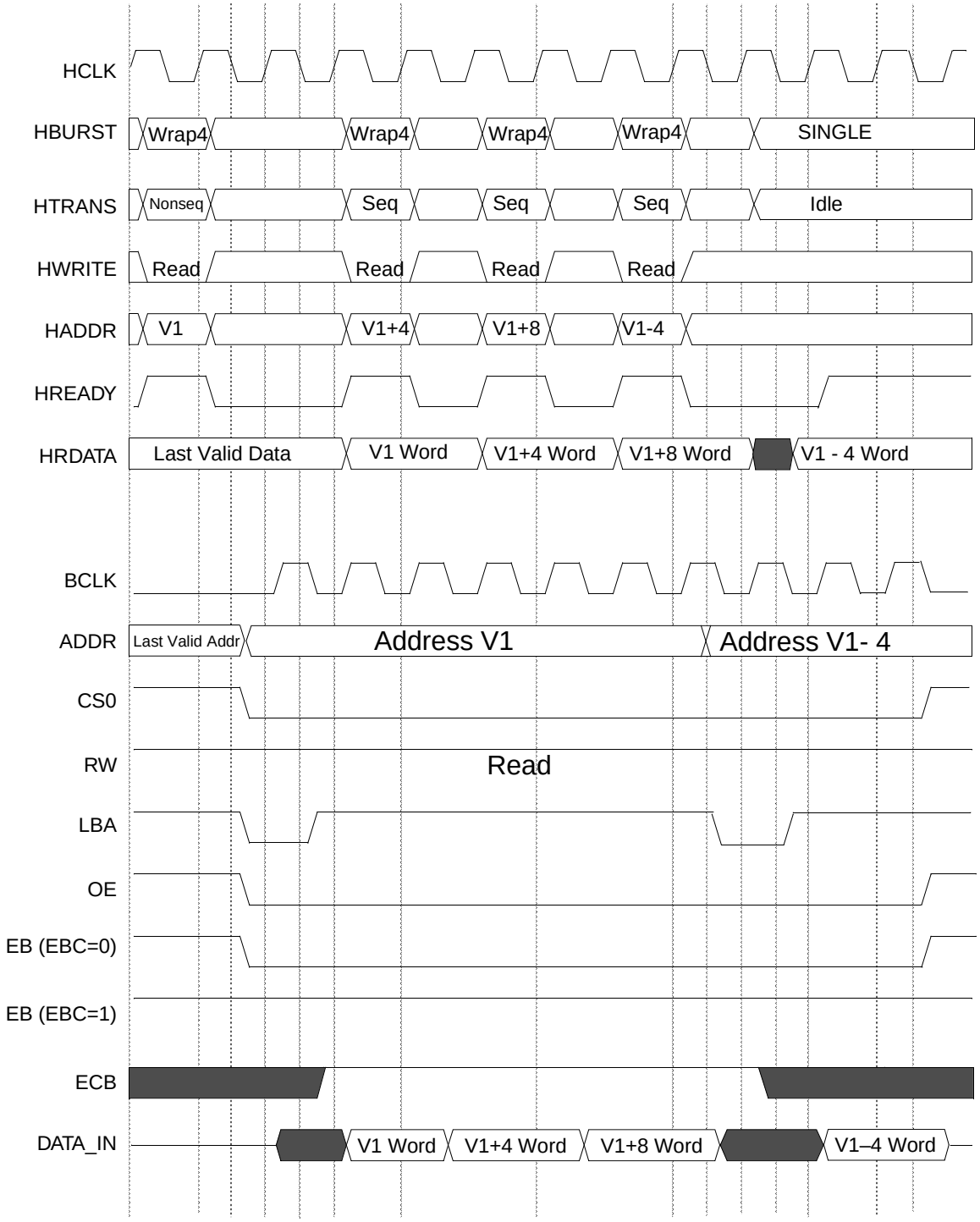


Figure 18-38. Wrap 4 AHB Read Access, WSC=2, SYNC=1, DOL=1, WRAP=0

In the accesses on Figure 18-38 (V1-4) is a four-word boundary address.

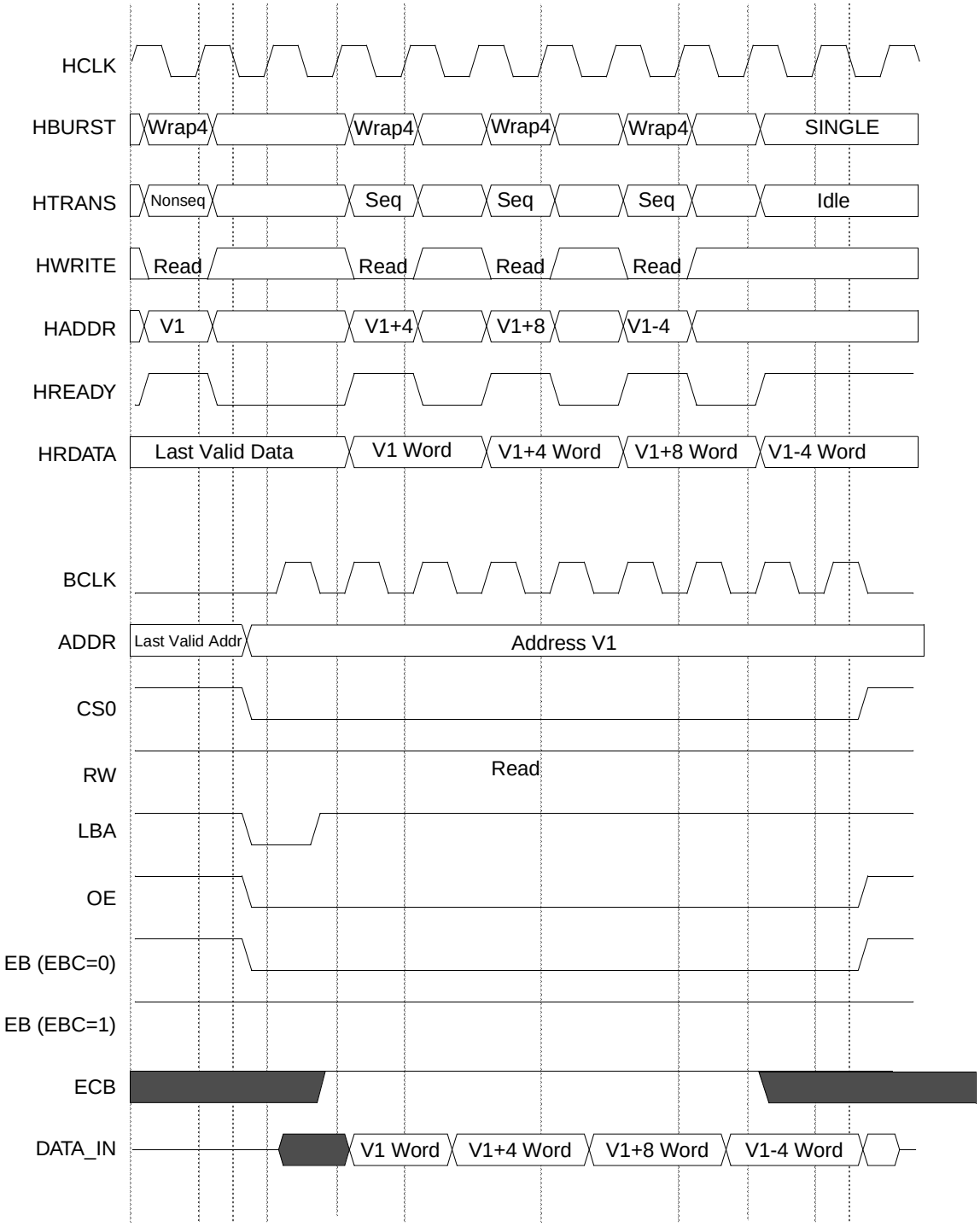


Figure 18-39. Wrap 4 AHB Read Access, WSC=2, SYNC=1, DOL=1, WRAP=1, PSZ=0

In the accesses on Figure 18-39 (V1-4) is a four-word boundary address.

18.7.5 Synchronous Accesses Timing Diagrams with PSRAM

18.7.5.1 AHB Sequential Accesses to Halfword Width PSRAM Memory

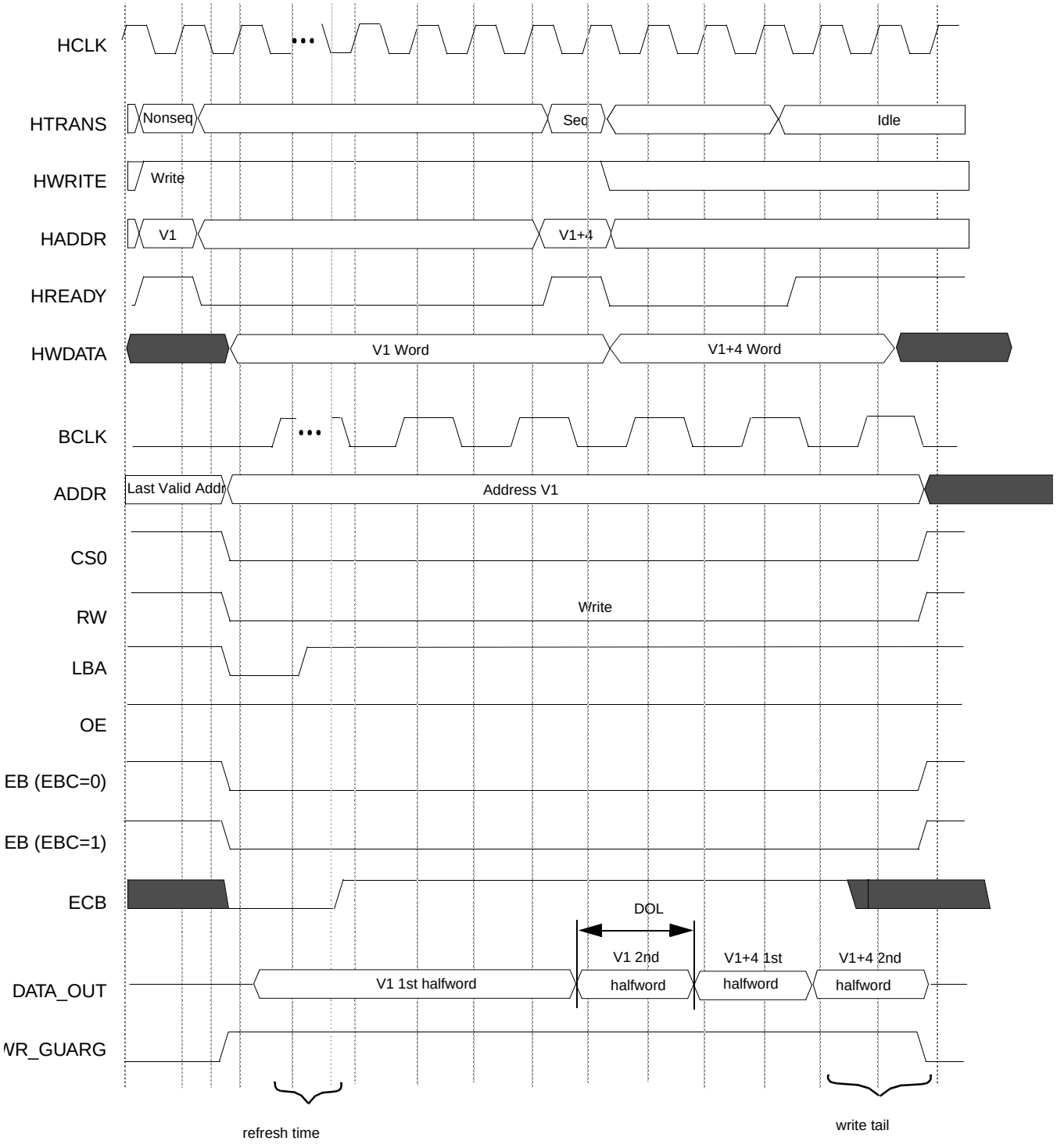


Figure 18-40. Write Access, BCD=1, BCS=1, WSC=3, SYNC=1, DOL=1, EW=1, PSR=1

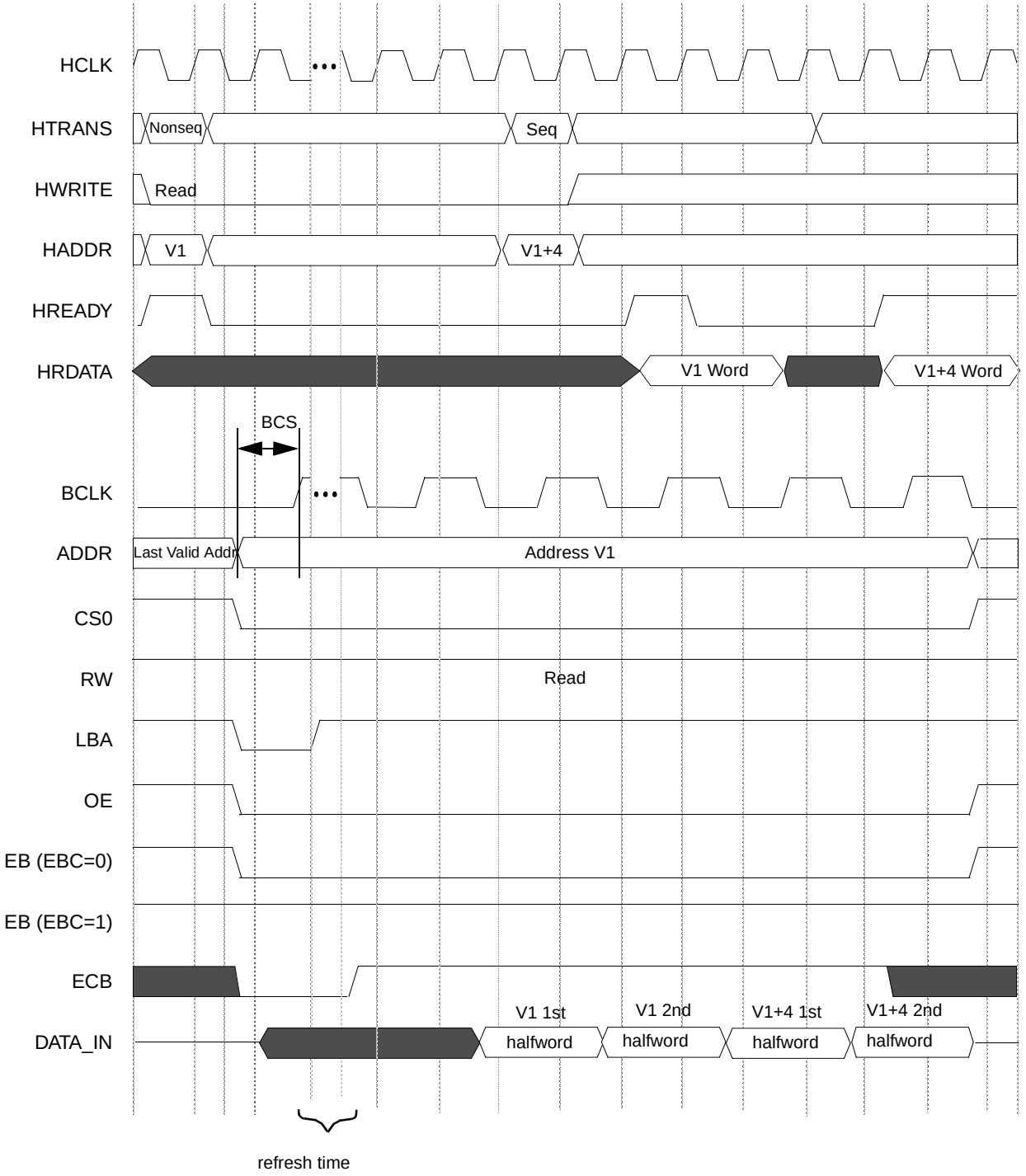


Figure 18-41. Read Access, BCD=1, BCS=1, WSC=3, SYNC=1, DOL=1, EW=1, PSR=1

18.7.5.2 AHB Sequential Accesses to Word-width PSRAM Memory

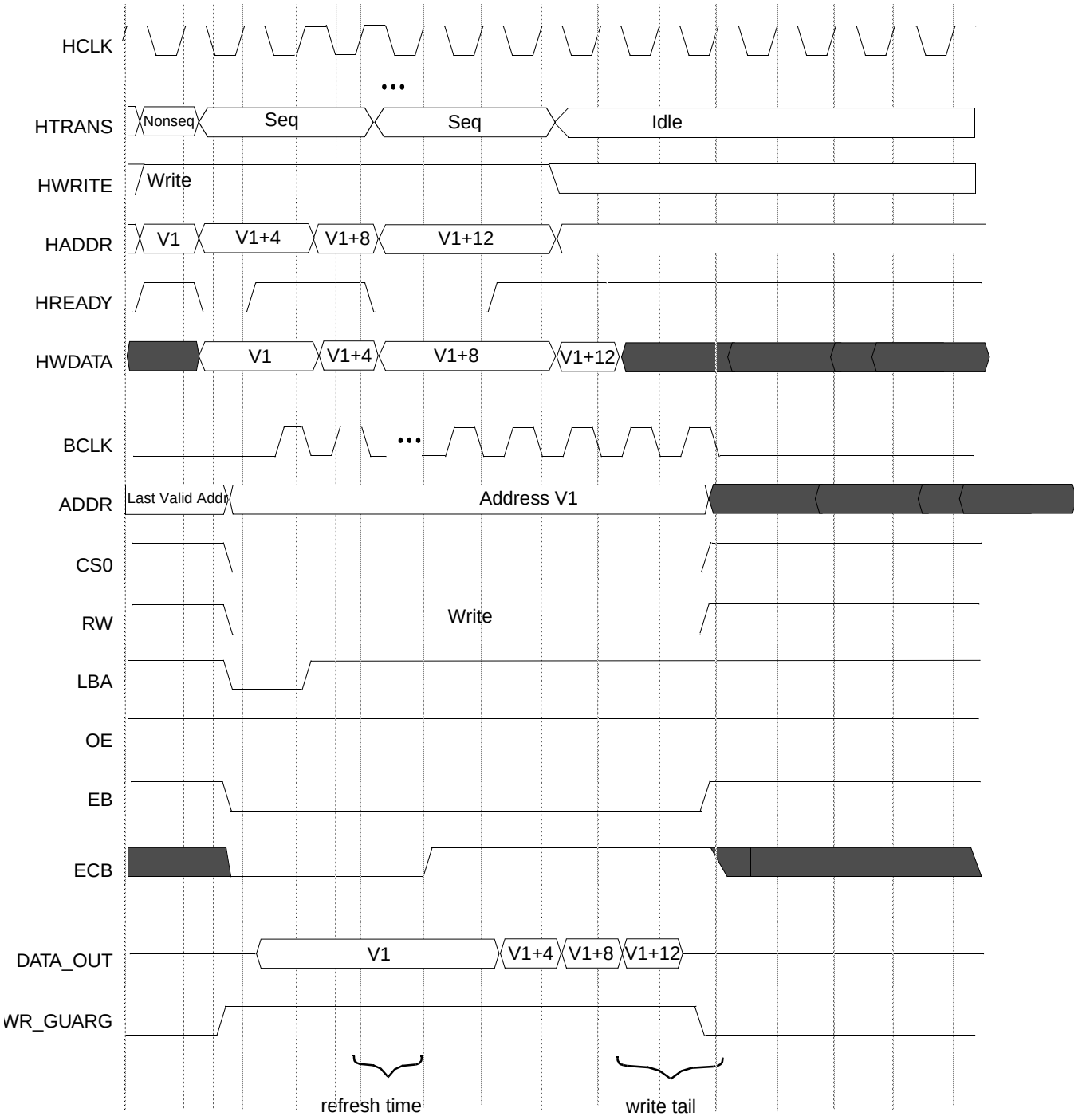


Figure 18-42. Write Access, BCS=1, WSC=4, SYNC=1, PSR=1

18.7.6 Muxed A/D Mode

18.7.6.1 Asynchronous Word Accesses to Word-Width Memory

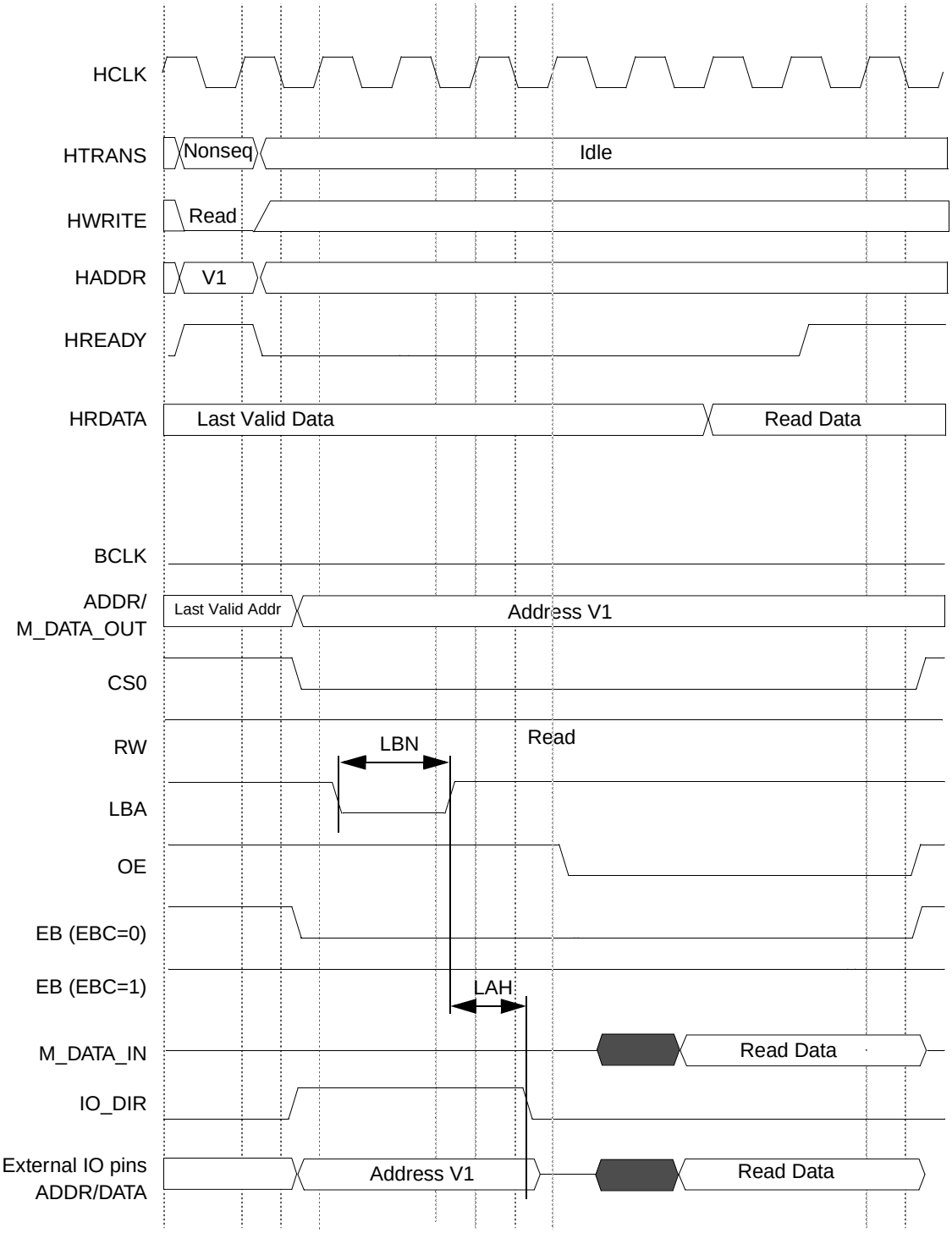


Figure 18-43. Read Access, WSC=7, LBA=1, LBN=1, LAH=1, OEA=7

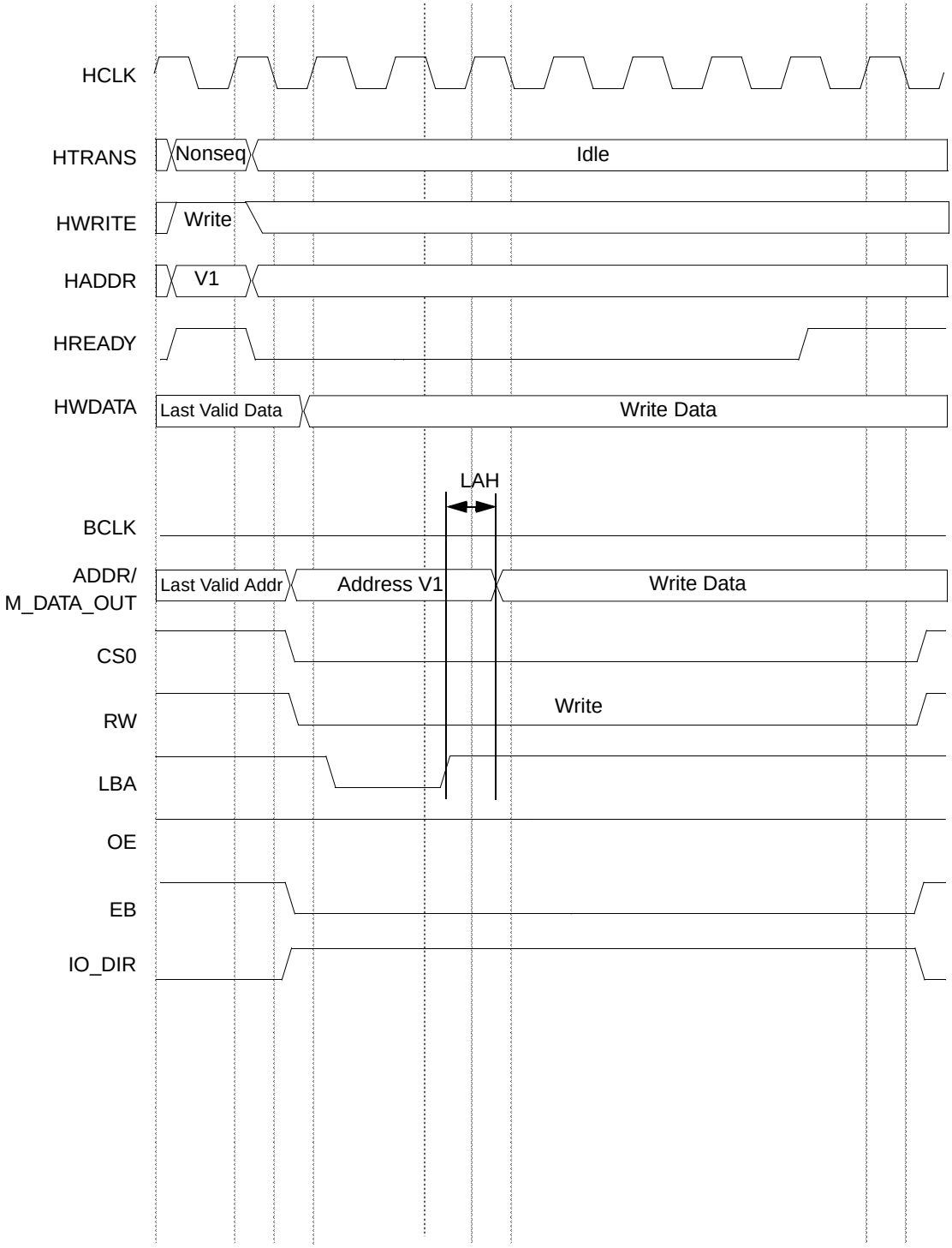


Figure 18-44. Write Access, WSC=7, LBA=1, LBN=1, LAH=1

18.7.6.2 Synchronous Accesses with Word-width Memory

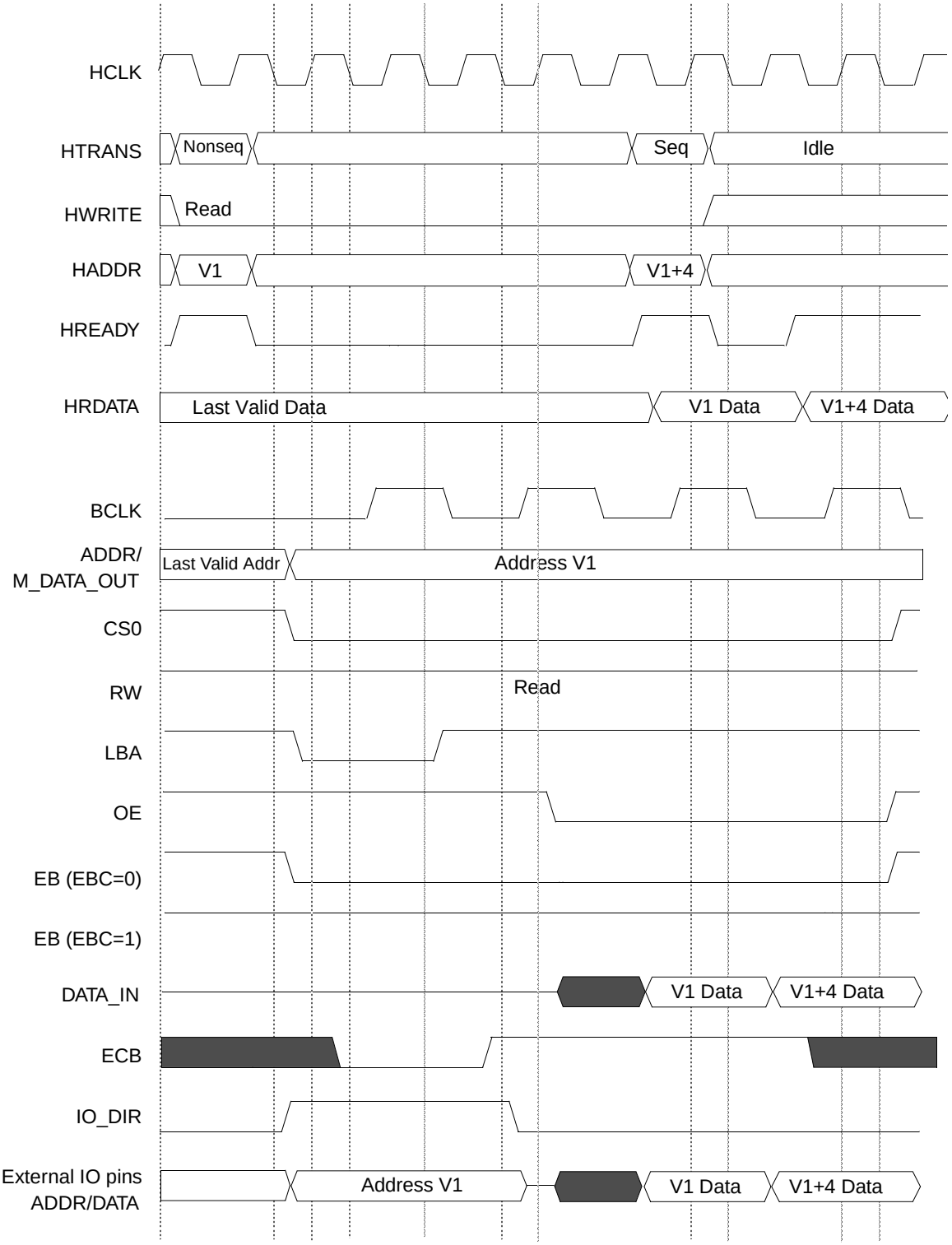


Figure 18-45. Read Access, BCD=1, SYNC=1, WCS=4, DOL=1, LBN=2, LAH=1, PSR=1

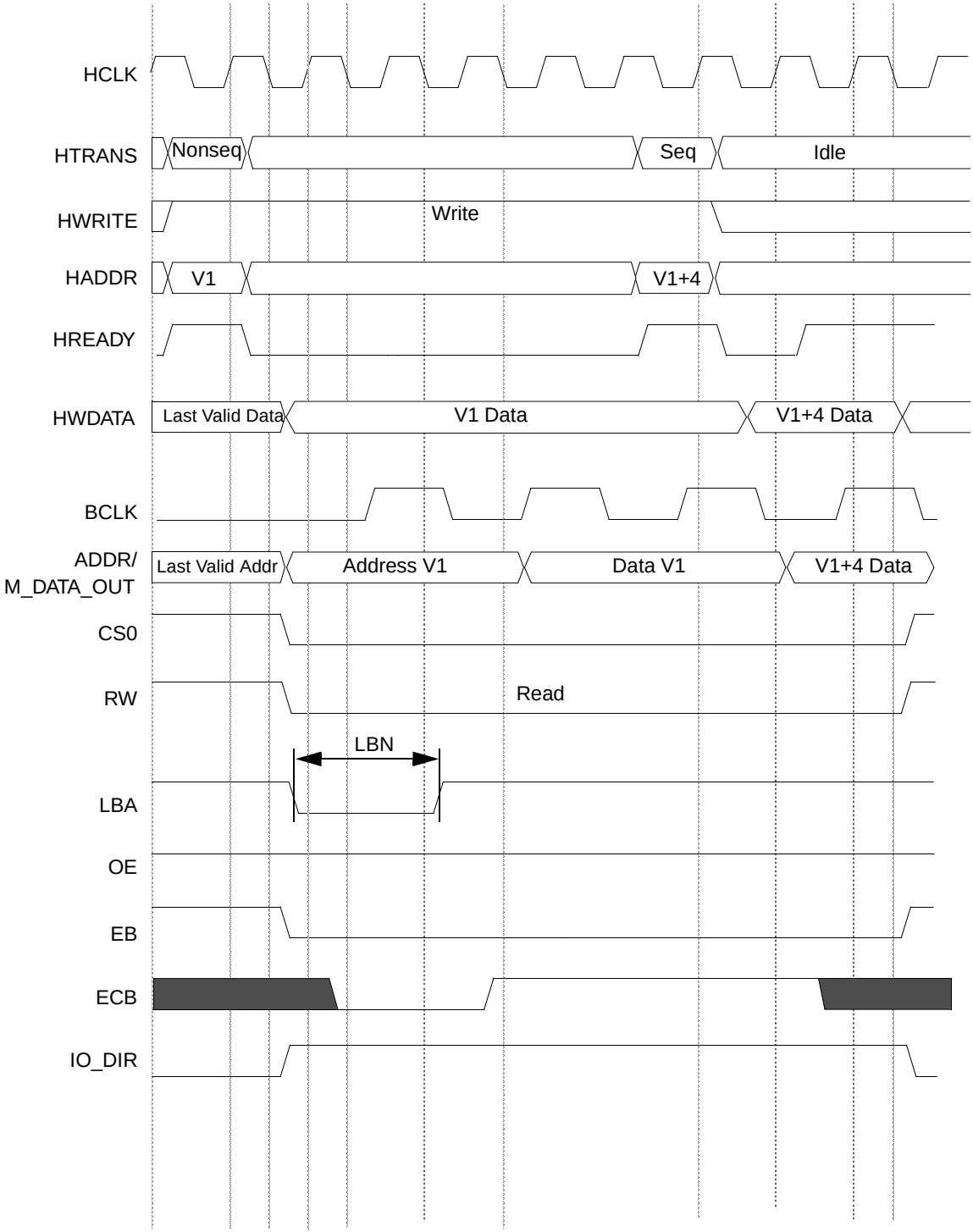


Figure 18-46. Write Access, BCD=1, SYNC=1, WCS=5, DOL=1, LBN=2, LAH=1, PSR=1

Chapter 19
Enhanced SDRAM Controller (ESDCTL)

The Enhanced Synchronous Dynamic RAM Controller (ESDCTL) provides interface and control for synchronous DRAM memories for the system. SDRAM memories use a synchronous interface with all signals registered on a clock edge. A command protocol is used for initialization, read, write, and refresh operations to the SDRAM and is generated on the signals by the controller when required due to external or internal requests. It has support for both single data rate RAMs and double data rate SDRAMs. It supports 64, 128, 256, and 512-Mbit, 1 Gbit, 2 Gbit, 4 bank synchronous DRAM by two independent chip selects and with up to 256 Mbytes addressable memory per chip select.

Figure 19-1 shows the Enhanced SDRAM Controller top-level diagram that shows the functional organization of the block.

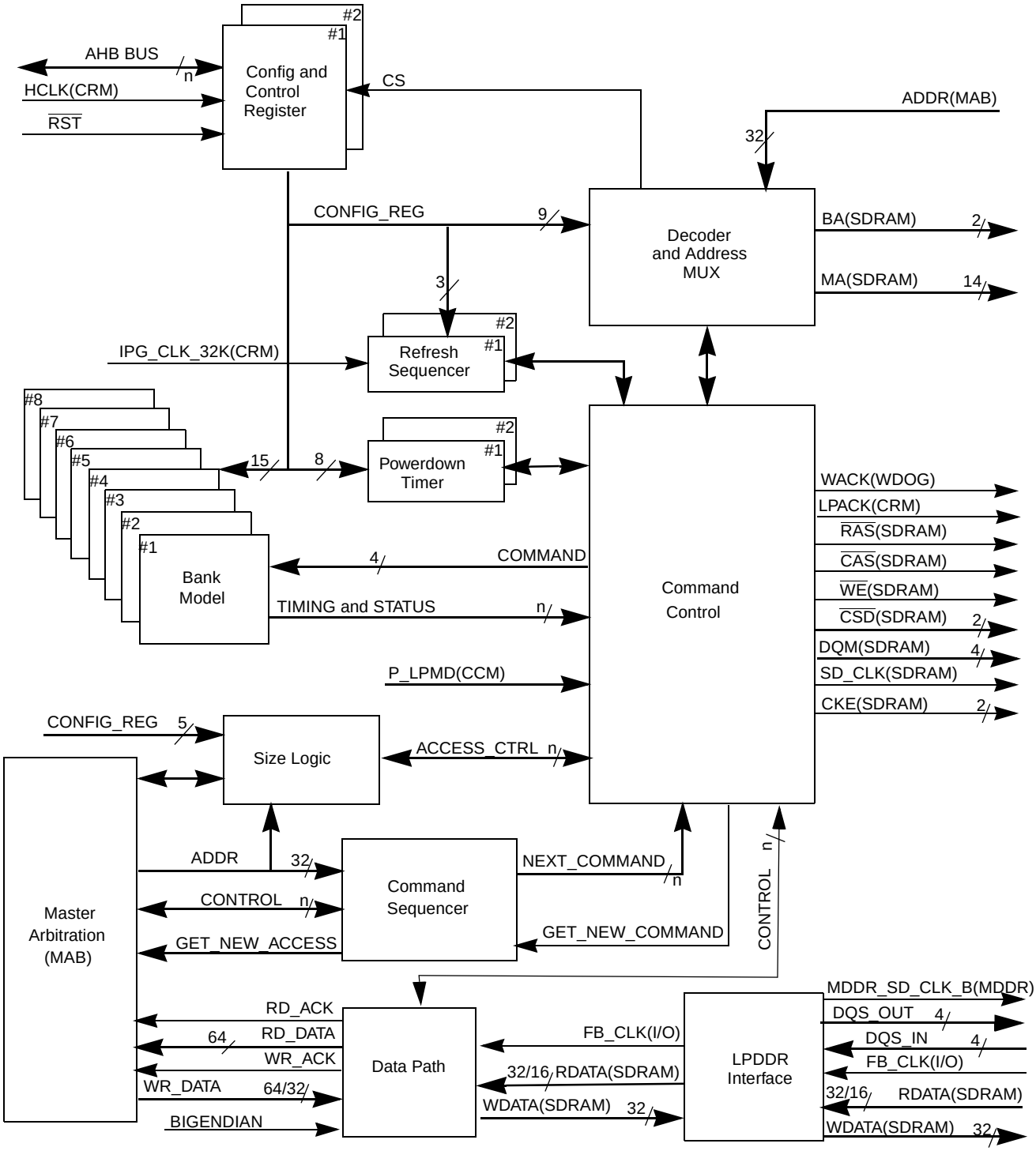


Figure 19-1. Enhanced SDR/LPDDR SDRAM Controller Block Diagram

19.1 Overview

The Enhanced SDRAM Controller consists of nine major blocks, including the SDRAM command state machine controller, bank register (page and bank address comparators), Row/Column Address Multiplexer, configuration registers, refresh request counter, command sequencer, size logic (splitting access), data path (data aligner/multiplexer), LPDDR interface, and the Power Down timer.

19.1.1 SDRAM Command Controller

This functional block controls the majority of the actions within the Enhanced SDRAM controller, including 12 FF indicating if the bus to the memory is busy for the next 12 cycles, and all the command to the memory are executing through this block.

19.1.2 Bank Model

There are a total of 8 address comparators, one comparator for each of the 4 banks within a chip select region. The comparators are used to determine if a requested access falls within the address range of a currently active SDRAM page. The bank model includes also all timing parameters comparators.

19.1.3 Decoder and Address MUX

All synchronous SDRAMs incorporate a multiplexed address bus, although the address folding points vary according to memory density, number of data I/O, and the processor data bus width. The Enhanced SDRAM Controller takes these variables into account and provides the proper alignment of the multiplexed address through the combination of the row/column address multiplexer, non-multiplexed address pins, and the connections between the controller and the memory devices.

19.1.4 ESDCTL Control and Configuration Registers

Control and Configuration registers determine the operating mode of the M3IF. Memory device density and bus width, number of memory devices, CAS latency, row to column delay, and burst length, and others are all configurable. Enable bits are provided for refresh and the Power Down timer. Mode bits provide a mechanism for software initiated SDRAM initialization, setting the device mode register, precharge, and auto-refresh cycles.

19.1.5 Refresh Sequencer

SDRAM memories require periodic refresh to retain data. The refresh request counter generates requests to the SDRAM Command Controller to perform those refresh cycles. Requests are scheduled according to a 32-kHz clock input. One, 2, 4, 8, or 16 refresh cycles are generated per a 32-kHz clock.

19.1.6 Command Sequencer

The Command Sequencer block send the commands be executed (precharge (all/bank), active, read, write, and burst terminate) to the Command Controller, after taking in account the bank's state of the access, the command controller execute signal and the command busy situation.

19.1.7 Size Logic

The Size Logic block gets as inputs from one side the address, access size, and wrap/incr access, and from the other side the configuration values—burst length and DSIZ. In case of a misaligned access, the size logic splits the access into multiple access as a functions of all inputs.

19.1.8 Mobile/Low Power DDR (LPDDR) Interface

The LPDDR interface adds ability to interface with Low Power Double Data Rate SDRAM. It converts the double data rate which is synchronized to both positive and negative edges of the DQS signals to a double width of data bus synchronized to the positive edge of the controller internal clock (which is derived from HCLK).

The interface uses a read FIFO which samples the data with delayed DQS (data strobe) in read cycles. Two read FIFOs are used: one for positive clock edge and the other is for negative clock edge. Delay lines are used to generate a delayed version of DQS input signals in order to sample the data at the middle of the data valid window. In write cycles DQS are output signals that are generated using a delay lines. The data at the input to the interface has a double width from the memory so it is divided into the memory width but with double frequency. For this purpose a MUX is used to pass the upper half and lower half of the data. This way in read cycles we receive double rate data with a DQS signal which is EDGE-aligned with read data and create double data rate and centered DQS in write cycles.

DQS delay lines are based on three units:

1. Measurement unit which measures one cycle time from positive edge to the next positive edge. The output of this unit is the number of small delay unit needed to delay a specific positive edge of the clock to overlap the following positive edge.
2. Dividing the result of the measurement by 4.
3. Delay unit take the result and selects the correct delay tap.

The delay unit is duplicated five times: four units for read (each byte has DQS signal) and one unit for write (to delay the data sampling).

19.1.8.1 Power Down Timer

A Power Down timer detects periods of inactivity to the SDRAM and disables the clock when the inactive period surpasses the selected time-out. Data is retained during the Power Down state. Subsequent requests to the SDRAM incur only a minimal added start-up delay (beyond the normal access time, as specified in Table 19-31). The Power Down timer may be programmed to expire anytime the controller is not actively reading/writing the memory, after 64 or 128 clocks of inactivity, or disabled entirely.

19.1.9 Features

The ESDCTL includes the following features:

- Optimization of consecutive memory accesses through memory command anticipation (latency hiding)
  - — Hiding latency by optimization the commands to both CS connected—command anticipation

  - Keeping track of open memory pages
  - Bank-wise memory address mapping
  - SDRAM burst length configuration of 4 or 8 (for 16-bit memory burst length 4 is not supported) or full page mode
  - LPDDR burst length configuration of 8
  - Support of different internal burst length (1/4/8 words) by using burst truncate commands
  - ARM AMBA AHB light compliant
  - Shared Address and command bus to SDRAM/LPDDR
- Supports 64-, 128-, 256-, 512-Mbit, 1-Gbit, 2-Gbit, 4-bank, single data rate, synchronous SDRAM, and LPDDR
  - Two independent chip selects
  - Up to 256 Mbytes per chip select
  - Up to four banks active simultaneously per chip select
  - JEDEC standard pinout/operation
- Support for 16- and 32-bit Mobile/Low Power DDR266 devices
- PC133-compliant Interface
  - 133 MHz system clock achievable with “-7” option PC133 compliant memories
  - Single fixed-length (4/8-word) burst or full page access
  - Access time of 9-1-1-1-1-1-1-1 at 133 MHz (for read access when memory bus is available, row is open and CAS latency configured to 3 cycles). The access time includes the M3IF delay (assuming no arbitration penalty).
- Software configurable for different system and memory devices requirements
  - 16- or 32-bit memory data bus width
  - Number of row and column addresses
  - Row cycle delay (t_{RC})
  - Row precharge delay (t_{RP})
  - Row to column delay (t_{RCD})
  - Column to data delay (CAS Latency)
  - Load mode register to active command (t_{MRD})
  - Write to precharge (t_{WR})
  - Write to read (t_{WTR}) for LPDDR memories only
  - LPDDR exit power down to next valid command delay (t_{XS})
  - Active to precharge (t_{RAS})
  - Active to active (t_{RRD})
- Built in auto-refresh timer and state machine
- Hardware and software supported self-refresh entry and exit
  - Keeps data valid during system reset and low power modes
  - Auto Power Down timer (one per Chip Select)

— Auto Precharge timer (one per bank in each chip select)

19.1.10 Modes of Operation

Each of the Enhanced SDRAM Controller (ESDCTL) operating modes are briefly described in this section. The ESDCTL's different operating modes for each chip select ($\overline{\text{CSD0}}$ and $\overline{\text{CSD1}}$) are defined by the SMODE field (3 bits) in the ESDCTL0 and ESDCTL1 registers respectively. In addition to the normal operating mode, the controller is capable of operating in the alternate operating modes primarily used for SDRAM/LPDDR initialization.

Any access to the SDRAM/LPDDR memory space, while in one of the alternate modes, results in the corresponding special cycle being run. Moving from Normal to any other mode does not close (precharge) any banks that may be open (activated). Under most circumstances, software should run a precharge-all cycle when transitional out of Normal Read/Write mode. Reset initializes the operating mode to "Normal Read/Write." This is a high level description only, detailed descriptions of operating modes are contained in Section 19.4, "Functional Description."

- Normal Read/Write Mode
  This is the normal operating mode used to READ/WRITE (single or burst accesses) from/to external SDRAM/LPDDR devices. The ESDCTL automatically drives the PRECHARGE/ACTIVE/BURST TERMINATE commands during the normal operating mode.
- Precharge Mode
  The manual PRECHARGE command is used to manually deactivate the open row (ACTIVE) in a particular bank or the open row in all banks. The bank(s) will be available for a subsequent row access a specified time (tRP) after the PRECHARGE command is issued. External memory device input A10 determines whether one or all banks are to be precharged, and in the case that only one bank is to be precharged, inputs BA0 and BA1 select the bank. The manual PRECHARGE command is used during SDRAM initialization, LOAD MODE REGISTER command and manual REFRESH cycles
- Auto Refresh Mode
  The AUTO REFRESH command is used to retain data in the SDRAM memory devices. This command is non persistent, so it must be issued each time a refresh is required. The ESDCTL has a REFRESH counter for each CSD_B (external memory region) and it automatically handles the REFRESH commands toward the memory. The manual AUTO REFRESH command is used during SDRAM initialization or in case that the REFRSH counters are not enabled.
- Load Mode Register Mode
  The Mode Register is used to define the specific mode of operation of the SDRAM device. This definition includes the selection of a burst length, a burst type, a CAS latency, an operating mode and a write burst mode. The Mode Register is programmed via the LOAD MODE REGISTER command and will retain the stored information until it is programmed again or the external memory device loses power. The Mode Register must be loaded when all banks are idle (after PRECHARGE ALL). The Enhanced SDRAM Controller will wait a specified ($t_{MRD)}$ period of time as configured in ESDCFG0 or ESDCFG1 registers. Violating either of these requirements by an incorrect controller register configuration results in unspecified operation.

19.2 External Signal Description

This section discusses input and output signals between the Enhanced SDRAM Controller and the external memory devices. Other than the chip select outputs ($\overline{CSD0}$ and $\overline{CSD1}$) and clock enables (CKE0 and CKE1), all signals are shared between the two chip select regions. Table 19-2 summarizes the interface signals, and is followed by a detailed description of signal functions. Interconnect and timing diagrams are included as part of the detailed discussion on controller operation in Section 19.4, "Functional Description."

Table 19-2. ESDCTL Signal Properties

Name	Port	Function	Reset State	Direction
SD_CLK	—	Clock to SDRAM (up to 133MHz)	1	Output
FB_CLK_IN	—	Feedback CLock	0	Input
CKE0	—	Clock enable to SDRAM 0	0	Output
CKE1	—	Clock enable to SDRAM 1	0	Output
$\overline{CSD}$[0]	—	Chip select to SDRAM Array 0	1	Output
$\overline{CSD}$[1]	—	Chip select to SDRAM Array 1	1	Output
RDATA[31:0]	—	Read Data from memories	0	Input
WDATA[31:0]	—	Write data to memories	0	Output
MA[13:0]	—	Multiplexed Address	0	Output
BA[1:0]	—	Bank address	0	Output
DQM3	—	Data Qualifier Mask byte 3 (D[31:24])	0	Output
DQM2	—	Data Qualifier Mask byte 2 (D[23:16])	0	Output
DQM1	—	Data Qualifier Mask byte 1(D[15:8])	0	Output
DQM0	—	Data Qualifier Mask byte 0 (D[7:0])	0	Output
$\overline{WE}$	—	Write Enable	1	Output
$\overline{RAS}$	—	Row Address Strobe	1	Output
$\overline{CAS}$	—	Column Address Strobe	1	Output
LPACK	—	Low Power Mode Acknowledge	1	Output
WACK	—	Memory wake up	0	Output
BIGEND	—	Big Endian signal	system dependent	Input
P_LPMD	—	Low Power Mode Entry Request	0	Input
Mobile LPDDR Signals				
MDDR_SD_CLK_B	—	Inverted Clock to LPDDR SDRAM	0	Output
DQS_OUT3	—	Data strobe byte 3 (D[31:24])	0	Output
DQS_OUT2	—	Data strobe byte 2 (D[23:16])	0	Output
DQS_OUT1	—	Data strobe byte 1 (D[15:8])	0	Output

Table 19-2. ESDCTL Signal Properties (continued)

Name	Port	Function	Reset State	Direction
DQS_OUT0	—	Data strobe byte 0 (D[7:0])	0	Output
DQS_IN3	—	Data strobe byte 3 (D[31:24])	0	Input
DQS_IN2	—	Data strobe byte 2 (D[23:16])	0	Input
DQS_IN1	—	Data strobe byte 1 (D[15:8])	0	Input
DQS_IN0	—	Data strobe byte 0 (D[7:0])	0	Input

19.2.1 Detailed Signal Descriptions

Table 19-3 lists the signals and descriptions of all of the I/O signals that interface with the ESDCTL.

Table 19-3. ESDCTL Detailed Signal Description

Signal	I/O	Description
SD_CLK	O	SDRAM CLock. The SD_CLK output provides the timing reference for the memory devices. All other SDRAM interface signals are referenced to this clock. SD_CLK is synchronous to the system clock, but is gated off during low power operating modes when both CKE0 and CKE1 are negated.
FB_CLK_IN	I	Feedback Clock. The FB_CLK_IN signal is used by the Enhanced SDRAM controller to sample the data during READ cycles. The FB_CLK_IN is a delayed SD_CLK, and at a good proximity is identical to the external memory device clock. A delay between SD_CLK and FB_CLK_IN compensates the PAD input/output buffers, chip route.
CKEO CKE1	O	SDRAM/MMDR Clock Enables. Clock enable outputs to the SDRAM memory devices. CKE0 corresponds to SDRAM/LPDDR array 0 and CKE1 to SDRAM/LPDDR array 1. Activates the memory's clock input when high, indicating a stable clock is being supplied. Deactivates the memory's clock input when low. CKEx low initiates Power Down and Self Refresh modes to the SDRAM.
$\overline{\text{CSD0}}$ $\overline{\text{CSD1}}$	O	$\overline{\text{SDRAM/LPDDR Chip Select. CSD0}}$ and $\overline{\text{CSD1}}$ are used to select SDRAM/LPDDR Array 0 and SDRAM/LPDDR Array 1, respectively. The chip select signals are used to indicate when a valid command is present on the other control signals and to which device the command is directed.
RDATA[31:0] WDATA[31:0]	I/O	Read/Write Data Bus. The 32 data pins are used to transfer data between the Enhanced SDRAM Controller and memory. Data bit 31 is the most significant bit. Bit 0 is the least significant. 16-bit memory alignment is selectable according to the programming of the DSIZ field in the ESDCTL register (see Section 19.3.3, "Register Descriptions").
MA[13:0]	O	Multiplexed Address Bus. The multiplexed address bus specifies the SDRAM page and location within the page targeted by the current access. Connections between the Enhanced SDRAM Controller and memory will vary depending on the SDRAM device density. See Section 19.4.2, "Address Multiplexing" and specifically Table 19-28 and Table 19-29 for details on supported SDRAM configurations.
BA[1:0]	O	Bank Address Bus. The bank address pins specify to which bank the current command is targeted. Table 19-28 and Table 19-29 document which address pins are used as the bank address bus for given device configuration.

Table 19-3. ESDCTL Detailed Signal Description (continued)

Signal	I/O	Description
DQM3, DQM2, DQM1, DQM0	O	Data Qualifier Mask. During read cycles, DQMx controls the SDRAM data output buffers. DQMx asserted high disables the output buffers leaving them in a high-impedance state. DQMx low allows the data buffers to drive normally. During write cycles, DQMx controls which bytes are written in the SDRAM. DQMx driven low enables a write to the corresponding byte, while DQMx asserted high leaves the byte unchanged. DQM0 corresponds to D[7:0] and DQM3 to D[31:24]. Sixteen bit memories require only two DQM connections. Memories aligned to the upper data bus (D[31:16]) connect to DQM2 and DQM3, while memories aligned to the lower data bus (D[15:0]) connect to DQM0 and DQM1. The ESDCTL takes care of the Endian operating mode, and drives the respective DQM strobe required. **Note:** When the controller is used to interface Mobile/Low Power DDR, DQM will change twice each cycle (like the data) and will be aligned to DQS edges.
$\overline{WE}$	O	Write Enable. Write enable is part of the three bit command field ($\overline{RAS}$ and $\overline{CAS}$ make up the other two bits) used by the SDRAM. Generally, $\overline{SDWE}$ is asserted low if a command transfers data to the memory. A detailed summary of the supported SDRAM commands is provided in Table 19-32.
$\overline{RAS}$	O	Row Address Strobe. Row address strobe is also part of the SDRAM command field. It is generally used to indicate an operation affecting an entire bank or row. RAS is an active low signal. Table 19-32 provides details on SDRAM command encoding.
$\overline{CAS}$	O	Column Address Strobe. The column address strobe is the third signal comprised in the command field. It generally signifies a column oriented command. CAS is an active low signal. Table 19-32 provides details on SDRAM command encoding.
LPACK	O	Low Power Mode Acknowledge. This signal indicates that the external memory devices operates in one of the low power mode. This signal is used by the system clock and reset module to enable the deactivation of the system clock during system low power operating mode. The default value of LPACK is high, not to affect in case that the memory devices are not used/disabled.
L_LPMD	I	Low Power Mode Entry Indication. This input signal (from the CRM) is used as a low power mode entry indication. During RUN mode P_LPMD value is 0. While there is a system request to enter low power mode (STOP) the P_LPMD value changes to 1. During STOP mode the ESDCTL clock is shut off, so the external memory devices need to enter self refresh. This change will cause the ESDCTL to complete all active/pending requests afterwards, the external memory devices will be placed in self refresh mode.
WACK	O	External memory device WakeUp. This signal indicates that the external memory device power up period is over, so the initialization commands can be issued. This signals goes down during reset or low power operating mode. This signal can be used by the system WatchDog module to disable/mask WatchDog reset/interrupt during the time WACK is low.
SD_CLK/$\overline{MDDR\_SD\_CLK}$	O	LPDDR SDRAM Inverted Clock. SD_CLK and $\overline{MDDR\_SD\_CLK}$ are Mobile/Low Power DDR Differential Clocks. All LPDDRs address and control input signals are sampled on the crossing of the positive edge of SD_CLK and negative edge of $\overline{MDDR\_SD\_CLK}$. Internal clock signals are derived from SD_CLK/$\overline{MDDR\_SD\_CLK}$.
DQS_OUT3, DQS_OUT2, DQS_OUT1, DQS_OUT0	O	Data Strobes Outputs. Data strobes are used for data capture for LPDDR memory. During write cycles, they are generated by the SDRAMC controller and are centered with write data. DQS3 corresponds to the most significant byte and DQS0 to the least significant byte.

Table 19-3. ESDCTL Detailed Signal Description (continued)

Signal	I/O	Description
DQS_IN3, DQS_IN2, DQS_IN1, DQS_IN0	I	Data Strobes Inputs. During read cycles, DQS_INx are generated by the memory devices and are edge aligned with read data. For read data, the controller receives a DQS signal that is edge aligned with the read data. The DQS delay module is used to delay the DQS signal and center it in the data valid window.
BIGEND	I	Big Endian. Defines the byte ordering. For example, it controls how multi-byte objects are represented by the underlying architecture. This signal is driven by the ARM platform, and defines two Endianess modes, Little and Big as illustrated in Figure 19-3.

19.3 Memory Map and Register Definition

The Enhanced SDRAM Controller programming model consists of control and configuration registers (32-bit length) for each chip select and a miscellaneous register, as shown in Table 19-5. The control register maintain system dependent information, while the configuration register maintain SDRAM/LPDDR memory device dependent information. ESDCFG0 defines the operating characteristics for the region selected by $\overline{\text{CSD0}}$, while ESDCFG1 does the same for the $\overline{\text{CSD1}}$ region. Bit field assignments within the registers are identical, so a single description will apply to both registers.

Both the control and configuration registers are 32 bits in length with bit fields defined in Figure 19-4 through Figure 19-7, respectively. All implemented bits are fully readable and writable. Reserved bit locations are unaffected by writes and always read back as zero. All register accesses must be SINGLE word (32-bit) operations through the AHB bus protocol. Accesses of any other size will have indeterminate results. All Enhanced SDRAM Controller registers can be access only by one master at a time. Multi access to the Enhanced SDRAM Controller register causes undetermined behavior.

The reset state of each bit is shown underneath the bit field name. An asterisk indicates that the value is dependent on the operating mode selected during reset. Details are provided in the following bit field descriptions.

19.3.1 Memory Map

The ESDCTL supports 64, 128, 256, 512-Mbit, 1-Gbit and 2-Gbit, 4 bank, single data rate, synchronous DRAMs on two independent chip selects. Each chip selects defines a specific memory address mapped, as shown in Table 19-4. Table 19-5 shows the ESDCTL memory map.

Table 19-4. ESDCTL Memory Map Overview

Address	Use	Access
0x8000_0000–0x8FFF_FFFF	CSD0 SDRAM or LPDDR memory region (256MB)	Read/Write
0x9000_0000–0x9FFF_FFFF	CSD1 SDRAM or LPDDR memory region (256MB)	Read/Write

Table 19-5. ESDCTL Memory Map

Address	Register	Access	Reset	Section/Page
0xB800_1000 (ESDCTL0)	Enhanced SDRAM Control Register 0	R/W	0x0111_0080	19.3.3.1/19-15
0xB800_1004 (ESDCFG0)	Enhanced SDRAM Configuration Register 0	R/W	0x0076_EB3A	19.3.3.2/19-20
0xB800_1008 (ESDCTL1)	Enhanced SDRAM Control Register 1	R/W	0x8112_0080	19.3.3.1/19-15
0xB800_100C (ESDCFG1)	Enhanced SDRAM Configuration Register 1	R/W	0x007A_C727	19.3.3.2/19-20
0xB800_1010 (ESDMISC)	Enhanced SDRAM Miscellaneous Register	R/W	0x0000_0000	19.3.3.3/19-33
0xB800_1020 (ESDCDLY1)	Enhanced MDDR Delay Line 1 Configuration Debug Register	R/W	0x001C_0000	19.3.3.4/19-35
0xB800_1024 (ESDCDLY2)	Enhanced MDDR Delay Line 2 Configuration Debug Register	R/W	0x001C_0000	19.3.3.5/19-36
0xB800_1028 (ESDCDLY3)	Enhanced MDDR Delay Line 3 Configuration Debug Register	R/W	0x001C_0000	19.3.3.8/19-39
0xB800_102C (ESDCDLY4)	Enhanced MDDR Delay Line 4 Configuration Debug Register	R/W	0x001C_0000	19.3.3.9/19-40
0xB800_1030 (ESDCDLY5)	Enhanced MDDR Delay Line 5 Configuration Debug Register	R/W	0x001C_0000	19.3.3.8/19-39
0xB800_1034 (ESDCDLYL)	Enhanced MDDR Delay Line Cycle Length Debug Register	R	N/A	19.3.3.9/19-40

19.3.2 Register Summary

Figure 19-2 shows the key to the register fields, and Table 19-6 shows the register figure conventions.

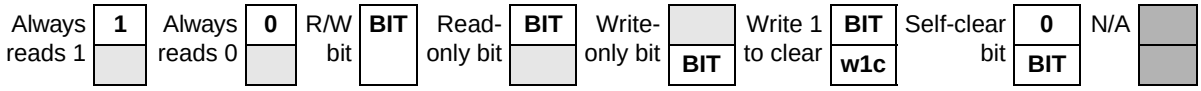


Figure 19-2. Key to Register Fields

Table 19-6. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
R	Read-only. Writing this bit has no effect.
W	Write-only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).

Table 19-6. Register Figure Conventions (continued)

Convention	Description
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 19-7 shows the ESDCTL register summary.

Table 19-7. ESDCTL Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xB800_1000 (ESDCTL0)	R	SDE	SMODE			SP	ROW			0	0	COL		0	0	DSIZ	
	W																
	R	SREFR			0	PWDT		0	FP	BL	0	PRCT					
	W																
0xB800_1004 (ESDCFG0)	R	0	0	0	0	0	0	0	0	0	tXP		tWTR	tRP		tMRD	
	W																
	R	tWR	tRAS			tRRD		tCAS		0	tRCD			tRC			
	W																
0xB800_1008 (ESDCTL1)	R	SDE	SMODE			SP	ROW			0	0	COL		0	0	DSIZ	
	W																
	R	SREFR			0	PWDT		0	FP	BL	0	PRCT					
	W																
0xB800_100C (ESDCFG1)	R	0	0	0	0	0	0	0	0	0	tXP		tWTR	tRP		tMRD	
	W																
	R	tWR	tRAS			tRRD		tCAS		0	tRCD			tRC			
	W																

Table 19-7. ESDCTL Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xB800_1010 (ESDMISC)	R	SDRAM_RDY	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	LHD	MDDR_MDIS	0	MDDREN	0	0
	W													MDDR_DL_RST		RST	
0xB800_1020 (ESDCDLY1)	R	SEL_DLY_REG_1	0	0	0	0	DLY_CORR_1										
	W																
	R	0	0	0	0	0	DLY_REG_1										
	W																
0xB800_1024 (ESDCDLY2)	R	SEL_DLY_REG_2	0	0	0	0	DLY_CORR_2										
	W																
	R	0	0	0	0	0	DLY_REG_2										
	W																
0xB800_1028 (ESDCDLY3)	R	SEL_DLY_REG_3	0	0	0	0	DLY_CORR_3										
	W																
	R	0	0	0	0	0	DLY_REG_3										
	W																
0xB800_102C (ESDCDLY4)	R	SEL_DLY_REG_4	0	0	0	0	DLY_CORR_4										
	W																
	R	0	0	0	0	0	DLY_REG_4										
	W																

Table 19-7. ESDCTL Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xB800_1030 (ESDCDLY5)	R	SEL_DLY_REG_5	0	0	0	0	DLY_CORR_5										
	W																
	R	0	0	0	0	0	DLY_REG_5										
	W																
0xB800_1034 (ESDCDLYL)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
	W																
	R	0	0	0	0	0	DLY_CYCLE_LENGTH										
	W																

19.3.3 Register Descriptions

This section consists of register descriptions in address order. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

NOTE

Memory may be viewed from either a Big Endian or Little Endian byte ordering perspective depending on the processor configuration (see Figure 19-3).

In Big Endian mode (the typical default operating mode), the most significant byte (byte 0) of word 0 is located at address 0.

For Little Endian mode, the most significant byte of word 0 is located at address 3. Within registers, bits are numbered within a word starting with bit 31 as the most significant bit. By convention, byte 0 of a register is the most significant byte regardless of Endian mode.

31	Big Endian Mode		0
Byte 0	Byte 1	Byte 2	Byte 3
Byte 4	Byte 5	Byte 6	Byte 7
Byte 8	Byte 9	Byte A	Byte B

Word at 0
Word at 4
Word at 8

⋮

31	Little Endian Mode		0
Byte 3	Byte 2	Byte 1	Byte 0
Byte 7	Byte 6	Byte 5	Byte 4
Byte B	Byte A	Byte 9	Byte 8

Word at 0
Word at 4
Word at 8

⋮

Figure 19-3. Data Organization in Memory

19.3.3.1 ESDCTL0 and ESDCTL1 Control Registers

This register contains the controlling various memory and control settings for the ESDCTL. The bit assignments for the register are shown in Figure 19-4 and Figure 19-5. The field descriptions for the bit assignments are listed in Table 19-8.

0xB800_1000 (ESDCTL0) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SDE	SMODE			SP	ROW			0	0	COL		0	0	DSIZ	
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SREFR			0	PWDT		0	FP	BL		PRCT					
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Figure 19-4. Enhanced SDRAM Control Register (ESDCTL0)

0xB800_1008 (ESDCTL1) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SDE	SMODE			SP	ROW			0	0	COL		0	0	DSIZ	
W																
Reset	1	0	0	0	0	0	0	1	0	0	0	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SREFR			0	PWDT		0	FP	BL		PRCT					
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Figure 19-5. Enhanced SDRAM Control Register (ESDCTL1)

Table 19-8. Enhanced SDRAM Control Register (ESDCTL0/1) Field Descriptions

Field	Description
31 SDE	Enhanced SDRAM Controller Enable. This control bit enables/disables the Enhanced SDRAM controller. The reset value of the SDE0 bit is “0”, means the module is disabled for CSD0. Writing a one to those control bits enables the module, for both chip selects. Clearing those bits disables the module. By disabling both bits (SDE0 and SDE1) all clocks within the module shuts off (with the exception of register accesses). 0 Disabled 1 Enabled
30–28 SMODE	SDRAM Controller Operating Mode. This bit field determines the operating mode of the Enhanced SDRAM controller. In addition to the normal operating mode, the controller is capable of operating in the alternate operating modes listed below. These modes are primarily used for SDRAM initialization. Any access to the SDRAM memory space, while in one of the alternate modes, will result in the corresponding special cycle being run. Moving from Normal to any other mode does not close (precharge) any banks that may be open (activated). Under most circumstances, software should run a precharge-all cycle when transitional out of Normal Read/Write mode. Operating mode details are provided in Section 19.4, “Functional Description.” Reset initializes the operating mode to “Normal Read/Write.” 000 Normal Read/Write 001 Precharge Command 010 Auto-Refresh Command 011 Load Mode Register Command 100 Manual Self Refresh 101 through 111 are reserved
27 SP	Supervisor Protect. This control bit is used to restrict User accesses within the chip select region. The default at reset are that both user and supervisor accesses are allowed. 0 User mode accesses are allowed to this chip select region. 1 User mode accesses are prohibited. An attempted access to this chip select region while in User mode will result in a high HRESP[1] being returned back to the CPU. The chip select will not be asserted. ESDCTL error response is generated by the M3IF arbitrator.

Table 19-8. Enhanced SDRAM Control Register (ESDCTL0/1) Field Descriptions (continued)

Field	Description
26–24 ROW	Row Address Width. This control field specifies the number of row addresses used by the memory array. This number does not include the bank, column, or data qualifier addresses. Parameters affected by the programming of this field include the page-hit address comparators and the bank address bit locations. 000 11 Row Addresses 001 12 Row Addresses 010 13 Row Addresses 011 14 Row Addresses 100 15 Row Addresses 101 through 111 are reserved.
23–22	Reserved
21–20 COL	Column Address Width. The COL control field is used to specify the number of column addresses in the memory array and will determine the break point in the address multiplexer. Column width is the number of multiplexed column addresses and does not include bank, and row addresses, or addresses used to generate the DQM signals. The settings for the COL bit field encoding are shown in Table 19-9.
19–18	Reserved
17–16 DSIZ	SDRAM Memory Data Width. This field defines the width of the SDRAM memory and its alignment on the external data bus. 16-bit ports may be aligned to either the high or low halfword to equalize capacitive loading on the bus. Data qualifier mask control outputs must be matched to the selected data bus alignment. Memories aligned to D[31:16] use DQM2 and DQM3. Memories aligned to D[15:0] use DQM0 and DQM1. 00 16-bit memory width aligned to D[31:16] 01 16-bit memory width aligned to D[15:0] (reset value for CSD0) 10 32-bit memory width (reset value for CSD1) 11 reserved
15–13 SREFR	SDRAM Refresh Rate. This control bit field enables/disables SDRAM refresh cycles and controls the refresh rate. Refresh cycles are referenced to a 32 kHz clock. At each rising edge 1, 2, 4, 8 or 16 rows are refreshed as determined by this bit field. Multiple refresh cycles will be separated by the row cycle delay specified in the SRC control field. Refresh is disabled by hardware reset. SREFR Bit Field Encoding settings are listed in Table 19-10. Usage example see Table 19-25.
12	Reserved
11–10 PWDT	Power Down Timer. This field determines whether the SDRAM will be placed in a Power Down condition after a selectable delay from the last access. The Power Down time-out can be triggered on either the absence of an active bank (PWDT=01) or a clock (HCLK) count from the last access (PWDT=10 or 11). Count based time-outs do not force the SDRAM into an idle condition (for example, any active banks remain open). The Power Down timers feature is disabled by hardware reset. See Section 19.4.5.3, “Precharge Power Down Mode” and Section 19.4.5.4, “Active Power Down Mode” for a comprehensive description of this operating mode. A listing of the PWDT Bit Field Encoding is shown in Table 19-11.
9	Reserved
8 FP	Full Page. This bit should be set to 1 if the Burst Length of the SDRAM connected to the CSD has been configured to Full-Page mode. This bit is needed since ESDCTL needs to induce a BURST TERMINATE (BT) command to terminate early all accesses that are less than Full Page. 0 Burst Length of the external memory device is not set to Full Page. 1 Burst Length of the external memory device is set to Full Page. **Note:** Full page mode is not supported when LPDDR external devices are used.

Table 19-8. Enhanced SDRAM Control Register (ESDCTL0/1) Field Descriptions (continued)

Field	Description
7 BL	Burst Length. This bit configures the access burst length. For proper operation the ESDCTL burst length configuration must match the external SDRAM/LPDDR memory device (configured via special operating mode Load Mode Register). If this bit is set to 1 means that the external memory device connected to the CSD have been configured to Burst Length of 8. If this bit is cleared to 0, means that the external memory device connected to the CSD have been configured to Burst Length of 4. The settings for the Burst Length Bit Field Encoding are listed in Table 19-12.
6	Reserved
5–0 PRCT	Precharge Timer. Precharges a bank after 2xPRCT clocks (HCLK, up to 133 MHz) of no activity. Table 19-13 illustrates the PRCT bit field encoding. “Closing” (due to precharge command) the last used/open row in any non active bank within a chip select reduces the power consumption of the external memory device. The power saving is device dependent, and one should consult/examine the external memory device specification for more details on power consumption reduction. If PRCT is enabled, a PRECHARGE command is issued after approximately number of cycles (as shown in Table 19-13) of non activity to one of the SDRAM/LPDDR banks. The number of cycles before the PRECHARGE command is issued depends on command bus (WE, RAS, CAS and CSD) availability (means there is no active access to other bank) and the memory timing parameters.

Table 19-9 through Table 19-13 show bit field encodings.

Table 19-9. COL Bit Field Encoding

COL[1:0]	Number of Column Addresses
00	8
01	9
10	10
11	Reserved

Table 19-10. SREFR Bit Field Encoding

SREFR[2:0]	Rows Each Refresh Clock	# Rows/64 mS @ 32 kHz	Row Rate @ 32 kHz
000	Refresh Disabled (bit field reset value)		
001	1	2048	31.25 µs
010	2	4096	15.62 µs
011	4	8192	7.81 µs
100	8	16384	3.91µs
101	16	32768	1.95 µs
110	Reserved		
111	Reserved		

Table 19-11. PWDT Bit Field Encoding

PWDT[1:0]	Power Down Time-out	Memory Device Operating Mode
00	Disabled (bit field reset value)	Run Mode
01	Any time no banks are active	Precharge Power Down
10	64 clocks (HCLK) after completion of last access[1]	Active Power Down
11	128 clocks (HCLK) after completion of last access[2]	Active Power Down

[1]This setting can not be used if the PRCT (precharge timer) is enabled.

Table 19-12. Burst Length Bit Field Encoding

BL	SDR SDRAM	LPDDR SDRAM
0	4[1]	Reserved
1[2]	8	8

1 For 16-bit SDRAM memory devices a BL setting of 4 is not supported.

2 Bit field reset value

Table 19-13. PRCT Bit Field Encoding

PRCT[5:0][1]	Precharge Timer[2]
000000	Disabled (Bit field reset value)
000001	2 clocks to precharge
000010	4 clocks to precharge
000011	6 clocks to precharge
000100	8 clocks to precharge
000101	10 clocks to precharge
000110	12 clocks to precharge
000111	14 clocks to precharge
001000	16 clocks to precharge
....	–.
010000	32 clocks to precharge
....	.....
100000	64 clocks to precharge
....	.....
111111	126 clocks to precharge

1 The PRCT can be used only if PWDT is disabled (“00”) or set to “any time no banks are active (“01”) PRCT cannot be used with any other PWDT settings.

[2] The number of clocks is approximate and it depends on the external bus availability.

19.3.3.2 ESDCTL Configuration Registers (ESDCFG0/ESDCFG1)

The bit assignments for the Configuration registers are shown in Figure 19-6 and Figure 19-7. The field descriptions for the registers is listed in Table 19-14.

0xB800_1004 (ESDCFG0) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	tXP		tWTR	tRP		tMRD	
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	1	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	tWR		tRAS		tRRD		tCAS		0	tRCD			tRC			
W																
Reset	1	1	1	0	1	0	1	1	0	0	1	1	1	0	1	0

Figure 19-6. Enhanced SDRAM Configuration Register 0 (ESDCFG0)

0xB800_100C (ESDCFG1) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	tXP		tWTR	tRP		tMRD	
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	1	1	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	tWR		tRAS		tRRD		tCAS		0	tRCD			tRC			
W																
Reset	1	1	0	0	0	1	1	1	0	0	1	0	0	1	1	1

Figure 19-7. Enhanced SDRAM Configuration Register 1 (ESDCFG1)

Table 19-14. ESDCFG0/ESDCFG1 Field Descriptions

Field	Description
31–23	Reserved
22–21 t_{XP}	LPDDR exit power down to next valid command delay. This control field determines the minimum delay between a valid command is issued to the LPDDR after exiting power down mode. The value programmed in t_{XP} is the number of clocks inserted after exiting power down mode and any subsequent new valid command. An example timing diagram for t_{XP} can be found in Figure 19-19. 00 1 clock delay before new COMMAND issued to LPDDR after power down mode exit 01 2 clock delay before new COMMAND issued to LPDDR after power down mode exit 10 3 clock delay before new COMMAND issued to LPDDR after power down mode exit 11 4 clock delay before new COMMAND issued to LPDDR after power down mode exit
20 t_{WTR}	tLPDDR WRITE to READ Command Delay. Data for any WRITE burst may be followed by a subsequent READ command. To follow a WRITE without truncating the WRITE burst, t_{WTR} should be set as shown in Figure 19-8. The LPDDRC will automatically induce a t_{WTR} number of idle cycles between a WRITE followed by a READ command. The t_{WTR} should be configured according to the LPDDR device being used. The t_{WTR} is referenced from the first positive clock edge after the last data-in pair. 0 1 clock 1 2 clocks
19–18 t_{RP}	SDRAM Row Precharge Delay. This control bit determines the number of idle clocks inserted between a precharge command and the next row activate command to the same bank. Hardware reset initializes the controller to insert 3 clocks. Following a PRECHARGE command, a subsequent command to the same bank cannot be issued until t_{RP} is met. 00 1 clock 01 2 clocks 10 3 clocks[1] 11 4 clocks
17–16 t_{MRD}	tMRD–SDRAM Load Mode Register to ACTIVE Command. This control bits determines the minimum number of idle clocks required between a Load-Mode-Register (LMR) command to ACTIVE. Hardware reset initializes the controller to insert 2 clocks. 00 1 clock 01 2 clocks[2] 10 3 clocks 11 4 clocks
15 t_{WR}	SDRAM WRITE to PRECHARGE Command. Data for a fixed length WRITE burst may be followed by, or truncated with, a PRECHARGE command to the same bank (provided that auto precharge was not activated), and a full-page WRITE burst may be truncated with a PRECHARGE command to the same bank. The PRECHARGE command should be issued t_{WR} after the clock edge at which the last desired input data element is registered. t_{WR} control bit determines the number of idle clocks inserted between the last desired input data element and the next PRECHARGE command, as shown in Figure 19-11. The t_{WR} Bit field encoding is listed on Table 19-15. **Note:** The auto precharge mode requires a t_{WR} of at least one clock plus time, regardless of frequency.

Table 19-14. ESDCFG0/ESDCFG1 Field Descriptions (continued)

Field	Description
14–12 t_{RAS}	SDRAM ACTIVE to PRECHARGE Command. These control bits determine the minimum number of clocks required between a ACTIVE to PRECHARGE command to the same bank. Hardware reset initializes the controller to insert 6 clocks. Following an ACTIVE command, a subsequent PRECHARGE command to the same bank cannot be issued until t_{RAS} is met. Figure 19-12 presents an example of a single read (without auto precharge). It should be noticed that the PRECHARGE command is not allowed at T3 and at T4, since t_{RAS} will be violated (for t_{RAS}= 4 clock cycles). **Note:** t_{RAS} also defines the minimum period of time that the SDRAM must remain in self refresh mode, as it shown in Figure 19-13. 000 1 clock 001 2 clocks 010 3 clocks 011 4 clocks 100 5 clocks 101 6 clocks[2] 110 7 clocks 111 8 clocks
11–10 t_{RRD}	ACTIVE Bank A to ACTIVE Bank B Command. A subsequent ACTIVE command to a different row in the same bank can only be issued after the previous active row has been “closed” (precharged). A subsequent ACTIVE command to another bank can be issued while the first bank is being accessed, which results in a reduction of total row-access overhead. The minimum interval between successive ACTIVE commands to different banks is defined by t_{RRD} as shown in Figure 19-14 (for t_{RRD}=3). The t_{RRD} bits field encoding is listed below and it determines the number of idle clocks inserted between consecutive ACTIVE commands to different banks. 00 1 clock active to active (different banks) 01 2 clocks active to active (different banks)[2] 10 3 clocks active to active (different banks) 11 4 clocks active to active (different banks)
9–8 t_{CAS}	SDRAM CAS Latency. This field determines the latency between a read command and the availability of data on the bus, as shown in Figure 19-15. This field does not affect the second and subsequent data words in a burst. This control field has no effect on write cycles. CAS latency is initialized to 3 clocks following a hardware reset. 00 Reserved 01 Reserved[2] 10 2 clocks SDR and LPDDR SDRAM CAS latency 11 3 clocks SDR and LPDDR SDRAM CAS latency[2]
7	Reserved

Table 19-14. ESDCFG0/ESDCFG1 Field Descriptions (continued)

Field	Description
6–4 t_{RCD}	SDRAM Row to Column Delay. This field determines the number of clocks inserted between a row activate command and a subsequent read or write command to the same bank. Hardware reset initializes the delay to 3 clocks. 000 1 clock row to column delay 001 2 clocks row to column delay 010 3 clocks row to column delay[3] 011 4 clocks row to column delay 100 5 clocks row to column delay 101 6 clocks row to column delay 110 7 clocks row to column delay 111 8 clocks row to column delay
3–0 t_{RC}	SDRAM Row Cycle Delay. This control field determines the minimum delay between a refresh and any subsequent refresh or read/write access. This delay corresponds to the minimum row cycle time captured in the t_{RC}/t_{RFC} memory timing specification. The value programmed in t_{RC} is the number of clocks inserted between the refresh and subsequent refresh/activate command. An example timing diagram for t_{RC} can be found in Figure 19-18. The bit field settings are listed on Table 19-17. **Note:** The tRC control field is not used to enforce tRC timing for row activate to row activate within the same bank as this is implicitly guaranteed by the sum of tRCD + tCAS + tRP.Use regular paragraphs to summarize register function, then use the following special styles to define bit and field function.

1 Reset value for CSD0: optimal @ 133 MHz

2 CAS1 is not supported due to shared DQM pads in a system

3 Reset value for ESDCFG1: optimal @ 133 MHz

NOTE

The ESDCTL configuration registers reset value defines the timing parameters to meet the memory device optimal timing requirements at 133 MHz. If the external memory device operates at a lower frequency the user should reconfigure the timing parameters (according to the device electrical characteristics and operating conditions) for optimal performance.

Table 19-15 lists the tWR bit field encoding. Table 19-16 lists and summarizes the Enhanced SDRAM Controller configurable set of timing parameters for the SDRAM and LPDDR devices.

Table 19-15. tWR Bit Field Encoding

tWR	Write to Precharge (SDRAM)	Write to Precharge (LPDDR)[1]
0	1 clock	2 clocks
1	2 clocks[2]	3 clocks[2]

[1]Relevant in case that MDDR_EN bit is set, that is, the external memory device is a LPDDR.
[2]Reset value for CSD0 and CSD1.

Table 19-16. Configurable SDRAM/LPDDR Timing Parameters

Symbol	Description	Relevancy	Optimal Values at 133 MHz
t_{MRD}	Load Mode Register command to ACTIVE or REFRESH command	always	2 cycles
t_{WR}	Write recovery time (write to precharge)	commands to same bank	2 cycles
t_{RAS}	ACTIVE to PRECHARGE command	commands to same bank	6 cycles
t_{RRD}	ACTIVE bank A to ACTIVE bank B command	commands to different banks	2 cycles
t_{CAS}	READ to DATA out period (known as CAS LATENCY)	always	3 cycles
t_{RP}	PRECHARGE command period	commands to same bank	3 cycles
t_{RCD}	ACTIVE to READ or WRITE delay	commands to same bank	3 cycles
t_{RC}	ACTIVE to ACTIVE command period	commands to same bank	10 cycles
t_{WTR}	LPDDR READ to WRITE command delay	command to same bank	2 cycles
t_{XP}	LPDDR EXIT power down to next valid command delay	always	4 cycles

Figure 19-8 through Figure 19-17 show the bit field encoding and timing diagrams.

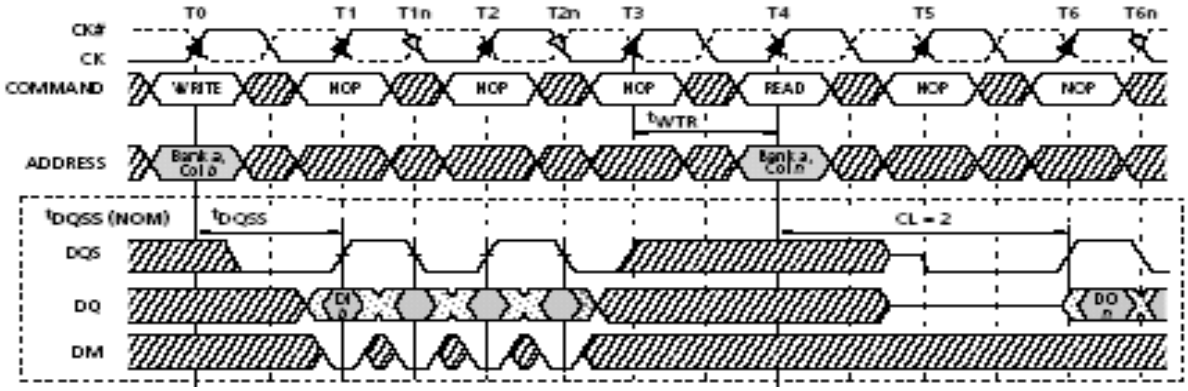


Figure 19-8. tWTRtRP Bit Field Encoding

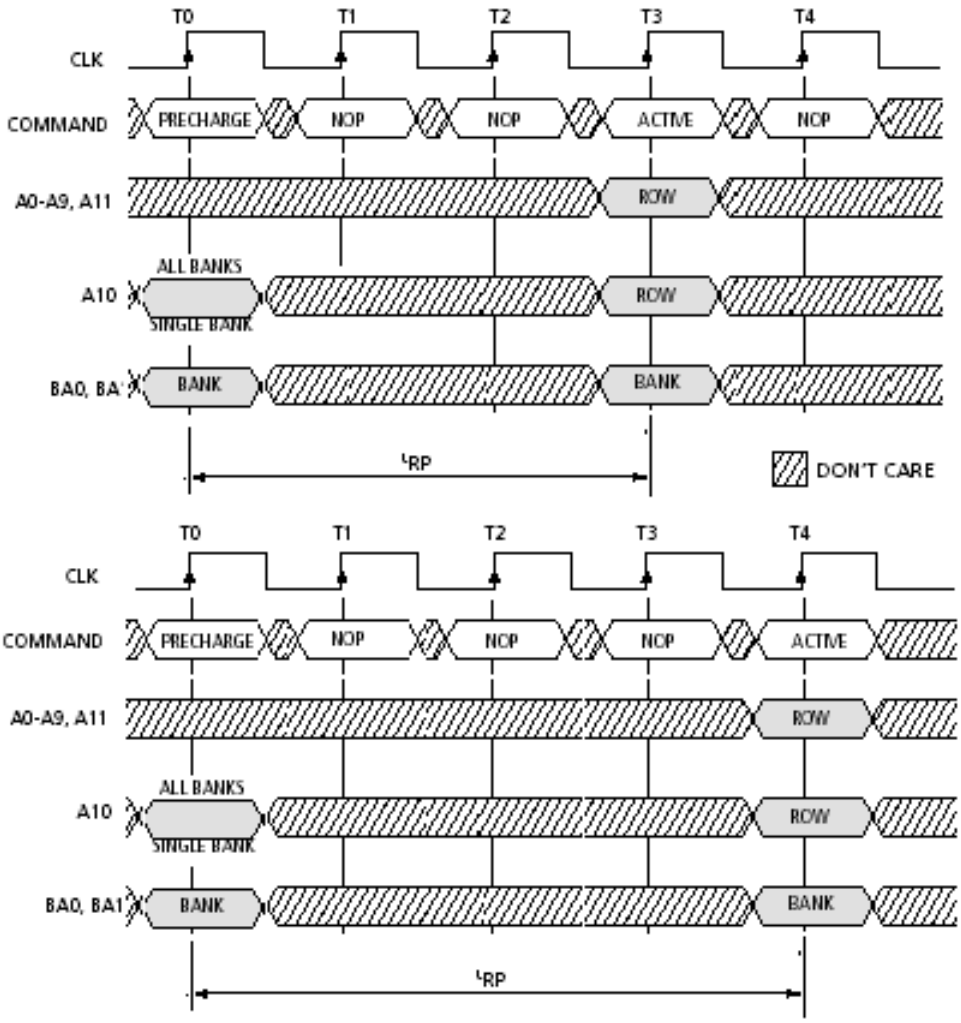


Figure 19-9. tRP—Precharge Delay Timing

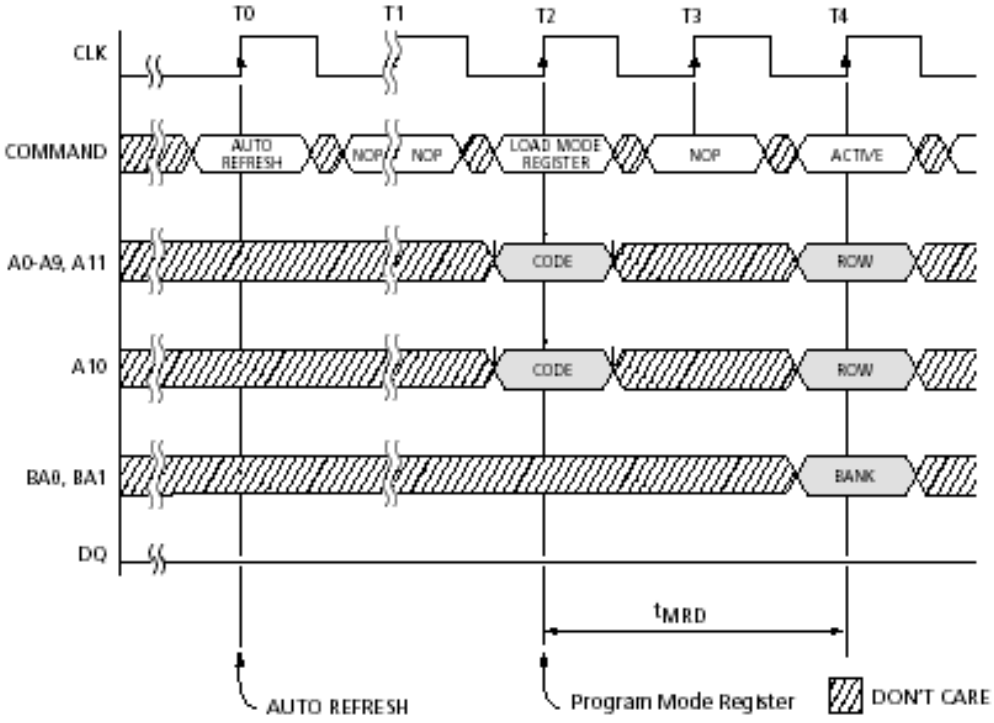


Figure 19-10. tMRD—SDRAM Load Mode Register to Active Command Timing Diagram

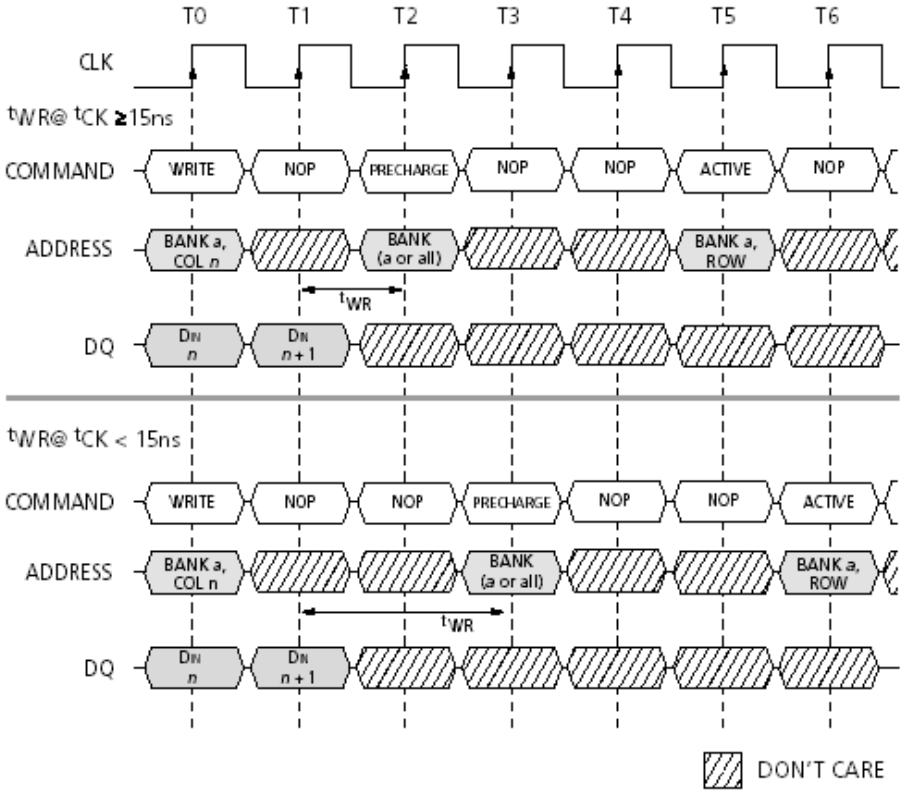


Figure 19-11. tWR—WRITE to PRECHARGE Timing Diagram

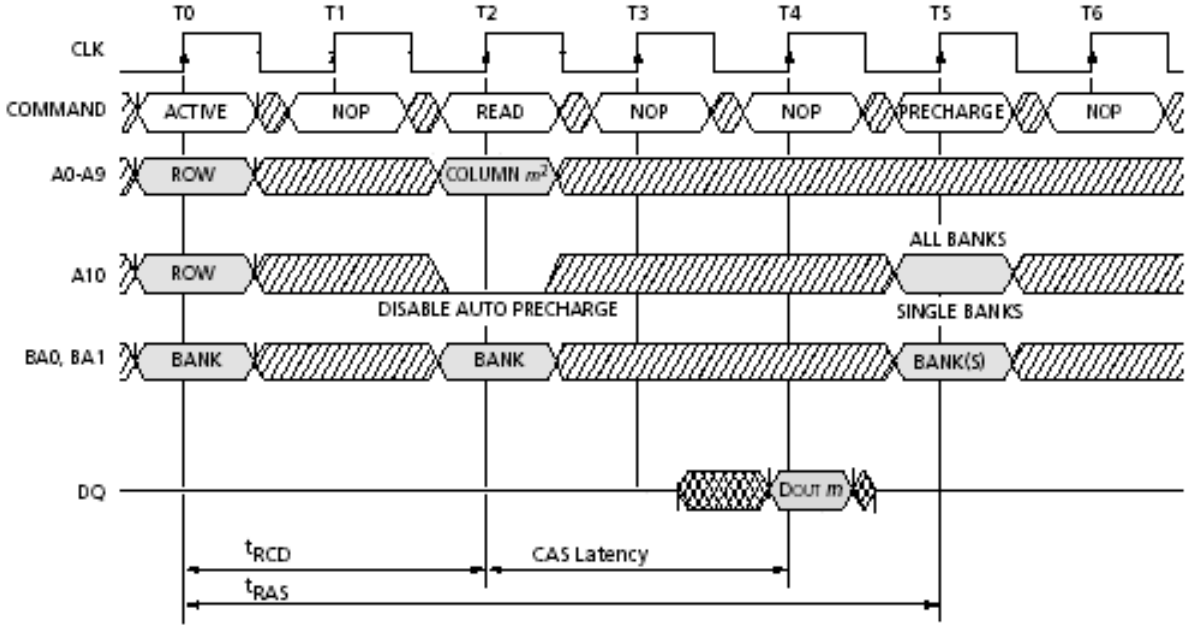


Figure 19-12. t_{RAS}—SDRAM ACTIVE to PRECHARGE Command Timing Diagram

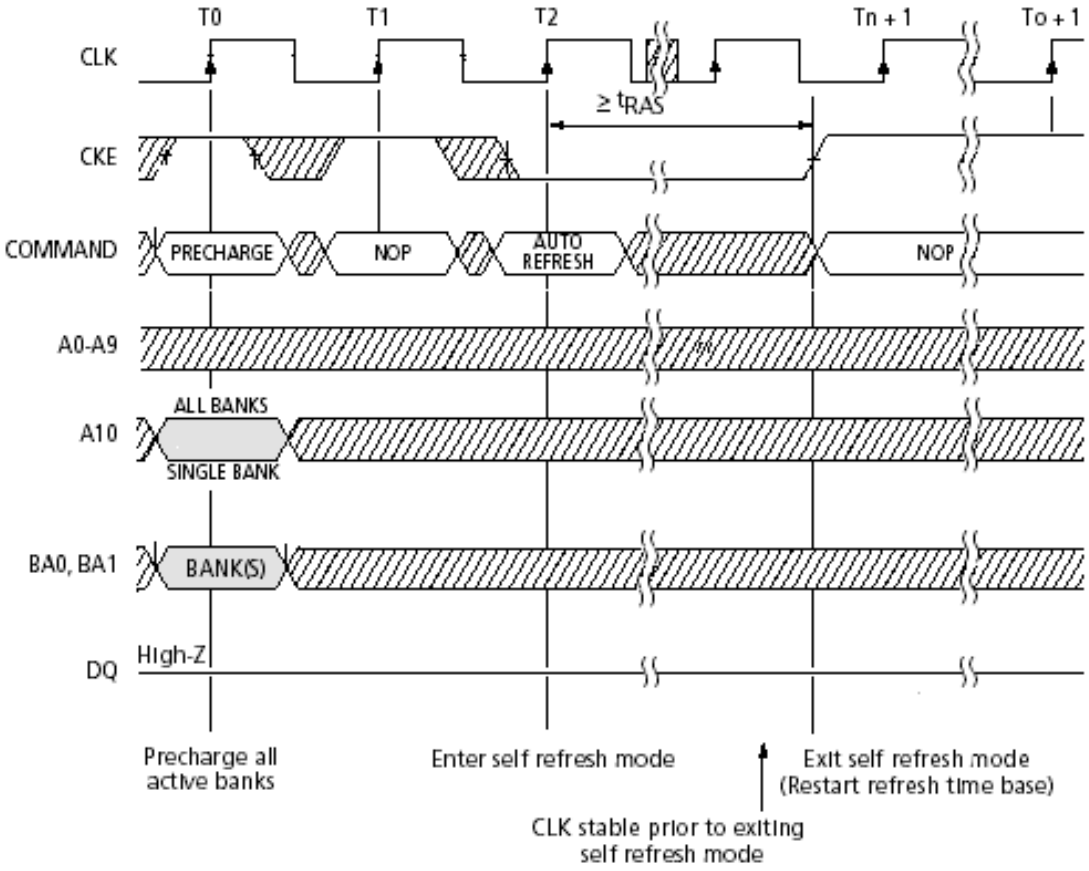


Figure 19-13. t_{RAS}—SELF REFRESH Mode Minimum Time Period

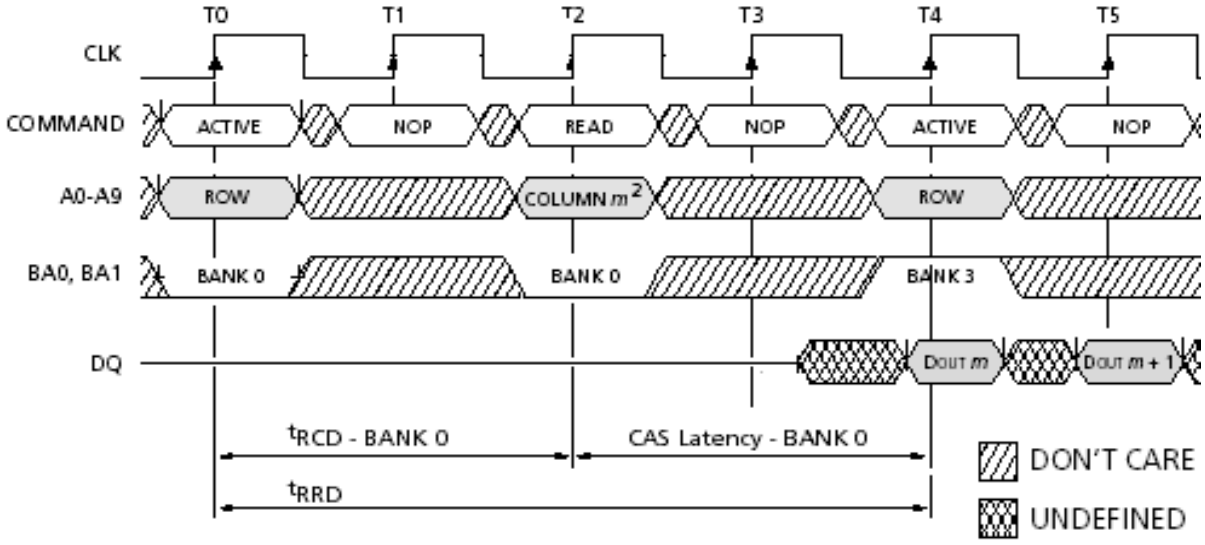


Figure 19-14. t_{RRD}—Alternating Bank Read Access

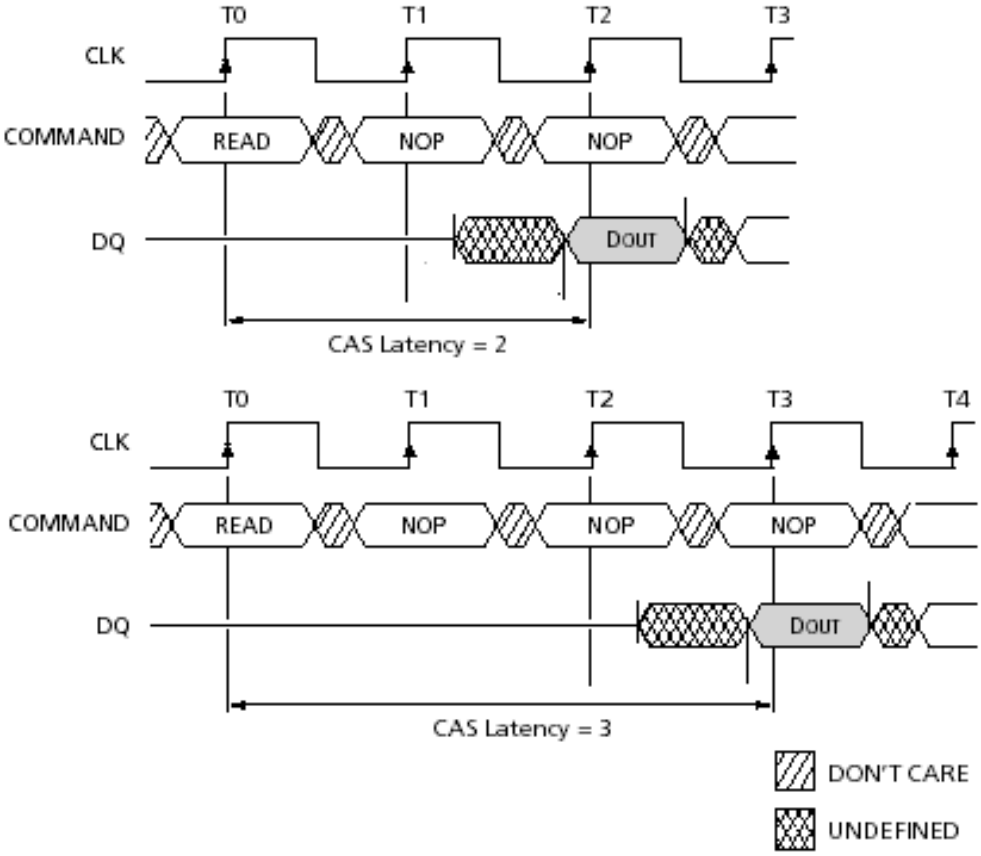


Figure 19-15. $_{SDR}$ CAS Latency Timing

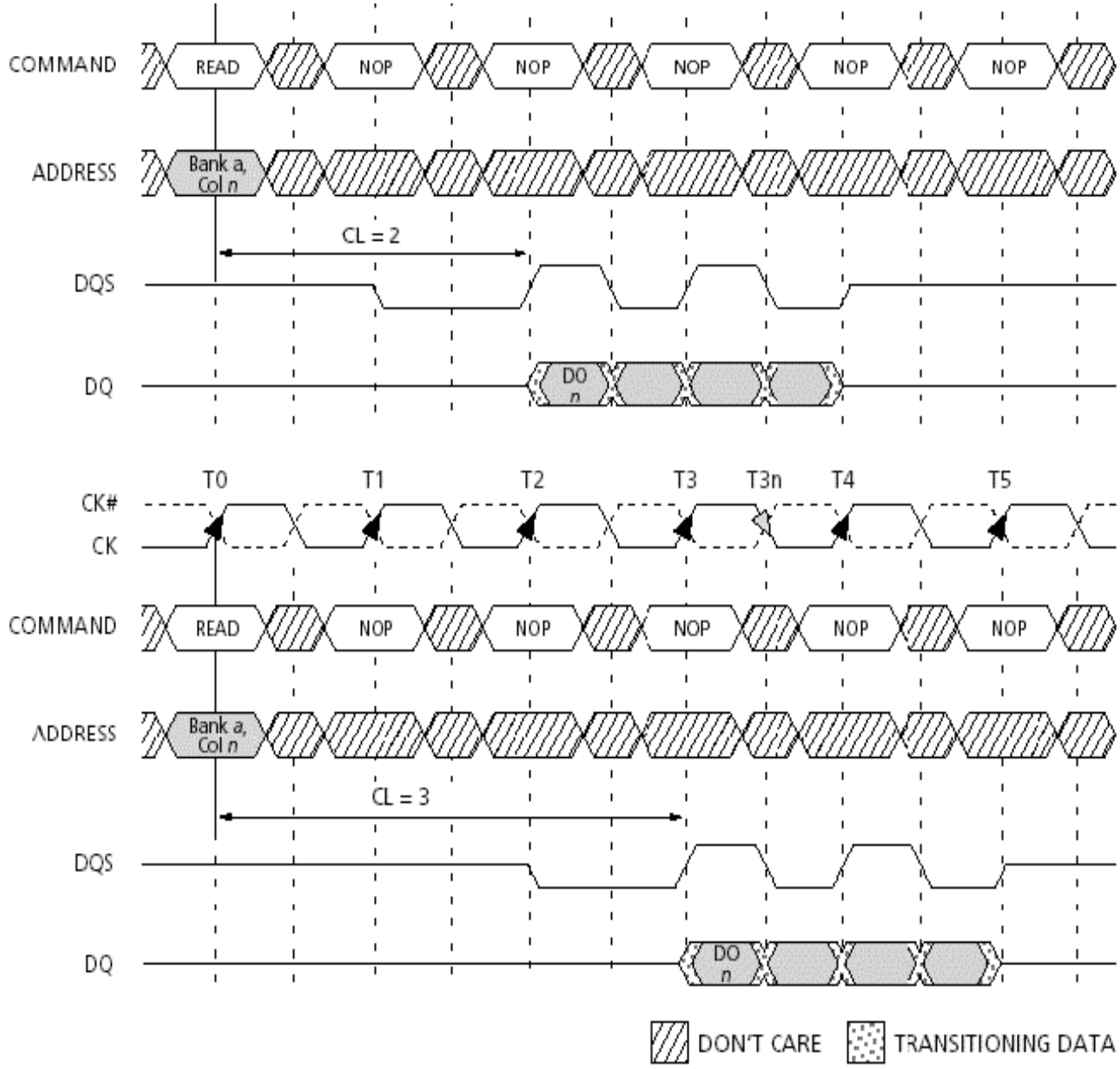


Figure 19-16. Mobile LPDDR CAS Latency Timing

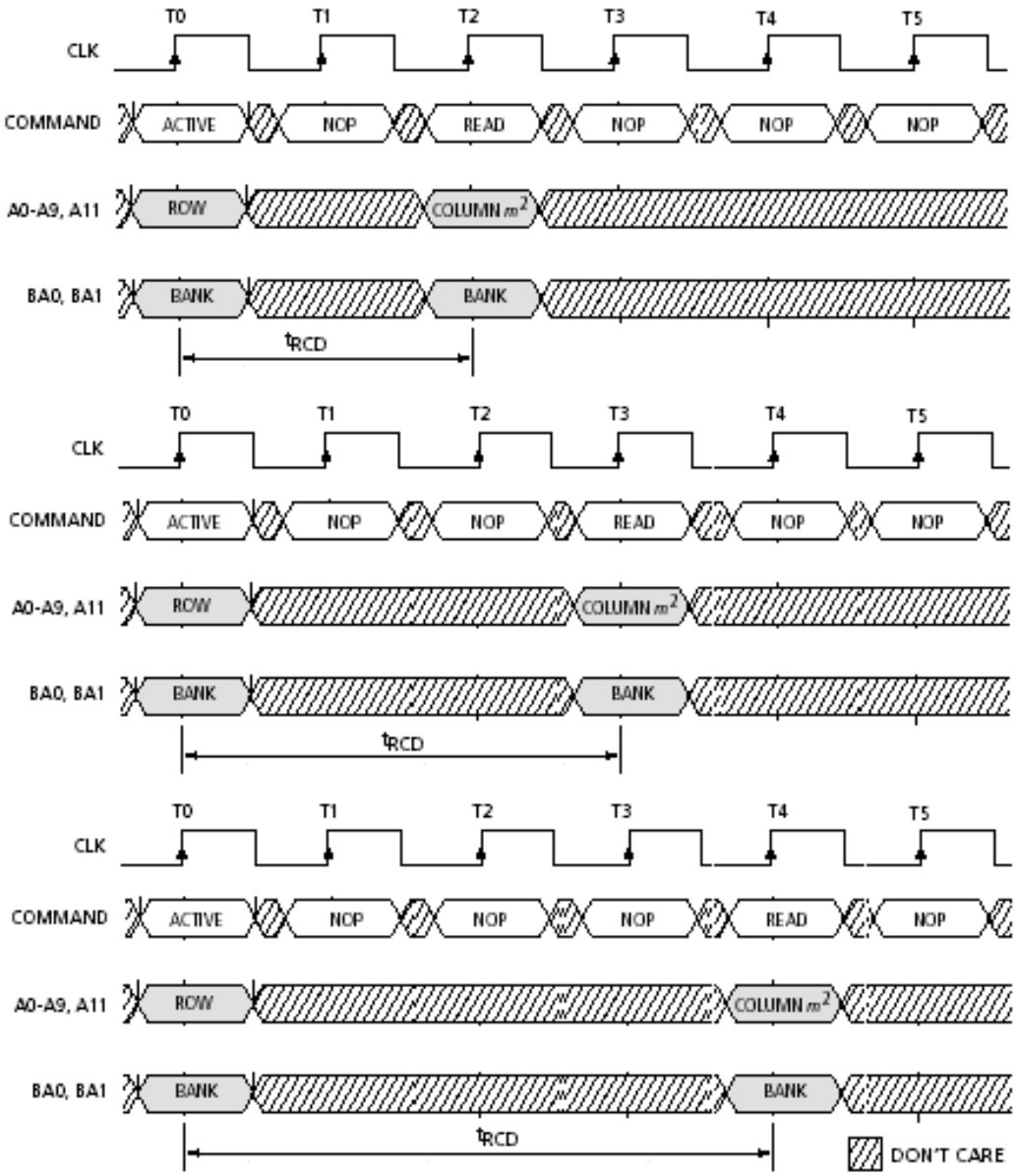


Figure 19-17. t_{RCD}—Row to Column Delay Timing

Table 19-17. t_{RC} Bit Field Encoding

t_{RC}[2:0]	Delay
0000	20 clocks
0001	2 clocks
0010	3 clocks
0011	4 clocks (reset value for CSD0)
0100	5 clocks
0101	6 clocks

Table 19-17. t_{RC} Bit Field Encoding (continued)

t_{RC}[2:0]	Delay
0110	7 clocks
0111	8 clocks
1000	9 clocks
1001	10 clocks[1]
1010	11 clocks
1011	12 clocks
1100	13 clocks
1101	14 clocks
1110	14 clocks
1111	16 clocks

[1]Reset value for CSD0.

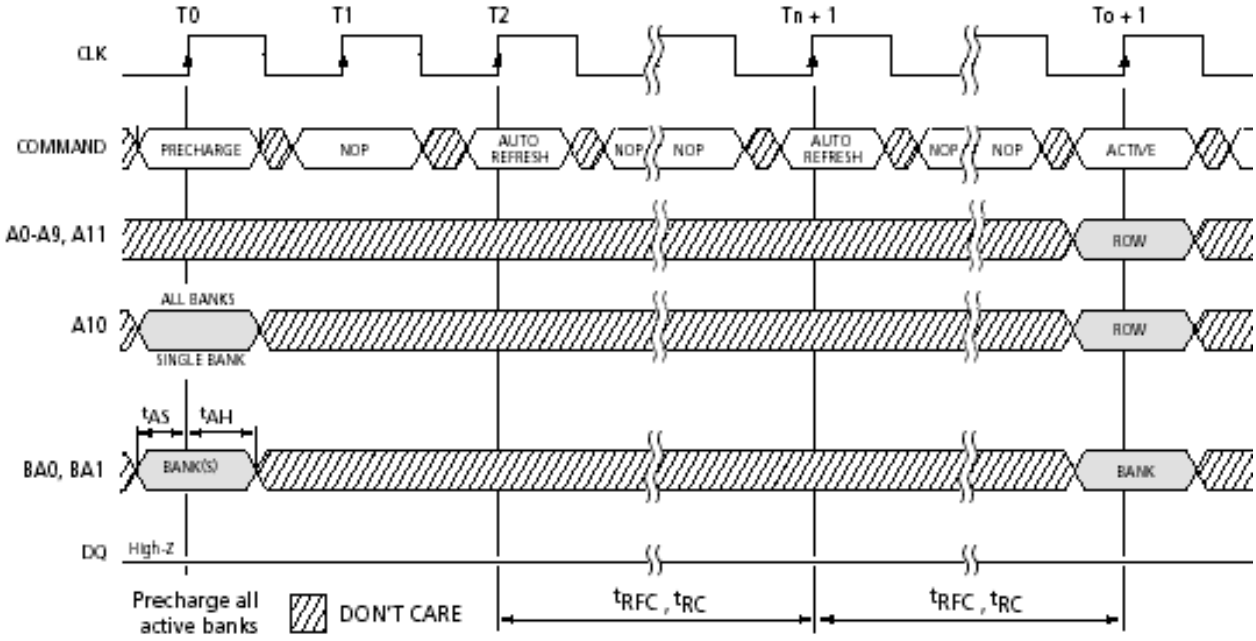


Figure 19-18. t_{RC}—Row Cycle Timing

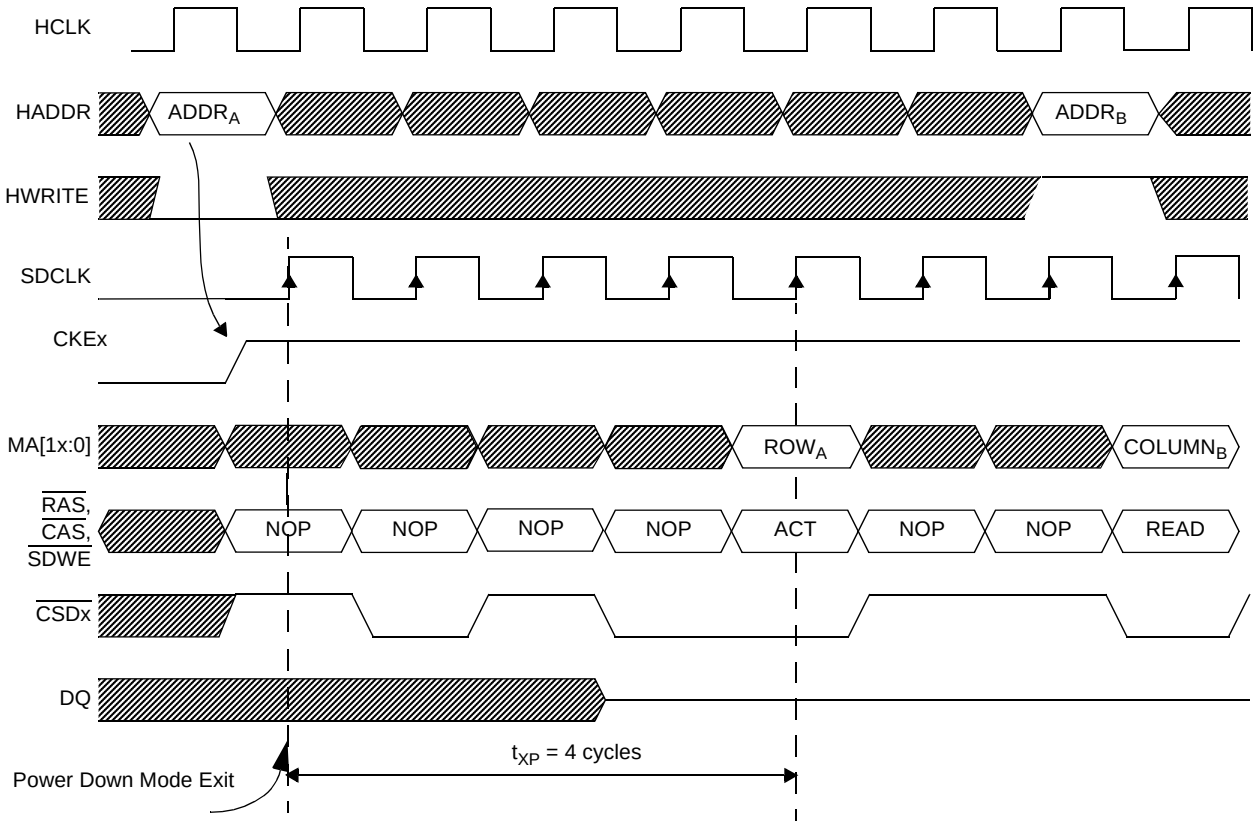


Figure 19-19. t_{XP}—New Command After Power Down Exit (4 Cycles)

19.3.3.3 ESDMISC Miscellaneous Register (ESDMISC)

This register contains the controlling various memory and control settings for the ESDCTL. The bit assignments for the register are shown in Figure 19-20, and the field descriptions for the bit assignments are listed in Table 19-18.

0xB800_1010 (ESDMISC) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SDRAMRDY	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	LHD	MDDR_MDIS	0	MDDR EN	0	0
W													MDDR_DL_RST		RST	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 19-20. ESDMISC Miscellaneous Register (ESDMISC)

Table 19-18. Enhanced SDRAM Control Register (ESDCTL0/1) Field Descriptions

Field	Description
31 SDRAMRDY	External SDRAM/LPDDR Device Status. This is a read-only status bit, that indicates the state of the external memory device(s). This bit is cleared at reset. This bit is set after the 200 µs SDRAM/LPDDR external memory wakeup period. After the wakeup period the software can start the external memory initialization (as described in Section 19.5.4, “SDRAM/LPDDR Initialization Sequence”). 0 SDRAM/LPDDR external device is not ready for use. 1 SDRAM/MMDR external device is ready for use.
30–6	Reserved
5 LHD	Latency Hiding Disable. This bit disables the command anticipation (latency hiding) mechanism. If this bit is set, the M3IF/ESDCTL will operate in MIF1 (non-optimized) mode as shown in Figure 19-29 and Figure 19-30. The first memory command of a new access is sent to the memory only after the previous access is completed, For example: the last data word of a burst has been read or written. The reset value of this bit is 0, meaning that the latency bidding is enabled (M3IF/ESDCTL works in MIF2 mode) as described in Section 19.4.1, “Enhanced SDRAM Controller Optimization Strategy”). 0 Latency Hiding Enable 1 Latency Hiding Disable
4 MDDR_MDIS	LPDDR Delay Line Measure Disable. This is a read/write bit, that, if set, disables the delay line measure unit. After reset, this bit is cleared, meaning the delay line measure unit is enabled. The measure time period is estimated to be around 2000 clock cycles of the AHB HCLK. 0 LPDDR delay line measure unit is enabled. 1 LPDDR delay line measure unit is disabled.
3 MDDR_DL_RST	LPDDR Delay Line Soft Reset. This is a write only bit, that if set the delay line unit is reset. After reset the delay unit will automatically (if LPDDR_MDIS is cleared) start a new measurement. 0 LPDDR Delay Line is not reset. 1 LPDDR Delay Line is reset.

Table 19-18. Enhanced SDRAM Control Register (ESDCTL0/1) Field Descriptions (continued)

Field	Description
2 MDDREN	Enable Mobile/Low Power DDR SDRAM. This bits activates the LPDDR Interface and enable the pipe line to work in LPDDR mode. 0 Enable Mobile SDR SDRAM operation 1 Enable Mobile DDR SDRAM operation
1 RST	Software Initiated Local Module Reset. This bit generate local module reset to the ESDCTL. Writing a 1 to RST bit results in a one cycle reset pulse to the controller. This bit is always read as 0. All ESDCTL registers are not affected by the software reset, in order to keep the REFRESH mechanism active as initially configured, so the SDRAM/LPDDR data is not violated. A burst terminate command is issued to the memory after the soft reset (to terminate any active bursts, in order to prevent potential contention on the DATA pads). During the software RESET an ERROR response (HRESP[1]=1) is broadcast to all masters with active access to the ESDCTL by the Multi Master Memory Interface (M3IF) module and the M3IF arbitration pipeline is cleared. For detailed information on error response refer to M3IF module specification. **Note:** After soft reset, a Precharge all command must be issued prior to normal usage of the ESDCTL. 0 Soft Reset is disabled. 1 Soft Reset is initiated.
0	Reserved

19.3.3.4 MDDR Delay Line 1 Configuration Debug Register

This debug register controls delay line 1 functionality, that is, DQS[0] delay used during READ cycles. It allows to override/manually set the delay of DQS[0] line, that is used during READ cycles of BYTE[0]. The delay line compensates for process variations, and produces a constant delay regardless of the process, temperature and voltage. The bit assignments for the register are shown in Figure 19-21, and the field descriptions for the bit assignments are listed in Table 19-19.

0xB800_1020 (ESDCDLY1) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SEL_DLY_REG_1	0	0	0	0	DLY_CORR_1										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	DLY_REG_1										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 19-21. MDDR Delay Line 1 Configuration Debug Register

Table 19-19. Enhanced MDDR Delay Line 1 Control Register (ESDCDLY1) Field Descriptions

Field	Description
31 SEL_DLY_REG_1	SEL_DLY_REG_1 bit selects the delay used by delay line 1. It selects between a quarter of a cycle (measured) minus the delay line 1 correction factor field (DLY_CORR_1) and delay line 1 register value (DLY_REG_1). 0 Delay line 1value is, a quarter of a cycle (measured) minus the delay line 1 correction factor field. 1 Delay line 1value is, the value of delay line 1 register field (skipping the measurement).
30–27	Reserved
26–16 DLY_CORR_1	This field is the delay line 1 correction factor. The correction factor is used only if SEL_DLY_REG_1 is cleared. The correction factor value is used to compensate a minimum delay (in number of inverters units) from the measured delay (the minimum delay varies with the process, voltage and temperature changes).
15–11	Reserved
10–0 DLY_REG_1	This field is the delay (in number of inverters units) that will be used by delay line 1, if the SEL_DLY_REG1 bit is set. Since the delay is process, temperature and voltage dependent, for a given value of this filed we get different delay values.

19.3.3.5 MDDR Delay Line 2 Configuration Debug Register

This debug register controls delay line 2 functionality, that is, DQS[1] delay used during READ cycles. It allows to override/manually set the delay of DQS[1] line, that is used during READ cycles of BYTE[1]. The bit assignments for the register are shown in Figure 19-22, and the field descriptions for the bit assignments are listed in Table 19-20.

0xB800_1024 (ESDCDLY2) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SEL_DLY_REG_2	0	0	0	0	DLY_CORR_2										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	DLY_REG_2										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 19-22. MDDR Delay Line 2 Configuration Debug Register

Table 19-20. Enhanced MDDR Delay Line 2 Control Register (ESDCDLY2) Field Descriptions

Field	Description
31 SEL_DLY_REG_2	SEL_DLY_REG_2 bit selects the delay used by delay line 2. It selects between a quarter of a cycle (measured) minus the delay line 2 correction factor field (DLY_CORR_2) and delay line 2 register value (DLY_REG_2). 0 Delay line 1value is, a quarter of a cycle (measured) minus the delay line 2 correction factor field. 1 Delay line 1value is, the value of delay line 2 register field (skipping the measurement).
30–27	Reserved
26–16 DLY_CORR_2	This field is the delay line 1 correction factor. The correction factor is used only if SEL_DLY_REG_2 is cleared. The correction factor value is used to compensate a minimum delay (in number of inverters units) from the measured delay (the minimum delay varies with the process, voltage and temperature changes).
15–11	Reserved
10–0 DLY_REG_2	This field is the delay (in number of inverters units) that will be used by delay line 1, if the SEL_DLY_REG2 bit is set. Since the delay is process, temperature and voltage dependent, for a given value of this filed we get different delay values.

19.3.3.6 MDDR Delay Line 3 Configuration Debug Register

This debug register controls delay line 3 functionality, for example, DQS[2] delay used during READ cycles. It allows to override/manually set the delay of DQS[2] line, that is used during READ cycles of BYTE[2]. The bit assignments for the register are shown in Figure 19-23, and the field descriptions for the bit assignments are listed in Table 19-21.

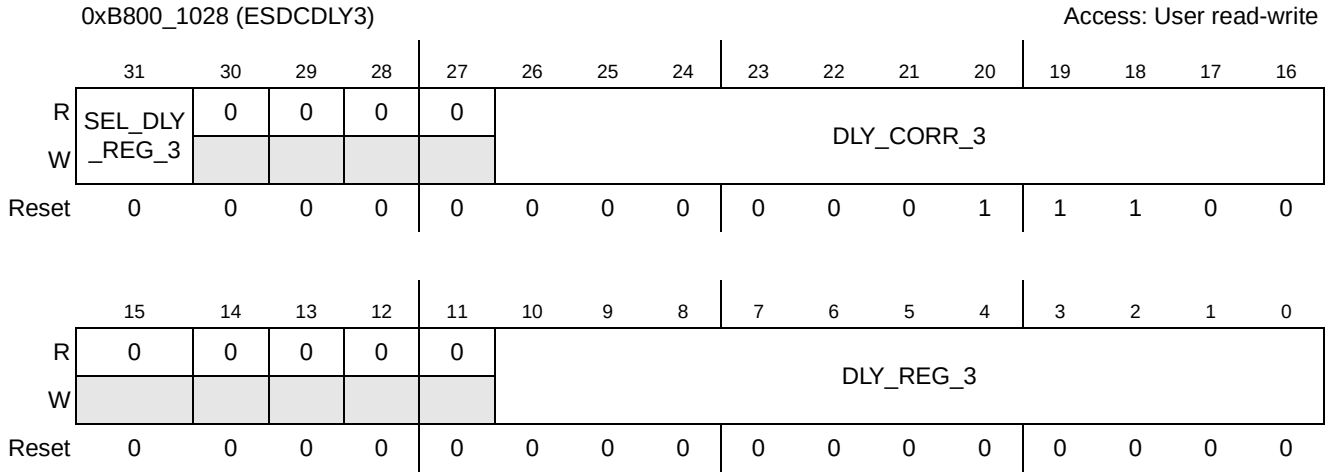


Figure 19-23. MDDR Delay Line 3 Configuration Debug Register

Table 19-21. Enhanced MDDR Delay Line 3 Control Register (ESDCDLY3) Field Descriptions

Field	Description
31 SEL_DLY_REG_3	SEL_DLY_REG_3 bit selects the delay used by delay line 3. It selects between a quarter of a cycle (measured) minus the delay line 3 correction factor field (DLY_CORR_3) and delay line 3 register value (DLY_REG_3). 0 Delay line 1value is, a quarter of a cycle (measured) minus the delay line 3 correction factor field. 1 Delay line 1value is, the value of delay line 3 register field (skipping the measurement).
30–27	Reserved
26–16 DLY_CORR_3	This field is the delay line 3 correction factor. The correction factor is used only if SEL_DLY_REG_3 is cleared. The correction factor value is used to compensate a minimum delay (in number of inverters units) from the measured delay (the minimum delay varies with the process, voltage and temperature changes)..
15–11	Reserved
10–0 DLY_REG_3	This field is the delay (in number of inverters units) that will be used by delay line 3, if the SEL_DLY_REG3 bit is set. Since the delay is process, temperature and voltage dependent, for a given value of this filed we get different delay values.

19.3.3.7 MDDR Delay Line 4 Configuration Debug Register

This debug register controls delay line 4 functionality, for example, DQS[3] delay used during READ cycles. It allows to override/manually set the delay of DQS[3] line, that is used during READ cycles of BYTE[3]. The bit assignments for the register are shown in Figure 19-24 and the field descriptions for the bit assignments are listed in Table 19-22.

0xB800_102C (ESDCDLY4) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SEL_DLY_REG_4	0	0	0	0	DLY_CORR_4										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	DLY_REG_4										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 19-24. MDDR Delay Line 4 Configuration Debug Register

Table 19-22. Enhanced MDDR Delay Line 4 Control Register (ESDCDLY4) Field Descriptions

Field	Description
31 SEL_DLY_REG_4	SEL_DLY_REG_4 bit selects the delay used by delay line 4. It selects between a quarter of a cycle (measured) minus the delay line 4 correction factor field (DLY_CORR_4) and delay line 4 register value (DLY_REG_4). 0 Delay line 1value is, a quarter of a cycle (measured) minus the delay line 4 correction factor field. 1 Delay line 1value is, the value of delay line 4 register field (skipping the measurement).
30–27	Reserved
26–16 DLY_CORR_4	This field is the delay line 4 correction factor. The correction factor is used only if SEL_DLY_REG_4 is cleared. The correction factor value is used to compensate a minimum delay (in number of inverters units) from the measured delay (the minimum delay varies with the process, voltage and temperature changes)..
15–11	Reserved
10–0 DLY_REG_4	This field is the delay (in number of inverters units) that will be used by delay line 4, if the SEL_DLY_REG4 bit is set. Since the delay is process, temperature and voltage dependent, for a given value of this filed we get different delay values.

19.3.3.8 MDDR Delay Line 5 Configuration Debug Register

This debug register controls delay line 5 functionality, that is, DATA bus delay used during WRITE cycles. It allows to override/manually set the delay of the DATA bus, during WRITE cycles. The bit assignments for the register are shown in Figure 19-25, and the field descriptions for the bit assignments are listed in Table 19-23.

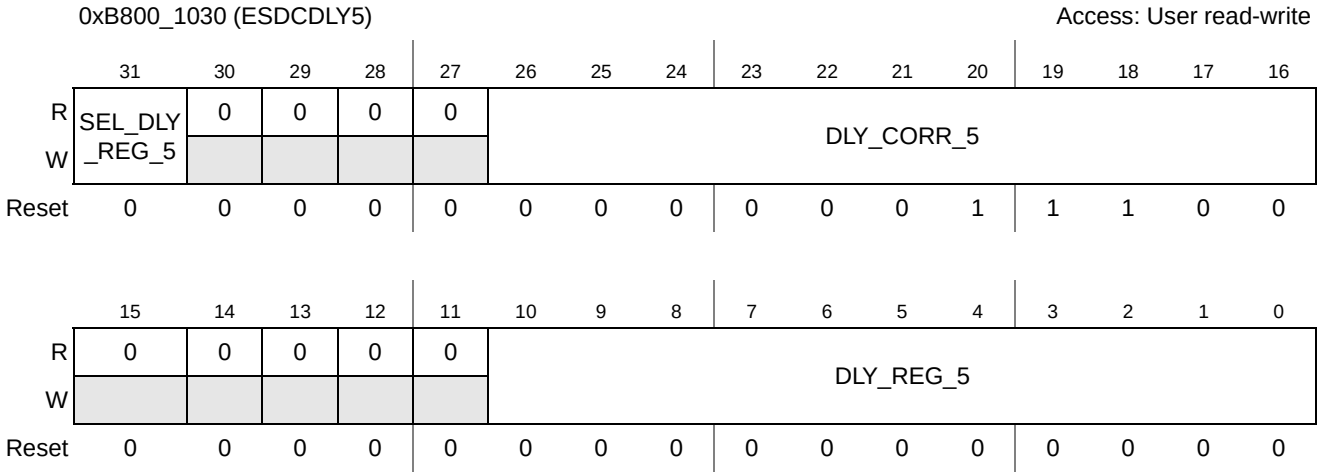


Figure 19-25. MDDR Delay Line 5 Configuration Debug Register

Table 19-23. Enhanced MDDR Delay Line 5 Control Register (ESDCDLY5) Field Descriptions

Field	Description
31 SEL_DLY_REG_5	SEL_DLY_REG_5 bit selects the delay used by delay line 5. It selects between a quarter of a cycle (measured) minus the delay line 5 correction factor field (DLY_CORR_5) and delay line 5 register value (DLY_REG_5). 0 Delay line 1value is, a quarter of a cycle (measured) minus the delay line 4 correction factor field. 1 Delay line 1value is, the value of delay line 4 register field (skipping the measurement).
30–27	Reserved
26–16 DLY_CORR_5	This field is the delay line 5 correction factor. The correction factor is used only if SEL_DLY_REG_5 is cleared. The correction factor value is used to compensate a minimum delay (in number of inverters units) from the measured delay (the minimum delay varies with the process, voltage and temperature changes).
15–11	Reserved
10–0 DLY_REG_5	This field is the delay (in number of inverters units) that will be used by delay line 5, if the SEL_DLY_REG4 bit is set. Since the delay is process, temperature and voltage dependent, for a given value of this filed we get different delay values.

19.3.3.9 MDDR Delay Line Cycle Length Debug Register

The MDDR Delay Line Cycle Length Debug Registers is a read only register that shows the number of inverters that “fit” in a cycle. The reset value is unknown, because after reset the register value is updated from the measured delay. The register value represents the number of inverters required to achieve a delay of one clock cycle, as a function of the IC conditions (temperature, voltage, frequency, process). The bit assignments for the register are shown in Figure 19-26, and the field descriptions for the bit assignments are listed in Table 19-24.

0xB800_1034 (ESDCDLYL) Access: User read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	DLY_CYCLE_LENGTH										
W																
Reset	0	0	0	0	0	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Figure 19-26. MDDR Delay Line Cycle Length Debug Register

Table 19-24. Enhanced MDDR Delay Line Cycle Length Debug Register (ESDCDLY1) Field Descriptions

Field	Description
31–11	Reserved
10–0 DLY_CYCLE_LENGTH	DLY_CYCLE_LENGTH shows the number of Inverters that “fit” in a cycle.

19.4 Functional Description

General Enhanced SDRAM Controller operating characteristics are addressed in this section. The discussion starts with the optimization strategy (latency hiding/command anticipation), continues with one of the most basic of all DRAM controller features, the address multiplexor. The following subsections explain operation out of reset, hardware refresh and the low power modes. More on, each of the Enhanced SDRAM Controller operating modes are described. The discussion includes details on basic operation, relationship to SDRAM/LPDDR operating modes, and any special precautions which need to be observed. State and timing diagrams are included where appropriate.

The Enhanced SDRAM Controller is designed to support a broad range of broad range of JEDEC standard SDRAM/LPDDR configurations including devices of 64-, 128-, 256-, 512-Mbit, 1-Gbit and 2-Gbit densities. Given the physical size constraints of the target applications, the design support memory devices with data widths of 16 and 32 bits. Table 19-25 summarizes the devices supported by the design. Only 4 bank devices are supported. 133-MHz system bus operation is possible with PC133 compliant Single or Double Data Rate memory devices.

Each of the Enhanced SDRAM Controller operating modes are described in this section. The discussion includes details on basic operation, relationship to SDRAM/LPDDR operating modes, and any special precautions which need to be observed. State and timing diagrams are included where appropriate.

Table 19-25. JEDEC Standard Single/Double Data Rate SDRAMs

	SDRAM Configurations—4-Bank Devices											
Size	64 MBit		128 MBit		256 MBit		512MBit[1]		1-GBit[1]		2-GBit[1]	
Bus size	16	32	16	32	16	32	16	32	16	32	16[2]	32
Depth	4M	2M	8M	4M	16M	8M	32M	16M	64M	32M	N/A	64M
Refresh Rows	4096	4096	4096	4096	8192	8192	8192	8192	16384	16384	N/A	16384
Refresh rate (us)	15.6	31.25	15.6	15.6	7.81	7.81	7.81	7.81	3.91	3.91	N/A	3.91
Refresh cycles	2	1	2	2	4	4	4	4	8	8	N/A	8
Row Address	12	11	12	12	13	13	13	13	14	14	N/A	14
Col. Address	8	8	9	8	9	8	10	9	10	9	N/A	10

[1]Not ratified by JEDEC, row—column organization may change.
[2]2 -GB SDRAM/LPDDR (16-bit) are not supported/available.

19.4.1 Enhanced SDRAM Controller Optimization Strategy

SDRAM (SDR and LPDDR) memories provide high speed access by hiding the latency of consecutive memory accesses through a pipeline interface architecture. The resulting high bandwidth is achieved with a rather complex command interface and a large number of timing constraints that must be kept by the memory controller.

SDRAM, LPDDR and memories are organized in several independent banks. By issuing a row address and bank number to the memory device, the corresponding memory page is activated (opened). Consecutive read commands (together with the column address) into this memory page (same row address) have a low latency. Accessing another page in the same bank requires to close the open page by a PRECHARGE command, followed by the activation (ACTIVE command) of the new memory page (new row address).

In Figure 19-27 and Figure 19-28 examples for an SDR and LPDDR SDRAM read bursts are shown. Initially the cell in [COL a, ROW a] is active/open. Trying now to access/open the cell position [COL b, ROW b] requires first to close the old sell in [COL a, ROW a]. This is done with a precharge command (P on Figure 19-27). Now the access row address (ROW b) can be activated (A on Figure 19-27) before the column address (COL b) will be passed (read command R). The timing restrictions are as follows:

- The ACTIVE command can be issued only tRP cycles after the PRECHARGE command.
- The READ command can be issued only tRCD cycles after the ACTIVE command.
- The first data is available only tCAS cycles after the READ command has been issued.

Hence, in those examples a 4-word (8-word in LPDDR will be converted to 4-word with twice data width) read burst requires 9 cycles (6 cycles latency from precharge command to the first word on the external bus) or 6-1-1-1. A read access from an already open row is shown as well, the read burst from target cell [ROW b, COL c] on takes only 5 cycles (2 cycles latency from the read command to the first word on the external bus) or 2-1-1-1.

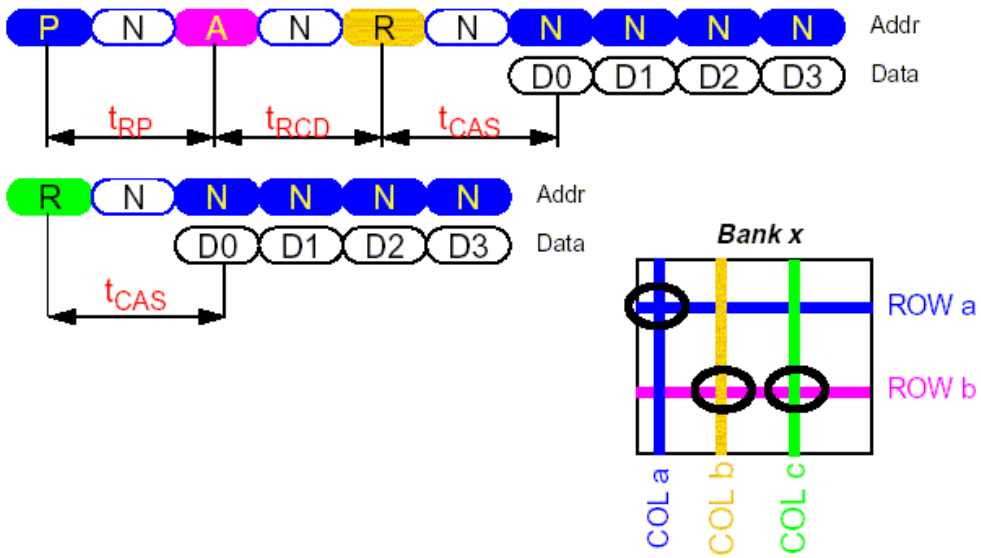


Figure 19-27. SDR SDRAM Read Burst Command Sequence Example

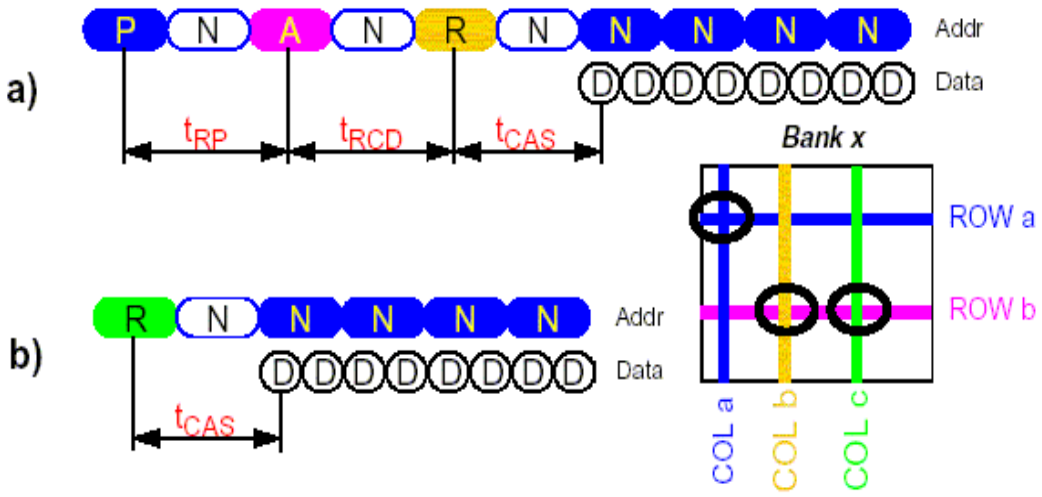


Figure 19-28. LPDDR SDRAM Read Burst Command Sequence Example

The fact that the Enhanced SDRAM Controller handles two devices and that each of them features 4 independent memory banks opens the opportunity for the controller to optimize the access timing of consecutive memory accesses by trying to hide as much as possible the latency of the first data word in a burst. The latency hiding is possible in the cases that are summarized in Table 19-26.

Table 19-26. Possibilities for Latency Hiding

Current Burst Access	Next Burst Access
SDRAM bank x	SDRAM bank y
SDRAM bank x (row y, col z)	SDRAM bank x (row y, col w)
LPDDR SDRAM bank x	LPDDR SDRAM bank y

Figure 19-29 (SDR) and Figure 19-30 (LPDDR) show two different optimization strategies. First “no optimization” strategy (referred as MIF1) and second the “medium level optimization” (referred as MIF2). For SDR SDRAM each strategy, two examples for two consecutive read accesses are given:

- An 8-word burst from the SDRAM with CAS latency of 2 cycles followed by an 8-word burst from the same bank and row (different column) with CAS latency of 2 cycles.
- An 8-word burst from one bank in the SDRAM followed by an 8-word burst from a different bank to the same SDRAM.

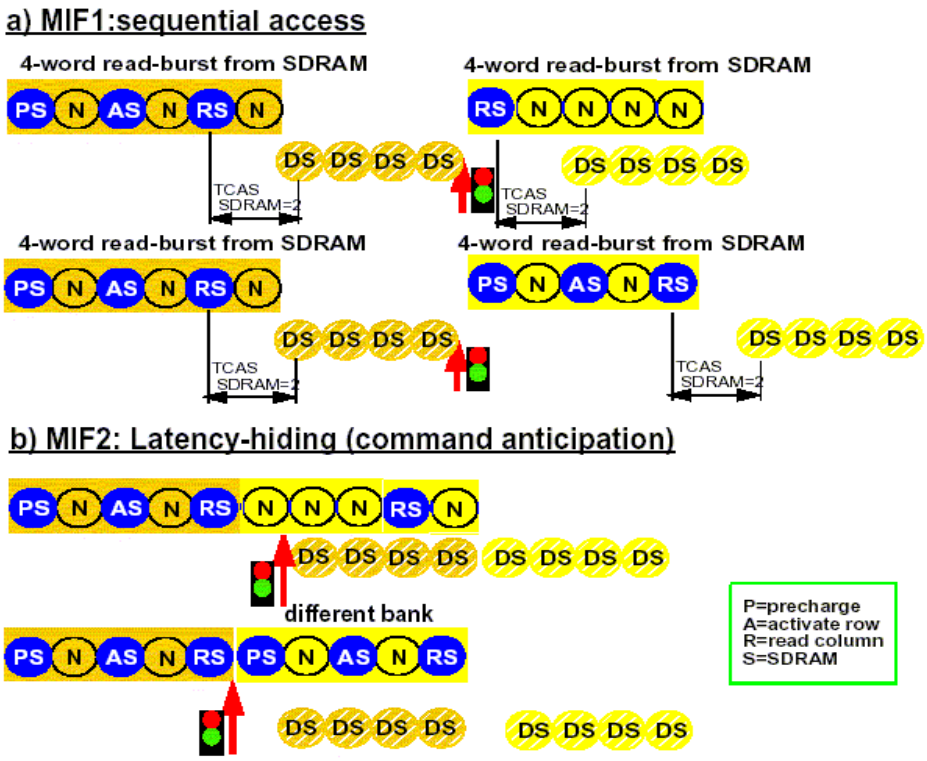


Figure 19-29. SDR SDRAM Optimization Strategies—MIF1 and MIF2 Examples

For LPDDR SDRAM each strategy, one example for two consecutive read accesses is given:

- An 8-word burst from one bank in the LPDDR SDRAM followed by a 8-word burst from a different bank to the same LPDDR SDRAM.

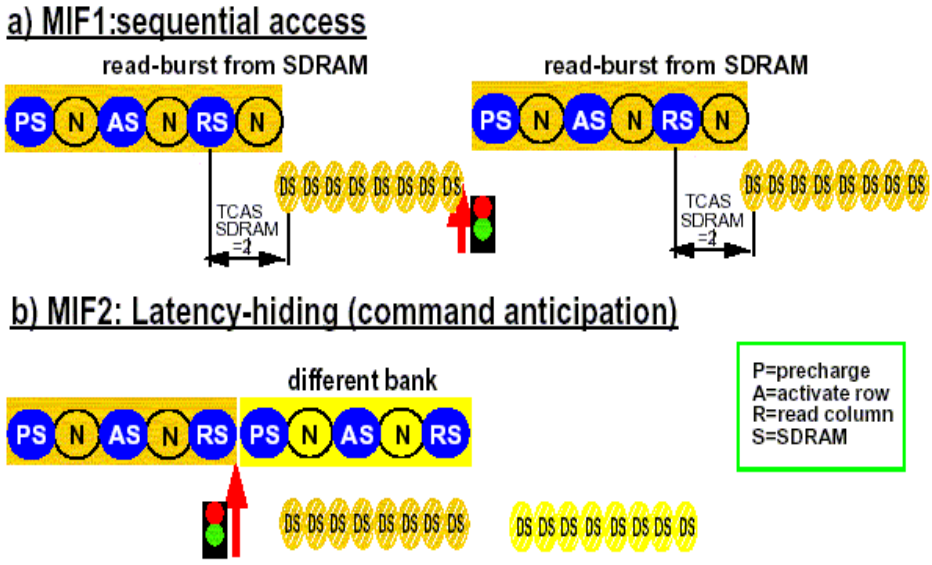


Figure 19-30. Mobile LPDDR SDRAM Optimization Strategies—MIF1 and MIF2 Examples

19.4.1.1 MIF1—No Optimization/Sequential Accesses

This is the non-optimized case shown in Figure 19-29 and Figure 19-30. The first memory command of a new access is sent to the memory only after the previous access is completed, for example, the last data word of a burst has been read or written. It can be seen in this example that although the 2 memory accesses follow with no delay between them, the bandwidth usage for the command (address) and data busses is far from being optimal. This no optimization/ sequential accesses occurs in the following cases:

- Only one master active/present in a given system—in this case only sequential commands are possible since the given master need to receive/send (READ/WRITE) all data's for one access before it can proceed to the next access, for example, command anticipation is not possible.
- Low density accesses, such as non-overlapped/consecutive/continuos requests, means that the next SDRAM request starts after the previous request is completed. In this low SDRAM utilization-only sequential accesses occurs.
- Large number of SINGLE or INCR (aborted after one data) accesses instead of burst type accesses. SINGLE/INCR refer to AMBA AHB bus protocol.
- The LHD (latency hiding disable) bit is set.

19.4.1.2 MIF2—Medium Level Optimization/Command Anticipation

This strategy is shown in Figure 19-29 and Figure 19-30. As soon as the address and command bus (refer as the SDRAM control bus) is no more used for issuing the previous memory access command, the controller can use this bus to start issuing the PRECHARGE/ACTIVE commands for the next scheduled memory access while the previous one is still active on the data bus. Two conditions limit the use of this optimization at a given time:

- Memory timing constraints must not be violated.
- The execution of the previous command should not be affected (for example, truncated).

This approach allows for hiding a part of the latency for the first data word or even the complete hiding of the latency in case that the burst length exceeds the maximal command sequence length.

19.4.1.3 Latency Hiding

The Enhanced SDRAM Controller optimization is based on command anticipation (MIF2), that is, the next access control phase (memory address and command) is driven during the previous access data phase (data flow from/to the memory), thus an overlap between accesses is created and latency is partially or fully hidden. Additional optimization (not implemented in the ESDCTL) can be achieved by control phase interleaving, that is, during idle cycles in the control phase (caused by the memory timing constraints like tRP, tRCD) two accesses control phases can be interleaved so the latency is fully hidden. The Enhanced SDRAM Controller optimization can occur only in a multi master system with a high SDRAM utilization (high density accesses).

As early mentioned, the ESDCTL optimize the command sequence toward the memories in order to hide latency, so the data bus will be used as much as possible under the access required and SDRAM/system initial configuration. At Figure 19-31 and Figure 19-32 latency hiding timing diagram example is shown when miss burst read from bank A is followed by a hit burst read request from the same chip select. The second read command is issued during the first access data phase, so first data of the second access (D10)

is valid immediately after the last data of the first access (D4 for the SDR and D8 for the LPDDR). The CAS latency (tCAS) is set to 2 cycles. The second access latency is fully hidden during the first access data phase.

The Mobile/Low Power DDR needs another cycle to prevent contention on the DQS signals when two different LPDDRs (two different chip selects) drive those data strobe signals (contention between the last two data cycles of the first transfer and the preamble of the second transfer).

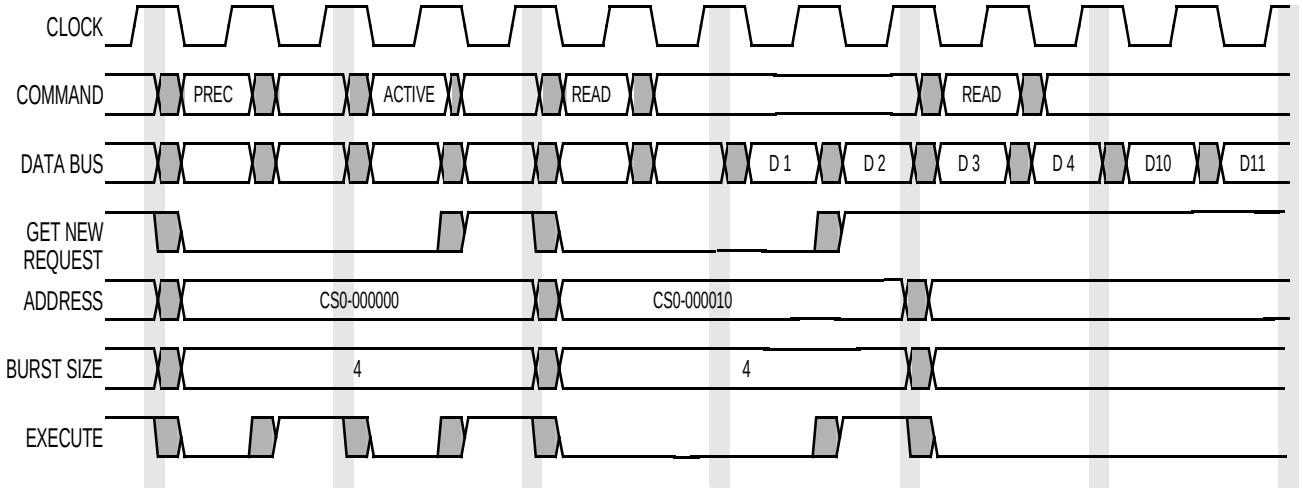


Figure 19-31. SDR Simple Read after Read Latency Hiding Timing Diagram

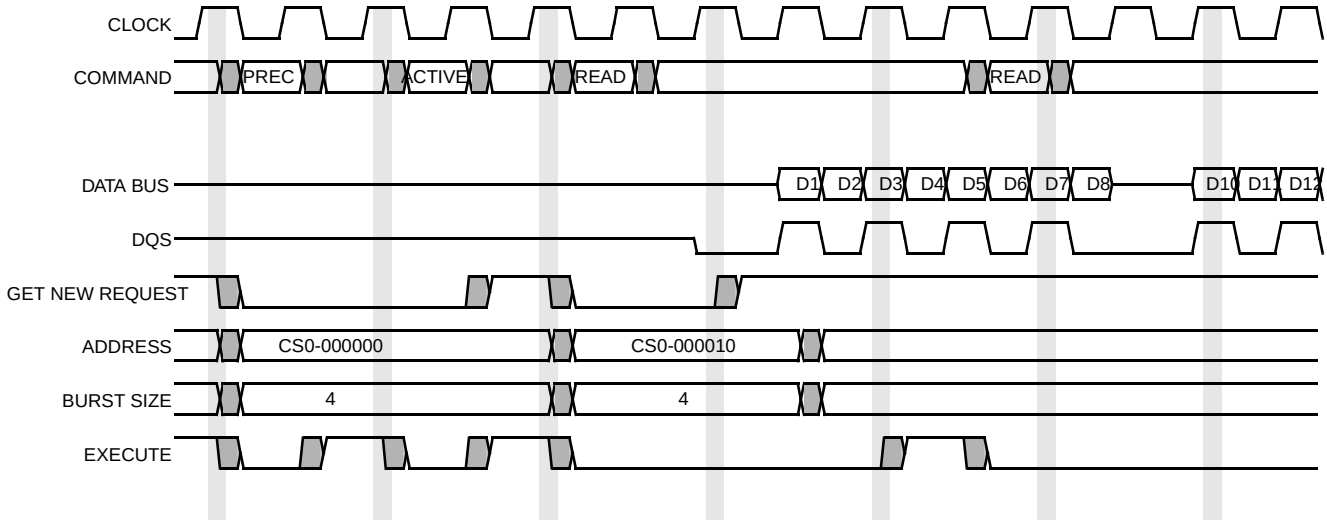


Figure 19-32. Mobile DDR Simple Read after Read Latency Hiding Timing Diagram

At Figure 19-33 and Figure 19-34 a burst read to CSD0 is followed by a miss burst write access to CSD1. Due to command anticipation the control phase of the WRITE command overlap the data phase of the READ command, so again the second access latency is fully hidden during the first access data phase. The first WRITE command to CSD1 (D10) is issued immediately after the last data (D4 for SDR and D8 for LPDDR) from CSD0, although the access to CSD1 is a miss access (CSD0 CAS latency is set to 3 cycles,

and both CSD burst length is set to 4 words. LPDDR burst length is 8 but is considered from the system point of view as a burst of 4 with double bus width).

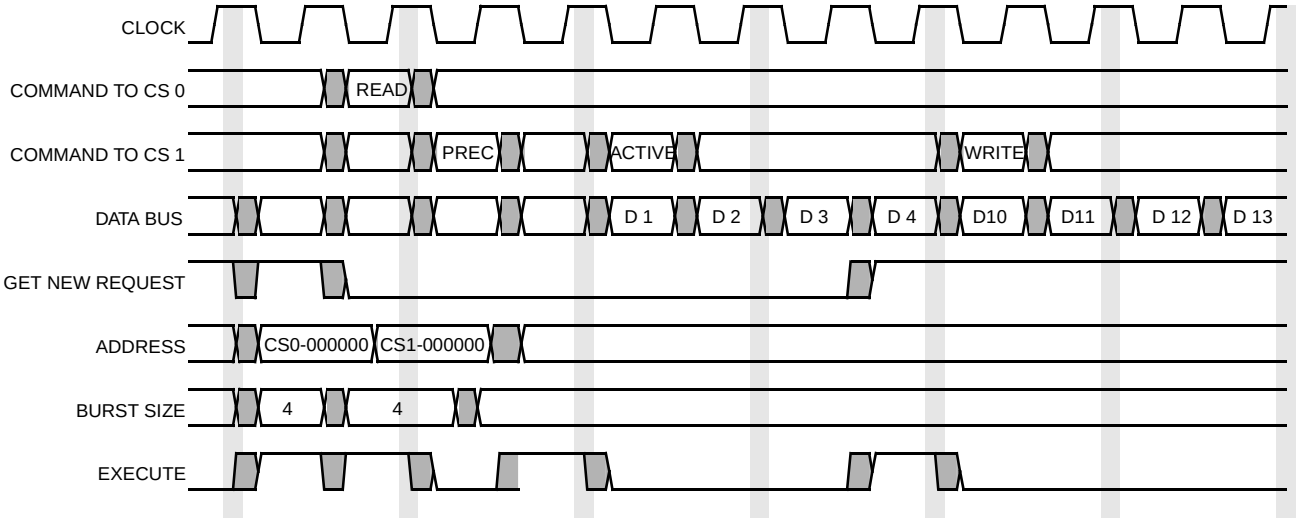


Figure 19-33. SDR Miss Write to CSD1 After Read from CSD0

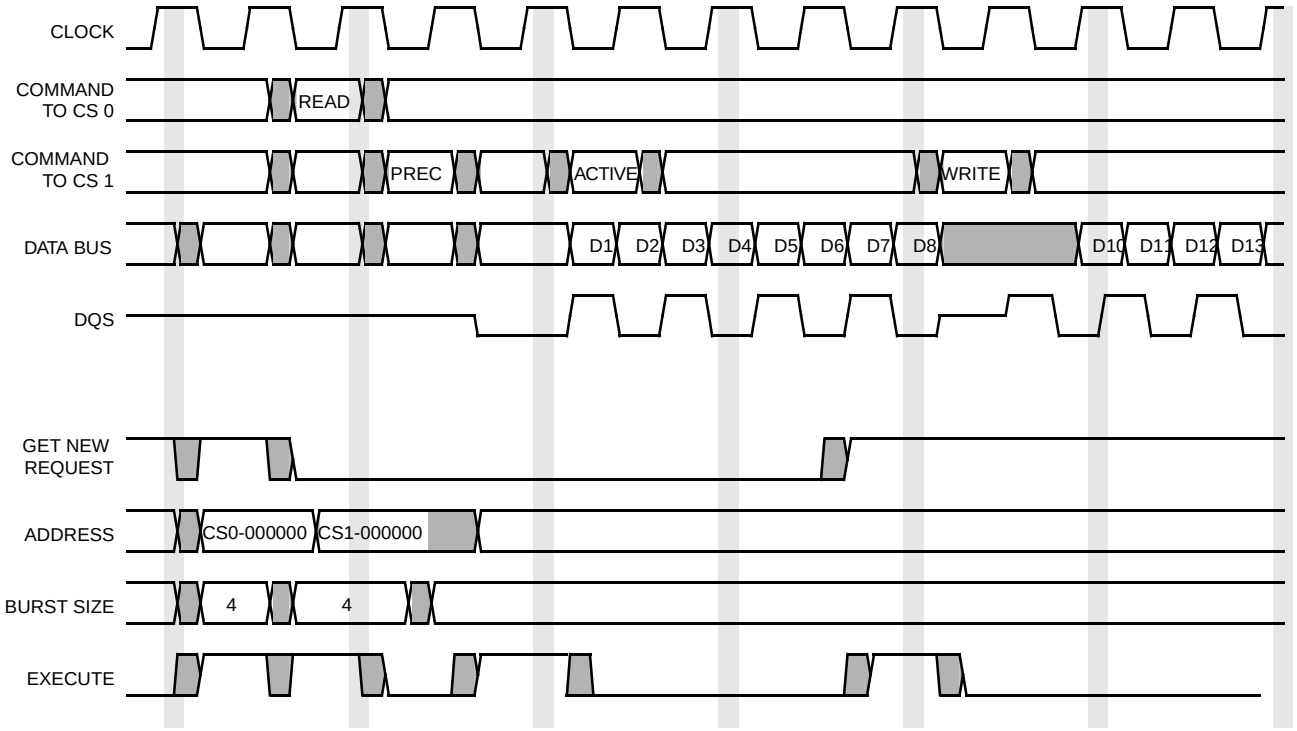


Figure 19-34. Mobile DDR Miss Write to CSD1 After Read from CSD0

19.4.2 Address Multiplexing

19.4.2.1 Multiplexed Address Bus

Table 19-27 illustrates several examples on how a CPU address is scrambled by the Enhanced SDRAM Controller to implement a contiguous address space.

Table 19-27. CPU to SDRAM/LPDDR Translation

CPU ADDRESS	16-bit SDRAM[1]	32-bit SDRAM[1]
A25	—	BA1
A24	BA1	BA0
A23	BA0	R11
A22	R11	R10
A21	R10	R9
A20	R9	R8
A19	R8	R7
A18	R7	R6
A17	R6	R5
A16	R5	R4
A15	R4	R3
A14	R3	R2
A13	R2	R1
A12	R1	R0
A11	R0	C9
A10	C9	C8
A9	C8	C7
A8	C7	C6
A7	C6	C5
A6	C5	C4
A5	C4	C3
A4	C3	C2
A3	C2	C1
A2	C1	C0
A1[2]	C0	-
A0[3]	-	-

1 For the SDRAM example a memory configuration with 10 columns and 12 rows is illustrated. The address translation is based on the following concept, COLUMN-ROW-BANK.

2 CPU A1 defines how the data masks are driven, that is, it is used as the byte enable for non word accesses. This bit has a regular/normal use only in case of 16-bit memory, while CPU A0 defines the 2 bytes (low and high) in the 16-bit word.

3 CPU A0 defines how the data masks are driven, that is, it is used as the byte enable for non word accesses to 32-bit memory device. Both CPU A0 and A1 defines the 4 bytes in the 32-bit word.

4. Legend: C=COLUMN, S=SEGMENT, R=ROW, BA=BANK

The Enhanced SDRAM Controller multiplexed address bus is aligned to the column addresses so that address line A1 always appears on pin MA0 for 16-bit memory devices and A2 always appears on MA0 for 32-bit memory devices. With this alignment, the “folding point” in the multiplexor is driven solely by the number of column address bits. Column bus widths of 8 to 11 bits are supported. Table 19-28 summarizes the multiplex options supported by the controller for 16 and 32-bit devices respectively. Column addresses through A10 are driven regardless of the multiplexor configuration, although some of the lines will be unused for the smaller page sizes.

Table 19-28. Address Multiplexing by Column/Row Width for 16-Bit Devices

Device Pins	ESDCTL Pins	16-Bit SDR and LPDDR SDRAM Memory Device									
		64 Mbit		128 Mbit		256 Mbit		512 Mbit		1 Gbit	
		8 Col 12 Row		9 Col 12 Row		9 Col 13 Row		10 Col 13 Row		10 Col 14 Row	
		Col	Row	Col	Row	Col	Row	Col	Row	Col	Row
BA1	BA1	A22	A22	A23	A23	A24	A24	A25	A25	A26	A26
BA0	BA0	A21	A21	A22	A22	A23	A23	A24	A24	A25	A25
MA13	MA13	-	-	-	-	-	-	-	-	-	A24
MA12	MA12	-	-	-	-	-	A22	-	A23	-	A23
MA11	MA11	-	A20	-	A21	-	A21	-	A22	-	A22
MA10	MA10	-	A19	-	A20	-	A20	-	A21	-	A21
MA9	MA9	-	A18	-	A19	-	A19	A10	A20	A10	A20
MA8	MA8	-	A17	A9	A18	A9	A18	A9	A19	A9	A19
MA7	MA7	A8	A16	A8	A17	A8	A17	A8	A18	A8	A18
MA6	MA6	A7	A15	A7	A16	A7	A16	A7	A17	A7	A17
MA5	MA5	A6	A14	A6	A15	A6	A15	A6	A16	A6	A16
MA4	MA4	A5	A13	A5	A14	A5	A14	A5	A15	A5	A15
MA3	MA3	A4	A12	A4	A13	A4	A13	A4	A14	A4	A14
MA2	MA2	A3	A11	A3	A12	A3	A12	A3	A13	A3	A13
MA1	MA1	A2	A10	A2	A11	A2	A11	A2	A12	A2	A12
MA0	MA0	A1	A9	A1	A10	A1	A10	A1	A11	A1	A11

Table 19-29. Address Multiplexing by Column/Row Width for 32-Bit Devices

Device Pins	ESDCTL Pins	32-Bit SDR and LPDDR SDRAM Memory Device											
		64 Mbit		128 Mbit		256 Mbit		512 Mbit		1 Gbit		2 Gbit	
		8 Col 11 Row		8 Col 12 Row		8 Col 13 Row		9 Col 13 Row		9 Col 14 Row		10 Col 14 Row	
		Col	Row	Col	Row	Col	Row	Col	Row	Col	Row	Col	Row
BA1	BA1	A22	A22	A23	A23	A24	A24	A25	A25	A26	A26	A27	A27
BA0	BA0	A21	A21	A22	A22	A23	A23	A24	A24	A25	A25	A26	A26
MA13	MA13	-	-	-	-	-	-	-	-	-	A24	-	A25
MA12	MA12	-	-	-	-	-	A22	-	A23	-	A23	-	A24
MA11	MA11	-	-	-	A21	-	A21	-	A22	-	A22	-	A23
MA10	MA10	-	A20	-	A20	-	A20	-	A21	-	A21	-	A22
MA9	MA9	-	A19	-	A19	-	A19	-	A20	-	A20	A11	A21
MA8	MA8	-	A18	-	A18	-	A18	A10	A19	A10	A19	A10	A20
MA7	MA7	A9	A17	A9	A17	A9	A17	A9	A18	A9	A18	A9	A19
MA6	MA6	A8	A16	A8	A16	A8	A16	A8	A17	A8	A17	A8	A18
MA5	MA5	A7	A15	A7	A15	A7	A15	A7	A16	A7	A16	A7	A17
MA4	MA4	A6	A14	A6	A14	A6	A14	A6	A15	A6	A15	A6	A16
MA3	MA3	A5	A13	A5	A13	A5	A13	A5	A14	A5	A14	A5	A15
MA2	MA2	A4	A12	A4	A12	A4	A12	A4	A13	A4	A13	A4	A14
MA1	MA1	A3	A11	A3	A11	A3	A11	A3	A12	A3	A12	A3	A13
MA0	MA0	A2	A10	A2	A10	A2	A10	A2	A11	A2	A11	A2	A12

19.4.2.2 Bank Addresses

Bank address connections are summarized in Table 19-30. Bank addressing utilizes the most-significant addresses to specify the active bank, the actual bits being dependent on the density of the memory system. Page size and density for a number of potential configurations are documented in Table 19-30. For undocumented configurations, the Equation 19-1 and Equation 19-2 can be used to calculate page size and density.

$$\text{Page Size (Bytes)} = 2^{\#\text{ Column Address Bits}} \times (\text{Memory Width in bits} / 8) \qquad \textit{Eqn. 19-1}$$

$$\text{Density (bytes)} = 2^{(\#\text{Column Address Bits} + \#\text{ Row Address Bits})} \times (\text{Memory Width in bits} / 2) \qquad \textit{Eqn. 19-2}$$

Table 19-30. Bank Address Bit Assignment

Density (Bytes)	Page Size (Bytes)	BA1	BA0
8MB	X	A22	A21
16MB		A23	A22
32MB		A24	A23
64MB		A25	A24
128MB		A26	A25
256MB		A27	A26

19.4.3 Multiplexed Address Bus—During "Special" Mode (SMODE 1 or 3)

During "special" mode, for example, precharge mode (SMODE=1) or load mode registers (SMODE=3) there is no address shifting, means that CPU address A0 is mapped on MA0 at all memory width. For example, in order to drive MA10 bit (for the precharge all command) the CPU A10 bit should be set (for both 16 or 32 bit external devices). The same logic is valid for the load mode register command, as can be seen on the initialization routine example on Section 19.5.4.1, "SDRAM Initialization."

NOTE

BYTE accesses are required (during precharge/load mode register modes) since the address can be non-aligned, depends on the load mode register data.

19.4.4 Refresh

Enhanced SDRAM Controller hardware satisfies all SDRAM refresh requirements after an initial configuration by the user software. Zero, 1, 2, 4, 8 or 16 refresh cycles are scheduled at 31.25 us (nominal 32 kHz clock) intervals, providing 0, 2048, 4096, 8192, 16384, or 32768 refresh cycles every 64 ms. The refresh rate is programmed through the REFR field in the ESDCTL0 and ESDCTL1 registers. Each array can have a different rate, allowing a mix of SDRAM/LPDDR devices, or different SDRAMs density. Refresh is disabled by hardware reset.

A refresh request is made pending at each rising edge on the 32 kHz clock. In response to this request, the hardware gains control of the SDRAM as soon as any in-process bus cycle completes. Once it has gained control of the memory, commands are issued to precharge all banks. Following a row precharge delay (t_{RP}), an auto-refresh command is issued. At t_{RC} intervals, additional auto-refresh cycles are issued until the specified number of cycles have been run. Figure 19-35 illustrates two refresh sequence. Burst transfers in progress when the refresh request arrives are allowed to complete prior to the refresh operation. SDRAM bus accesses queued after the refresh request are held off until the refresh completes.

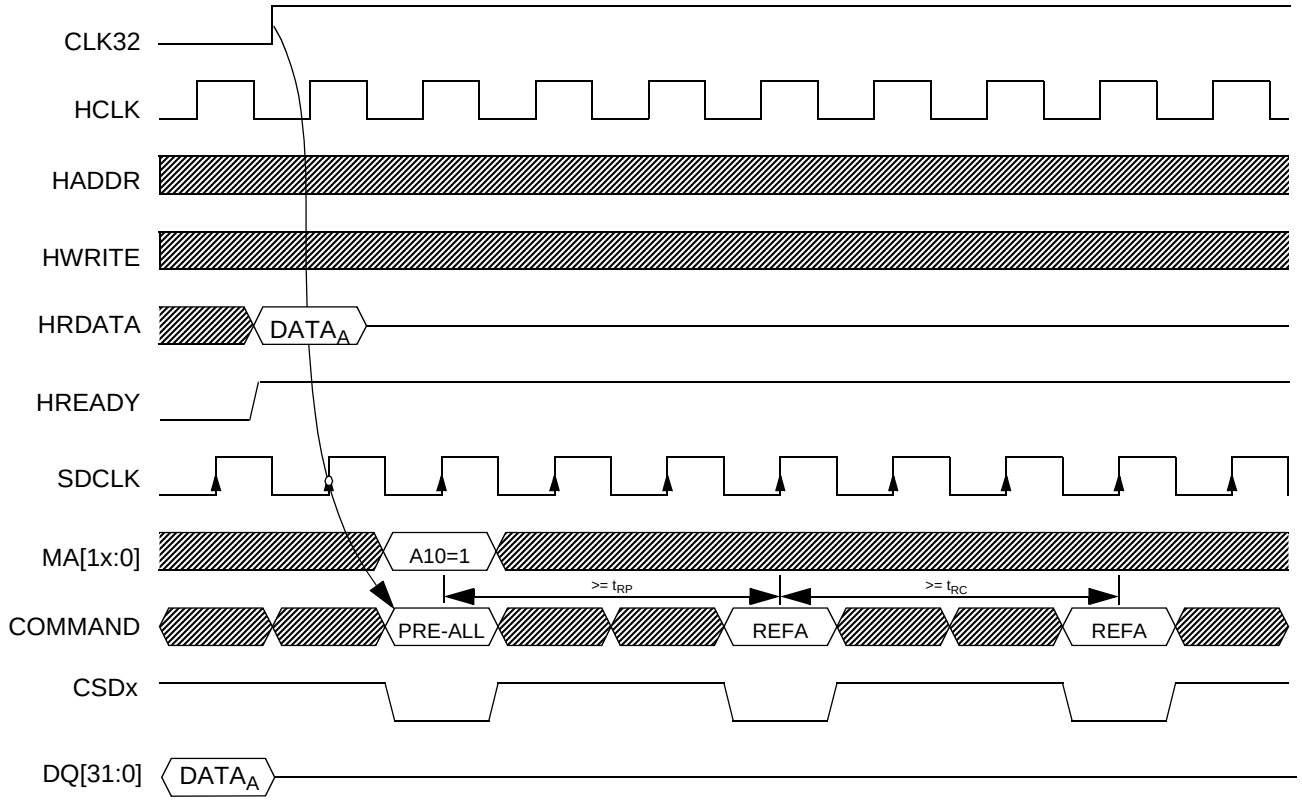


Figure 19-35. Hardware Refresh Timing Diagram

In Figure 19-36, an access is queued just as the refresh begins. This cycle is delayed until the precharge and single refresh (REFR=01) cycles are run. Bus cycles targeted to other memory or peripheral devices are allowed to progress normally while the refresh is in progress. None of the pins shared between the SDRAM and other devices are required for the refresh operation.

NOTE

Since REFRESH commands (requires all banks to be in IDLE state, achieved by PRECHARGE ALL) are issued automatically by the Enhanced SDRAM Controller at each 32 kHz clock, address bits A10 (for both 16 and 32-bit devices) cannot be shared with other peripherals address bus in the system.

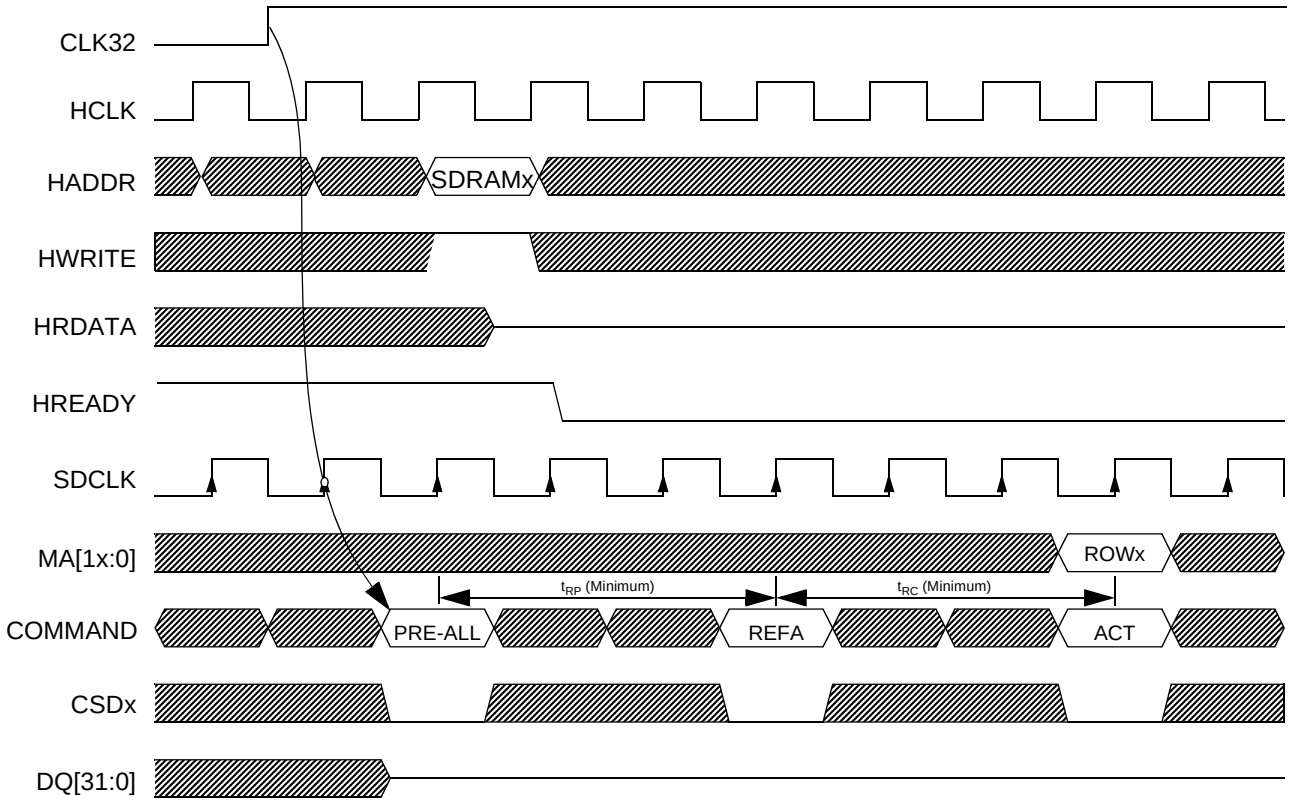


Figure 19-36. Hardware Refresh with Pending Bus Cycle Timing Diagram

19.4.5 Low Power Operating Modes

The following section describes the low power operating modes of the Enhanced SDRAM Controller as a functions of the various memory devices. Table 19-31 lists and summarizes the low power modes supported by the Enhanced SDRAM Controller.

Table 19-31. ESDCTL Low Power Operating Modes

Memory Device	System Operating Mode	Memory Device Low Power Operating Mode	WakeUp Penalty
SDRAM	RUN	POWER DOWN MODE	1 clock cycle
	RUN	PRECHARGE BANK(s)	1 clock cycle
	RUN	MANUAL SELF REFRESH MODE[1] (SMODEx=100)	2 Refresh Period
	STOP	SELF REFRESH MODE	2 Refresh Period

Table 19-31. ESDCTL Low Power Operating Modes (continued)

Memory Device	System Operating Mode	Memory Device Low Power Operating Mode	WakeUp Penalty
LPDDR	RUN	POWER DOWN MODE	tXP
	RUN	PRECHARGE BANK(s)	1 clock cycle
	RUN	MANUAL SELF REFRESH MODE (SMODEx=100)	tXS + 2 Refresh Period
	STOP	SELF REFRESH MODE	tXS + 2 Refresh Period

[1] SDCLK stops, only if both chip selects are in manual self refresh mode.

19.4.5.1 Self Refresh Mode for SDRAM/LPDDR Devices

This operating mode (see Figure 19-37 and Figure 19-38) allows the software/user to control a Self-refresh mode entry of the external SDRAM/LPDDR device if refresh has been enabled, during system RUN mode. When this mode is selected (SMODE=100 in the respective CSD control register) and refresh is enabled the Enhanced SDRAM Controller will complete any active access and a self refresh command to the external device will be issued. No access is allowed to the respective CSD during manual self refresh mode. If refresh has not been enabled, the Enhanced SDRAM Controller places the memory in a low power consumption mode known as power down. The LPACK signal (low power mode acknowledge) will not be asserted if only one CSD enters manual self refresh mode.

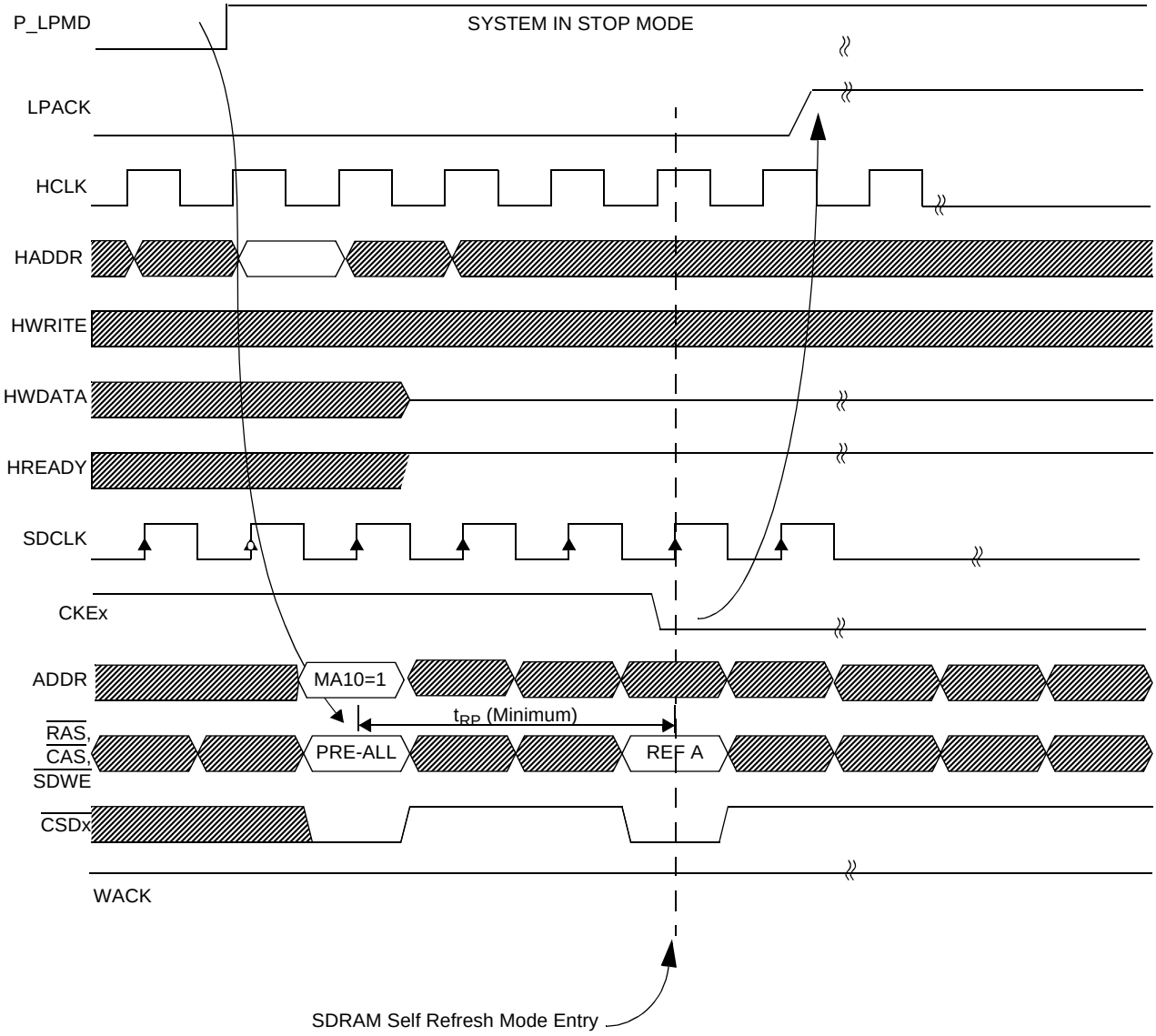


Figure 19-37. SDRAM/LPDDR Enter Self Refresh Mode During System STOP Mode

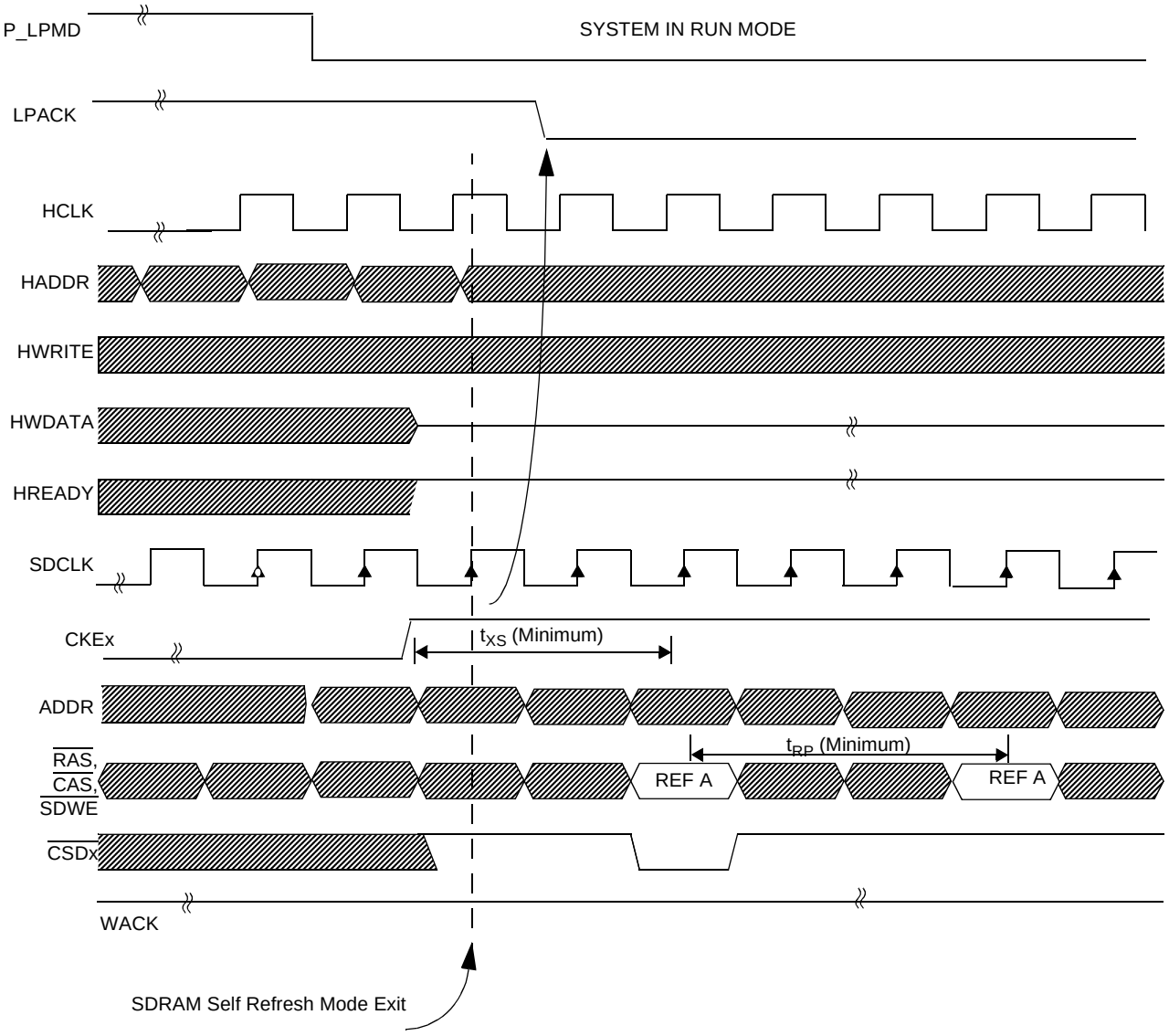


Figure 19-38. SDRAM/LPDDR Exit Self Refresh Mode During System STOP Mode

19.4.5.2 Manual Self Refresh Mode for SDRAM/LPDDR Devices

This operating mode allows the software/user to control a Self refresh mode entry of the external SDRAM/LPDDR device if refresh has been enabled, during system RUN mode. When this mode is selected (SMODE=100 in the respective CSD control register) and refresh is enabled the Enhanced SDRAM Controller will complete any active access and a self refresh command to the external device will be issued. No access is allowed to the respective CSD during manual self refresh mode. If refresh has not been enabled, the Enhanced SDRAM Controller places the memory in a low power consumption mode known as power down. The LPACK signal (low power mode acknowledge) will not be asserted if only one CSD enters manual self refresh mode.

NOTE

A manual precharge all should be initiated by the user before a manual self refresh.

To exit manual self refresh mode, a different operating mode need to be selected by changing SMODE bits in the respective chip select control register. When a different mode is selected, the controller will take the SDRAM device out of self refresh mode and will begin issuing auto refresh cycles (if the refresh has been enabled). illustrates the entry and exit from manual self refresh mode. See Figure 19-39 and Figure 19-40 for timing information.

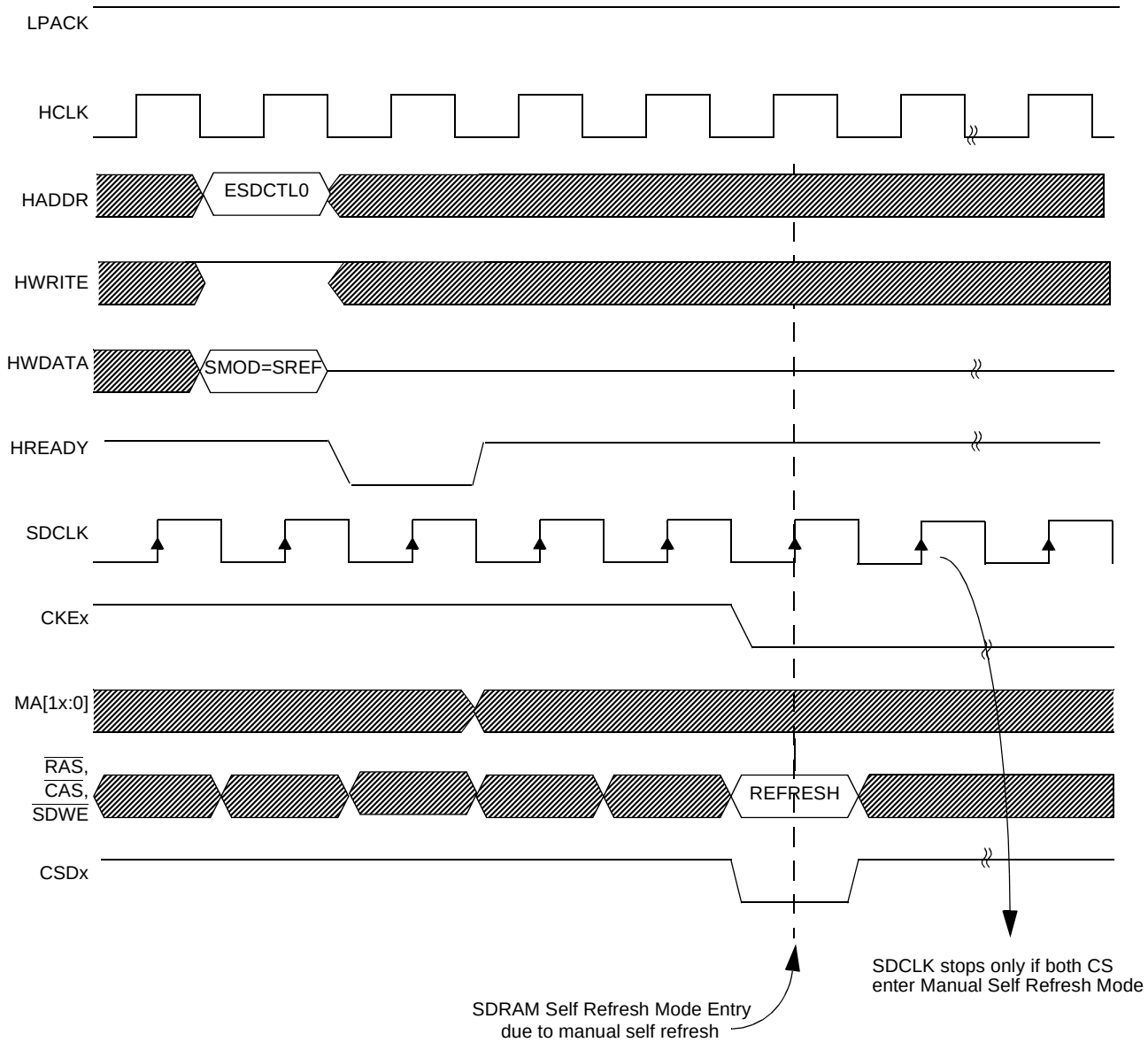


Figure 19-39. Manual Self Refresh Entry Timing Diagram

NOTE

SDCLK stops, only if both chip selects are in manual self refresh. This is in order to allow the usage of one chip select, while the other is in manual self refresh (in case both chip-selects are in use).

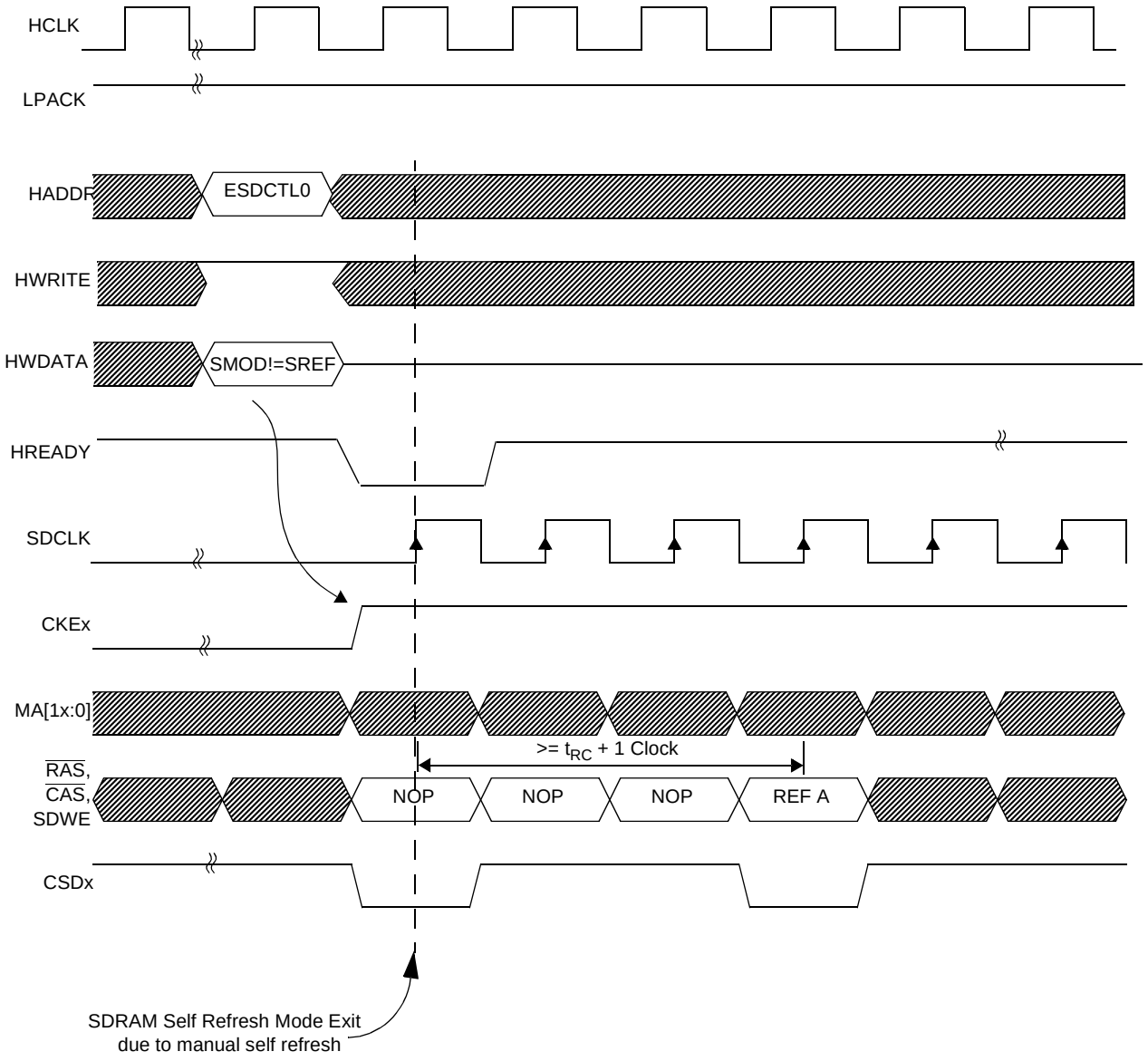


Figure 19-40. Manual Self Refresh Exit Timing Diagram

19.4.5.3 Precharge Power Down Mode

19.4.5.3.1 SDRAM Precharge Power Down Mode

All the low power operating mode described in the above paragraphs will be activated only if the system enters low power operating mode, for example, STOP mode. The Enhanced SDRAM Controller has the

capability to reduce power consumption if the SDRAM/LPDDR memory utilization is low, by setting the SDRAM device in Power Down Mode. This mode is activated through the PWDT bits in the ESDCTL0 and/or ESDCTL1 registers. During this operating mode the ESDCTL automatically issues the REFRESH commands toward the SDRAM/LPDDR memories at the rate defined by the SREFR bits in the ESDCTL0 and/or ESDCTL1 registers.

Programming PWDT[1:0] = 01 causes the Enhanced SDRAM Controller to place the memories in power down mode anytime the controller detects that no banks are active. This mode is useful in applications where a memory array is accessed infrequently and the chances of another access to the same page are minimal.

Reading or writing to memory activates a page within the addressed bank. Reset, software generated precharge, and hardware initiated refresh are three ways to close an active bank. The periodically occurring refresh will be the normal means that invokes the power down mode. At each refresh interval, all banks will be closed by a precharge-all command, followed by the refresh operation. The controller will then issue the power down command to the memories. A few cycle delay is incurred with the first read or write cycle in order to restart the clocks, but only on the first cycle. After that, the clocks will continue to run until the next refresh operation or until any active banks are manually precharged.

Page misses on read and write cycles cause the addressed bank to be closed (precharged) and a new page opened within the bank. This operation does not cause the clocks to stop, nor does manually precharging only a single bank within the memory. All banks within the memory must be inactive before the power down mode is invoked.

Power Down Mode occurs if CKE is registered LOW coincident with a NOP or COMMAND INHIBIT, when no accesses are in progress. Entering power down will deactivate the input and output buffers (excluding CKE) of the device. The Power Down Mode state is exited by registering a NOP or COMMAND INHIBIT and CKE HIGH at the desired clock edge. For SDR SDRAM Figure 19-41 and Figure 19-42 illustrates the power down mode entry and exit respectively. For LPDDR SDRAM Figure 19-43 and Figure 19-44 illustrates the power down mode entry and exit respectively.

NOTE

Since the ESDCTL does not issue AUTO PRECHARGE commands toward the SDRAM, the software will have to issue a PRECHARGE ALL command in order to enter Precharge Low Power Down Mode, to wait for PRECHARGE timer (PRCT) to close/precharge all active banks, or to wait for the next REFRESH cycle in order to enter this low power mode. (During the REFRESH cycle, the ESDCTL automatically issue the PRECHARGE ALL command).

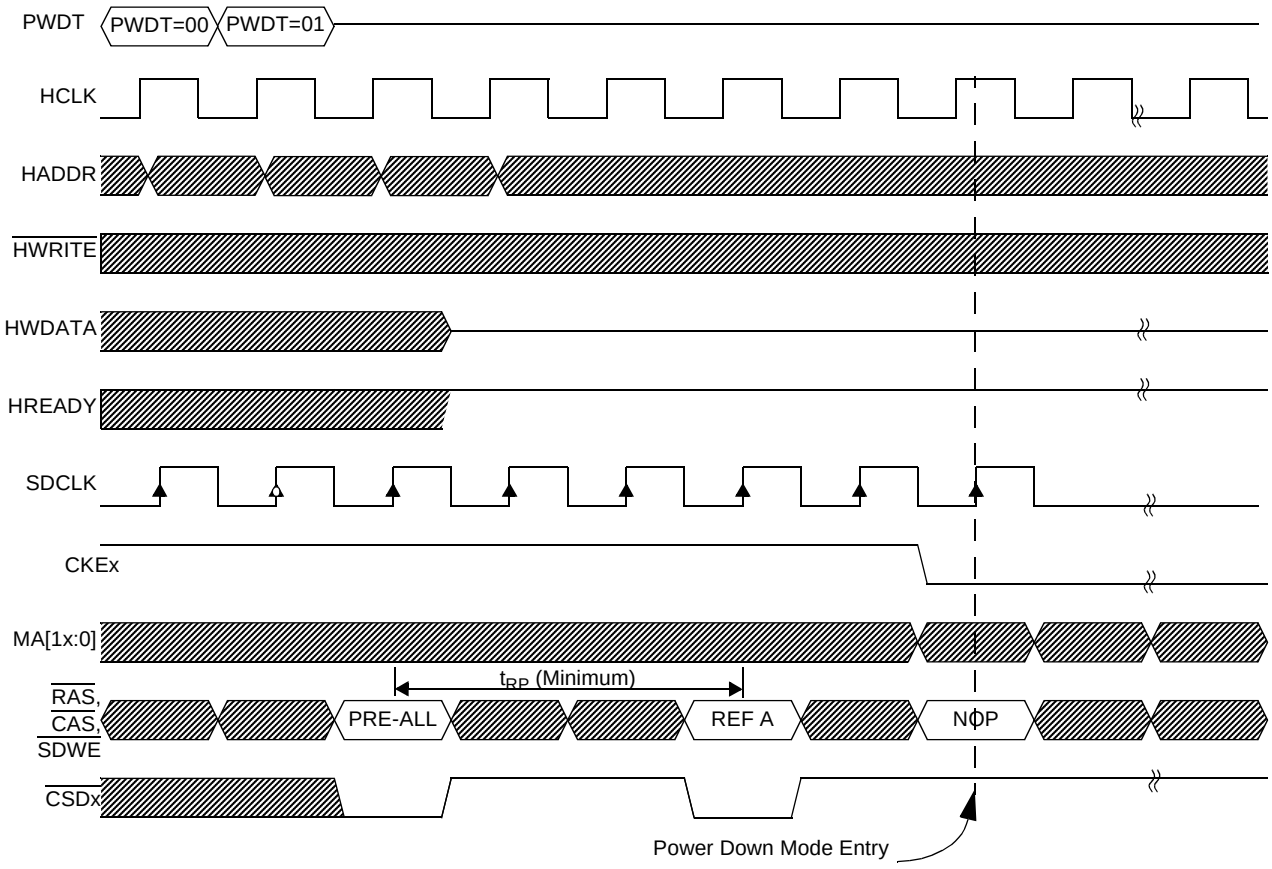


Figure 19-41. SDR SDRAM Precharge Power Down Mode Entry Timing Diagram

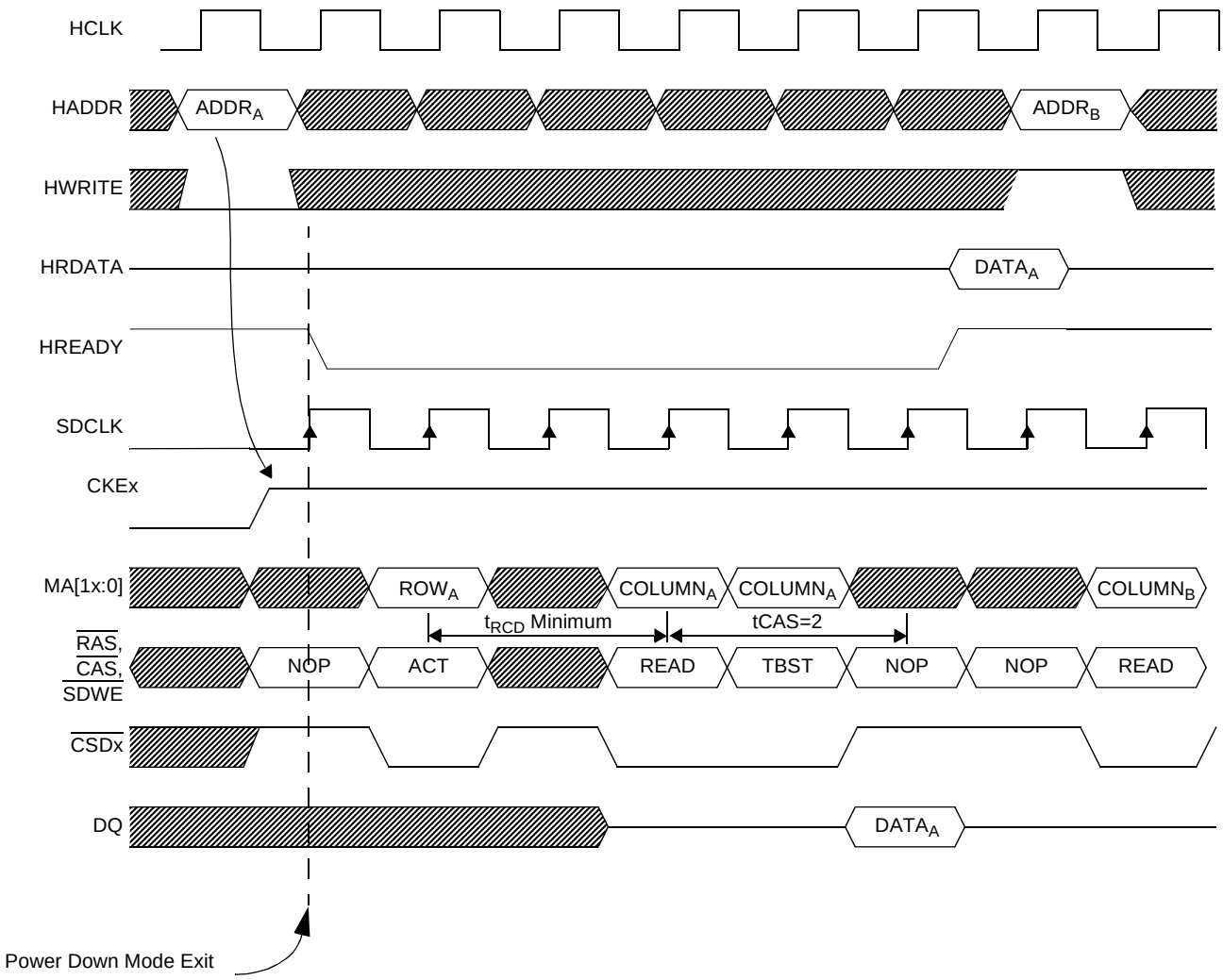


Figure 19-42. SDR SDRAM Precharge Power Down Mode Exit Timing Diagram

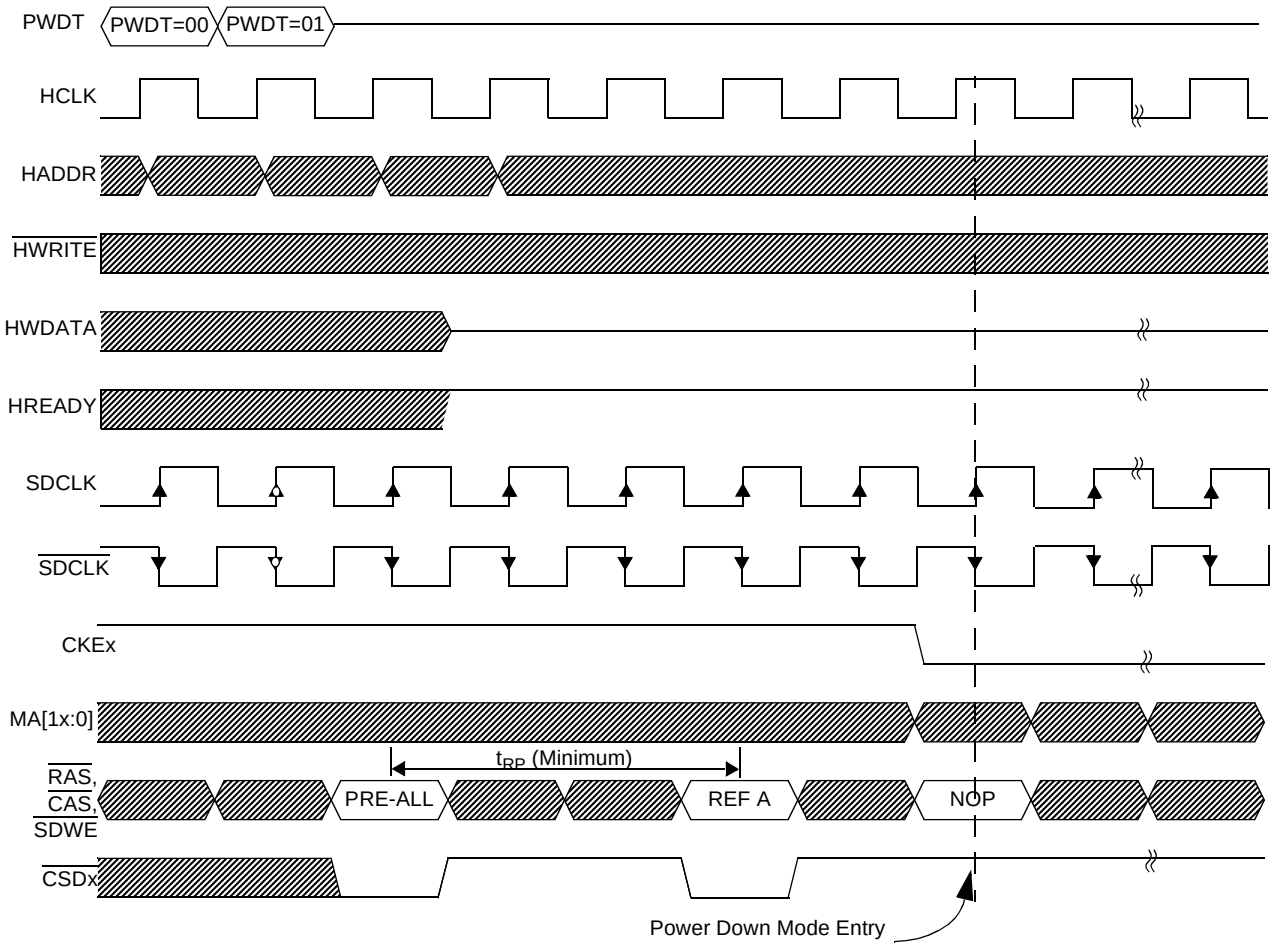


Figure 19-43. Mobile DDR SDRAM Precharge Power Down Mode Entry Timing Diagram

Power Down Mode for several Mobile/Low Power DDRs require the clock CK (and $\overline{CK}$) to continue running. (The PWR CK EN (Power Down Clock Enable for Mobile/Low Power DDR SDRAM) should be set to “1” in order to enable this option)

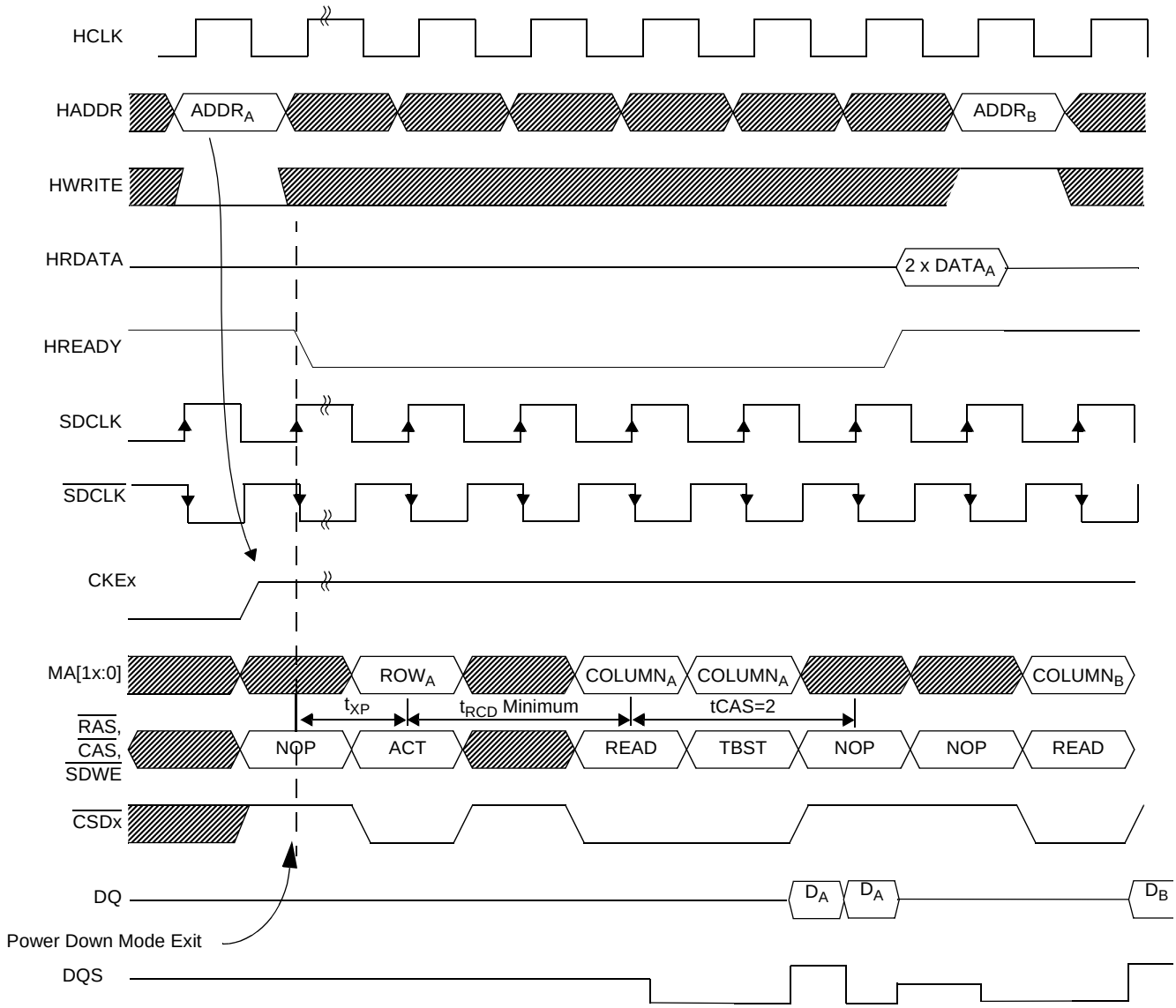


Figure 19-44. Mobile DDR SDRAM Precharge Power Down Mode Exit Timing Diagram

19.4.5.4 Active Power Down Mode

19.4.5.4.1 SDRAM/LPDDR Active Power Down Mode

The second clock suspend mode is selected whenever PWDT[1:0] = 1x. In this mode the SDCLK is stopped after a selectable delay from the last access to the array. Active banks are not closed prior to disabling the SDRAM/LPDDR clock. Either 64 (PWDT[1:0] = 10) or 128 (PWDT[1:0] = 11) cycle delays are possible. SDRAM/LPDDR clocks are counted from the end of the last read or write access. Subsequent read or write accesses, and self-refresh modes reset the counter. Auto-refresh cycles do not affect the counter; however, if the counter expires during a refresh operation the clock will be disabled immediately following the refresh.

The distinguishing factor between Precharge Power Down Mode and Active Power Down Mode is whether banks remain active while the clock is stopped. Active Power Down allows banks to remain activated, while Precharge Power down does not. Figure 19-45 and Figure 19-46 illustrates SDR and LPDDR SDRAM Active Power Down Mode entry and exit timing diagram.

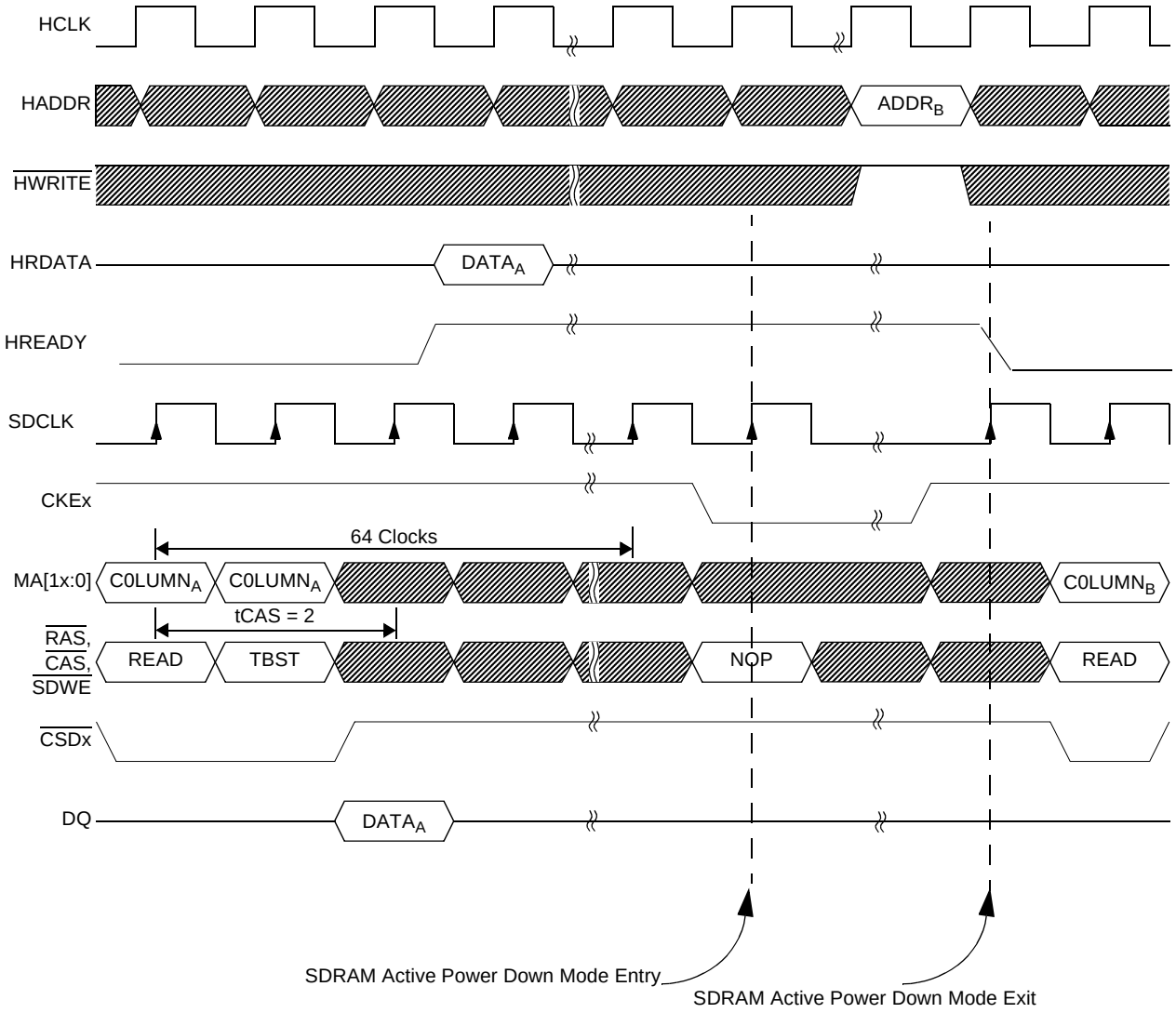


Figure 19-45. SDR SDRAM Active Power Down Mode Timing Diagram

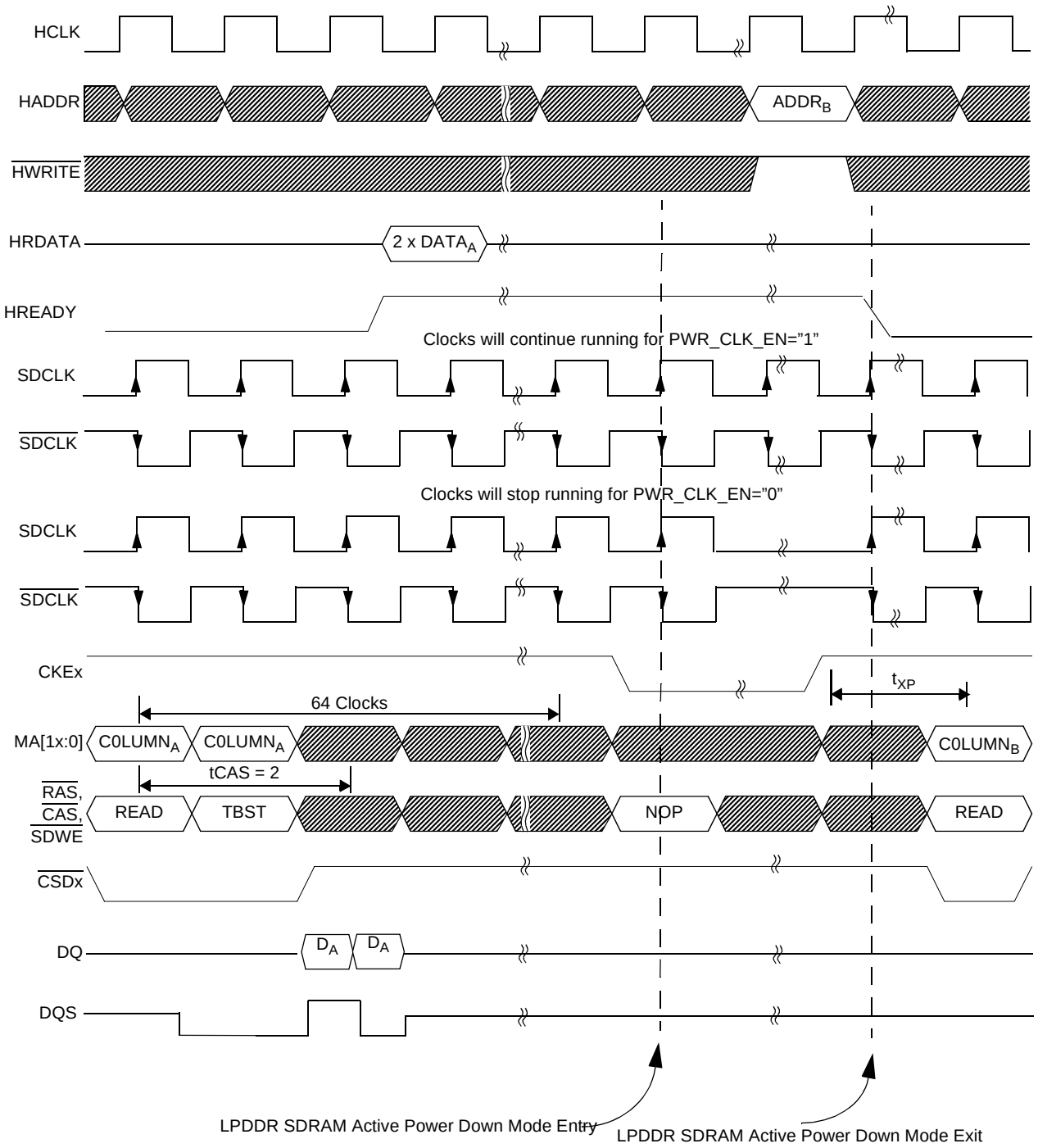


Figure 19-46. Mobile DDR SDRAM Active Power Down Mode Timing Diagram

Power Down Mode for several Mobile DDRs require the clock CK (and $\overline{CK}$) to continue running. (The PWR CK EN (Power Down Clock Enable for Mobile DDR SDRAM) should be set to “1” in order to enable this option)

19.4.5.5 Precharge bank(s)—Low Power Mode

"Closing" (due to precharge command) the last used/open row in any non active bank within a chip select reduces the power consumption of the external memory device. The power saving is device dependent, and one should consult/examine the external memory device specification for more details on power consumption reduction. The precharge bank is activated if PRCT is enabled. A PRECHARGE command is issued after 2xPRCT clocks (HCLK, up to 133 MHz) of no activity (as shown in Table 19-13) to one of the SDRAM/LPDDR banks. The number of cycles before the PRECHARGE command is issued depends on command bus (WE, RAS, CAS and CSD) availability (means there is no active access to other bank) and the memory timing parameters.

19.4.5.6 LPDDR Frequency Change

The following steps need to be performed prior to a frequency change in a LPDDR based system, in order for the DDRC delay line re-calibration.

1. Issue PRECHARGE_ALL command.
2. Enter the external memories in SELF_REFRESH operating mode.
3. Change system frequency.
4. Reset the delay line, by setting the MDDR_DL_RST bit in the ESDCTL miscellaneous register.
5. Wait ~4500 HCLK cycles in order for the delay line to lock on the new frequency.
6. Exit SELF_REFRESH mode.

After the above 6 steps are performed the LPDDR is ready for normal operation at the new frequency.

19.4.6 SDRAM (SDR and LPDDR) Command Encoding

Table 19-32 summarizes the command encoding utilized by this controller. These commands represent a subset of the commands defined by the JEDEC standard.

Table 19-32. SDRAM (SDR and LPDDR) Command Encoding

Function	Symbol	CKE n-1	CKE n	CS	RAS	CAS	WE	A11	A10	BA[1:0]	A[13:0]
Deselect	DSEL	H	X	H	X	X	X	X	X	X	X
No Operation	NOP	H	X	L	H	H	H	X	X	X	X
Read	READ	H	X	L	H	L	H	V	L	V	V
Write	WRIT	H	X	L	H	L	L	V	L	V	V
Bank Activate	ACT	H	X	L	L	H	H	V	V	V	V
Burst Terminate[1]	TBST	H	X	L	H	H	L	X	X	V	X
Precharge Select Bank	PRE	H	X	L	L	H	L	V	L	V	X
Precharge All Banks	PALL	H	X	L	L	H	L	X	H	X	X
Auto-Refresh	CBR	H	X	L	L	L	H	X	X	X	X
Self Refresh Entry	SLFRSH	H	L	L	L	L	H	X	X	X	X

Table 19-32. SDRAM (SDR and LPDDR) Command Encoding (continued)

Function	Symbol	CKE n-1	CKE n	CS	RAS	CAS	WE	A11	A10	BA[1:0]	A[13:0]
Self Refresh Exit	SLFRSHX	L	H	H	X	X	X	X	X	X	X
Power-Down Entry	PWRDN	H	L	H	X	X	X	X	X	X	X
Power-Down Exit	PWRDNX	L	H	H	X	X	X	X	X	X	X
Mode Register Set[2]	MRS	H	X	L	L	L	L	L	L	V	V

[1]For Mobile DDR, applies only to read bursts (with auto precharge disabled).
[2]BA0-BA1 select either the mode register, the extended mode register or the LOW POWER extended mode register.

19.4.6.1 Reset

Assertion of the $\overline{\text{RST}}$ signal initializes the controller into the idle state, and disables the module. While disabled, the controller remains in the idle state with the internal clocks stopped. The reset state of the control register allows for basic read/write operations sufficient to fetch the reset vector and execute the initialization code. A complete initialization of the controller should be performed as part of the start-up code sequence.

Read/write cycles, refresh and low-power mode requests, and Power Down time-outs will all trigger transitions out of the idle state. As shown in the simplified Enhanced SDRAM Controller state diagram pictured in Figure 19-47, state transitions due to a read or write request depend on the operating mode. Other transitions require the corresponding function to be enabled in the ESDCTL registers. Some state transitions have been removed from Figure 19-47 to minimize complexity and allow an easier understanding of the basic controller operation.

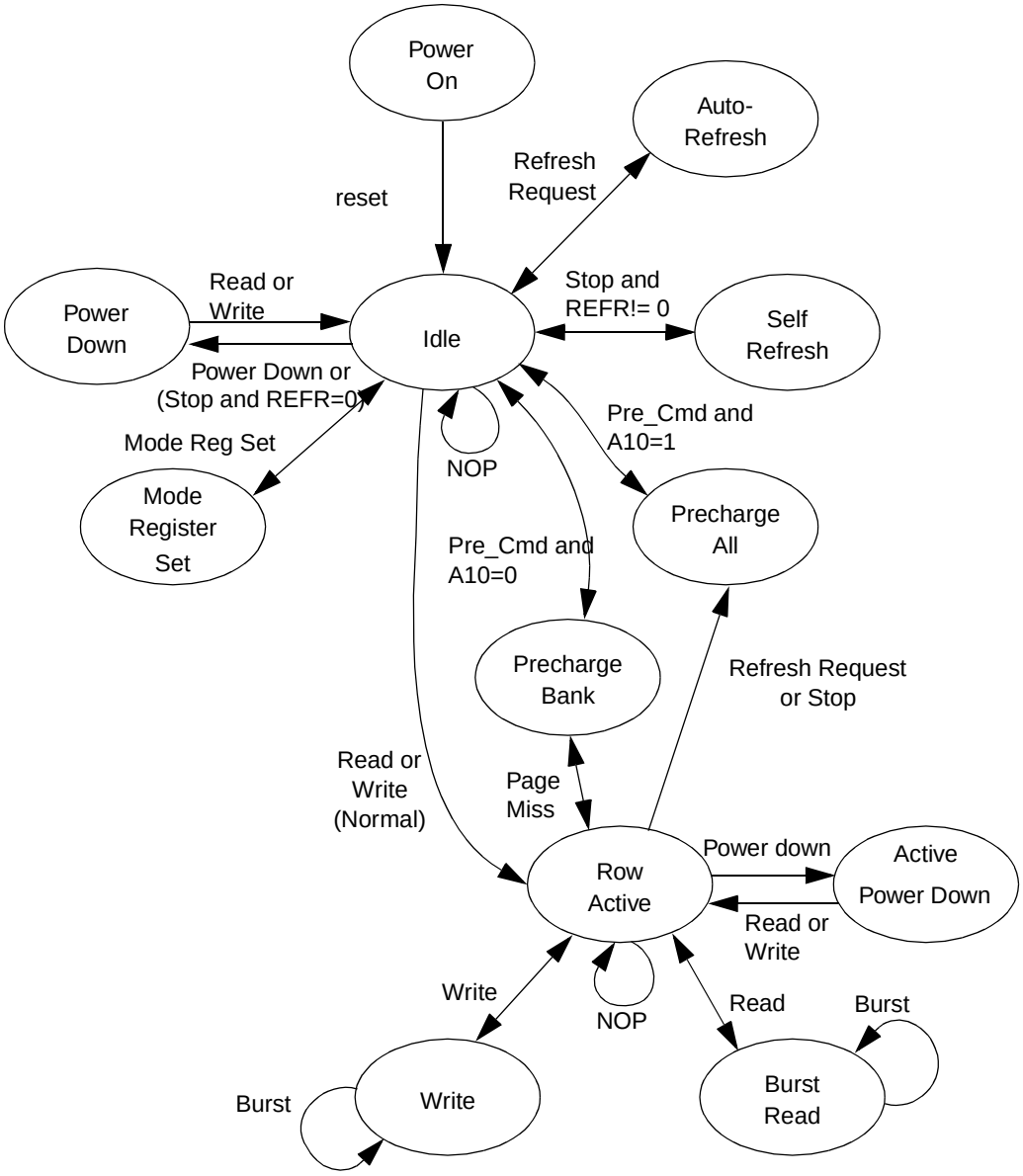


Figure 19-47. Simplified Enhanced SDRAM Controller State Diagram

The following subsections document the operation of each of the operating modes.

19.4.7 Normal READ/WRITE Mode

The Normal Read/Write mode (SMODE = 000) is used for general read and write accesses (AHB light compliant) to the SDRAM/LPDDR. Single and read burst accesses are supported for both SDRAM/LPDDR memories (although bursts requests are limited as shown in Table 19-33). For SDRAM/LPDDR memories single and burst write accesses are supported as well.

Table 19-33. SDRAM/LPDDR Burst Access Support

Internal AHB WORD Access (32bit)		External Memory Device[3]						Description
		16-Bit			32-Bit			
HBURST	TYPE	BL=4	BL=8	BL=FP	BL=4	BL=8	BL=FP[2]	
000	SINGLE[1,2]	No	Yes	Yes	Yes	Yes	Yes	Single transfer
001	INCR	No	Yes	Yes	Yes	Yes	Yes	Incrementing burst
010	WRAP4	No	Yes	Yes	Yes	Yes	Yes	4-beat wrapping burst
011	INCR4	No	Yes	Yes	Yes	Yes	Yes	4-beat incrementing burst
100	WRAP8	No	Yes	Yes	Yes	Yes	Yes	8-beat wrapping burst
101	INCR8	No	Yes	Yes	Yes	Yes	Yes	8-beat incrementing burst
110	WRAP16	No	No	No	No	No	No	16-beat wrapping burst
111	INCR16	No	No	No	No	No	No	16-beat incrementing burst

1 ESDCTL supports only burst of 32-bit (word size). Byte (8 bits) or halfwords (16 bits) accesses are supported only in case of single transfer (SINGLE). The ESDCTL automatically splits (see example in Table 19-34) the AHB bursts as a function of the external memory device, configured via the ESDCTL configuration registers (size and burst length), in such a way that a continuous flow of data is obtained (for both READ or WRITE bursts).

2 BL = Burst Length field in device mode register; FP = Full Page (wrap at external memory device ROW boundary). For both LPDDR 16 and 32-bit devices only BL=8 is supported

Read or write requests to the Enhanced SDRAM Controller initiate a check to see whether the page is already open. This check consists of comparing the request address against the last row accessed within the corresponding bank. If the rows are different, a precharge has occurred since the last access, or there has never been an access to the bank, then the access must follow the “off-page” sequence. If the requested and last row match, the shorter “on-page” access is used. An off-page sequence must first activate the requested row, an operation which is analogous to a conventional DRAM RAS cycle. An activate cycle is the first operation depicted in Figure 19-48. During the activate cycle, the appropriate chip select is driven low, the row addresses are placed on the multiplexed address pins, the non-multiplexed addresses are driven to their respective values, write enable is driven high, CAS is driven high, and RAS is driven low. These latter three pins form the SDRAM command word. The data bus is unused during the activate command.

Once the selected row has been activated, the read operation begins after the row to column delay (tRCD) has been met. This delay is either 2 or 3 clocks, as determined by the tRCD control field. During the read cycle, the chip select is once again asserted, the column addresses are driven onto the multiplexed address bus, the non-multiplexed addresses remain driven to the value presented during the activate cycle, the write enable is driven high (read), RAS is driven high, and CAS is driven low. After the CAS latency has expired, data is transferred across the data bus. CAS latency is programmable via the tCAS control field. As data is being returned across the AHB, transfer acknowledge is asserted back to the CPU indicating that the CPU should latch data. While data is still on the bus, the Enhanced SDRAM Controller must begin monitoring transfer request since the CPU is free to issue the next bus request on the same edge that data is being latched.

Data transfers can be either single operand or a burst (WRAP or INCR) of up to a full page. Burst requests are designated as such by the HBURST bus indicating the length of the access, When HBURST equal to 0 a single access is required, otherwise the access is a burst of HBURST words.

SDRAM memories assume that all transfers are burst transfers unless terminated early. Burst transfers can be terminated by a variety of mechanisms: another read or write cycle, a precharge operation, or through a burst terminate command. Burst terminate commands are the general mechanism used by the ESDCTL for early burst termination, The burst terminate command is subject to the CAS latency and must be pipeline similar to the Read command, as shown in Figure 19-48 to Figure 19-68.

NOTE

The signals displayed are internal signals, for example, interface signals between the M3IF and the ESDCTL. All those figures are not cycle accurate and meant to show mainly the external pins activity. For cycle accurate diagrams refer to Figure 19-70 through Figure 19-72.

SDRAM write cycles are different than read cycles in one important aspect. Whereas read data was delayed by the CAS latency, write data has no delay and is supplied at the same time as the Write command. Figure 19-56 illustrates an off-page write cycle followed by one that hits on-page. Note that the write data is driven during the same clock that the Write command is issued. A Burst Terminate command cancels the burst operation, but again without the CAS latency.

NOTE

ESDCTL handles the HUNALIGN accesses in the following way. The M3IF module converts the original access address to a word align address toward the ESDCTL. The HBSTRB are used by the ESDCTL to drive the correct value on the DQM signals toward the external memory.

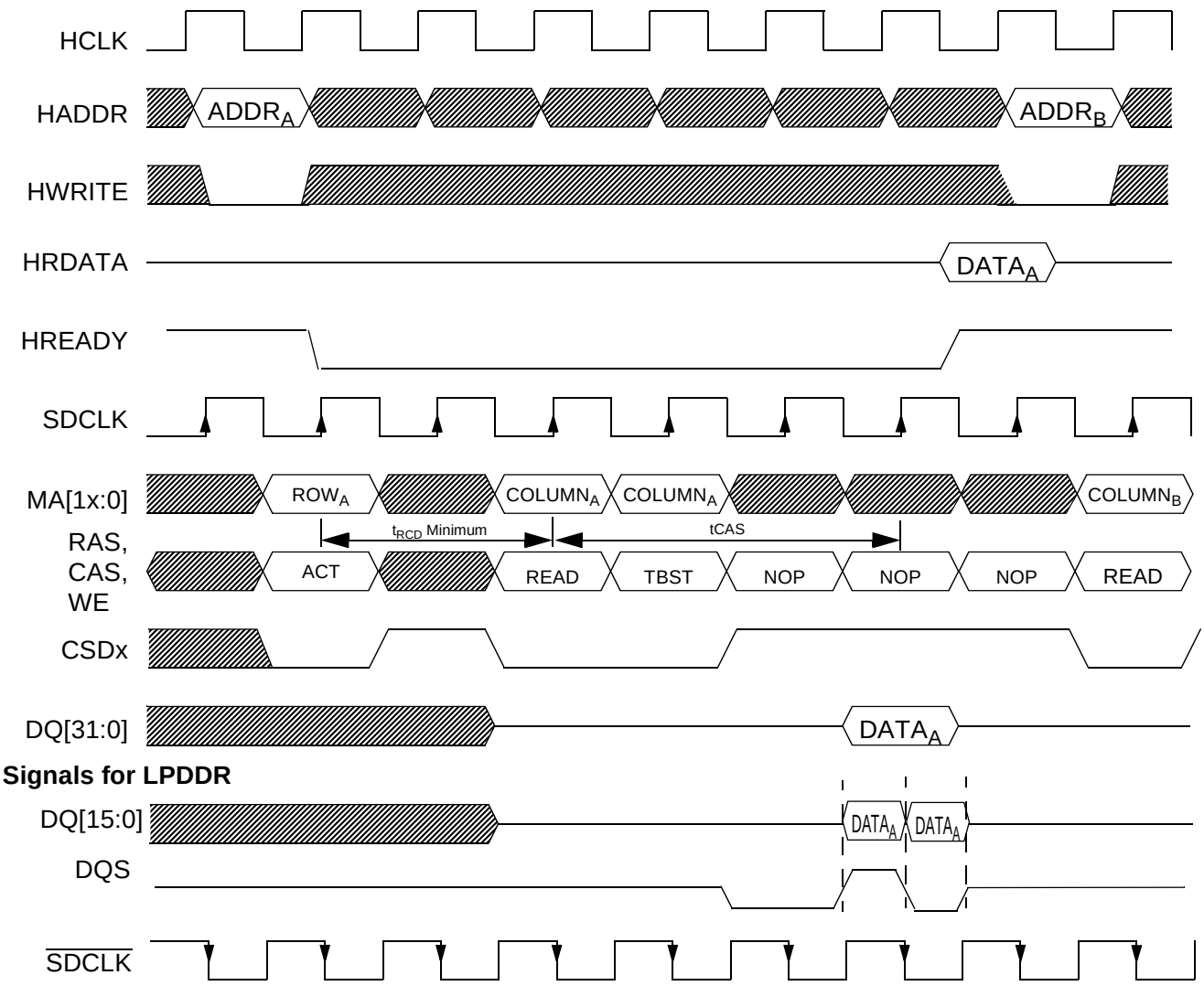


Figure 19-48. SDR and LPDDR Off-Page Single Read Timing Diagram (32-Bit Memory for SDR and 16-Bit for LPDDR)

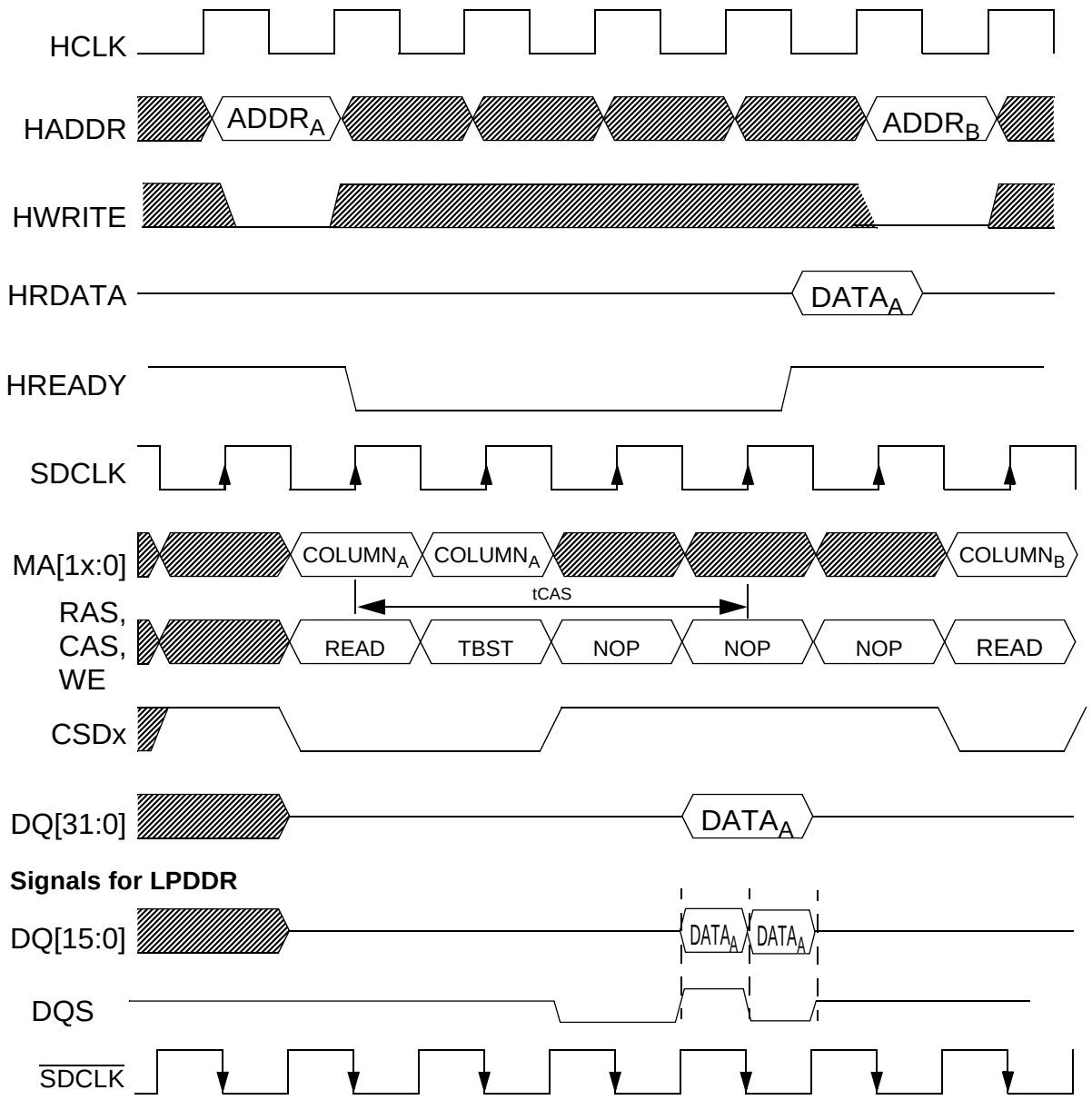


Figure 19-49. SDR and LPDDR On-Page Single Read Timing Diagram (32-Bit Memory for SDR, 16-Bit for LPDDR)

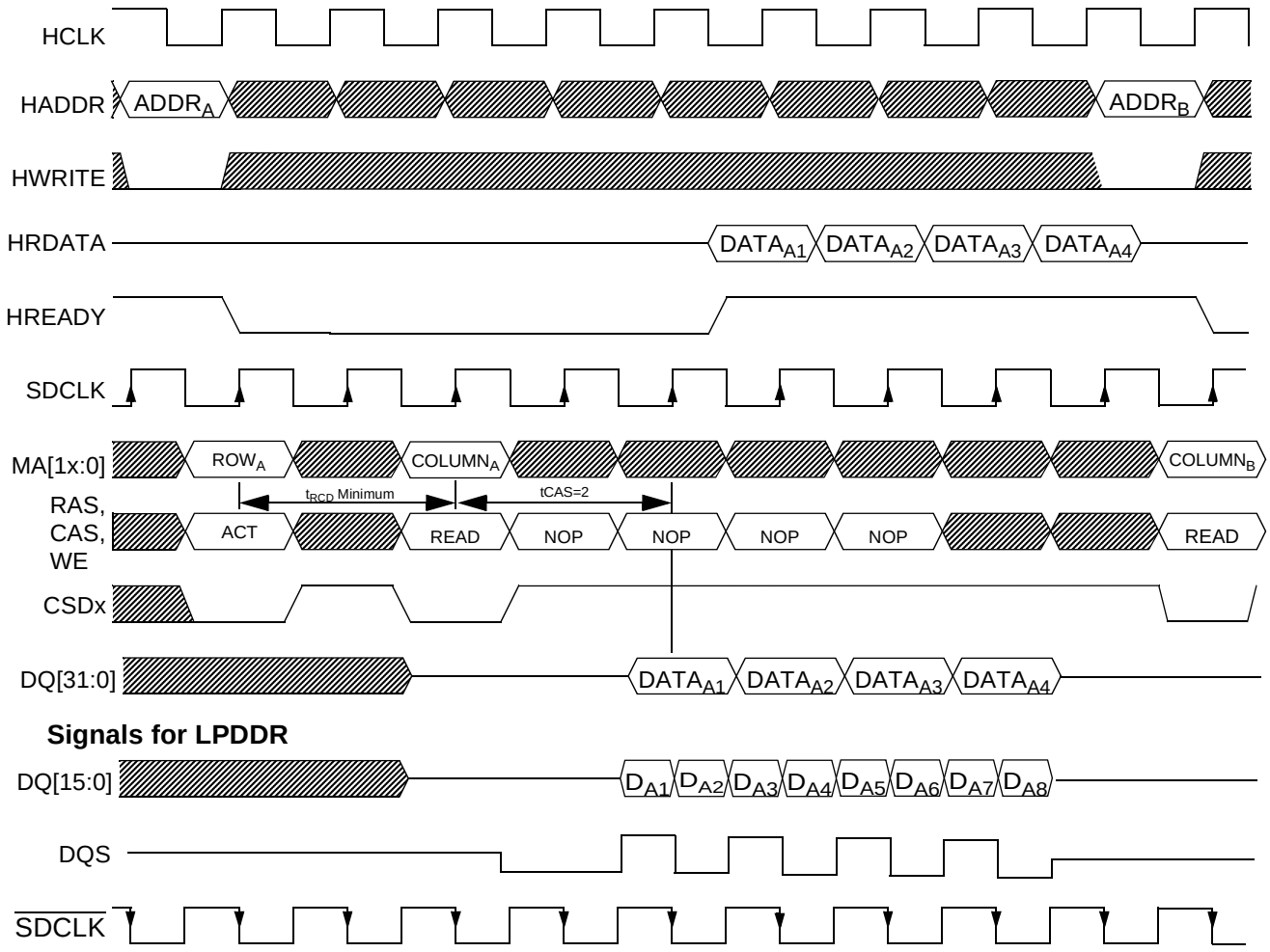


Figure 19-50. SDR and LPDDR Off-Page Burst Read Timing Diagram (32-Bit Memory for SDR or 16-Bit for LPDDR)

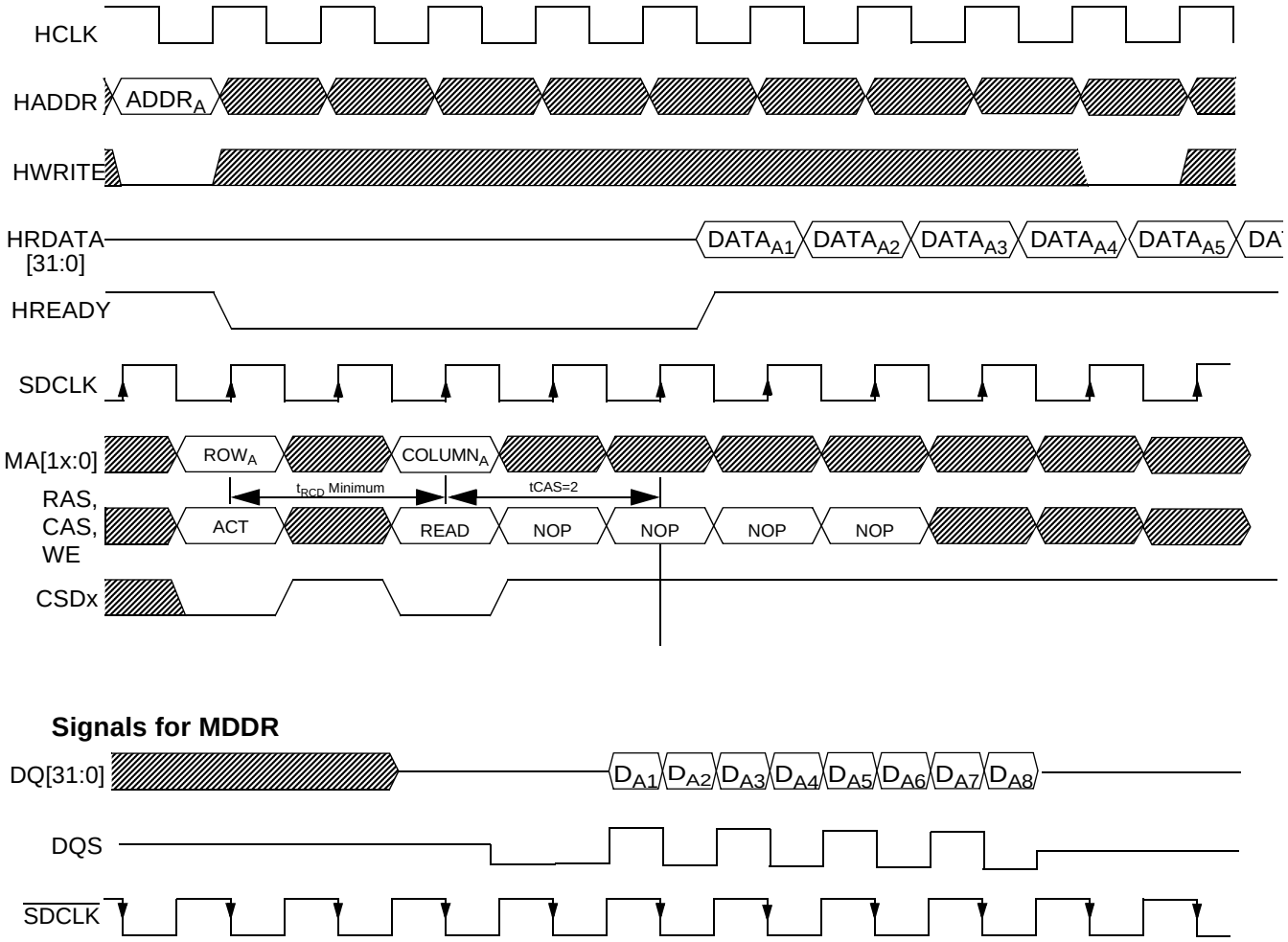


Figure 19-51. AHB 32-Bit Read from a LPDDR: Off-Page Burst Read Timing Diagram (32-Bit)

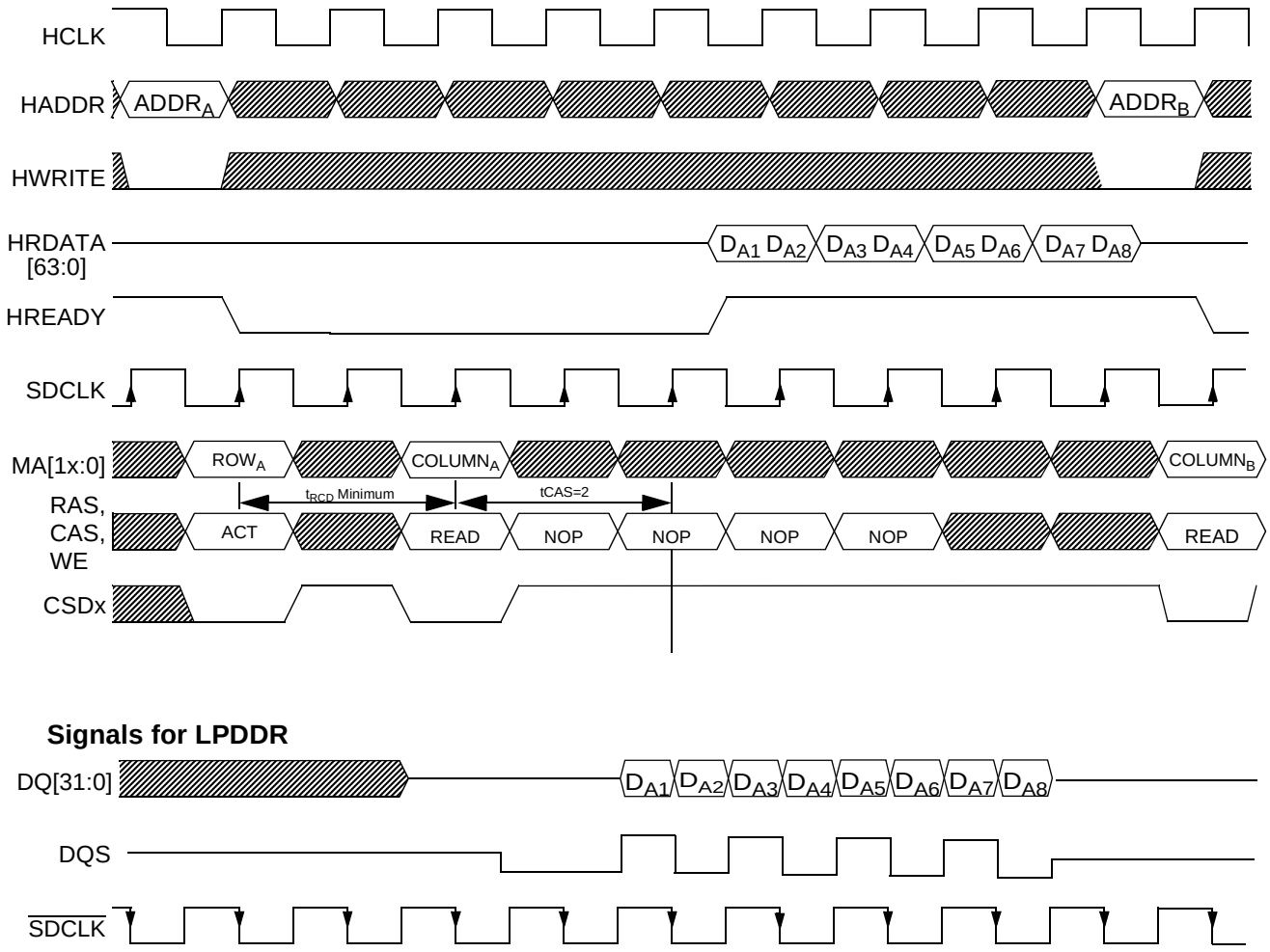


Figure 19-52. AHB 64-Bit Read from a LPDDR: Off-Page Burst Read Timing Diagram (32-Bit)

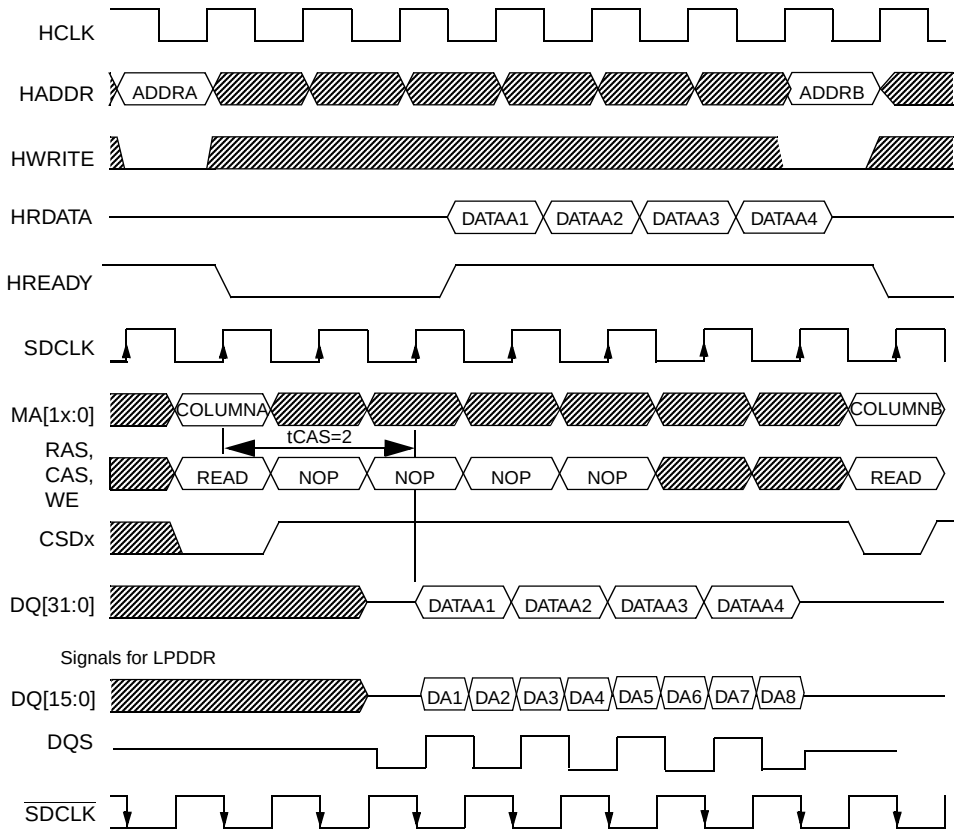


Figure 19-53. SDR and LPDDR On-Page Burst Read Timing Diagram (32-Bit Memory for SDR and 16-Bit for LPDDR)

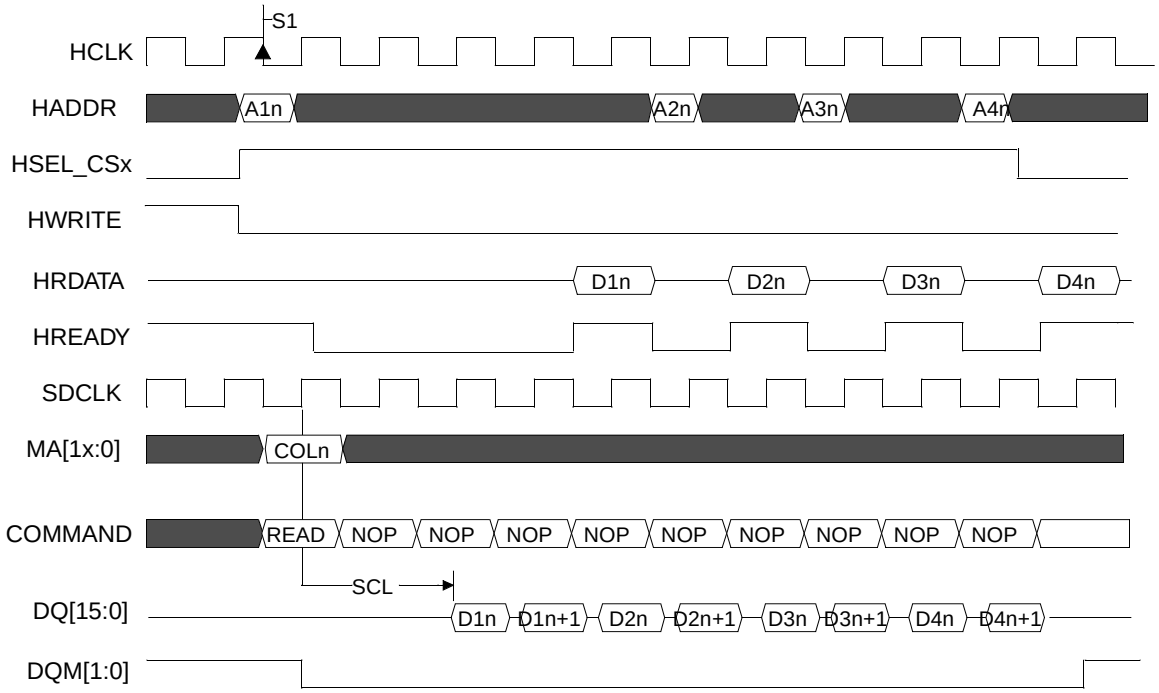


Figure 19-54. On-Page Burst Read Timing Diagram (16-Bit Memory for SDR, LPDDR 8-Bit is Not Supported)

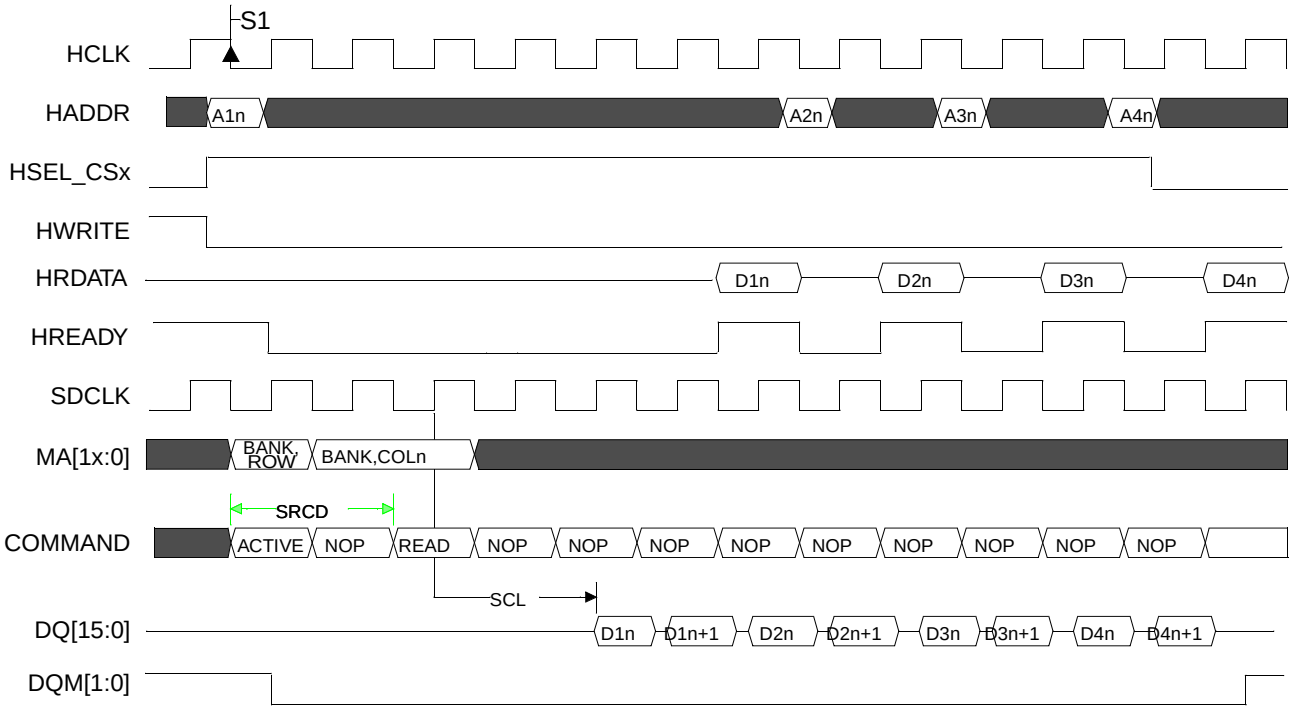


Figure 19-55. Off-Page Burst Read Timing Diagram (16-Bit Memory, LPDDR 8-Bit is Not Supported)

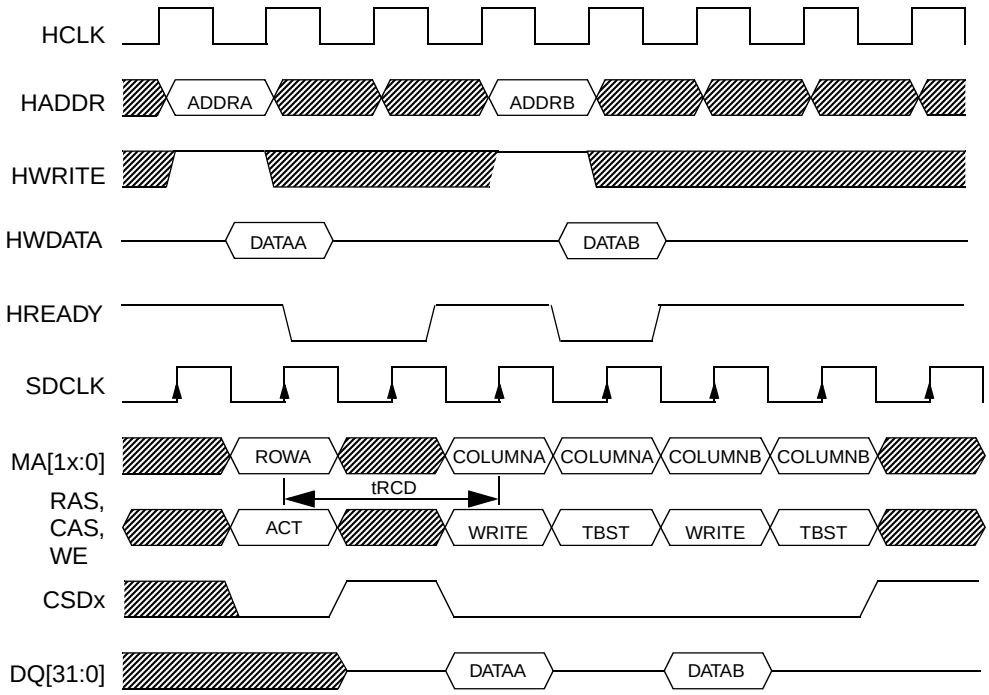
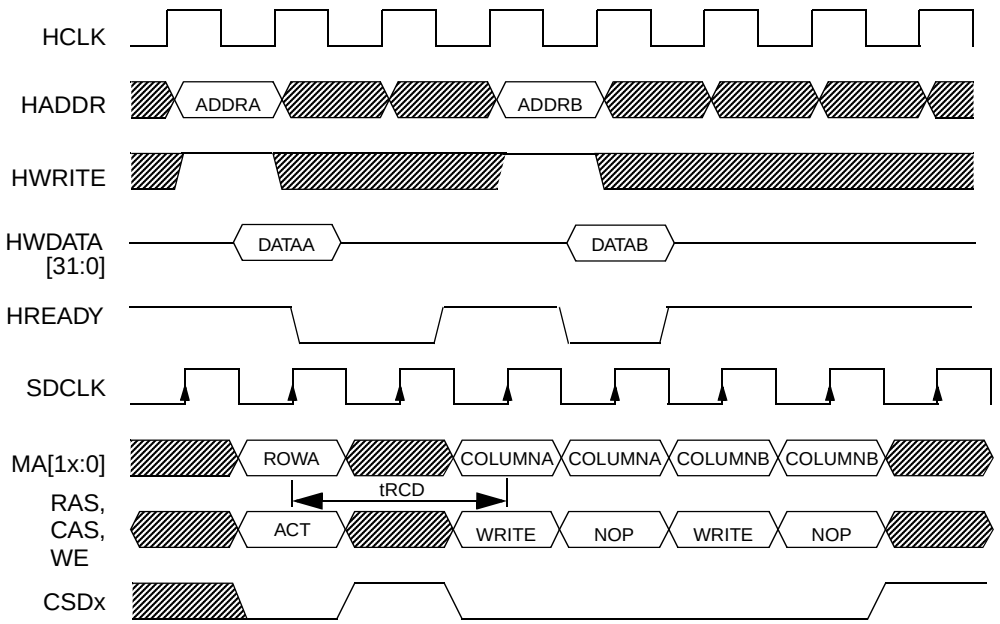


Figure 19-56. SDR Off-Page Write Followed by On-Page Write Timing Diagram



Signals for LPDDR

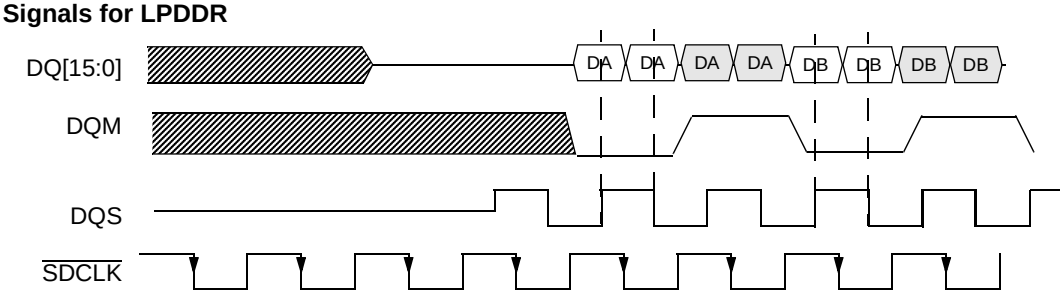


Figure 19-57. LPDDR Off-Page Write Followed by On-Page Write Timing Diagram

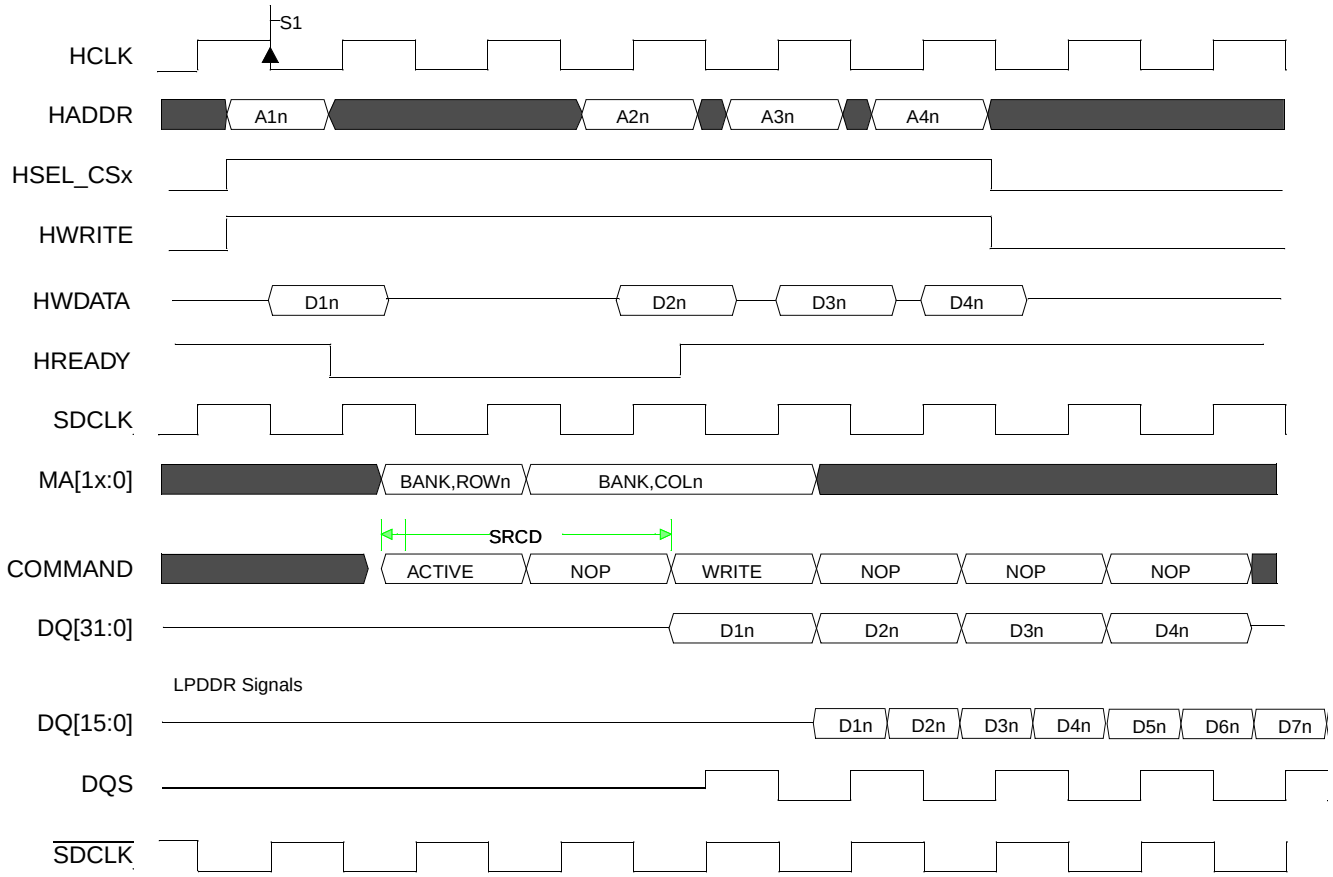


Figure 19-58. Off-Page Burst Write Timing Diagram (32-Bit Memory for SDR and 16-Bit for LPDDR)

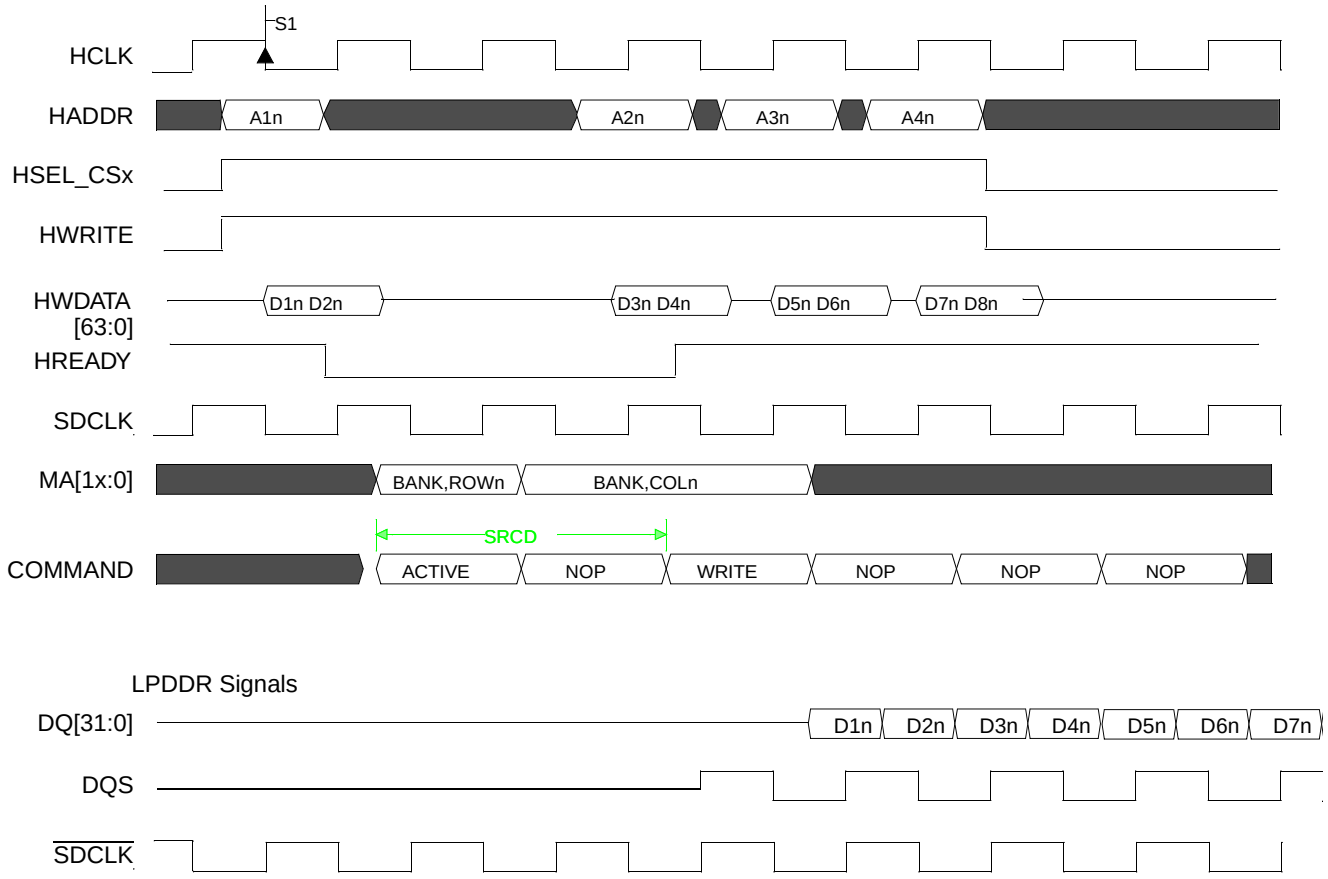


Figure 19-59. AHB 64-Bit Write to LPDDR: Off-Page Burst Write Timing Diagram (32-Bit Memory)

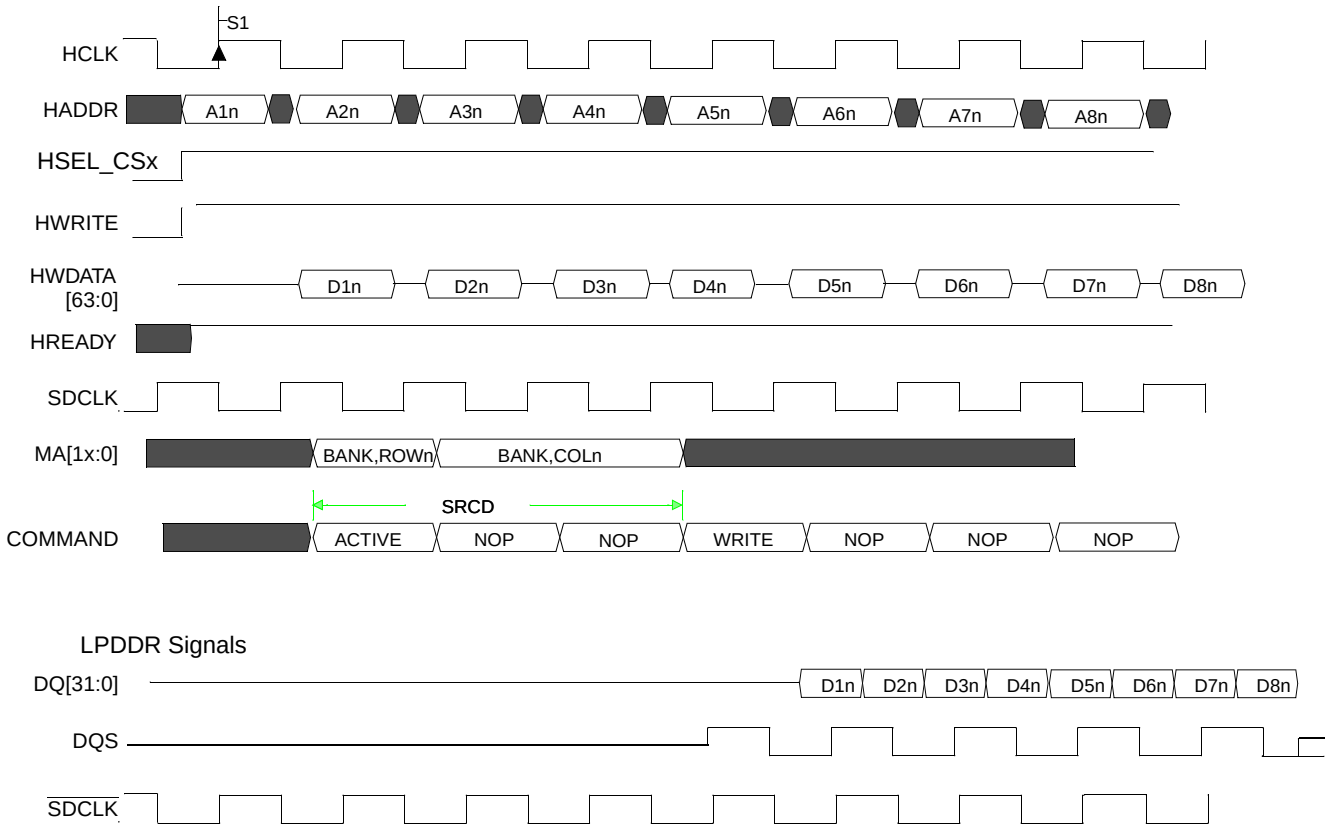


Figure 19-60. AHB 32-Bit Write to LPDDR: Off-Page Burst Write Timing Diagram (32-Bit Memory)

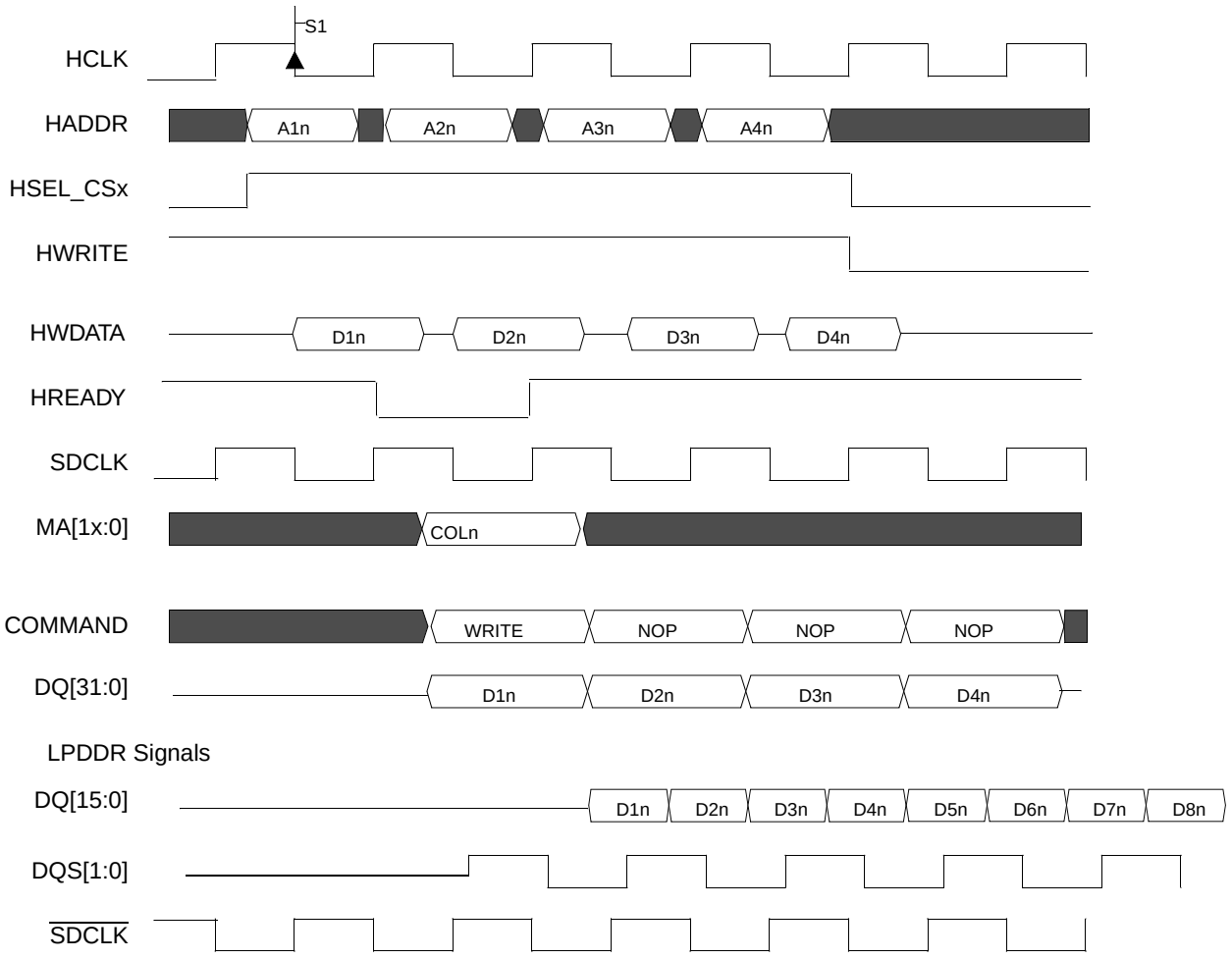


Figure 19-61. On-Page Burst Write Timing Diagram (32-Bit Memory for SDR and 16-Bit for LPDDR)

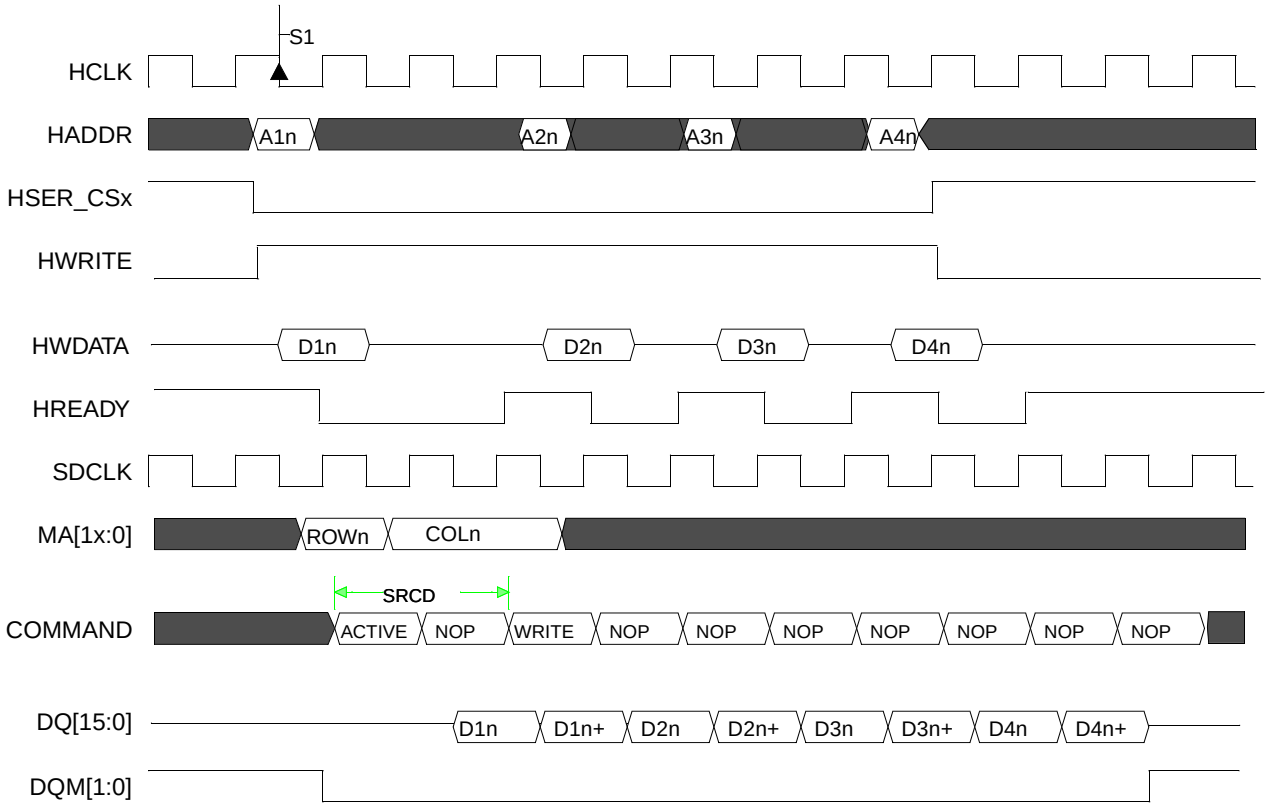


Figure 19-62. Off-Page Burst Write Timing Diagram (SDR 16-bit Memory, LPDDR 8-Bit is Not Supported)

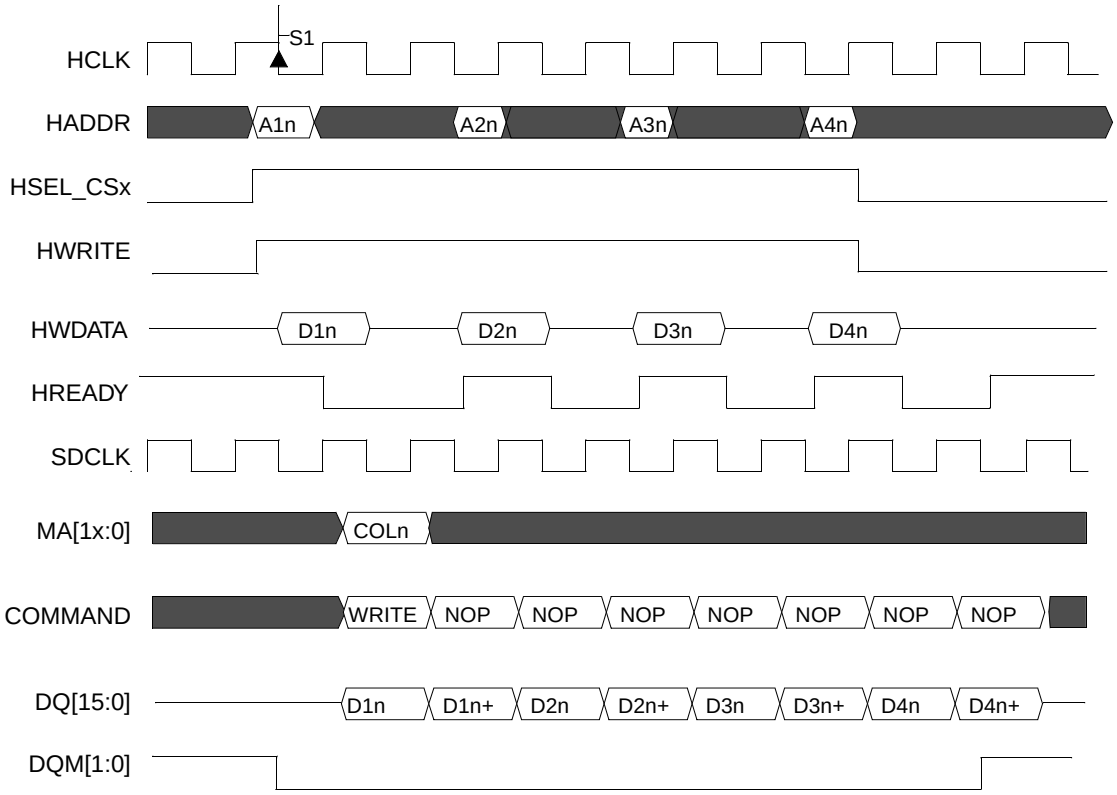


Figure 19-63. On-Page Burst Write Timing Diagram (SDR 16-Bit Memory, LPDDR 8-Bit Memory is Not Supported)

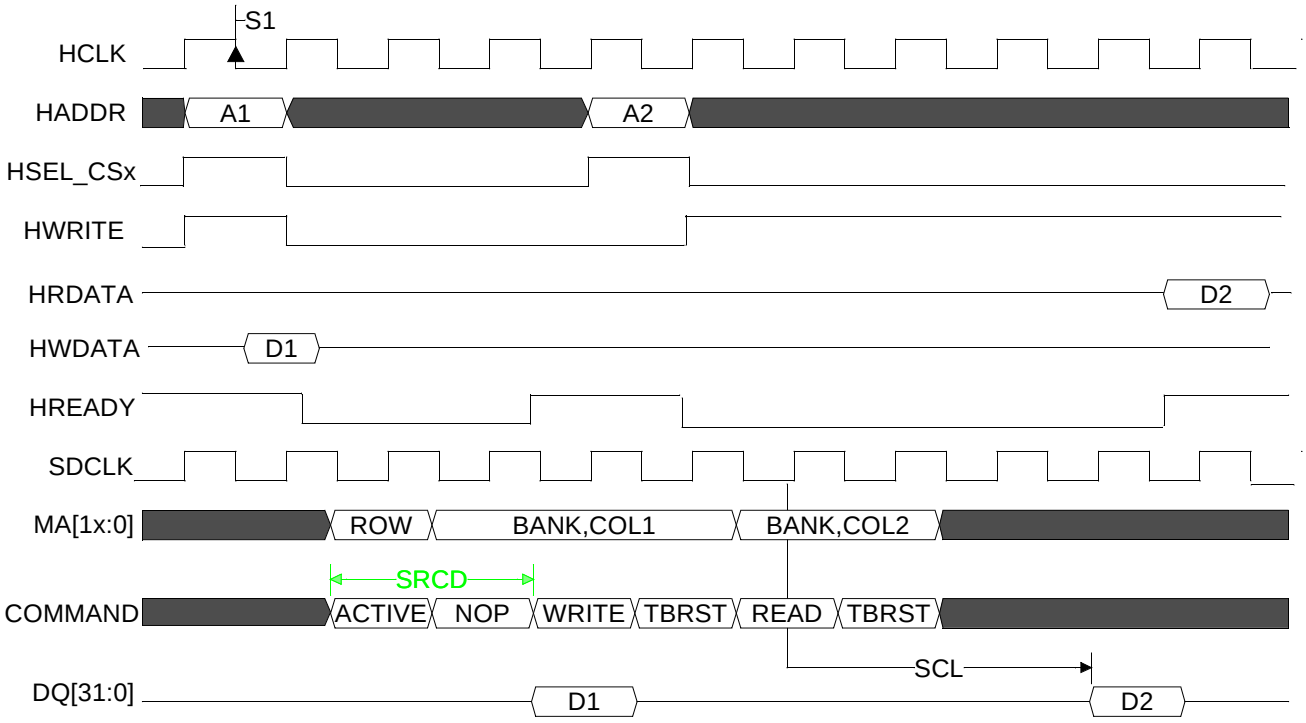


Figure 19-64. SDR Single Write Followed by On-Page Read Timing Diagram

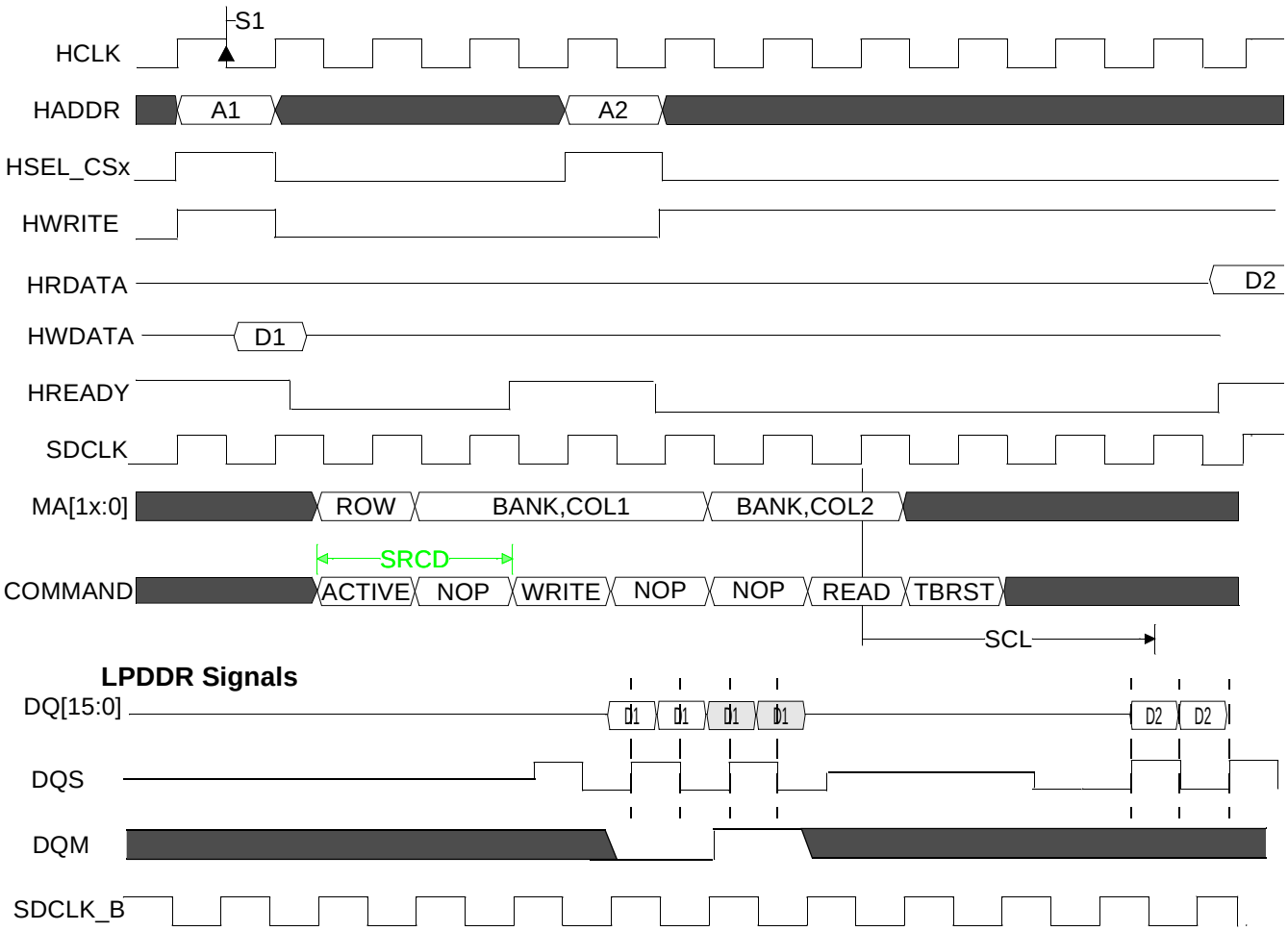


Figure 19-65. LPDDR Single Write Followed by On-Page Read Timing Diagram

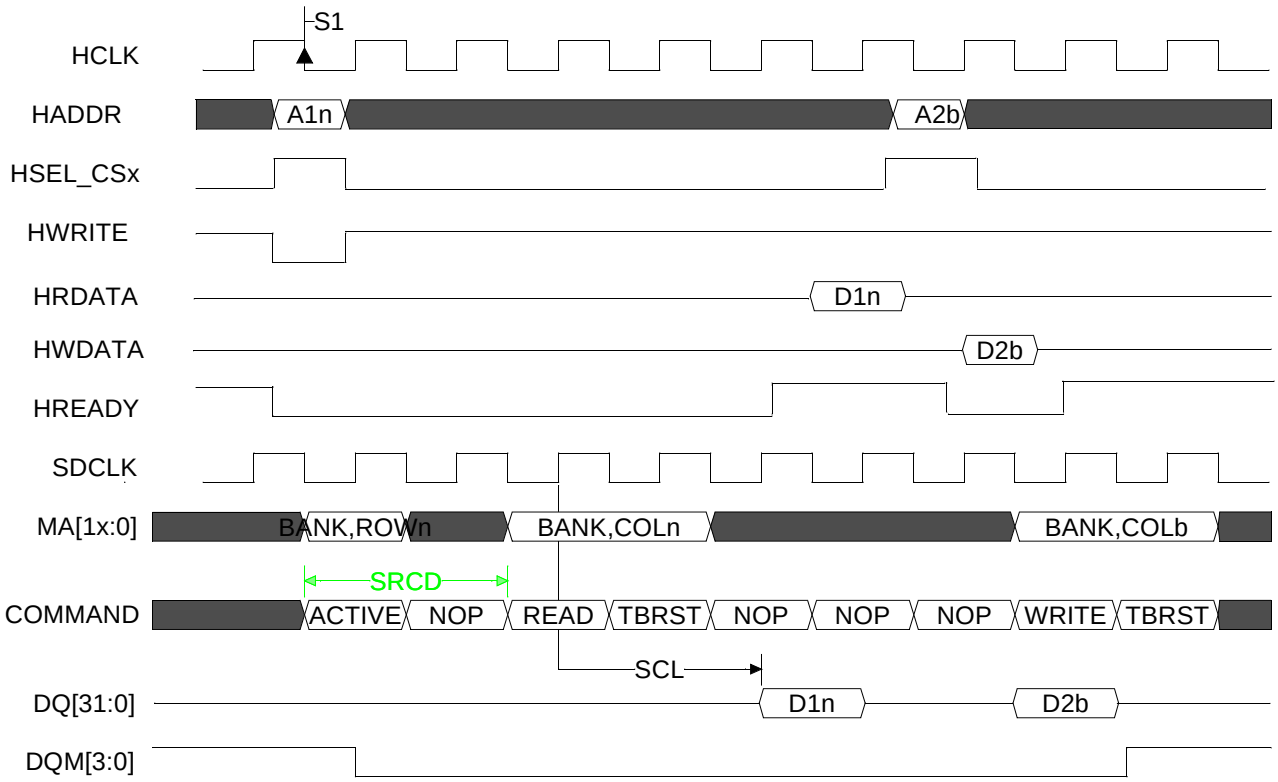


Figure 19-66. SDR Single Read Followed by On-Page Write Timing Diagram

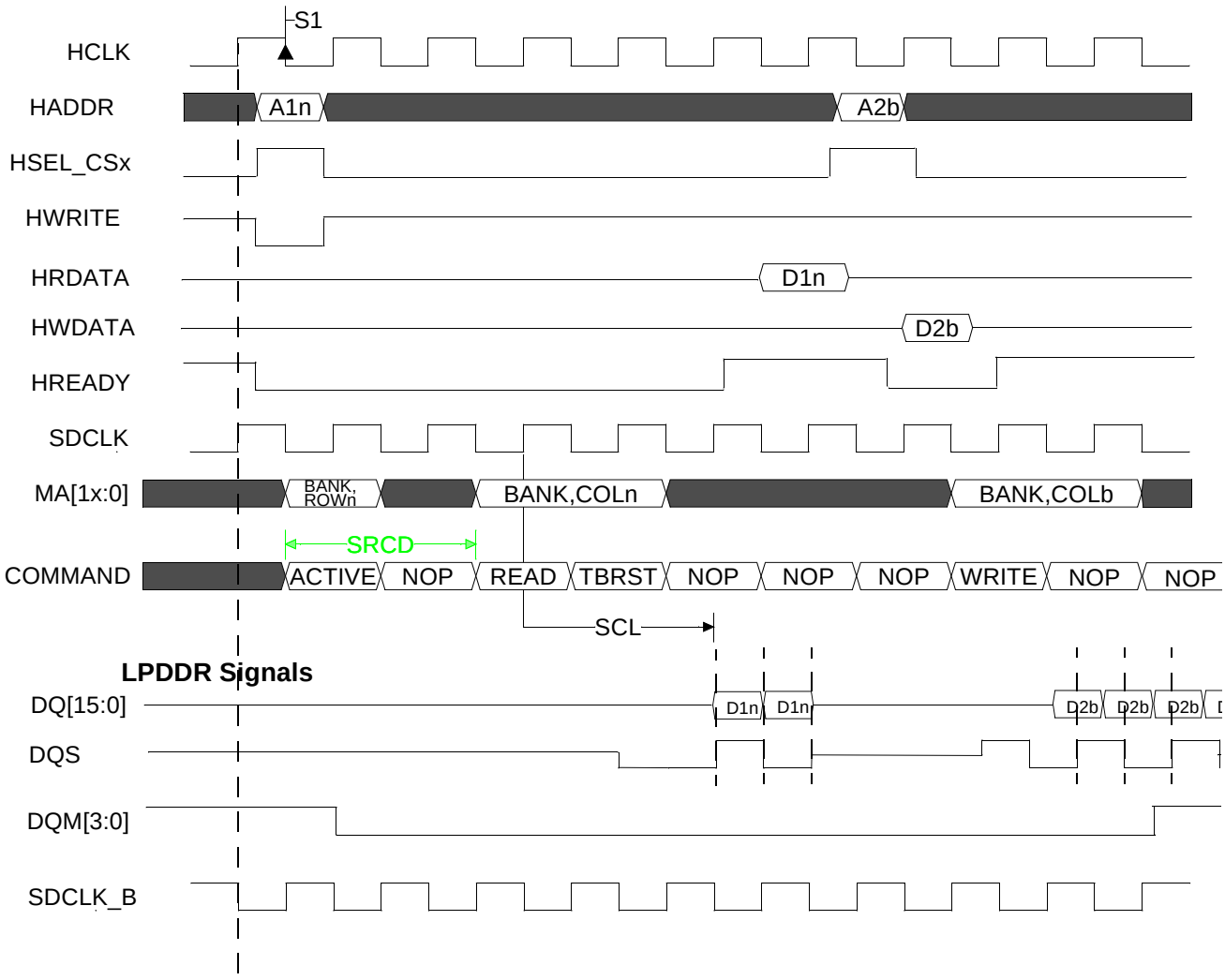


Figure 19-67. LPDDR Single Read Followed by On-Page Write Timing Diagram

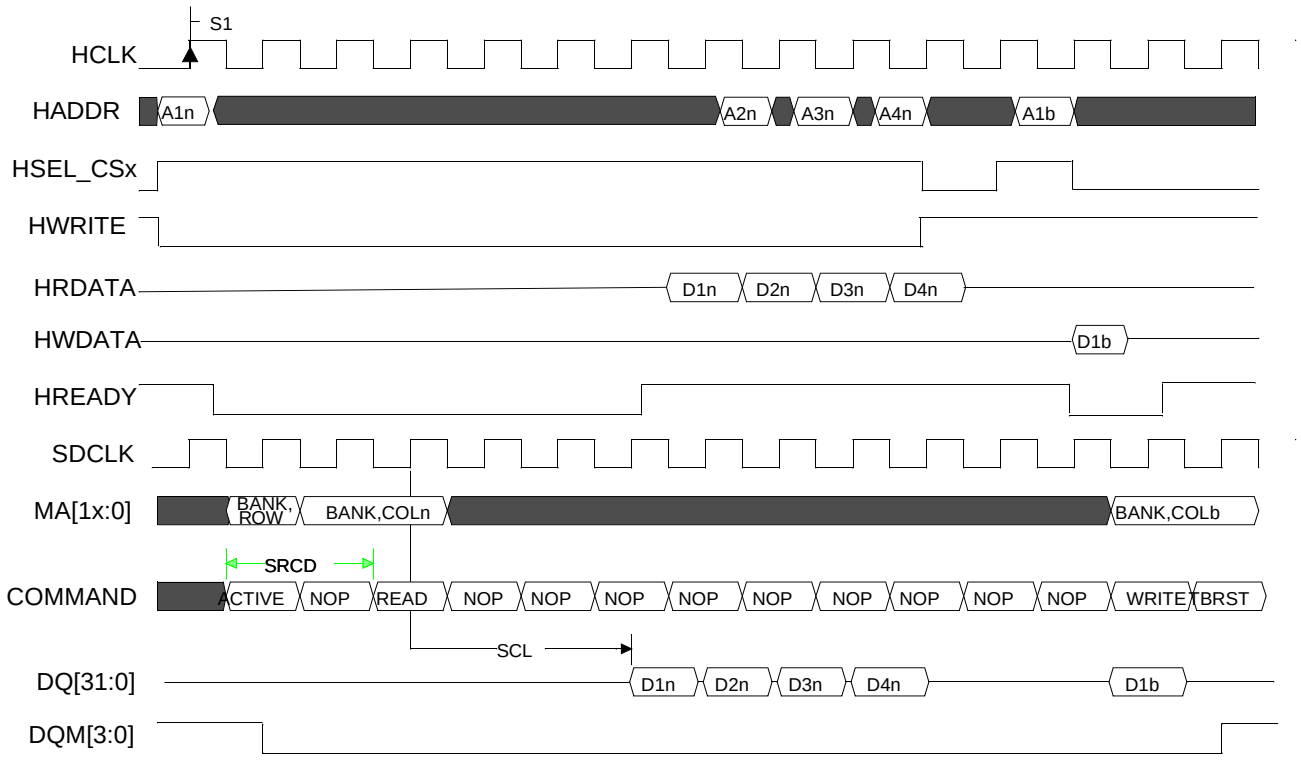


Figure 19-68. SDR Burst Read Followed by On-Page Write Timing Diagram

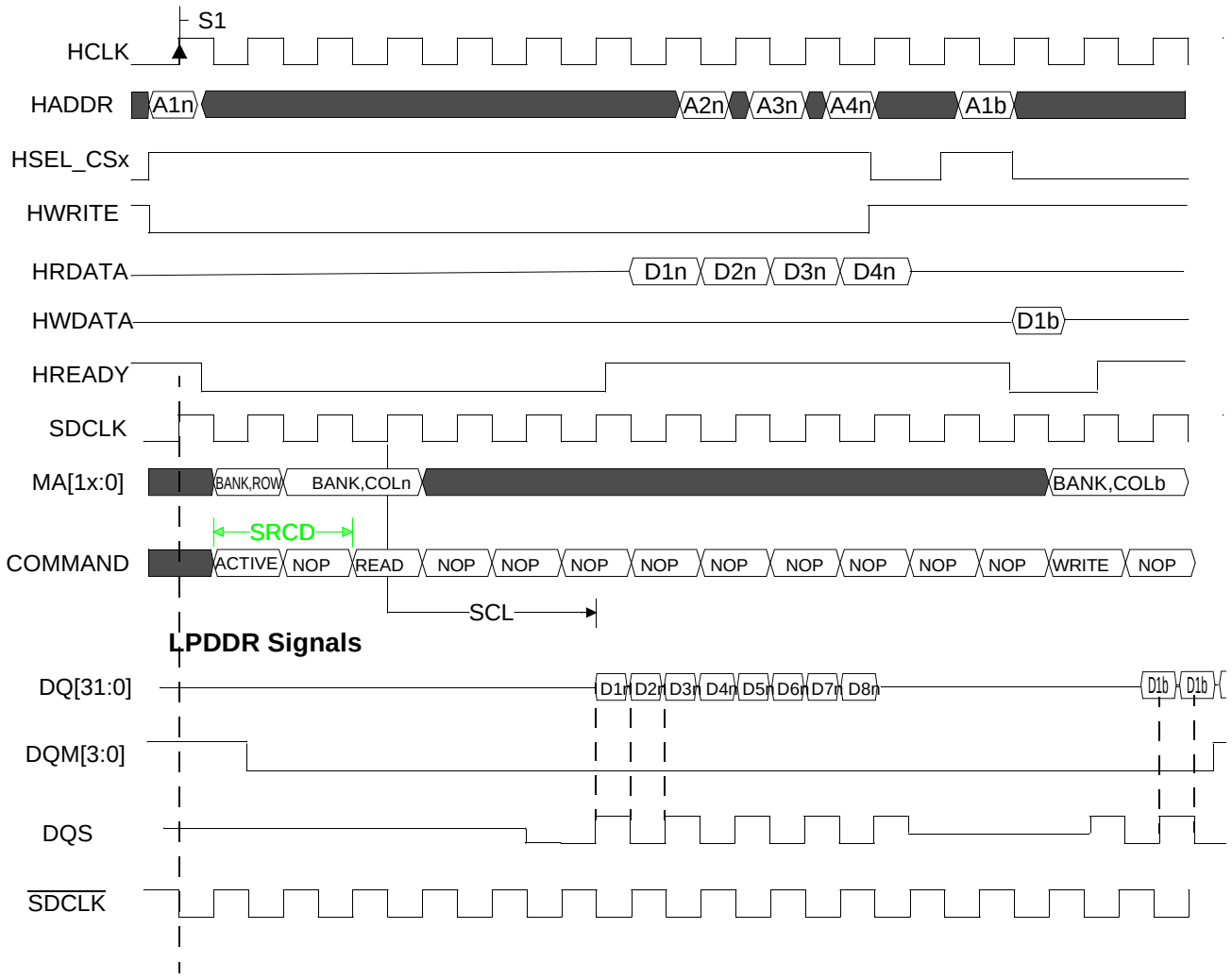


Figure 19-69. LPDDR Burst Read Followed by On-Page Write Timing Diagram

19.4.7.1 SDR Cycle Accurate Enhanced SDRAM Controller Accesses

This section provides cycle accurate timing diagrams for several ESDCTL (AMBA AHB Lite) supported read and write accesses to both 16 and 32-bit data width SDR memory devices from only one master. This diagrams are provided to emphasis ESDCTL performance for single master hit (ACTIVE row) requests. The CAS latency for all diagrams is 2 cycles and burst length is set to 4 words for 16-bit memory and 8 words for 32-bit memory.

19.4.7.1.1 Single Read Word Access to 16-Bit Memory

The markers in Figure 19-70 marks the request access time, for example, the time period between HREADY goes LOW (the ESDCTL starts to execute the request) and HREADY goes HIGH (the ESDCTL request execution is completed).

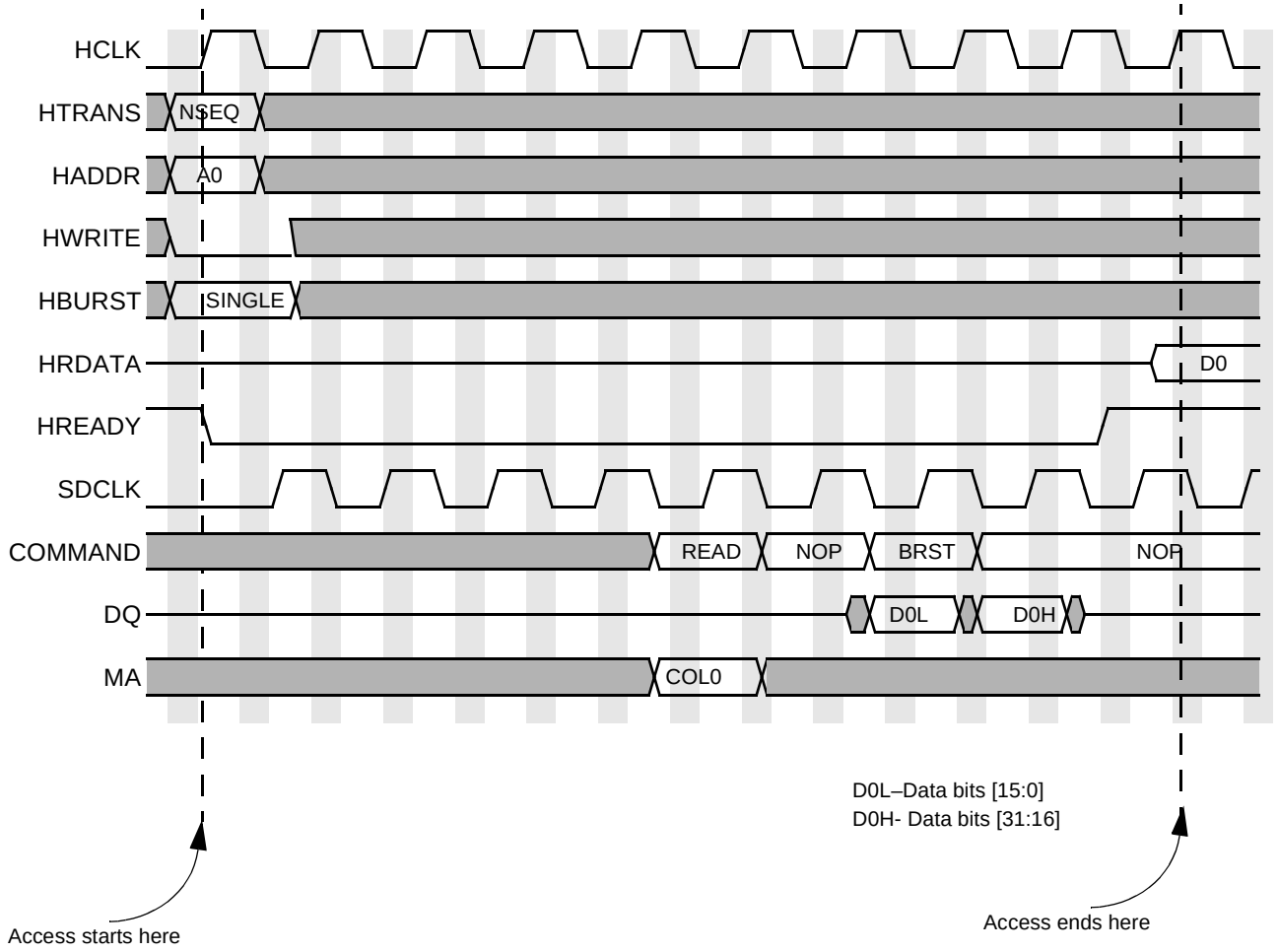


Figure 19-70. Single on Page Read-Word Access to 16-Bit Memory (Cycle Accurate)

19.4.7.1.2 Misaligned INCR4 Burst Read Access to 16-Bit Memory

The markers in Figure 19-71 marks the request access time, for example, the time period between HREADY goes LOW (the ESDCTL starts to execute the request) and HREADY goes HIGH (the ESDCTL request execution is completed).

Two READ commands are issued toward the SDRAM memory, since the misaligned read burst crosses the 16-bit SDRAM (4 words) memory boundary. The read addresses are 0x04, 0x08, 0x0C, and 0x10. Without issuing the second read the last SDRAM data will come from address 0x00. The ESDCTL issue the second READ command in such a way (at a specific timing) so continuous data flow is obtained. There is no timing difference between an aligned and a misaligned access although the second one requires more commands.

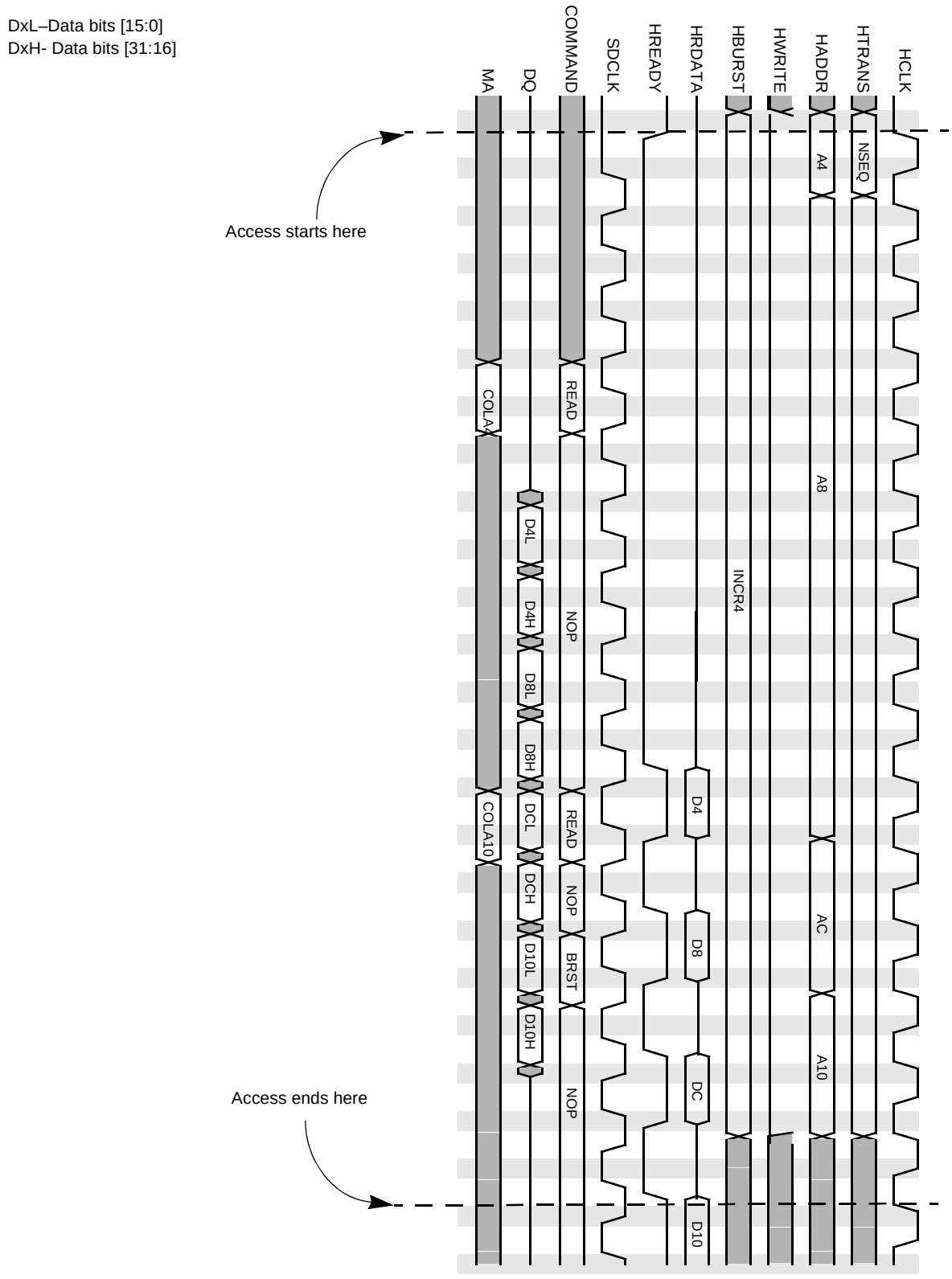


Figure 19-71. Misaligned on Page INCR4 Burst Read Access to 16-Bit Memory

19.4.7.1.3 Misaligned WRAP8 Burst Read Access to 32-Bit Memory

The markers in Figure 19-72 marks the request access time, that is, the time period between HREADY goes LOW (the ESDCTL starts to execute the request) and HREADY goes HIGH (the ESDCTL request execution is completed).

Only one READ command is issued toward the SDRAM memory, since the misaligned read burst crosses the 32-bit SDRAM memory (8 words) boundary at the memory “natural” boundary. The read addresses are 0x04, 0x08, 0x0C, 0x10, 0x14, 0x18, 0x1C and 0x00. Without issuing the second read the last SDRAM data will come from address 0x00 since this is the “natural” 32-bit memory device boundary as well.

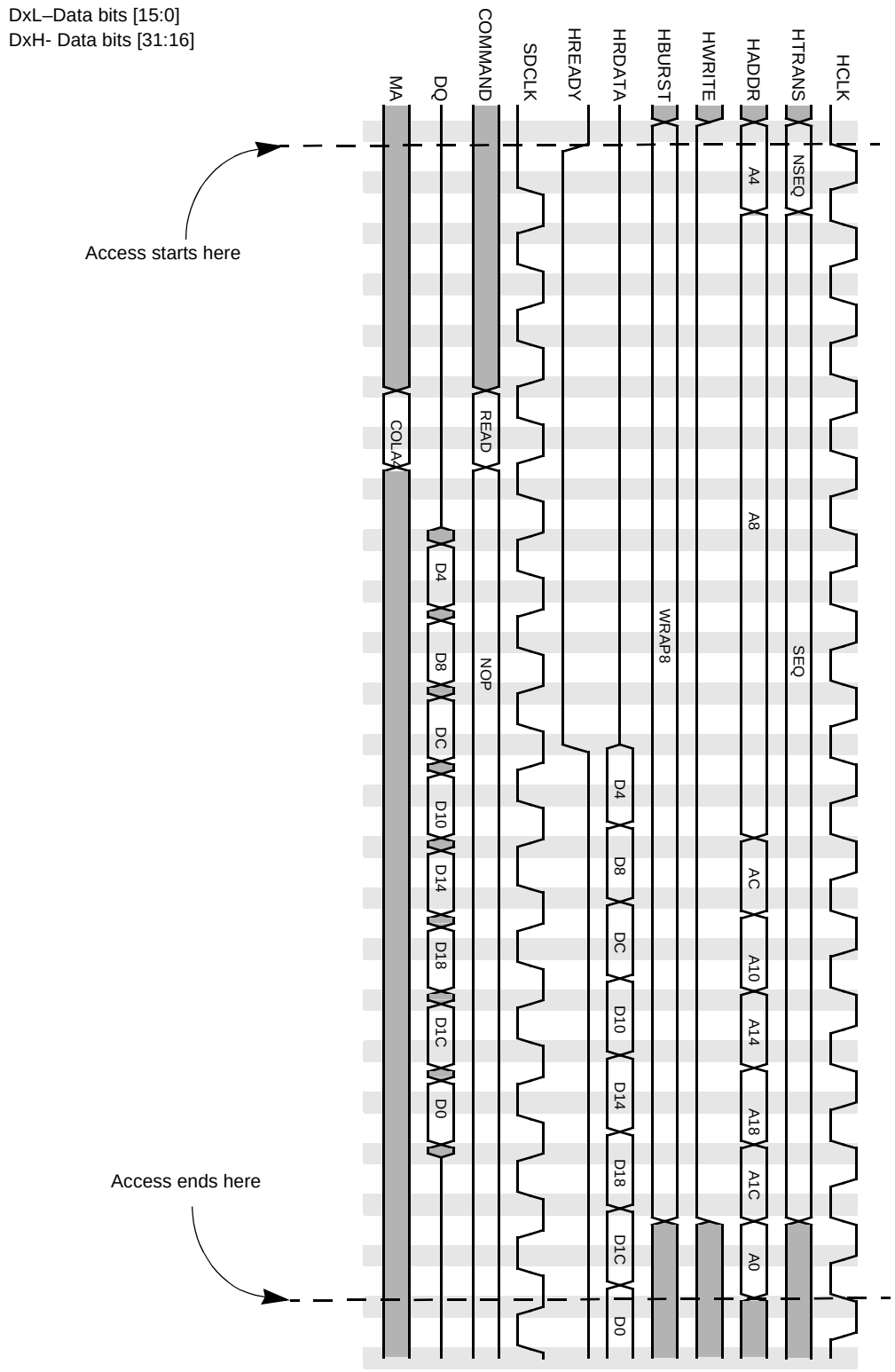


Figure 19-72. Misaligned WRAP8 Burst Read Access to 32-Bit Memory

19.4.7.2 Single Write Word Access to 32-Bit Memory

The markers in Figure 19-73 mark the request access time, for example, the time period between HREADY goes LOW (the ESDCTL starts to execute the request) and HREADY goes HIGH (the ESDCTL request execution is completed).

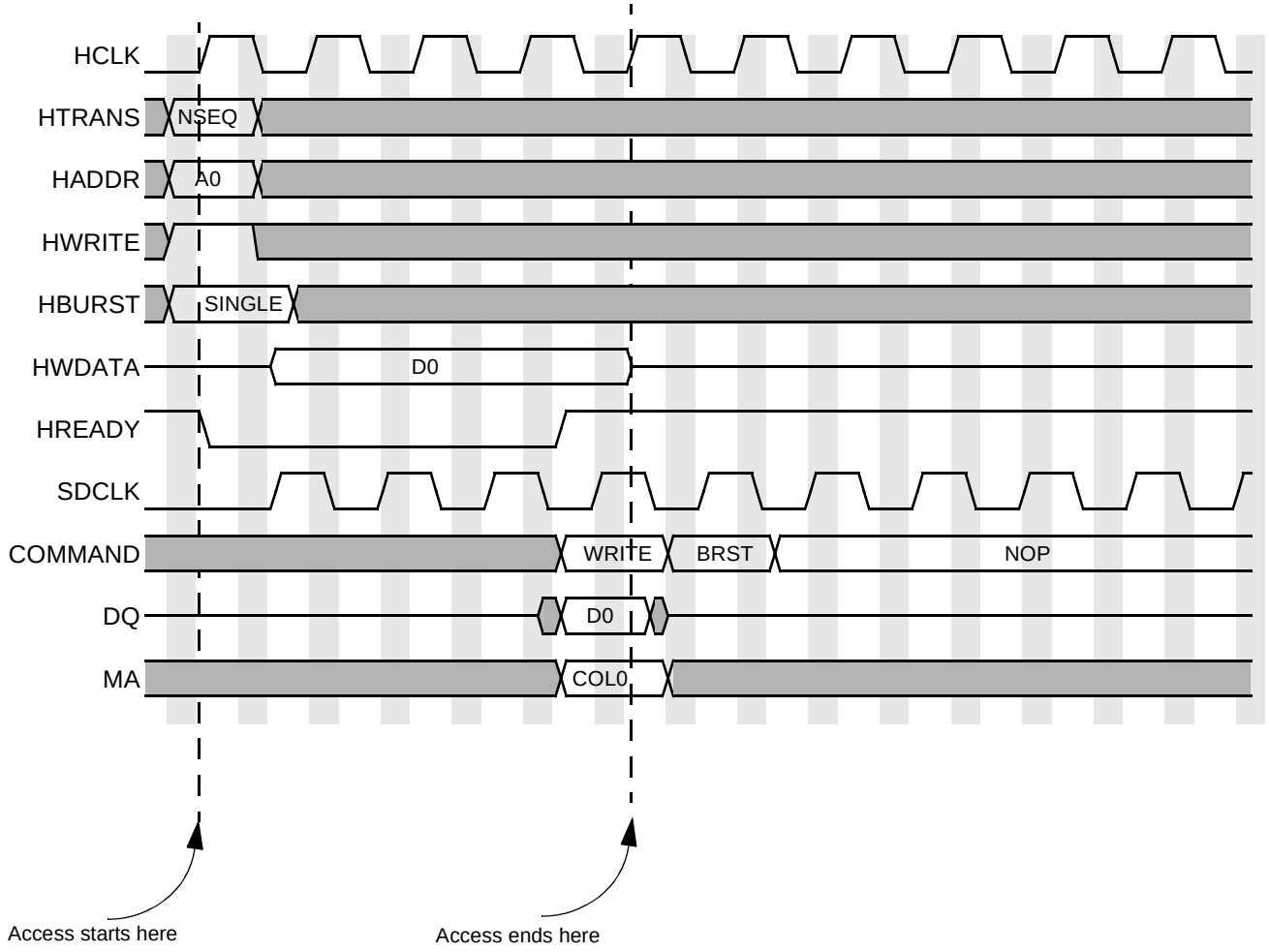


Figure 19-73. Single on Page Write-Word Access to 32-Bit Memory (Cycle Accurate)

19.4.7.2.1 INCR4 Burst Write Word Access to 32-Bit Memory

Two WRITE commands are issued toward the SDRAM memory, since the misaligned write burst crosses the 32-bit SDRAM (8 words) memory boundary. The write addresses are 0x14, 0x18, 0x1C, and 0x20. Without issuing the second write the last SDRAM data will be written to address 0x00. The ESDCTL issue the second WRITE command in such a way (at a specific timing) so continuous data flow is obtained. There is no timing difference between an aligned and a misaligned access although the second one requires more commands. The markers in Figure 19-74 mark the request access time.

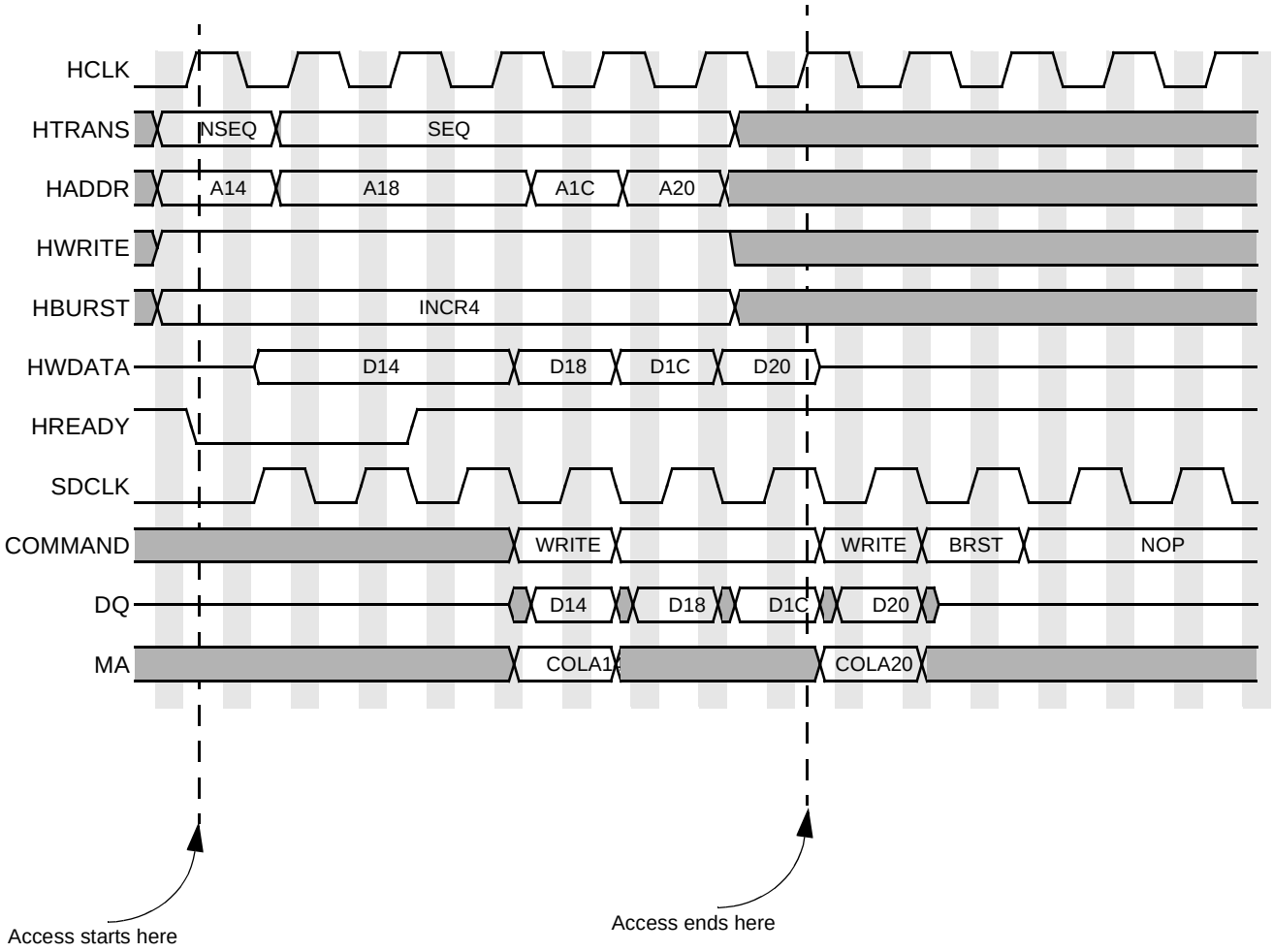


Figure 19-74. INCR4 burst on Page Write-Word Access to 32-Bit Memory (Cycle Accurate)

19.4.7.3 SDRAM Command Sequence for Burst Accesses

Table 19-34 show the commands that the ESDCTL will perform for WRAP and INCR accesses from the AHB. The controller will split the transaction when needed in cases that the access cross WRAP boundaries of the SDRAM. The memory configured to 8 beat burst (BL=8).

Unspecified INCR burst accesses will be translated to single accesses to allow the burst to terminate at any length with no additional delays.

The number in the brackets represent the address for READ command and the last Burst Address for BTERM command.

Table 19-34. SDRAM Command Sequence for Burst Accesses

Type	Bus	Address							
		0	4	8	C	10	14	18	1C
WRAP8	16-bit	READ(0)	READ(4)	READ(8)	READ(C)	READ(10)	READ(14)	READ(18)	READ(1C)
		READ(10)	READ(10)	READ(10)	READ(10)	READ(0)	READ(0)	READ(0)	READ(0)
			READ(0)	READ(0)	READ(0)		READ(10)	READ(10)	READ(10)
			BTERM(0)	BTERM(4)	BTERM(8)		BTERM(10)	BTERM(14)	BTERM(18)
	32-bit	READ(0)	READ(4)	READ(8)	READ(C)	READ(10)	READ(14)	READ(18)	READ(1C)
INCR8	16-bit	READ(0)	READ(4)	READ(8)	READ(C)	READ(10)	READ(14)	READ(18)	READ(1C)
		READ(10)	READ(10)	READ(10)	READ(10)	READ(20)	READ(20)	READ(20)	READ(20)
			READ(20)	READ(20)	READ(20)		READ(30)	READ(30)	READ(30)
			BTERM(20)	BTERM(24)	BTERM(28)		BTERM(30)	BTERM(34)	BTERM(38)
	32-bit	READ(0)	READ(4)	READ(8)	READ(C)	READ(10)	READ(14)	READ(18)	READ(1C)
			READ(20)	READ(20)	READ(20)	READ(20)	READ(20)	READ(20)	READ(20)
			BTERM(20)	BTERM(24)	BTERM(28)	BTERM(2C)	BTERM(30)	BTERM(34)	BTERM(38)
WRAP4	16-bit	READ(0)	READ(4)	READ(8)	READ(C)	READ(10)	READ(14)	READ(18)	READ(1C)
	32-bit	READ(0)	READ(4)	READ(8)	READ(C)	READ(10)	READ(14)	READ(18)	READ(1C)
		BTERM(C)	BTERM(10)	READ(0)	READ(0)	READ(0)	READ(10)	READ(10)	READ(10)
				BTERM(4)	BTERM(8)	BTERM(C)	BTERM(10)	BTERM(14)	BTERM(18)
INCR4	16-bit	READ(0)	READ(4)	READ(8)	READ(C)	READ(10)	READ(14)	READ(18)	READ(1C)
			READ(10)	READ(10)	READ(10)		READ(20)	READ(20)	READ(20)
			BTERM(10)	BTERM(14)	BTERM(18)		BTERM(20)	BTERM(24)	BTERM(28)
	32-bit	READ(0)	READ(4)	READ(8)	READ(C)	READ(10)	READ(14)	READ(18)	READ(1C)
		BTERM(C)	BTERM(10)	BTERM(14)	BTERM(18)	BTERM(1C)	READ(20)	READ(20)	READ(20)
							BTERM(20)	BTERM(24)	BTERM(28)

19.4.8 Precharge Command Mode

The Precharge Command Mode (SMODE=001) is used during SDRAM/LPDDR device initialization, and to manually deactivate an active bank(s). While in this mode, an access (either read or write) to the SDRAM/LPDDR address space will generate a precharge command cycle. SDRAM/LPDDR address bit A10 determines whether a single bank, or all banks, are precharged by the command. Accessing an address with the SDRAM/LPDDR address A10 low will precharge only the bank selected by the bank addresses, as illustrated in Figure 19-75. Conversely, accesses with A10 high will precharge all banks regardless of the bank address, as illustrated in Figure 19-76. Note that A10 is the SDRAM pin, not the A10 bit ARM address bus. Translation of the SDRAM A10 to the corresponding ARM address is dependent on the

memory configuration. The precharge command access is two clocks in length on the ARM, and one cycle to the SDRAM/LPDDR.

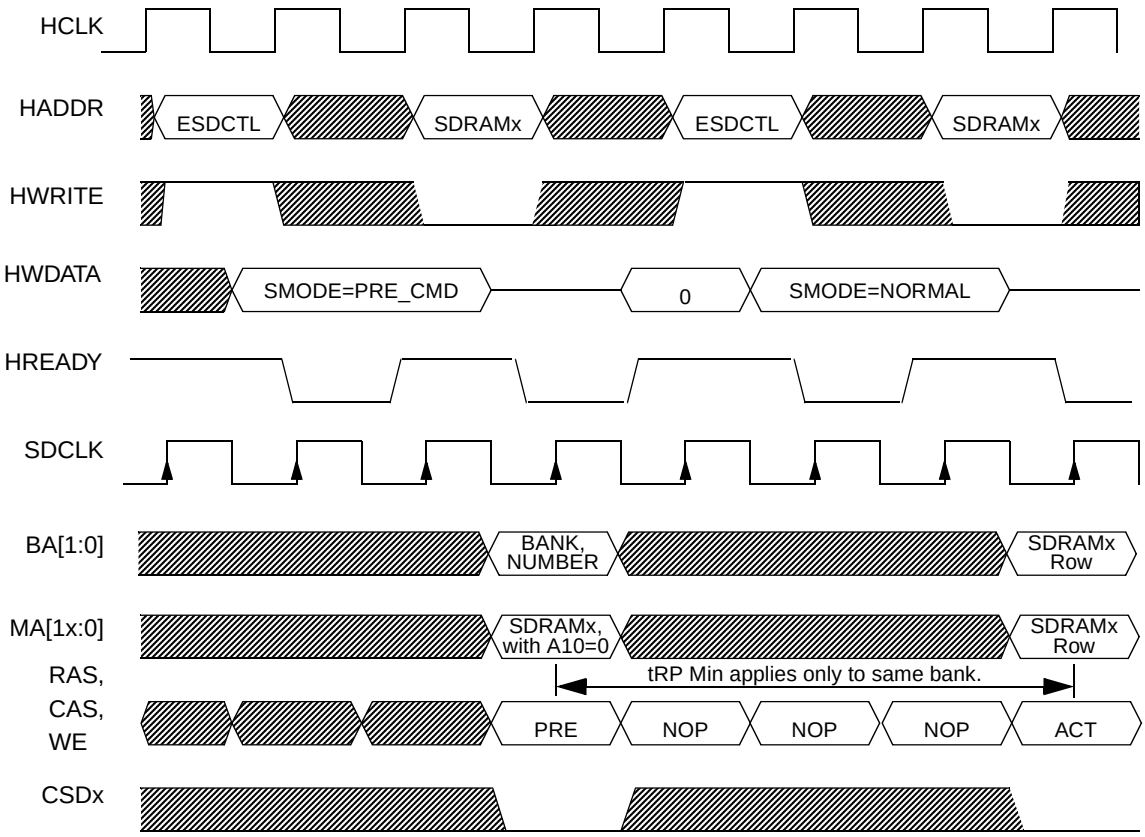


Figure 19-75. Precharge Specific Bank Timing Diagram

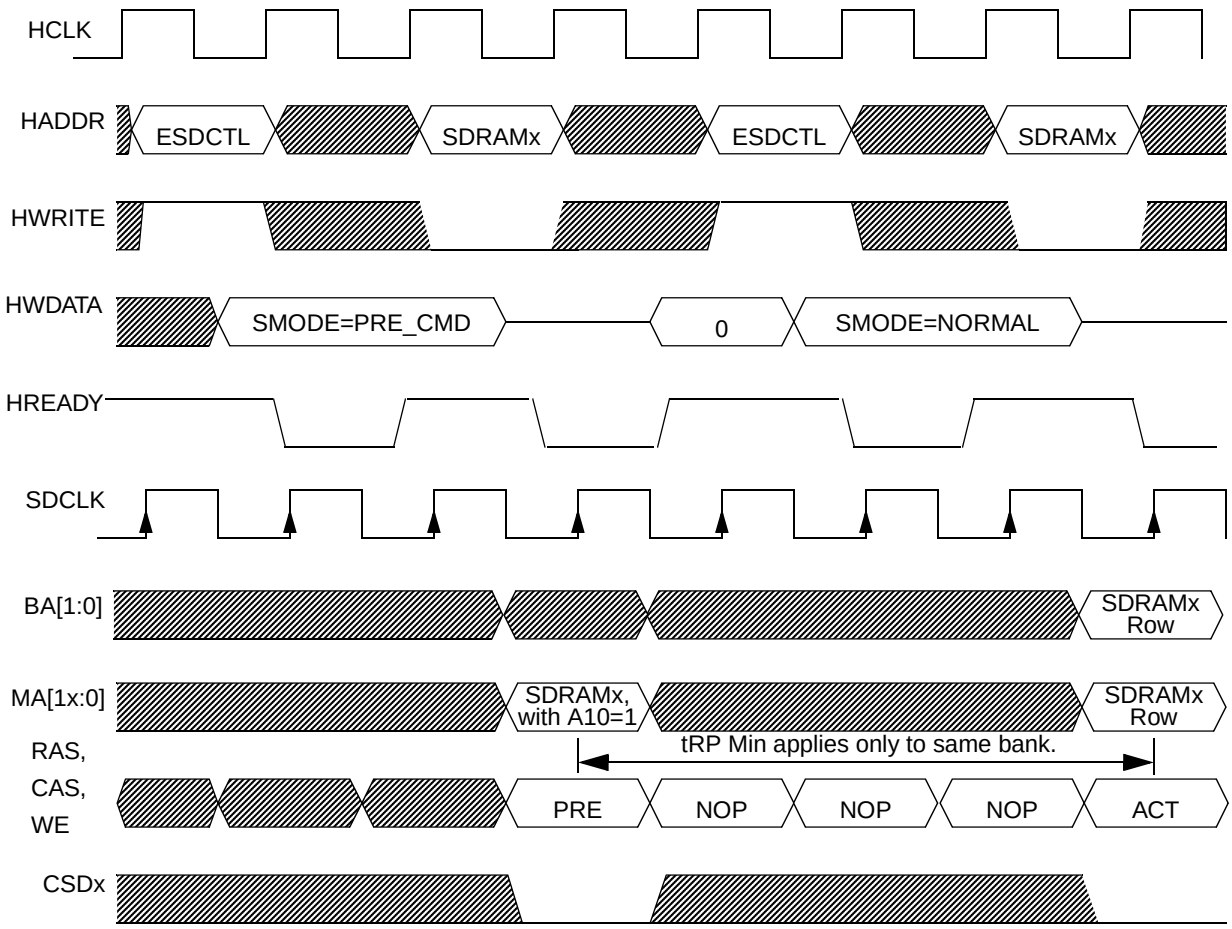


Figure 19-76. Precharge All Banks Timing Diagram

19.4.9 Auto-Refresh Mode

The Auto-Refresh Mode (SMODE=010) is used to manually request SDRAM/LPDDR refresh cycles and is normally used only during device initialization, since the ESDCTL will automatically generate refresh cycles when properly configured. The auto-refresh command (see Figure 19-77) refreshes all banks in the device, therefore the address supplied during the refresh command need only specify the correct SDRAM/LPDDR device. The lower address lines are a *Don't Care*. Either a read or write cycle may be used. If a write is used, the data will be ignored and the external data bus will not be driven. The cycle will be 2 clocks on the ARM and a single clock to the SDRAM/LPDDR device.

The ESDCTL does not guarantees that the SDRAM/LPDDR is in the idle state before the auto-refresh command is given. If one or more rows are active, a precharge-all command should be issued by the software prior to the auto-refresh command.

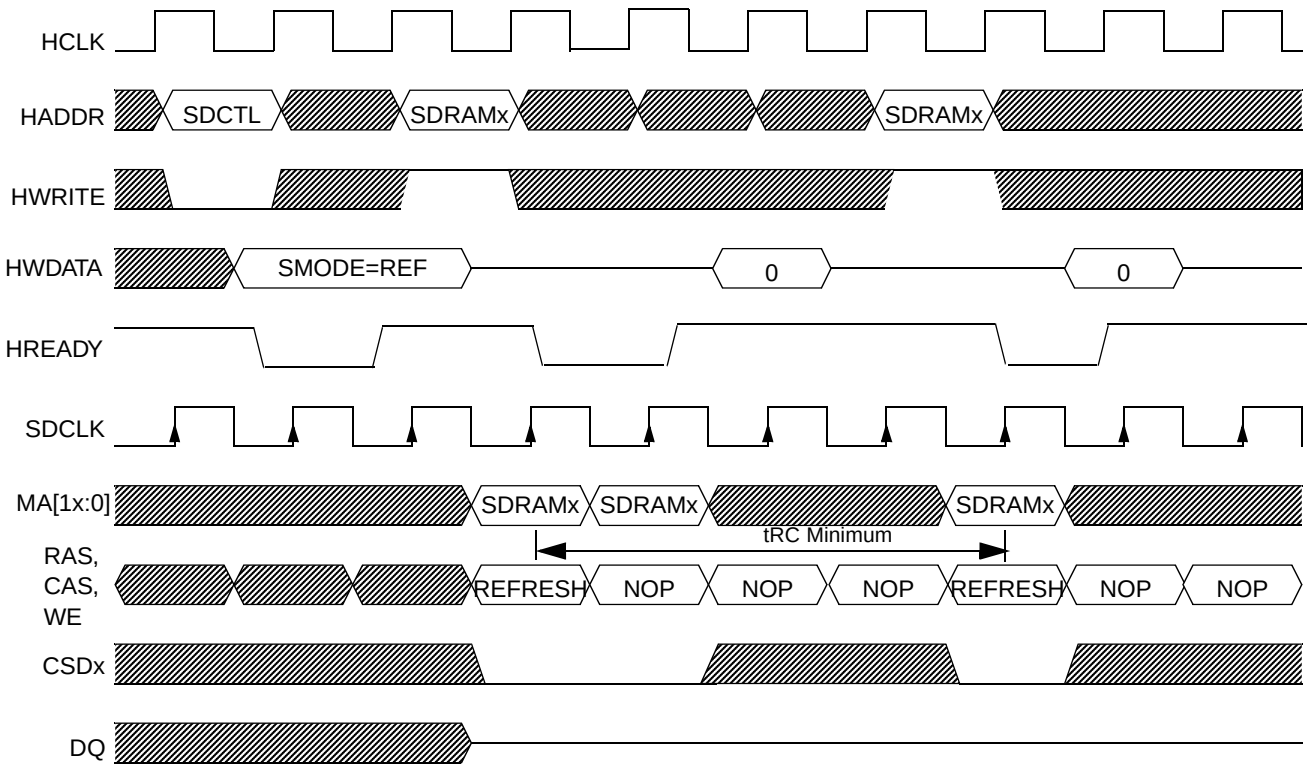


Figure 19-77. Software Initiated Auto-Refresh Timing Diagram

19.4.10 Manual Self Refresh Mode

The Manual Self Refresh Mode (SMODE=100) is used to enter the SDRAM/LPDDR external device in self refresh low power operating mode during the RUN system operating mode. Details regarding this operating mode are provided in Section 19.4.5.2, "Manual Self Refresh Mode for SDRAM/LPDDR Devices."

19.4.11 Set Mode Register Mode

The Set Mode Register mode (SMODE=011) is used to program the SDRAM/LPDDR mode register (see Example 19-1 and Section 19.5.4.2, "SDR SDRAM Load Mode Register"). This mode differs from normal SDRAM write cycles because the data to be written is transferred across the address bus. Reads of the mode register are not allowed.

After SMODE bits are set to 011, either a read or write cycle may be used to write the external memory device mode register. In both cases (read or write), the ARM data will be ignored and the external data bus will not be driven. The row and bank address signals are used to transfer the data toward the external memory device mode register (see Example 19-1 and Section 19.5.4.2, "SDR SDRAM Load Mode Register"). The cycle will be 2 clocks on the ARM and a single clock to the SDRAM device.

Figure 19-78 illustrates the bus sequence for a mode register set operation. Mode register set commands must be issued while the SDRAM/LPDDR is idle. The Enhanced SDRAM Controller does not guarantee that the SDRAMs have been returned to the idle state before issuing the mode register set command.

Software must generate a precharge all sequence before issuing the mode register set command if there is any possibility that one or more banks could be active. Also note, the row cycle time (t_{RC}) must be met before the mode register set command is issued.

Section 19.5.4.2, "SDR SDRAM Load Mode Register" provides a detailed example of the mode register data value calculation and mapping to the ARM address.

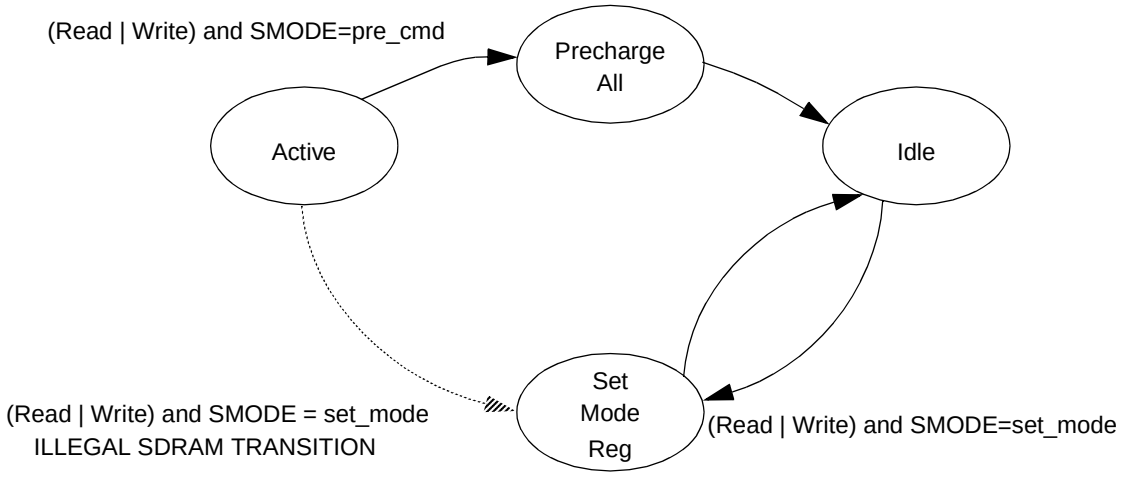
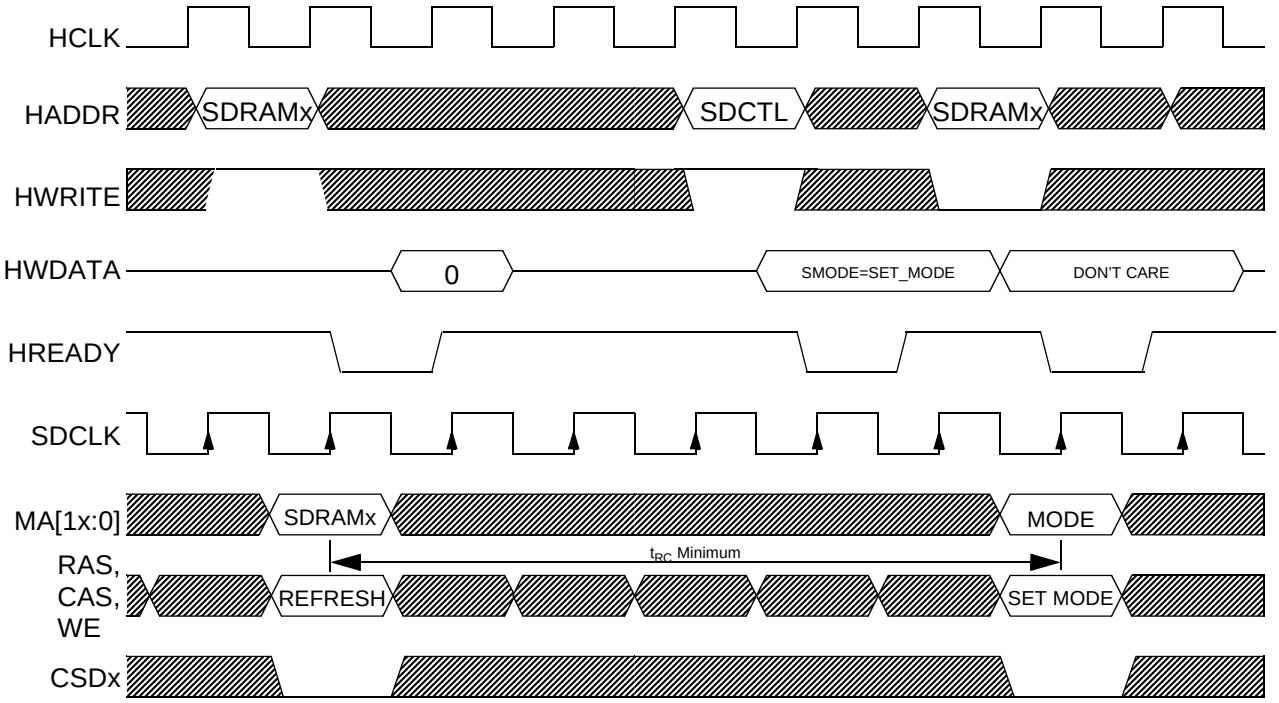


Figure 19-78. Set Mode Register State Diagram



For LPDDR:

To program the "Mode Register"

To program the "Extended Mode Register"

To program the "Low Power Extended Mode Register"

Figure 19-79. SDR and LPDDR Set Mode Register Timing Diagram

19.5 Initialization/Application Information

The following paragraphs provide details on selecting compatible SDRAM memories and configuring the controller to work with the memory system.

19.5.1 Memory Device Selection

Many SDRAM/LPDDR memory types are supported by the Enhanced SDRAM Controller. Important characteristics to consider when choosing a memory device are as follows:

- The page comparators expect 4 bank memories. 2 bank devices are not supported. Their use will result in the memory and controller losing synchronization when crossing page boundaries.
- Page (column) addressing must match one of the supported sizes.
- Memory density can be larger or smaller than those directly supported, although some memory may be inaccessible or redundantly mapped. The bank addresses are the most significant addresses and connecting a memory smaller than the selected density will result in one or more banks being inaccessible.
- The controller is designed for memories meeting PC133 timing specifications up to 133 MHz system operation. Use of non-compliant memories require a thorough analysis of all timing parameters.

19.5.2 Configuring Controller for SDRAM Memory Array

Configuration register programming options and controller-memory physical connections provide flexibility to accommodate different memory types and system configurations. Options are broadly grouped into 3 categories:

- Physical Characteristics: row and column address bus widths and data bus width.
- Timing Parametric: CAS latency, row precharge, cycle delays, refresh rate, and so on.
- Functional Features: clock suspend timer and supervisor/user protection.

Table 19-37 through Table 19-47 are provided to assist the designer with the selection of the correct physical parameters for a number of preferred memory configurations. Timing parametric are addressed in the following subsections.

19.5.3 CAS Latency

CAS latency is determined by the operating frequency and access time of the memories. For a 133 MHz system clock frequency and PC133 compliant memories, the CAS latency will generally be programmed to 3 clocks, although the memory specifications should always be consulted to confirm this value. CAS latency must be programmed in two places: the chip select Control Register and the device Mode Register. See Table 19-18 for a description of the control register encoding and Section 19.4.11, “Set Mode Register Mode” for the details on programming the SDRAM mode register.

19.5.4 SDRAM/LPDDR Initialization Sequence

Prior to normal operation (read/write accesses), the external memory device must be initialized. The following paragraphs provide detailed information covering device initialization. Register definition, command descriptions and device operation information has been thoroughly described throughout the chapter.

19.5.4.1 SDRAM Initialization

SDR and LPDDR SDRAMs must be powered up and initialized in a predefined manner. Operational procedures other than those specified by the SDRAM manufacture specification may results in undefined operation.

19.5.4.1.1 SDR SDRAM Initialization

Once power is applied to the device and the clock is stable, the SDRAM requires a 200µs delay prior to issuing any command other than a COMMAND INHIBIT or a NOP. Starting at some point during this 200µs period and continuing at least through the end of this period, COMMAND INHIBIT or NOP commands should be applied.

Once the 200µs delay has been satisfied with at least one COMMAND INHIBIT or NOP command have been applied (the SDRAM_RDY status bit will be asserted), a PRECHARGE command should be applied. All banks must then be precharged, thereby placing the device in the all banks idle state.

Once in idle state, several (manufacture dependent) AUTO REFRESH cycles must be performed. After the AUTO REFRESH cycles are complete, the SDRAM is ready for Mode Register programming. Because the Mode Register will power up in an unknown state, it should be loaded prior to applying any operational command. Figure 19-80 illustrates a SDRAM initialization routine with 8 AUTO REFRESH cycles. Table 19-1 shows an initialization SDRAM example code.

It is crucial that the dual parameters (those parameters that are defined both in the SDRAM device register and in ESDCTL registers, like CAS latency, burst length, and so on) to be identical for proper operation of both SDRAM memory and Enhanced SDRAM Controller.

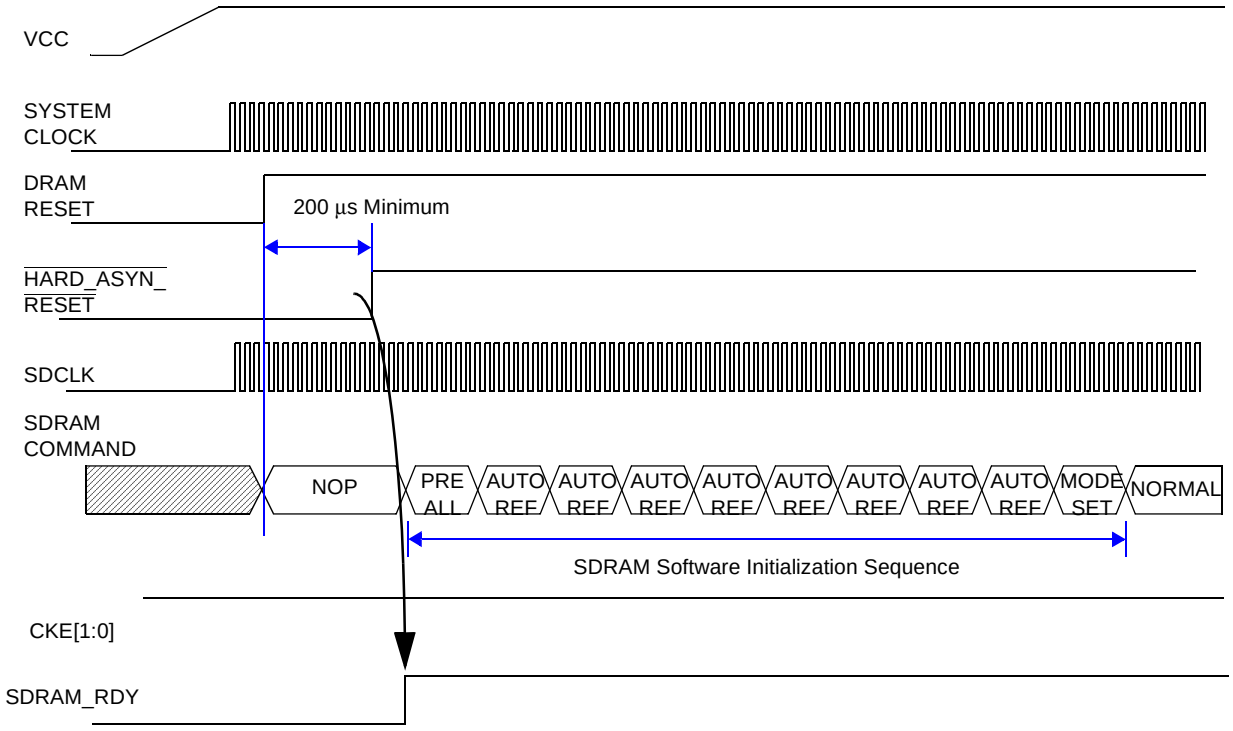


Figure 19-80. SDR SDRAM Initialization and Load Mode Register Sequence

Example 19-1. INIT_SDRAM Example Code (ARM Assembler)

```
init_sdram:
                ldr     r2, =ESD_ESDCTL0        // base address of registers
                ldr     r3, =PRE_ALL_CMD        // SMODE=001
                str     r3,(r2,#0x0)            // put CSD0 in precharge command mode
                ldr     r4, =SDRAM_CSD0         // CSD0 precharge address (A10=1)
                str     r1,(r4,#0x0)            // precharge CSD0 all banks
                ldr     r3, =AUTO_REF_CMD       // SMODE=010
                str     r3,(r2,#0x0)            // put array 0 in auto-refresh mode
                ldr     r4, =SDRAM_CSD0_BASE    // CSD0 base address
                ldr     r6,=0x7                 // load loop counter
0:              ldr     r5,(r4,#0x0)            // run auto-refresh cycle to array 0
                subs    r6,r6,#1                // decrease counter value
                bne     0b
                ldr     r3, =SET_MODE_REG_CMD // SMODE=011
                str     r3,(r2,#0x0)            // setup CSD0 for mode register write
                ldr     r3, =MODE_REG_VAL0      // array 0 mode register value
                ldrb    r5,(r3,#0x0)            // New mode register value on address bus

                ldr     r3, =NORMAL_MODE        // SMODE=000
                str     r3,(r2,#0x0)            // setup CSD0 for normal operation
```

```
ESD_ESDCTL0           .long   0xXXXX_XXXX    // system/external device dependent data
SDRAM_CSD0:           .long   0xXXXX_XXXX    // system/external device dependent data
SDRAM_CSD0_BASE:      .long   0xXXXX_XXXX    // system/external device dependent data
PRE_ALL_CMD           .long   0xXXXX_XXXX    // system/external device dependent data (SMODE=001)
AUTO_REF_CMD          .long   0xXXXX_XXXX    // system/external device dependent data (SMODE=010)
SET_MODE_REG_CMD      .long   0xXXXX_XXXX    // system/external device dependent data (SMODE=011)
MODE_REG_VAL0         .long   0xXXXX_XXXX    // system/external device dependent data
NORMAL_MODE           .long   0xXXXX_XXXX    // system/external device dependent data (SMODE=000)
```

NOTE

To do the LOAD MODE REGISTER the address starts from bit 0, so LRDB should be used.

19.5.4.1.2 LPDDR SDRAM Initialization

The DDR Mobile SDRAM must be powered up and initialized in a predefined manner. Operational procedures other than those specified may result in undefined operation. Power must first be applied to VDD and VDDQ according to the LPDDR SDRAM manufacture data sheet. Clock enable mast be driven through the SDRAM controller registers to cs0 and/or cs1.

After all power supply voltages are stable, and the clock is stable, the DDR Mobile-SDRAM requires a 200µs delay prior to applying a command other than DESELECT or NOP.

CKE is driven HIGH by the SDRAM controller on the first edge of the clock.

Once the 200us delay has been satisfied, the following command sequence shall be applied using the SDRAM controller registers:

1. A DESELECT or NOP command.
2. A PRECHARGE ALL command should then be applied, placing the device in the all banks idle state.
3. Once in the idle state, two AUTO REFRESH cycles must be performed (tRFC must be satisfied).
4. Two MODE REGISTER SET commands for the Mode Register and Extended Mode Register.

Following these cycles, the DDR Mobile-SDRAM is ready for normal operation.

NOTE

A WRITE access should be performed before the first READ access to the LPDDR (in order to assure that a 0 value will be driven on the DQS pins and held by the keeper of the DDR pads).

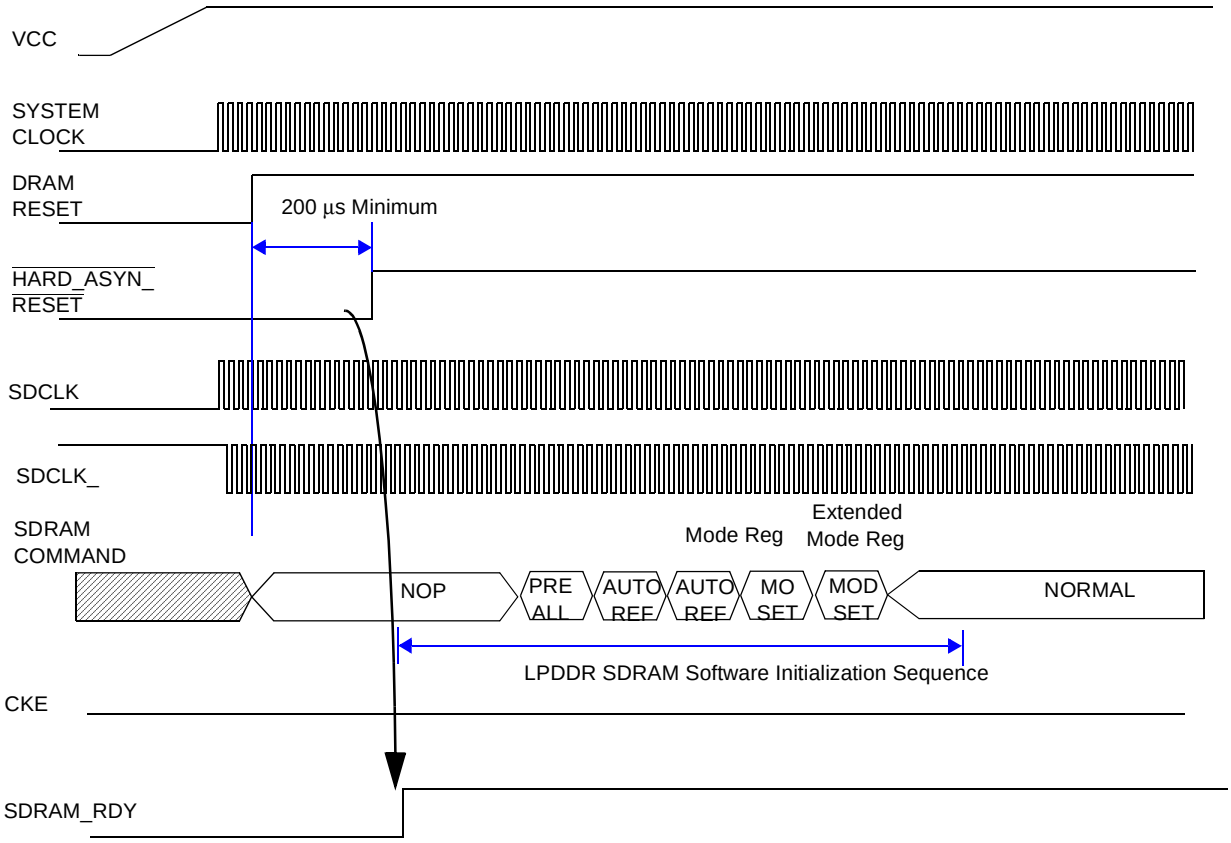


Figure 19-81. Simplified LPDDR SDRAM Initialization and Load Mode Register Sequence

19.5.4.2 SDR SDRAM Load Mode Register

The mode register is used to set the SDRAM operating characteristics including CAS latency, burst length, burst mode, and write data length. The settings depend on system characteristics including the operating frequency, memory device type, burst buffer/cache line length, and bus width. Operating characteristics vary by device type, so the data sheet must be consulted to determine the actual value to be written. In order to demonstrate the procedure, the following system characteristics will be used:

- Micron MT48LC4M32B2 128 Mbit (1M x 32 x 4 banks) SDR SDRAMs
- 133 MHz System Clock Frequency
- Sequential burst, burst length of 8
- Single word writes (for example, no bursting on writes)

Figure 19-82 shows the Mode Register bit assignments for the micron 128 Mbit SDRAM and the bit field descriptions are listed in Table 19-35.

128 Mbit SDRAM Mode Register												
	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
TYPE	Reserved*		WB	Op Mode		CAS Latency			BT	Burst Length		

Figure 19-82. 128 Mbit SDR SDRAM Mode Register

Table 19-35. SDRAM Mode Register Description

Name	Description
A11–A10	Reserved
A9 Write Burst	Write Burst Mode (WB). Selects between burst writes and single location writes. 0 Programmed Burst Length 1 Single Location Access[1]
A8–A7 Op Mode	Operating Mode. Defines operating modes. 00 Standard operation All other states reserved
A6–A4 CAS Latency	CAS Latency (CL). Sets latency between column address and data 000 Reserved 001 1 clock[1] 010 2 clocks 011 3 clocks 1xx Reserved
A3 Burst Type	Burst Type (BT). Selects burst type 0 Interleave 1 Sequential
A2–A0 Burst Length	Burst Length (BL). A 16-bit wide SDRAM requires a burst length of eight because the four 32-bit line fill cycles will be decomposed into eight 16-bit accesses. 000 = 1[1] 001 = 2[1] 010 = 4[2] 011 = 8 111 = Full Page 10x = Reserved 1x0 = Reserved

1 Not supported by the ESDCTL.

2 Not supported by the ESDCTL for 16-bit external memory device.

For this example:

- Sequential burst (BT = 0)
- Burst length of 8 (BL = 011)
- Programmed burst length (during writes) (WB = 0)
- 3 Clock Latency (CAS Latency= 011)

Once the mode register value has been determined, it must be converted to an address. The mode register is written via the address bus and the memory data sheet will specify the SDRAM address bits on which to place the data. The Enhanced SDRAM Controller drives the LSB address bits to the pins, so the memory

density and bus width do not need to be taken into account during the conversion. Table 19-36 provides an example conversion using the same system characteristics used in the previous example.

Table 19-36. Example Address Calculation for Mode Register

Mode Register	0	0	WB	0	0	CAS LATENCY			BT	BL		
Program Value	0	0	0	0	0	0	1	1	0	0	1	1
	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
SDRAM Pin	MA11	MA10	MA9	MA8	MA7	MA6	MA5	MA4	MA3	MA2	MA1	MA0
ARM Address	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

19.5.4.3 SDRAM Memory Configuration Examples

15 different SDRAM (SDR and LPDDR) configurations are demonstrated. These examples are 64 Mbit, 128 Mbit, and 256 Mbit SDRAM memories in single x16, dual x16, and single x32 configurations.

All single-configuration 16-bit examples are shown connected to the lower half of the data bus. Alternatively, the memory can be connected to the upper half of the data bus by swapping the data connections to D [31:16] and the data qualifier mask connections to DQM3 and DQM2. In this case, it will be necessary to program the DSIZ field in the Control Register to a value of 0 (configurations shown require a value of 1).

19.5.4.3.1 Single 64 Mbit (4Mx16) SDRAM Configuration

Table 19-37. Single 4Mx16 Control Register Value

Control Field	Value
Density	8 MB
Page Size	512
ROW	12
COL	8
DSIZ	16 (D [15:0])
SREFR	2

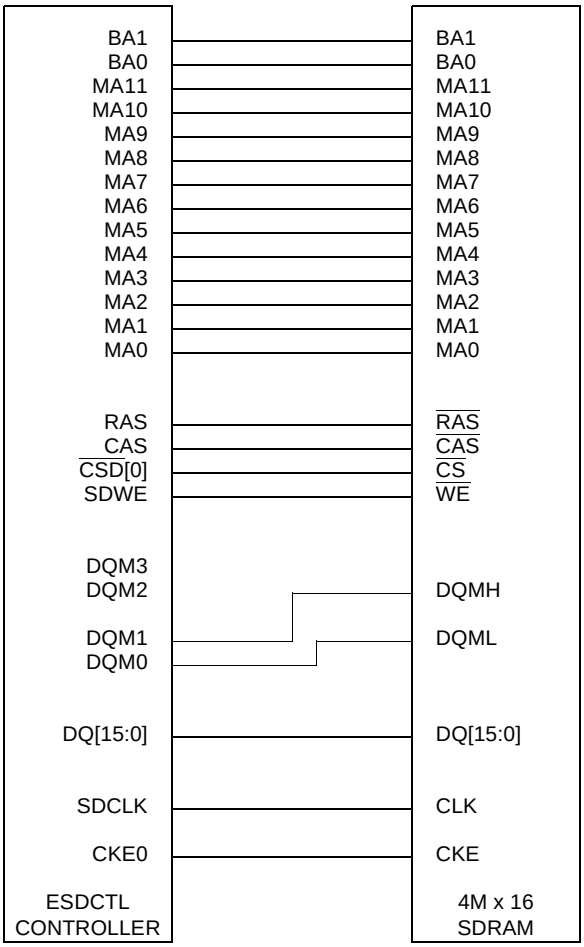


Figure 19-83. Single 64 Mbit (4M x 16) SDRAM Connection Diagram

19.5.4.3.2 Single 128 Mbit (8Mx16) SDRAM Configuration

Table 19-38. Single 8Mx16 Control Register Value

Control Field	Value
Density	16 Mbyte
Page Size	1024
ROW	12
COL	9
DSIZ	16 (D [15:0])
SREFR	4

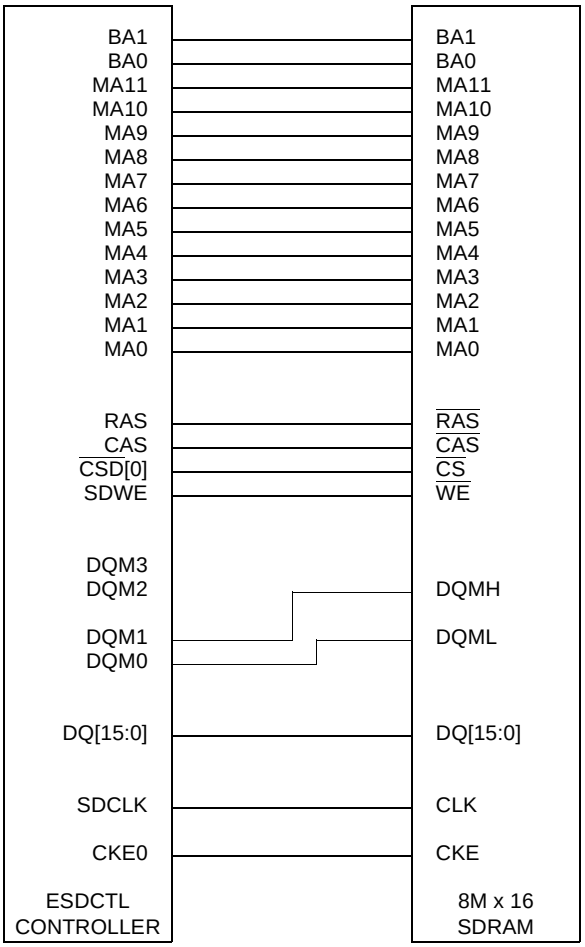


Figure 19-84. Single 128 Mbit (8M x 16) SDRAM Connection Diagram

19.5.4.3.3 Single 256 Mbit (16Mx16) SDRAM Configuration

Table 19-39. Single 16Mx16 Control Register Value

Control Field	Value
Density	32 Mbyte
Page Size	1024
ROW	13
COL	9
DSIZ	16 (D [15:0])
SREFR	4

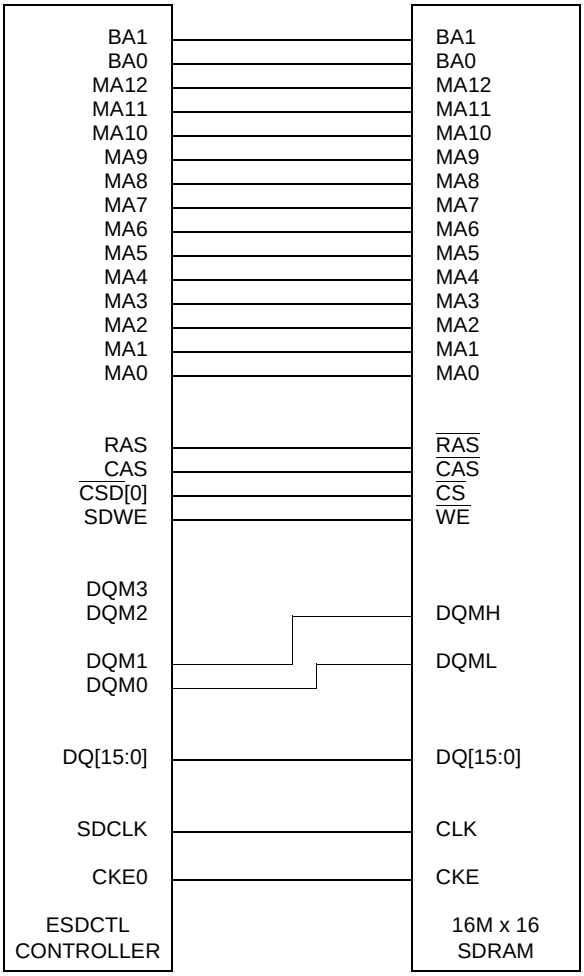


Figure 19-85. Single 256 Mbit (16M x 16) Connection Diagram

19.5.4.3.4 Single 512 Mbit (32Mx16) SDRAM Configuration

Table 19-40. Single 32Mx16 Control Register Value

Control Field	Value
Density	32 Mbyte
Page Size	1024
ROW	13
COL	10
DSIZ	16 (D [15:0])
SREFR	4

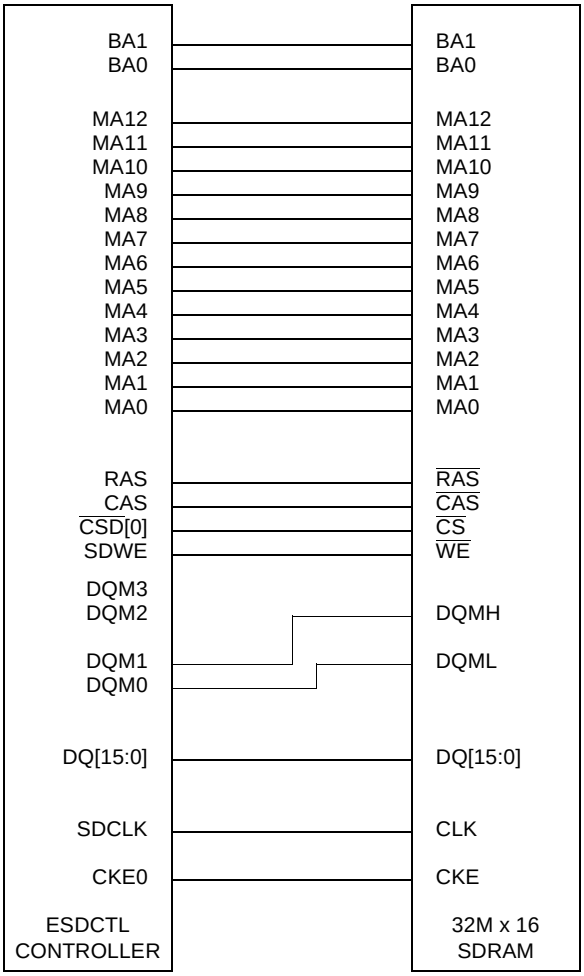


Figure 19-86. Single 512 Mbit (32M x 16) SDRAM Connection Diagram

19.5.4.3.5 Single 1-Gbit (64Mx16) SDRAM Configuration

Table 19-41. Single 64Mx16 Control Register Value

Control Field	Value
Density	64 Mbyte
Page Size	2048
ROW	14
COL	10
DSIZ	16 (D [15:0])
SREFR	8

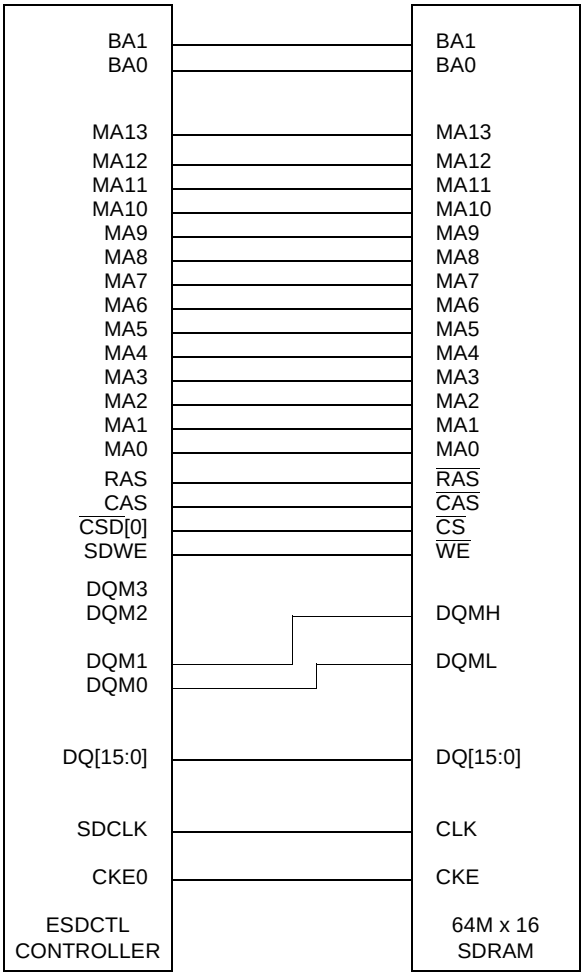


Figure 19-87. Single 1-Gbit (64M x 16) SDRAM Connection Diagram

19.5.4.3.6 Dual 64 Mbit (4Mx16) SDRAM Configuration

Table 19-42. Dual 4Mx16 Control Register Value

Control Field	Value
Density	16 Mbyte
Page Size	1024
ROW	12
COL	8
DSIZ	32 (D [31:0])
SREFR	4

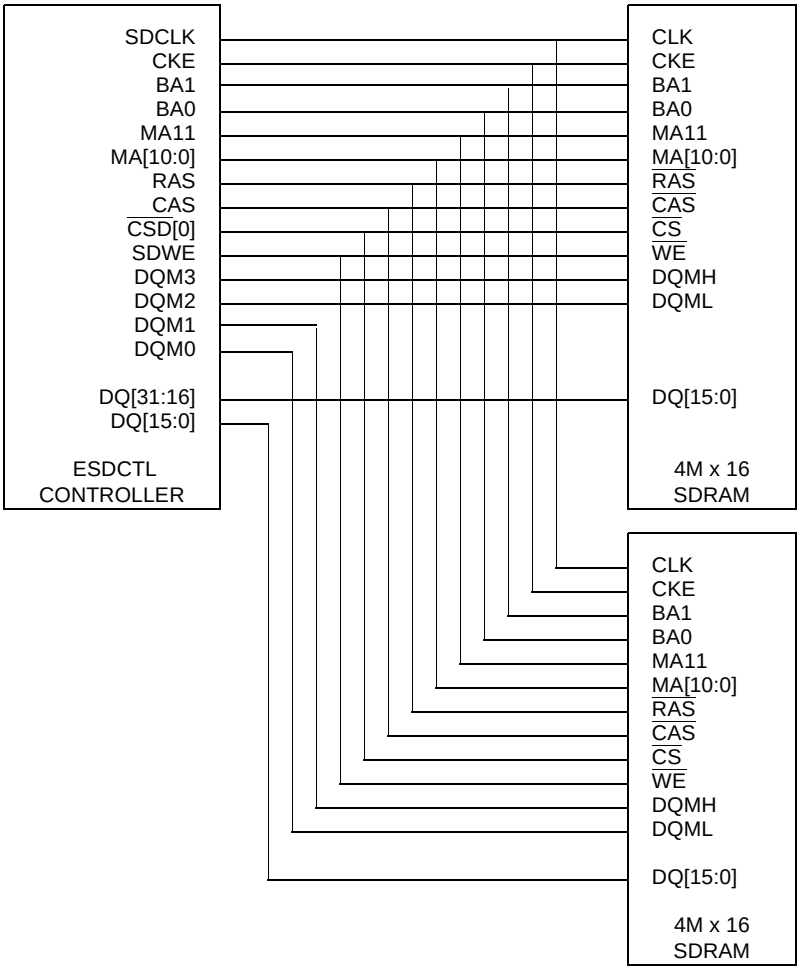


Figure 19-88. Dual 64 Mbit (4M x 16 x 2) SDRAM Connection Diagram

19.5.4.3.7 Dual 128 Mbit (8Mx16) SDRAM Configuration

Table 19-43. Dual 8Mx16 Control Register Value

Control Field	Value
Density	32 Mbyte
Page Size	2048
ROW	12
COL	9
DSIZ	32 (D [31:0])
SREFR	4

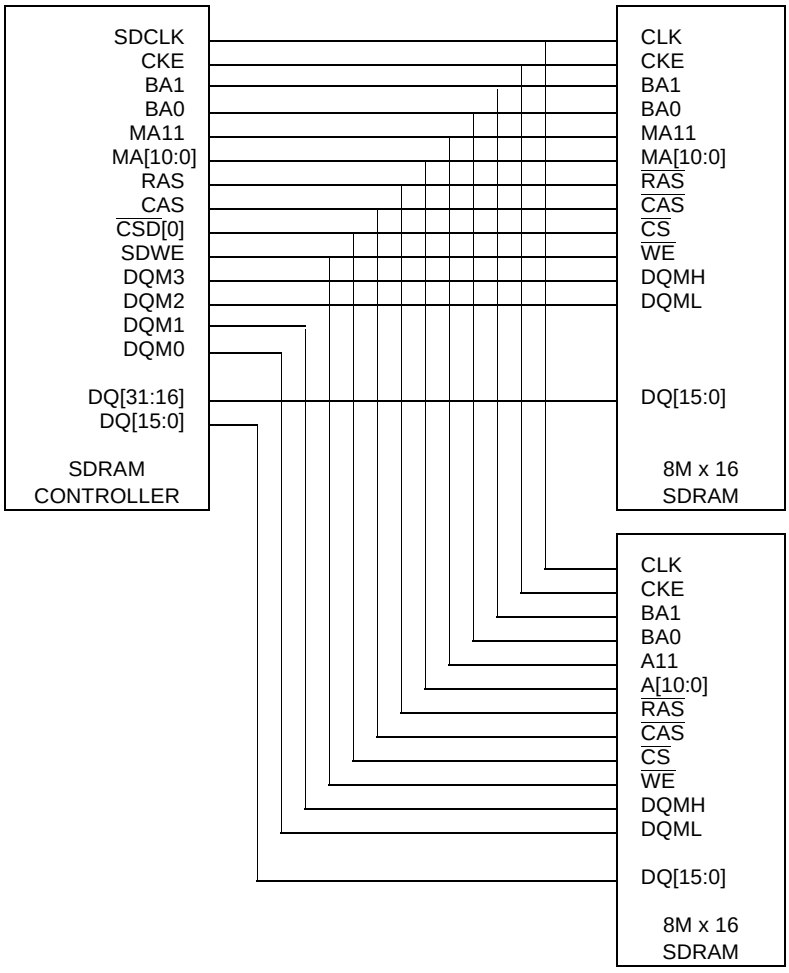


Figure 19-89. Dual 128 Mbit (8M x 16 x 2) SDRAM Connection Diagram

19.5.4.3.8 Dual 256 Mbit (16Mx16) SDRAM Configuration

Table 19-44. Dual 16Mx16 Control Register Value

Control Field	Value
Page Size	2048
ROW	13
COL	9
DSIZ	32 (D [31:0])
SREFR	4

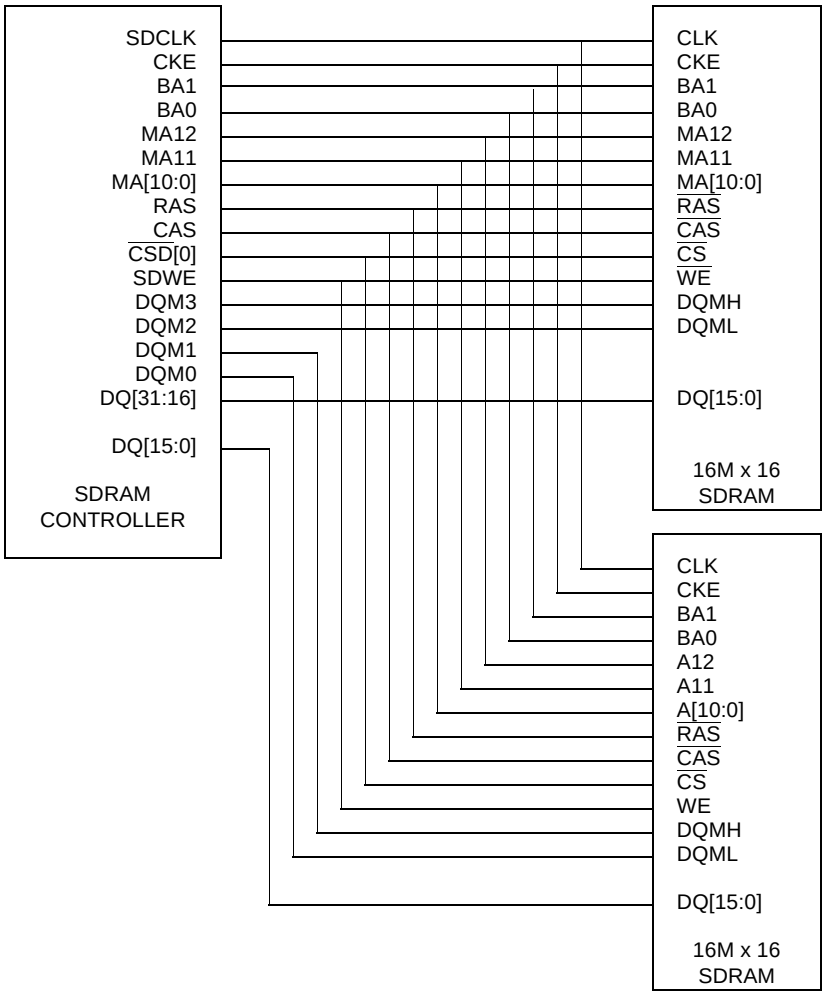


Figure 19-90. Dual 256-Mbit (16M x 16 x 2) SDRAM Connection Diagram

19.5.4.3.9 Single 64-Mbit (2Mx32) SDRAM Configuration

Table 19-45. Single 2Mx32 Control Register Value

Control Field	Value
Density	2 Mbyte
Page Size	1024
ROW	11
COL	8
DSIZ	32 (D [31:0])
SREFR	1

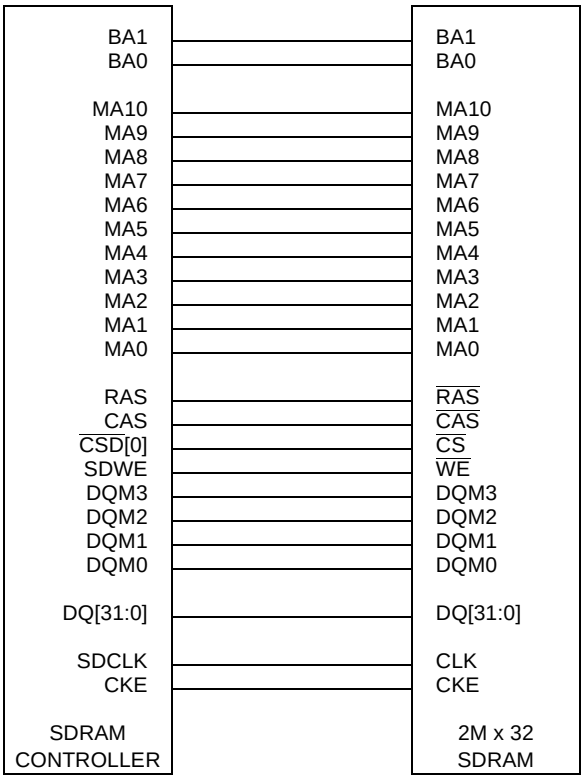


Figure 19-91. Single 64-Mbit (2Mx32) SDRAM Connection Diagram

19.5.4.3.10 Single 128-Mbit (4Mx32) SDRAM Configuration

Table 19-46. Single 4Mx32 Control Register Value

Control Field	Value
Density	4 Mbyte
Page Size	1024
ROW	12
COL	8
DSIZ	32 (D [31:0])
SREFR	2

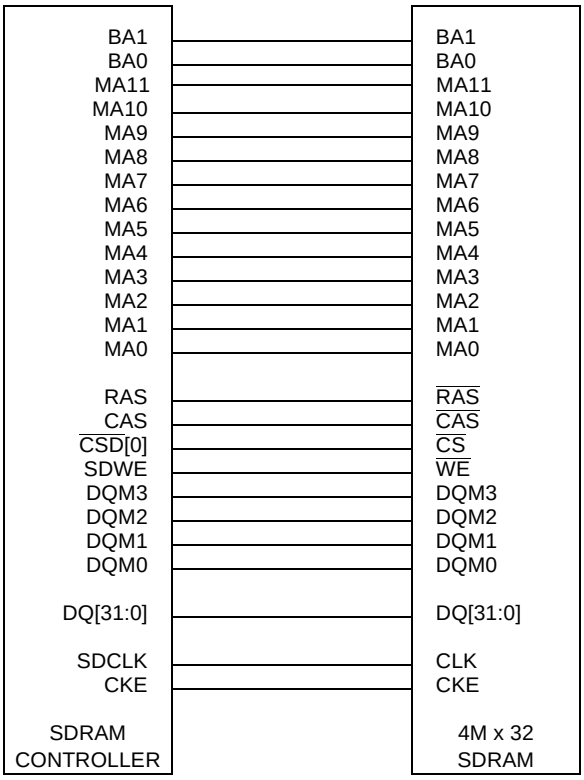


Figure 19-92. Single 128-Mbit (4Mx32) SDRAM Connection Diagram

19.5.4.3.11 Single 256-Mbit (8Mx32) SDRAM Configuration

Table 19-47. Single 8Mx32 Control Register Value

Control Field	Value
Density	8 Mbyte
Page Size	1024
ROW	13
COL	8
DSIZ	32 (D [31:0])
SREFR	4

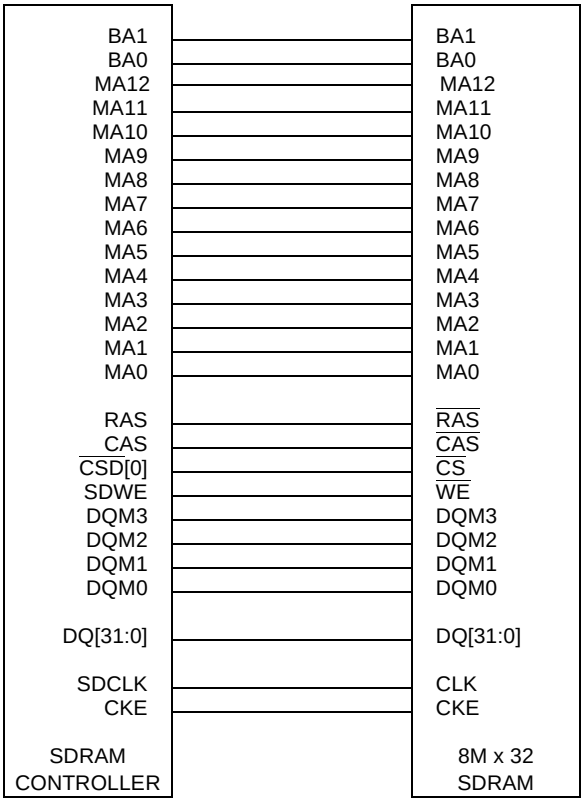


Figure 19-93. Single 256-MB (8Mx32) SDRAM Connection Diagram

19.5.4.3.12 Single 512-Mbit (16Mx32) SDRAM Configuration

Table 19-48. Single 16Mx32 Control Register Value

Control Field	Value
Density	16 Mbyte
Page Size	1024
ROW	13
COL	9
DSIZ	32 (D [31:0])
SREFR	4

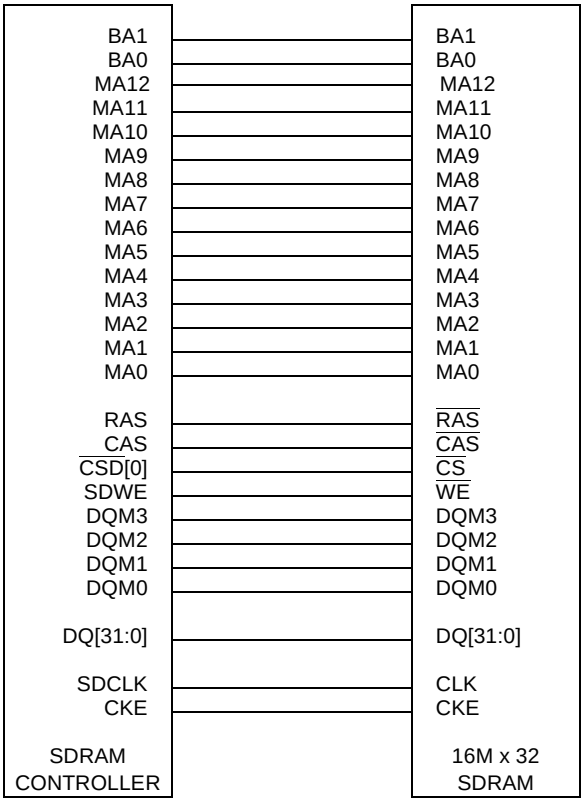


Figure 19-94. Single 512-Mbit (16Mx32) SDRAM Connection Diagram

19.5.4.3.13 Single 1-Gbit (32Mx32) SDRAM Configuration

Table 19-49. Single 32Mx32 Control Register Value

Control Field	Value
Density	32 Mbyte
Page Size	1024
ROW	14
COL	9
DSIZ	32 (D [31:0])
SREFR	8

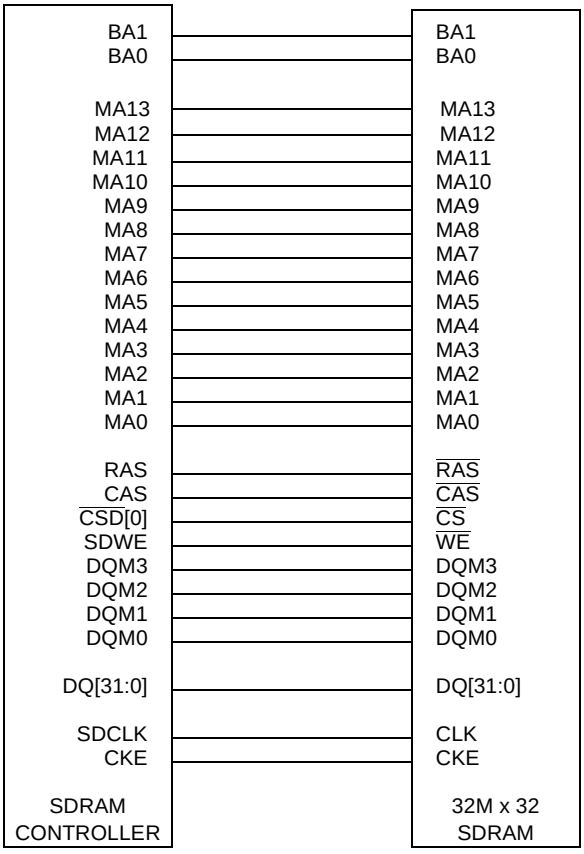


Figure 19-95. Single 1-Gbit (32Mx32) SDRAM Connection Diagram

19.5.4.3.14 Single 2-Gbit (64Mx32) SDRAM Configuration

Table 19-50. Single 64Mx32 Control Register Value

Control Field	Value
Density	64 Mbyte
Page Size	1024
ROW	14
COL	10
DSIZ	32 (D [31:0])
SREFR	8

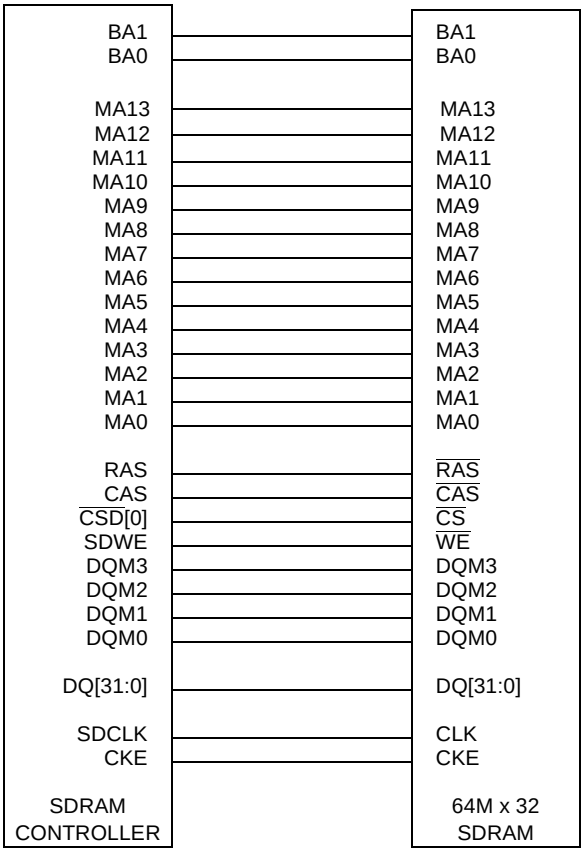


Figure 19-96. Single 2-Gbit (64Mx32) SDRAM Connection Diagram

19.5.4.3.15 Single 512-Mbit (16Mx32) Mobile DDR SDRAM Configuration

Table 19-51. Single 16Mx32 Control Register Value

Control Field	Value
Density	16 Mbyte
Page Size	1024
ROW	13
COL	9
DSIZ	32 (D [31:0])
SREFR	4

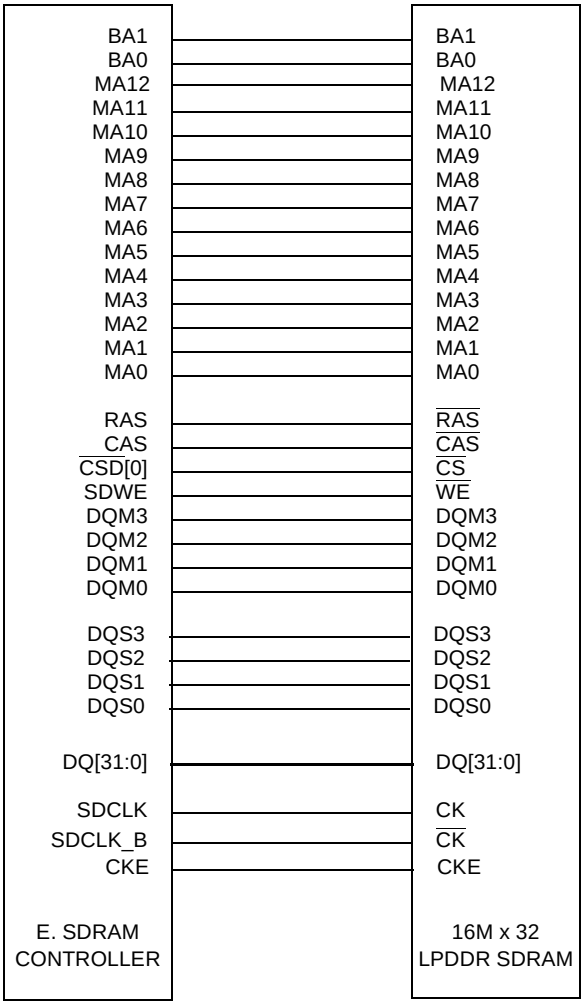


Figure 19-97. Single 512-Mbit (16Mx32) LPDDR SDRAM Connection Diagram

19.5.4.3.16 Single 512-Mbit (32Mx16) Mobile DDR SDRAM Configuration

Table 19-52. Single 32Mx16 Control Register Value

Control Field	Value
Density	32 Mbyte
Page Size	1024
ROW	13
COL	10
DSIZ	16 (D [15:0])
SREFR	4

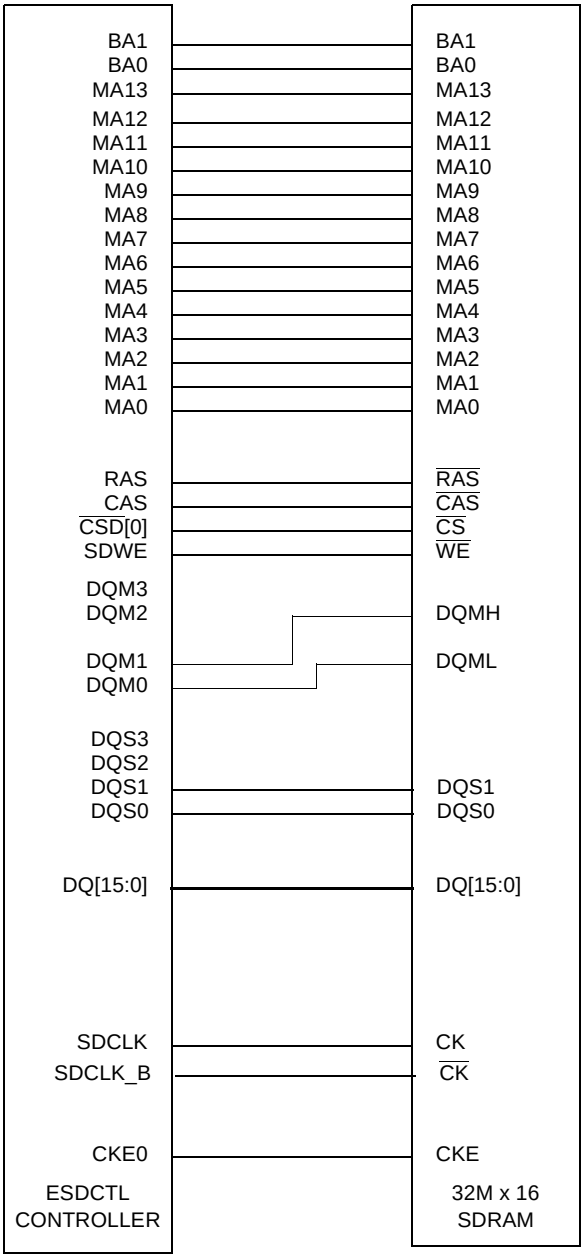


Figure 19-98. Single 512-Mbit (32Mx16) LPDDR SDRAM Connection Diagram

Chapter 20
NAND Flash Controller (NANDFC)

The NAND Flash Controller (NANDFC or NFC) device is a type of flash memory that is optimized for data storage applications with its unique cell structure, providing significant cost advantages over conventional NOR flash memory. The NANDFC integrates the functionality necessary for access to these devices. NAND Flash has a smaller bit cell, but has a fast sequential access as compared to a NOR flash, which has a large bit cell but a fast random access. This makes NAND Flash perfect as a data memory for storing audio/video files in NAND Flash, because these files are typically stored in a sequential manner while access to processor code is random. See Figure 20-1 for a block diagram of the NANDFC.

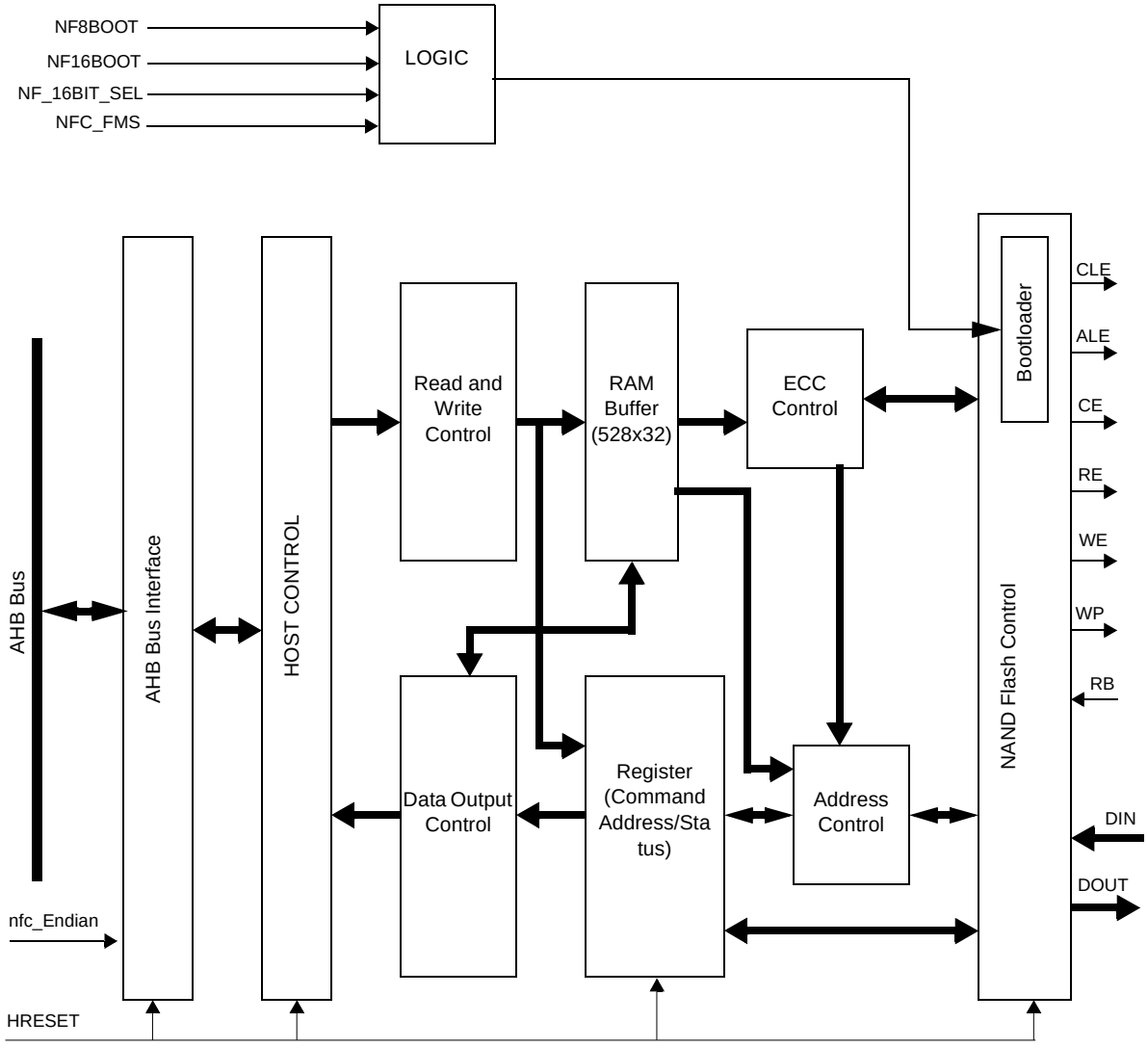


Figure 20-1. NAND Flash Controller Block Diagram

20.1 Overview

The NANDFC interfaces standard NAND Flash devices to the i.MX31 and i.MX31L, and hides the complexities of accessing the NAND Flash. It provides a glueless interface to both 8-bits and 16-bits NAND Flash parts with page sizes of 512 bytes or 2 Kbytes, and densities up to 2 Gbytes.

20.2 Operation

The NANDFC implements the interface to standard NAND Flash memory devices and contains a control logic and 2 Kbytes of internal RAM buffer. This 2-Kbyte RAM buffer is used as BootRAM during a cold reset (if a boot from NAND Flash is chosen), and is used as buffer RAM after a boot procedure.

The controller has X16-bit and X32-bit interface with the AHB bus and X8-bit/X16-bit interface with the NAND Flash. The controller supports NAND Flash devices with page sizes of 512 byte/2 Kbyte.

When the AHB Host needs to read from NAND Flash, it should configure the controller and wait for an interrupt. The NAND Flash Controller automatically brings a page from the NAND FLASH and then generate an interrupt to the AHB Host. When the interrupt occurs the host reads the content from the internal RAM buffer.

When AHB Host needs to program (write) the NAND Flash, it configures the controller and writes the program content to the internal RAM buffer and writes a program command.

20.3 Features

The NANDFC includes the following features:

- 8-bits/16-bits (pin option) NAND Flash Interface
- Internal RAM buffer (2 Kbytes + 64 bytes)
  - — Can be configured as Boot RAM and operates as a buffer during normal operation
  - — Memory mapped (to the same AHB region) registers and internal RAM buffer
- Manual interface with NAND Flash devices
  - — Supports all NAND Flash products regardless of density/organization (with pages of 512 Bytes/2 Kbytes)
- AHB Host Interface type
  - — Read/Write Burst
  - — 16-bits/32-bits bus transfers
- Programmable read latency for internal bus (directly affects AHB bus)
- ECC mode/Bypass ECC
- Multiple Reset
  - — Cold Reset/ Warm Reset/Hot Reset (reset of NANDFC and NAND Flash device)
- Internal Bootcode loader during power-up (can be enabled/disabled), providing advanced data protection
  - — Data Protection
  - — Write Protection mode for RAM buffer: Write protection of RAM buffer (LSB 1 Kbyte of RAM buffer). For more details, see Section 20.7.12, “NAND Flash Write Protection Status (NAND_FLASH_WR_PR_ST).”
  - — Write Protection mode for NAND Flash device: Block based write protection of NAND Flash
- Automatic Write protection for RAM buffer and NAND Flash during power-up
- Write Protection: automatic write protection for RAM buffer and NAND Flash during power-up, in addition to run-time write protection modes for both the RAM buffer and the NAND Flash device.
- Handshaking Feature: INT pin indicates ready/busy status of NANDFC

- IO pins sharing support
  - Allows sharing of the IO pins with other memory controllers through special arbitration logic.

20.4 External Signal Description

This section describes the NANDFC external signals.

20.4.1 Overview

The following signals shown in Table 20-1 are used to configure and control the NANDFC and its attached Flash device.

Table 20-1. NANDFC Signal Properties

Name	Port	Function	I/O	Reset
hclk_in	—	AHB clock of 133 MHz	I	enable
$\overline{\text{hreset}}$	—	WARM Reset (active low)	I	0
$\overline{\text{ipi\_int\_nfc}}$	—	NAND Flash Controller interrupt	O	1
ipp_flash_clk	—	Clock for the flash side	I	enable
ipp_nfc_ale_out	NFALE	Flash Address Latch Enable	O	0
ipp_nfc_ce_out	$\overline{\text{NFCE}}$	Flash Chip Enable	O	1
ipp_nfc_cle_out	NFCLE	Flash Command Latch Enable	O	0
ipp_nfc_rb_in	NFRB	Flash Ready/Busy	I	1
ipp_nfc_re_out	$\overline{\text{NFRE}}$	Flash Read Enable	O	1
ipp_nfc_read_data_in [15:0]	IO[15:0]	NANDFC data input from the NAND Flash	I	xxxx
ipp_nfc_we_out	$\overline{\text{NFWE}}$	Flash Write Enable	O	1
ipp_nfc_wp_out	$\overline{\text{NFWP}}$	Flash Write Protect	O	1
ipp_nfc_write_data_out [15:0]	IO[15:0]	NANDFC data output towards the NAND Flash	O	0000
$\overline{\text{ipp\_reset}}$	—	Power on Reset for booting	I	1
nf_16bit_sel	—	Use 8- or 16-bit NAND Flash	I	1
$\overline{\text{nf8boot}}$	—	Boot from 8-bit NAND Flash	I	1
nfc_fms	—	Flash Memory Select (512B/2Kbyte page size)	I	0
$\overline{\text{ng16boot}}$	—	Boot from 16-bit NAND Flash	I	1

20.4.2 Detailed Signal Descriptions

Table 20-2 gives a detailed description of the NANDFC signals.

Table 20-2. NANDFC Detailed Signal Descriptions

Signal	I/O	Description
NFCE	O	Flash Chip Enable. This signal indicates the NAND Flash selection. When the NAND Flash device is in the Busy state, or when the NAND Flash device is accessed, this signal is low.
NFRE	O	Flash Read Enable. This output is the NAND Flash device serial data output control. When active, this signal drives the data from the NAND Flash device onto the NAND Flash I/O bus, allowing the NANDFC to read the data. When reading a burst from the NAND Flash device, this signal increments the NAND Flash internal column address counter by one.
NFWE	O	Flash Write Enable.This output controls writes to the NAND Flash I/O port, thus allowing the NAND Flash device to read data. Commands, address and data are latched on the rising edge of the WE signal.
NFCLE	O	Flash Command Latch Enable. The CLE output controls the activating path for commands sent to the command register of NAND Flash (NAND_Flash_CMD). When active high, commands are latched into the command register of NAND Flash through the I/O ports on the rising edge of the WE signal.
NFALE	O	Flash Address Latch Enable. The ALE output controls the activating path for addresses sent to the address register of NAND Flash (NAND_Flash_Add). When active high, addresses are latched into the NAND FC address register of NAND Flash through the I/O ports on the rising edge of the WE signal.
NFWP	O	Flash Write Protect. This signal provides inadvertent program/erase protection during power transitions and is automatically controlled by the NANDFC. This pin status is only active (held low) during power-up.
NFRB	I	Flash Ready/Busy. This signal indicates the status of the NAND Flash operation. When low, it indicates that a program, erase, or random read operation of NAND Flash is in process. Upon completion of the process, this signal returns to high state. **Note:** This signal is connected to an open drain output, via a 100-KΩ pull-up resistor (outside the external NAND Flash memory device).
hreset		Warm Reset. This signal produces a Warm reset causing the NANDFC and the NAND Flash device to cease current operation, and set all internal registers to their default state. See Figure 20-2 for a timing diagram of warm reset operation. The AHB bus interface is connected directly to this signal (hreset) and will cause a reset immediately when this line goes to low state. The NANDFC will not be reset if the hreset pulses are shorter than two ipp_flash_clk cycles (50 ns if the clock period is 25 ns), but the NANDFC will be reset if the hreset pulse is longer than 20 ipp_flash_clk cycles (500 ns if this clock period is 25 ns). Warm reset has no effect on the contents of main/spare area buffers.
NF8BOOT NF16BOOT NF_16BIT_SEL		The NF8BOOT and NF16BOOT are boot signals that determine when the chip will boot from the NAND Flash device, in addition to indicating boot selection it is also used to controls the bus width of the NAND Flash (8-bit or 16-bit). If one of the boot inputs is asserted (NF8BOOT or NF16BOOT is low) at System Power-On reset (ipp_reset rising), a 2-Kbyte sized Bootcode is copied from the NAND Flash device to the RAM buffer. If none of the boot signals are asserted, then the input signal NF_16BIT_SEL is read. This is part of the logic of the operating modes of the NANDFC. For more information on operating modes, see Section 20.8.1, “Modes of Operation.” **Note:** The boot signals should remain at the same value before and after the boot process.

Table 20-2. NANDFC Detailed Signal Descriptions (continued)

Signal	I/O	Description
NFC_FMS	I	Flash Memory Select. The NFC_FMS signal from the Clock Control Module (CCM) indicates the size of the NAND Flash page (512 bytes or 2 Kbytes). 0 The NAND Flash page size is 512 bytes. (64 Mbyte/128 Mbyte/256 Mbyte/512 Mbyte/1 Gbyte DDP 1 The NAND Flash page size is 2 Kbytes.
$\overline{\text{ipp\_reset}}$	I	This input is the Power On Reset (POR) signal in the NANDFC. When it is asserted high, a POR takes place.
$\overline{\text{ipi\_int\_nfc}}$	O	NANDFC Interrupt. This output is the NANDFC interrupt, and is asserted when an NANDFC event takes place. It sets itself to “1” when basic operation and boot loading is done, or when a warm or hot reset is released. In addition, it is asserted when any of the following occur: • NAND Flash command input • NAND Flash address input • NAND Flash data input • NAND Flash data output
hclk_in	I	H Clock Input. This is the clock signal for the NANDFC, which arrives from the AHB side. Its value can up to 133 MHz.
ipp_nfc_write_data_out[15:0]	O	The AHB Host uses this path to write data to registers or to internal memory.
ipp_nfc_read_data_in[15:0]	I	The AHB Host uses this path to read data from registers or from internal memory.
ipp_flash_clk		This clock signal controls the NAND flash controller’s state machine when interfacing with a NAND flash device.

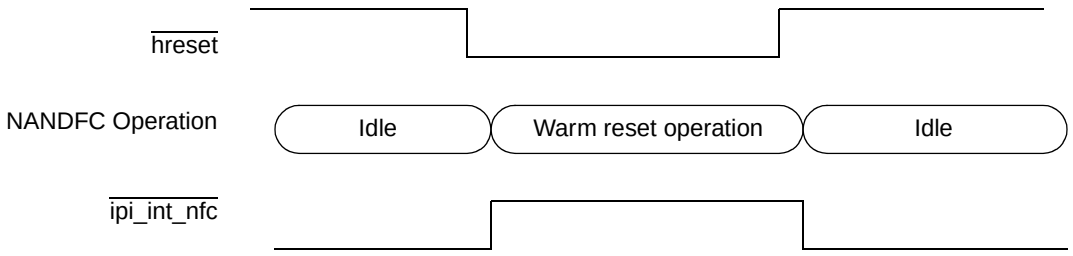


Figure 20-2. Warm Reset Operation

20.5 NANDFC Buffer Memory Space

Table 20-3 shows the organization of the buffer memory space in the NANDFC.

Table 20-3. Data (Buffer) Organization in Memory

Address	Use	Access
0xCC00_0000–0xCC00_01FE	Main area Buffer 0	R/W
0xB800_0200–0xB800_03FE	Main area Buffer 1	R/W

Table 20-3. Data (Buffer) Organization in Memory (continued)

Address	Use	Access
0xB800_0400–0xB800_05FE	Main area Buffer 2	R/W
0xB800_0600–0xB800_07FE	Main area Buffer 3	R/W
0xB800_0800–0xB800_080E	Spare area Buffer 0	R/W
0xB800_0810–0xB800_081E	Spare area Buffer 1	R/W
0xB800_0820–0xB800_082E	Spare area Buffer 2	R/W
0xB800_0830–0xB800_083E	Spare area Buffer 3	R/W
0xB800_0840–0xB800_0BFE	Reserved	—
0xB800_0E00–0xB800_0E1C	Registers	R/W

20.5.1 Main and Spare Area Buffers

The main area buffer is a general data block. The spare area buffer is used for a variety of functions including Error Correction. The memory is organized in a different manner depending on if the Flash bus width is 8-bit or 16-bit. Table 20-4 shows an 8-bit organization, and Table 20-5 shows a 16-bit configuration. The host can use all of the spare area except for the BI and ECC code areas. For example, the AHB Host can write data to a reserved area of the spare area buffer upon program operation.

The NANDFC automatically generates ECC code for both main and spare data during NANDFC's data loading to NAND Flash, and NANDFC updates ECC code to NAND Flash spare area, but does not update ECC code to spare buffer. When programming/reading spare area, the spare area buffer number (SB0–SB3) must be selected using the RAM buffer address register (NFC_RAM_BUFF).

Table 20-4. Spare Area Buffer (with X8 I/O Bus)

Name	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
0xB800_0800 (SB0)	LSN(2nd)[1]								LSN(1st)[1]							
0xB800_0802 (SB0)	WC(1st)[2]								LSN(3rd)[1]							
0xB800_0804 (SB0)	BI[3]								WC(2nd)[2]							
0xB800_0806 (SB0)	ECC Code for Main area data (2nd)								ECC Code for Main area data (1st)							
0xB800_0808 (SB0)	ECC Code for Spare area data (1st)								ECC Code for Main area data (3rd)							
0xB800_080A (SB0)	Reserved								ECC Code for Spare area data (2nd)							
0xB800_080C (SB0)	Reserved								Reserved							
0xB800_080E (SB0)	Reserved								Reserved							

Table 20-4. Spare Area Buffer (with X8 I/O Bus) (continued)

Name	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
0xB800_0810–0xB800_081E (SB1)	SB1–SB3 have same assignment like SB0.															
0xB800_0820–0xB800_082E (SB2)																
0xB800_0830–0xB800_083E (SB3)																

1 LSN: Logical Sector Number

2 WC: Wrap Count and other bytes have the same wrap count information and are used as error correction for wrap count itself.

3 BI: Bad Block Information

Table 20-5. Spare Area Buffer (with X16 I/O Bus)

Name	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
0xB800_0800 (SB0)	LSN(2nd)[1]								LSN(1st)[1]							
0xB800_0802 (SB0)	WC(1st)[2]								LSN(3rd)[1]							
0xB800_0804 (SB0)	Reserved								WC(2nd)[2]							
0xB800_0806 (SB0)	ECC Code for Main area data (2nd)								ECC Code for Main area data (1st)							
0xB800_0808 (SB0)	ECC Code for Spare area data (1st)								ECC Code for Main area data (3rd)							
0xB800_080A (SB0)	BI[3]								ECC Code for Spare area data (2nd)							
0xB800_080C (SB0)	Reserved								Reserved							
0xB800_080E (SB0)	Reserved								Reserved							

Table 20-5. Spare Area Buffer (with X16 I/O Bus) (continued)

Name	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
0xB800_0810 – 0xB800_081E (SB1)	SB1–SB3 have same assignment like SB0.															
0xB800_0820 – 0xB800_082E (SB2)																
0xB800_0830 – 0xB800_083E (SB3)																

1 LSN: Logical Sector Number

2 WC: Wrap Count and other bytes have the same wrap count information and are used as error correction for wrap count itself.

3 BI: Bad Block Information

20.6 Memory Map and Register Definition

Section 20.7, “Register Descriptions” provides the detailed descriptions for all of the NANDFC registers.

20.6.1 Memory Map

Table 20-6 shows the NANDFC memory map.

Table 20-6. NANDFC Register Memory Map

Address	Register	Reset Value	Reset Value	Section/Page
0xB800_0E00 (NFC_BUFSIZ)	NAND FC Buffer Size Register	R	0x0000_0001	20.7.1/20-12
0xB800_0E02	Reserved	—	—	—
0xB800_0E04 (RAM_BUFFER_ADDRESS)	RAM Buffer Address Register	R/W	0x0000_0000	20.7.2/20-13
0xB800_0E06 (NAND_FLASH_ADD)	NAND Flash Address Register	R/W	0x0000_0000	20.7.2/20-13
0xB800_0E08 (NAND_FLASH_CMD)	NAND Flash Command Register	R/W	0x0000_0000	20.7.3/20-13
0xB800_0E0A (NFC_CONFIGURATION)	NANDFC Internal Buffer Lock Control	R/W	0x0000_0001	20.7.4/20-14
0xB800_0E0C (ECC_STATUS_RESULT)	Controller Status and Result of Flash Operation	R	0x0000_0000	20.7.5/20-14
0xB800_0E0E (ECC_RSLT_MAIN_AREA)	ECC Error Position Main Area Data Error x8 ECC Error Position Main Area Data Error x16	R	0x0000_0000	20.7.6/20-15

Table 20-6. NANDFC Register Memory Map (continued)

Address	Register	Reset Value	Reset Value	Section/Page
0xB800_0E10 (ECC_RSLT_SPARE_AREA)	ECC Error Position Spare Area Data Error x8 ECC Error Position Spare Area Data Error x16	R	0x0000_0000	20.7.7/20-16
0xB800_0E12 (NF_WR_PROT)	NAND Flash Write Protection	R/W	0x0000_0002	20.7.9/20-18
0xB800_0E14 (UNLOCK_START_BLK_ADD)	Address to Unlock in Write Protection Mode–Start	R/W	0x0000_0000	20.7.10/20-18
0xB800_0E16 (UNLOCK_END_BLK_ADD)	Address to Unlock in Write Protection Mode–End	R/W	0x0000_0000	20.7.11/20-19
0xB800_0E18 (NAND_FLASH_WR_PR_ST)	NAND Flash Write Protection Status	R/W	0x0000_0002	20.7.12/20-19
0xB800_0E1A (NAND_FLASH_CONFIG1)	NAND Flash Operation Configuration1	R/W	0x0000_0008	20.7.13/20-20
0xB800_0E1C (NAND_FLASH_CONFIG2)	NAND Flash Operation Configuration 2	R/W	0x0000_0000	20.7.14/20-21

20.6.2 Register Summary

Figure 20-3 shows the key to the register fields, and Table 20-7 shows the register figure conventions.

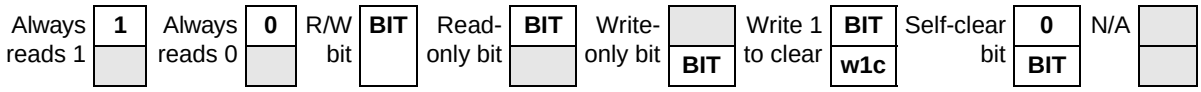


Figure 20-3. Key to Register Fields

Table 20-7. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.

Table 20-7. Register Figure Conventions (continued)

Convention	Description
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 20-8 shows the NANDFC register summary.

Table 20-8. NANDFC Register Summary

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xB800_0E00 (NFC_BUFSIZ)	R	0	0	0	0	0	0	0	0	0	0	0	0	BUFSIZE			
	W																
0xB800_0E04 (RAM_BUFFER_ADDRESS)	R	0	0	0	0	0	0	0	0	0	0	0	0	RBA			
	W																
0xB800_0E06 (NAND_FLASH_ADD)	R	ADD															
	W																
0xB800_0E08 (NAND_FLASH_CMD)	R	CMD															
	W																
0xB800_0E0A (NFC_CONFIGURATION)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	BLS	
	W																
0xB800_0E0C (ECC_STATUS_RESULT)	R	0	0	0	0	0	0	0	0	0	0	0	0	ERM		ERS	
	W																
0xB800_0E0E (ECC_RSLT_MAIN_AREA)	R	0	0	0	0	ECC Result 1									ECC Result2		
	W																
0xB800_0E10 (ECC_RSLT_SPARE_AREA)	R	0	0	0	0	0	0	0	0	0	0	0	ECC Result 4		ECC Result 3		
	W																
0xB800_0E12 (NF_WR_PROT)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	WPC		
	W																
0xB800_0E14 (UNLOCK_START_BLK_ADD)	R	USBA															
	W																
0xB800_0E16 (UNLOCK_END_BLK_ADD)	R	UEBA															
	W																
0xB800_0E18 (NAND_FLASH_WR_PR_ST)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	US	LS	LTS
	W																

Table 20-8. NANDFC Register Summary (continued)

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xB800_0E1A (NAND_FLASH_CONFIG1)	R	0	0	0	0	0	0	0	0	NF_CE	NFC_RST	NF_BIG	INT_MSK	ECC_EN	SP_EN	0	0
	W																
0xB800_0E1C (NAND_FLASH_CONFIG2)	R	INT	0	0	0	0	0	0	0	0	0	FDO			FDI	FADD	FCMD
	W	INT															

20.7 Register Descriptions

20.7.1 Internal SRAM SIZE (NFC_BUFSIZE)

This 16-bit read-only register contains the size of the internal SRAM installed in the IC. The bit assignments for the register are shown in Figure 20-4 and the field descriptions are shown in Table 20-9.

0xB800_0E00 (NFC_BUFSIZ) Access: User read-only

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	BUFSIZE			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 20-4. NFC_BUFSIZE Register

Table 20-9. NFC_BUFSIZE Register Field Descriptions

Name	Description
15–4	Reserved
3–0 BUFSIZE	Buffer Size. The size of internal RAM buffer. 0000 1 Kbyte 0001 2 Kbytes (Default) 0010–1111Reserved

20.7.1.1 Buffer Number for Page Data Transfer (RAM_BUFFER_ADDRESS)

RBA specifies which part of the RAM Buffer is transferred to/from flash memory. The bit assignments for the register are shown in Figure 20-5, and the field descriptions are shown in Table 20-10.

0xB800_0E04 (RAM_BUFFER_ADDRESS) Access: User Read/Write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	RBA			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 20-5. RAM Buffer Address Register

Table 20-10. RAM Buffer Address Field Descriptions

Field	Description
15–4	Reserved
3–0 RBA	RAM Buffer Address. Specifies the RAM buffer number to use for data transfers to or from the NAND FLASH device. 0000 1st internal RAM buffer 0001 2nd internal RAM buffer 0010 3rd internal RAM buffer 0011 4th internal RAM buffer

20.7.2 NAND Flash Address (NAND_FLASH_ADD)

The NAND FLASH Address (NAND_FLASH_ADD) register is a Read/Write register containing the address of the NAND Flash device that will be read, programmed or erased. The address in the NAND_FLASH_ADD register is written to the Flash device. The bit assignments for the register are shown in Figure 20-6, and the field descriptions are shown in Table 20-11.

0xB800_0E06 (NAND_FLASH_ADD) Access: User Read/Write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 20-6. NAND Flash Address Register

Table 20-11. NAND Flash Address Register Field Descriptions

Field	Description
15–0 ADD	NAND Flash Address. NAND Flash address which will be read, programmed, or erased. This address is written to the NAND Flash device.

20.7.3 NAND Flash Command (NAND_FLASH_CMD)

The bit assignments for the register are shown in Figure 20-7, and the field descriptions are shown in Table 20-12.

0xB800_0E08 (NAND_FLASH_CMD) Access: User Read/Write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CMD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 20-7. NAND_Flash_CMD Register

Table 20-12. NAND_Flash_CMD Register Field Descriptions

Field	Description
15–0 CMD	NAND Flash Command. This field contains the CMD that is written to the NAND Flash device.

20.7.4 NANDFC Internal Buffer Lock Control (NFC_CONFIGURATION)

The bit assignments for the register are shown in Figure 20-8, and the field descriptions are shown in Table 20-13.

0xB800_0E0A (NFC_CONFIGURATION) Access: User Read/Write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	BLS	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 20-8. NFC_Configuration Register

Table 20-13. NFC_Configuration Register Field Descriptions

Field	Description
15–2	Reserved
1–0 BLS	Buffer Lock Set. This field specifies the buffer lock status of first 2 pages in the internal buffer. The other 2 pages are always Unlocked. (For more details, see Section 20.9.3, “Write Protection Operation.”) 00 Locked 01 Locked (default) 10 Unlocked 11 Locked

20.7.5 Controller Status and Result of Flash Operation (ECC_STATUS_RESULT)

This register shows the number of errors in a page for Spare and the Main Area as a result of the ECC check upon a page read.The bit assignments for the register are shown in Figure 20-9, and the field descriptions are shown in Table 20-14.

0xB800_0E0C (ECC_STATUS_RESULT) Access: User read-only

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	ERM		ERS	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 20-9. ECC_Status_Result

Table 20-14. ECC_STATUS_RESULT Register Field Descriptions

Field	Description
15–4	Reserved
3–2 ERM	ECC Error for Main Area Data. This field shows the number of errors in a page as a result of the ECC check upon page read. The ECC algorithm of NANDFC does not correct if there are greater than two fault bits per page. It interprets any ECC error count greater than two as uncorrectable. 00 No error 01 1-bit Error (Correctable Error) 10 2-bits Error (Uncorrectable Error) 11 Reserved
1–0 ERS	ECC Error for Spare Area Data. This field shows the number of errors in a page as a result of the ECC check upon page read. The ECC algorithm of NANDFC does not correct if there are greater than two fault bits per page. It interprets any ECC error count greater than two as uncorrectable. 00 No error 01 1-bit Error (Correctable Error) 10 > 2-bits Error (Uncorrectable Error) 11 Reserved

20.7.6 ECC Error Position of Main Area Data Error x8 (ECC_RSLT_MAIN_AREA)

(NAND Flash X8 data bus case): This register contains the ECC error position address which is used to select the bit to repair in Main Area for 8-bit NAND Flash. The bit assignments for the register are shown in Figure 20-10, and the field descriptions are shown in Table 20-15.

0xB800_0E0E (ECC_RSLT_MAIN_AREA) Access: User read-only

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	ECC RESULT1									ECC RESULT2		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 20-10. ECC_RSLT_MAIN_AREA Register

Table 20-15. ECC_RSLT_MAIN_AREA Register Field Descriptions

Field	Description
15–12	Reserved
11–3 ECC RESULT1	ECC Result 1: ECC error position address which selects one of Main area data bytes (one of 512 Bytes).
2–0 ECC RESULT2	ECC Result 2: ECC error position address which selects one of the 8 data bits in the byte.

20.7.7 ECC Error Position of Main Area Data Error x16 (ECC_RSLT_MAIN_AREA)

(NAND Flash X16 data bus case): This register contains the ECC error position address which is used to select the bit to repair in Main Area for 16-bit NAND Flash. The bit assignments for the register are shown in Figure 20-11, and the field descriptions are shown in Table 20-16.

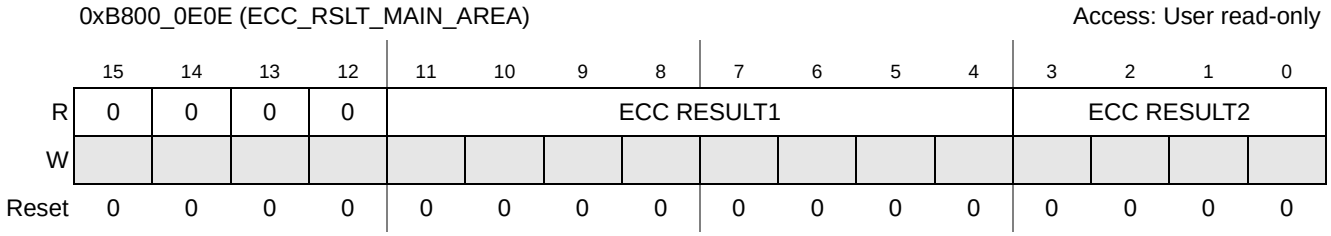


Figure 20-11. ECC_RSLT_MAIN_AREA Register

Table 20-16. ECC_RSLT_MAIN_AREA Register Field Descriptions

Field	Description
15–12	Reserved
11–4 ECC RESULT1	ECC Result 1: ECC error position address which selects one of the 16 data bits in the halfword
3–0 ECC RESULT2	ECC Result 2: ECC error position address which selects one of the 8 data bits in the byte.

20.7.8 ECC Error Position of Spare Area Data Error x8 (ECC_RSLT_SPARE_AREA)

(NAND Flash X8 data bus case): This register contains the ECC error position address which is used to select the bit to repair in Spare Area for 8-bit NAND Flash. The bit assignments for the register are shown in Figure 20-12, and the field descriptions are shown in Table 20-17.

0xB800_0E10 (ECC_RSLT_SPARE_AREA) Access: User read-only

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	ECC Result 4		ECC Result 3		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 20-12. ECC_Rslt_Spare_Area Register

Table 20-17. ECC_Rslt_Spare_Area Descriptions

Field	Description
15–5	Reserved
4–3 ECC RESULT4	ECC Result 4. ECC error position address which selects one of Logical Sector Number (3 Bytes)
2–0 ECC RESULT3	ECC Result 3. ECC error position address which selects one of 8 data bits in the byte.

20.7.8.1 ECC Error Position of Spare Area Data Error x16 (ECC_RSLT_SPARE_AREA)

(NAND Flash X16 data bus case): This register contains the ECC error position address which is used to select the bit to repair in Spare Area for 16-bit NAND Flash. The bit assignments for the register are shown in Figure 20-13, and the field descriptions are shown in Table 20-18.

0xB800_0E10 (ECC_RSLT_SPARE_AREA) Access: User read-only

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	Result4	Result 3			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 20-13. ECC_Rslt_Spare_Area Register

Table 20-18. ECC_Rslt_Spare_Area Descriptions

Field	Description
15–5	Reserved
4 ECC RESULT4	ECC Result 4. ECC error position address which selects one of Logical Sector Number (3 Bytes)
3–0 ECC RESULT3	ECC Result 3. ECC error position address which selects one of the 8 data bits in the byte.

20.7.9 NAND Flash Write Protection (NF_WR_PROT)

This register specifies the Protection command which the controller will perform: Lock, Unlock, or Lock Tight. The bit assignments for the register are shown in Figure 20-14, and the field descriptions are shown in Table 20-19.

0xB800_0E12 (NF_WR_PROT) Access: User Read/Write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	WPC		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Figure 20-14. NAND Flash Write Protection Register

Table 20-19. NAND Flash Write Protection Register Field Descriptions

Field	Description
15–3	Reserved
2–0 WPC	Write Protection Command. The Command field specifies the operation which the controller will perform. 100 Unlock NAND Flash block(s) according to given block address range 010 Lock all NAND Flash block(s) 001 Lock-tight locked blocks(s) (see also Section 20.9.3, “Write Protection Operation”)

20.7.10 Address to Unlock in Write Protection Mode—Start (UNLOCK_START_BLK_ADD)

Starting address of block memory in the NAND Flash that is unlocked in Write Protection mode. The bit assignments for the register are shown in Figure 20-15, and the field descriptions are shown in Table 20-20.

0xB800_0E14 (UNLOCK_START_BLK_ADD) Access: User Read/Write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	USBA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 20-15. Unlock_Start_Blk_Add Register

Table 20-20. Unlock_Start_Blk_Add Register Field Descriptions

Field	Description
15–0 USBA	Unlock Start Block Address. Starting address of block memory in the NAND Flash that is unlocked in Write Protection mode. For more details, see Section 20.9.3.4, “Write Protection Status.”

20.7.11 Address to Unlock in Write Protection Mode—End (UNLOCK_END_BLK_ADD)

Ending address of block memory in the NAND Flash that is unlocked in Write Protection mode. The bit assignments for the register are shown in Figure 20-16, and the field descriptions are shown in Table 20-21.

0xB800_0E16 (UNLOCK_END_BLK_ADD) Access: User Read/Write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	UEBA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 20-16. UNLOCK_END_BLK_ADD Register

Table 20-21. UNLOCK_END_BLK_ADD Register Field Descriptions

Field	Description
15–0 UEBA	Unlock End Block Address. Ending address of block memory in the NAND Flash that is unlocked in Write Protection mode. For more details, see Section 20.9.3.4, “Write Protection Status.”

20.7.12 NAND Flash Write Protection Status (NAND_FLASH_WR_PR_ST)

This register is a status register which read the NAND Flash Write Protection Status. Lock, Unlock or Lock Tight status.The bit assignments for the register are shown in Figure 20-17, and the field descriptions are shown in Table 20-22.

0xB800_0E18 (NAND_FLASH_WR_PR_ST) Access: User read-only

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	US	LS	LTS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Figure 20-17. NAND_FLASH_WR_PR_ST Register

Table 20-22. NAND_FLASH_WR_PR_ST Register Field Descriptions

Field	Description
15–3	Reserved
2 LS	Lock-tight Status: Indicates if any of the locked block(s) is (are) have a lock-tight status. 0 Locked block(s) is (are) not lock-tight. 1 Locked block(s) is (are) lock-tight.

Table 20-22. NAND_FLASH_WR_PR_ST Register Field Descriptions (continued)

Field	Description
1 LS	Locked Status. This bit indicate whether all NAND Flash blocks are in locked status or not. 0 Not all NAND Flash blocks are in locked status. 1 There are unlocked block(s) in NAND Flash.
0 US	Unlocked Status. This bit indicates whether there are any unlocked blocks in the NAND Flash. 0 There are no unlocked blocks in NAND Flash. 1 There are unlocked block(s) in NAND Flash.

20.7.13 NAND Flash Operation Configuration (NAND_FLASH_CONFIG1)

This register is a configuration register for the NAND Flash device to control the ECC Enable or Disable, Mask Interrupt. The bit assignments for the register are shown in Figure 20-18. and the field descriptions are shown in Table 20-23.

0xB800_0E1A (NAND_FLASH_CONFIG1) Access: User Read/Write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	$\overline{\text{NF\_CE}}$	NFC_RST	NF_BIG	INT_MSK	ECC_EN	SP_EN	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Figure 20-18. NAND_FLASH_CONFIG1 Register

Table 20-23. NAND_FLASH_CONFIG1 Register Field Descriptions

Field	Description
15–8	Reserved
7 $\overline{\text{NF\_CE}}$	NAND Flash Force CE. This bit forces the $\overline{\text{CE}}$ signal to the NAND Flash device to 0 when enabled. This bit allows a greater range of support new NAND Flash devices. 0 $\overline{\text{CE}}$ signal operates normally. 1 $\overline{\text{CE}}$ signal is asserted as long as this bit is set to 1.
6 NFC_RST	NANDFC Reset. This bit resets the NANDFC state machine. 0 Does not reset the NANDFC state machine 1 Resets the NANDFC state machine
5 NF_BIG	NAND FLASH Big Endian Mode. This bit enables Big Endian mode when writing from internal RAM to the NAND Flash device or reading from NAND Flash device to internal RAM. 0 Little Endian mode 1 Big Endian mode
4 INT_MSK	Mask interrupt Bit. This bit enables the interrupt by masking or not masking the interrupt bit. 0 Mask interrupt is disabled (interrupt enabled). 1 Mask interrupt is enabled (interrupt disabled).
3 ECC_EN	ECC operation Enable. This bit determines whether ECC operation is executed or bypassed. 0 ECC operation is bypassed. 1 ECC operation is executed.

Table 20-23. NAND_FLASH_CONFIG1 Register Field Descriptions (continued)

Field	Description
2 SP_EN	NAND Flash Spare Enable. This bit determines whether host reads/writes are to NAND Flash spare data only or NAND Flash main and spare data. 0 NAND Flash main and spare data is enabled. 1 NAND Flash spare only data is enabled.
1–0	Reserved

20.7.14 NAND Flash Operation Configuration 2 (NAND_FLASH_CONFIG2)

This register controls the NAND FLASH signals: CLE, ALE, WE, RE, $\overline{CE}$. and sets the interrupt after command completion. The bit assignments for the register are shown in Figure 20-19, and the field descriptions are shown in Table 20-24.

0xB800_0E1C (NAND_FLASH_CONFIG2) Access: User Read/Write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	INT	0	0	0	0	0	0	0	0	0	FDO			FDI	FADD	FCMD
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Figure 20-19. NAND_FLASH_CONFIG2 Register

Table 20-24. NAND_FLASH_CONFIG2 Register Field Descriptions

Field	Description
15 INT	Interrupt. This field determines the state of the interrupt output of the NAND Flash Controller. It is set by the controller when a basic operation is done. It is set cleared by the Host writing “0” to this field. (Host can also set this bit by writing “1” to this field). 0 Basic operation or boot loading is still running. 1 Basic operation or boot loading is done.
14–7	Reserved
6–3 FDO	NAND Flash Data Output. This bit enables NAND Flash Data Output. 001 One page data out[1] 010 NAND Flash ID data out 100 NAND Flash Status Register data out
2 FDI	NAND Flash Data Input. This field enables NAND Flash Data Input. 0 No NAND Flash data input operation 1 Enable NAND Flash data input operation
1 FADD	NAND Flash Address Input. This field enables NAND Flash Address Input. 0 No NAND Flash Address input operation 1 Enable NAND Flash Address input operation
0 FCMD	NAND Flash Command Input. This field enables the NAND Flash Command Input. 0 No NAND Flash Command input operation 1 Allow NAND Flash Command input operation

[1] Page size is determined by SP_EN register bit (main + spare or spare only). It is 528 bytes (main+spare) or 16 bytes (spare) regardless of the NFC_FMS setting.

NOTE

INT (Bit 15) reset value is 0, but soon after powerup it will change to 1. When performing boot from NAND FLASH, the INT bit will change from 0 to 1 after the transfer of boot code has been accomplished. For more information, see Section 20.8.1, "Modes of Operation."

NOTE

When the basic operation is completed, the FCMD/FADD/FDI/FDO bits change to LOW automatically.

Only one of the bit fields (FCMD/FADD/FDI/FDO) can be set at any given time.

20.8 Functional Description

This section provides the functional description for the NAND Flash Controller.

20.8.1 Modes of Operation

The NAND Flash Controller operating modes are described in this section. The operating mode is determined by four input lines: NFC_FMS, $\overline{\text{NF8BOOT}}$, $\overline{\text{NF16BOOT}}$, NF_16BIT_SEL, as shown in Table 20-25.

It is possible to configure the i.MX31 to boot from a NAND Flash device. For this to occur, *one* of the signals $\overline{\text{NF8BOOT}}$ or $\overline{\text{NF16BOOT}}$ must be low (0). If both of these signals are high, a boot from the NAND Flash device does not occur. If both of these signals are low, the situation is undefined, and should not be used.

The value of the NFC_FMS determines the page size of the NAND Flash–512 Bytes if NFC_FMS is low (0), and 2 Kbyte if NFC_FMS is high (1).

If booting from the NAND Flash device, the bus width is determined by the bus width used during boot.

If not booting from the NAND Flash device, the bus width is determined by the value of the NF_16BIT_SEL signal (0=8-bit bus, 1=16-bit bus).

Table 20-25. NAND FLASH Controller Operating Modes

NFC_FMS	$\overline{\text{NF8BOOT}}$	$\overline{\text{NF16BOOT}}$	NF_16BIT_SEL	Function
0	1	1	0	Do not Boot from NAND Flash. NAND Flash is configured to 8-bits I/O bus width and page size is 512 Bytes
0	1	1	1	Do not Boot from NAND Flash. NAND Flash is configured to 16-bits I/O bus width and page size is 512 Bytes.
0	1	0	X	Boot from 16-bit NAND Flash NAND Flash is configured to the same value as it booted from (16-bits) and page size is 512 Bytes.

Table 20-25. NAND FLASH Controller Operating Modes (continued)

NFC_FMS	$\overline{\text{NF8BOOT}}$	$\overline{\text{NF16BOOT}}$	NF_16BIT_SEL	Function
0	0	1	X	Boot from 8-bit NAND Flash NAND Flash is configured to the same value as it booted from (8-bits) and page size is 512 Bytes.
1	1	1	0	Do not Boot from NAND Flash. NAND Flash is configured to 8-bits I/O bus width and a page size is 2 Kbyte.
1	1	1	1	Do not Boot from NAND Flash. NAND Flash is configured to 16-bits I/O bus width and a page size is 2 Kbyte.
1	1	0	X	Boot from 16-bit NAND Flash NAND Flash is configured to the same value as it booted from (16-bits) and page size is 2 Kbyte.
1	0	1	X	Boot from 8-bit NAND Flash NAND Flash is configured to the same value as it booted from (8-bits) and page size is 2 Kbytes.
X	0	0	X	NOT DEFINED (do not use this setting)

20.8.2 Booting From a NAND Flash Device

Booting from NAND Flash device proceeds as follows[1]:

1. BOOTLOADER copies 1 page of 2 Kbyte or four pages of 528B (depending on the value of the NFC_FMS input) from the NAND Flash to the NANDFC internal RAM buffer. The transfer is done in the following order:
   - For NAND Flash with 528B page depth case:
     1st page read => 2nd page read => 3rd page read => 4th page read
   - For NAND Flash with 2 Kbyte page depth case:
     One page read
2. The AHB Host then reads (after exiting from reset state) the first code from the internal NAND Flash Controller RAM buffer.

1. A Boot from the NAND Flash device will only occur if one of the Boot inputs is asserted ($\overline{\text{NF8BOOT}}$ or $\overline{\text{NF16BOOT}}$ is low) at System Power-On reset (ipp_resetb rising).

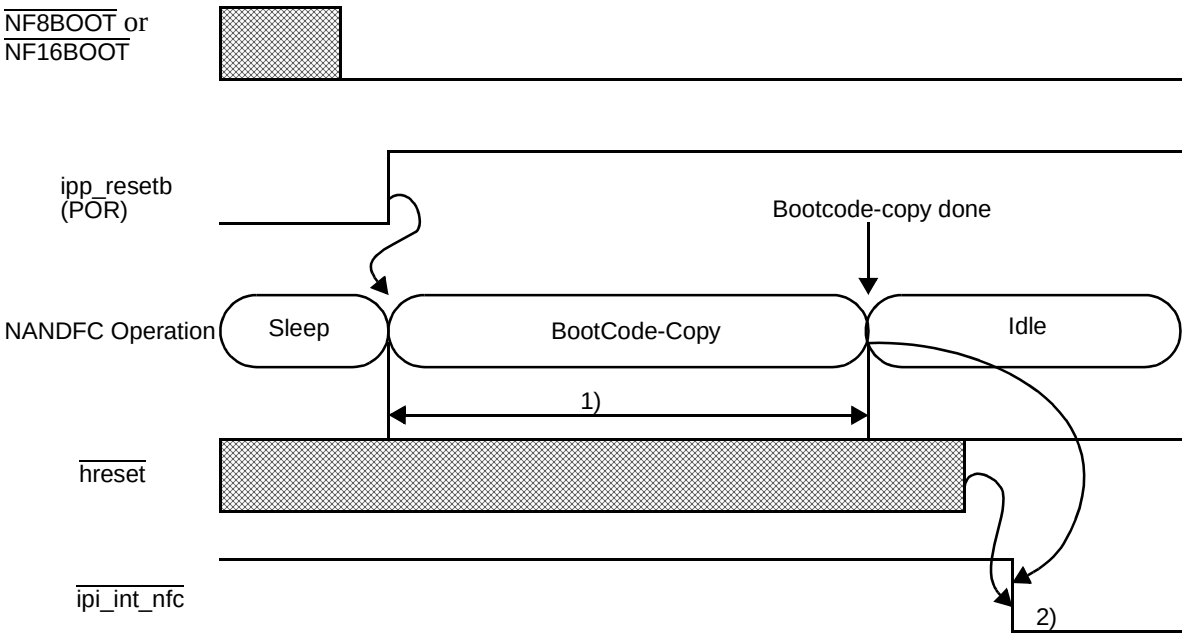


Figure 20-20. Boot Mode Operation

NOTE

2-Kbyte bootcode copy takes about 160 µs. Host must read Bootcode in RAM buffer (2 Kbyte) after Bootcode copy completion.

The Interrupt pin (ipi_int_nfc) goes high to low when the Bootcode-copy is completed, and upon the hreset rising edge. If hreset goes Low to High before the Bootcode-copy is done, the Interrupt pin (ipi_int_nfc) goes from High to Low as soon as Bootcode-copy is completed.

The interrupt can be relevant for cases of secured boot (booting from ROM and then enabling the NANDFC boot).

20.8.3 NAND Flash Control

The NAND Flash Control generates all the control signals that control the NAND Flash: CE (Flash Chip Enable), RE (Read Enable for read operations), WE (Flash Write Enable), CLE (Flash Command Latch Enable), ALE (Flash Address Latch Enable). It monitors the R/nB (Flash Ready/Busy indication) signal to check if the NAND Flash is in the middle of operation.

The BOOTLOADER is part of the NAND Flash Control Block.

Figure 20-21, Figure 20-22, and Figure 20-23 show the NAND Flash read, program, and erase timing diagrams.

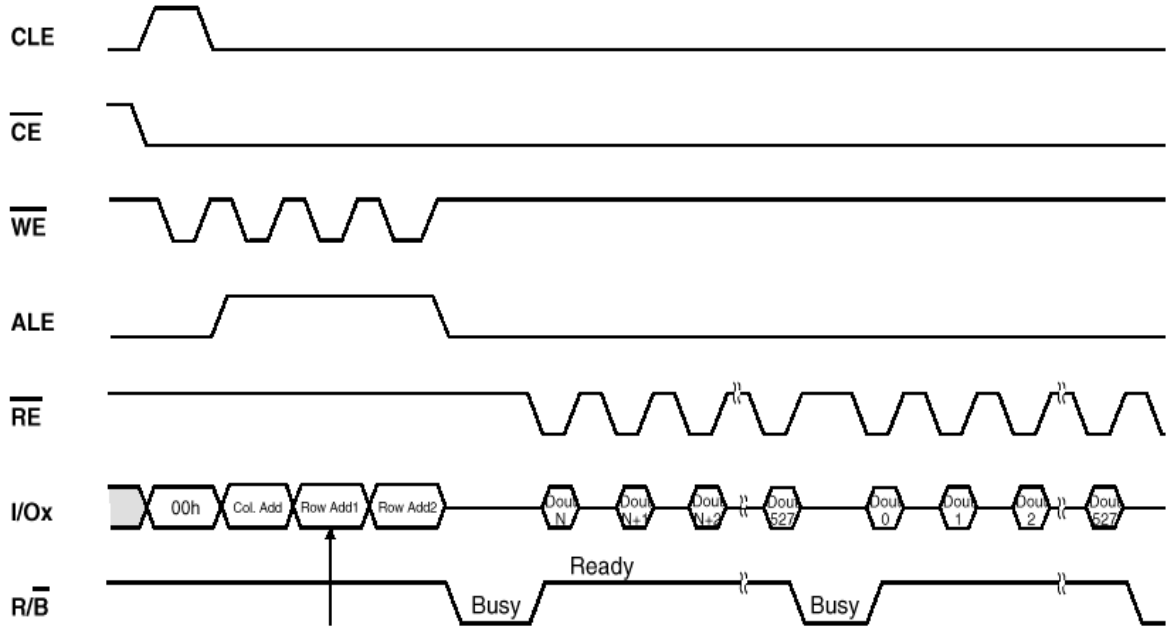


Figure 20-21. Read Operation

PAGE PROGRAM OPERATION

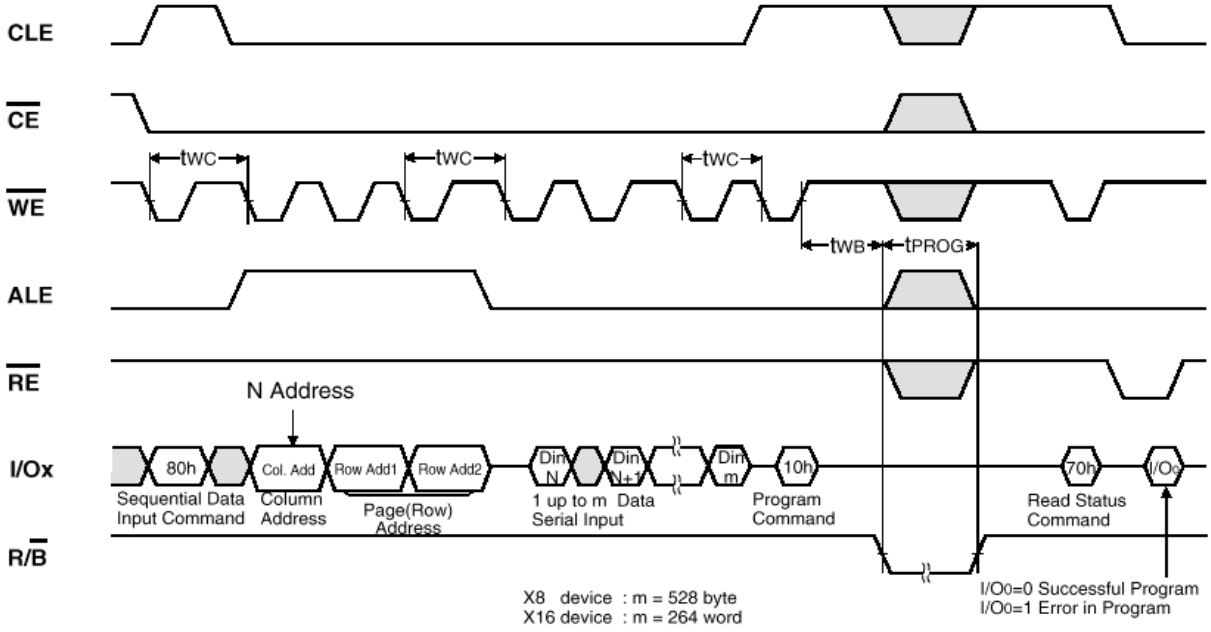


Figure 20-22. Program Operation

BLOCK ERASE OPERATION (ERASE ONE BLOCK)

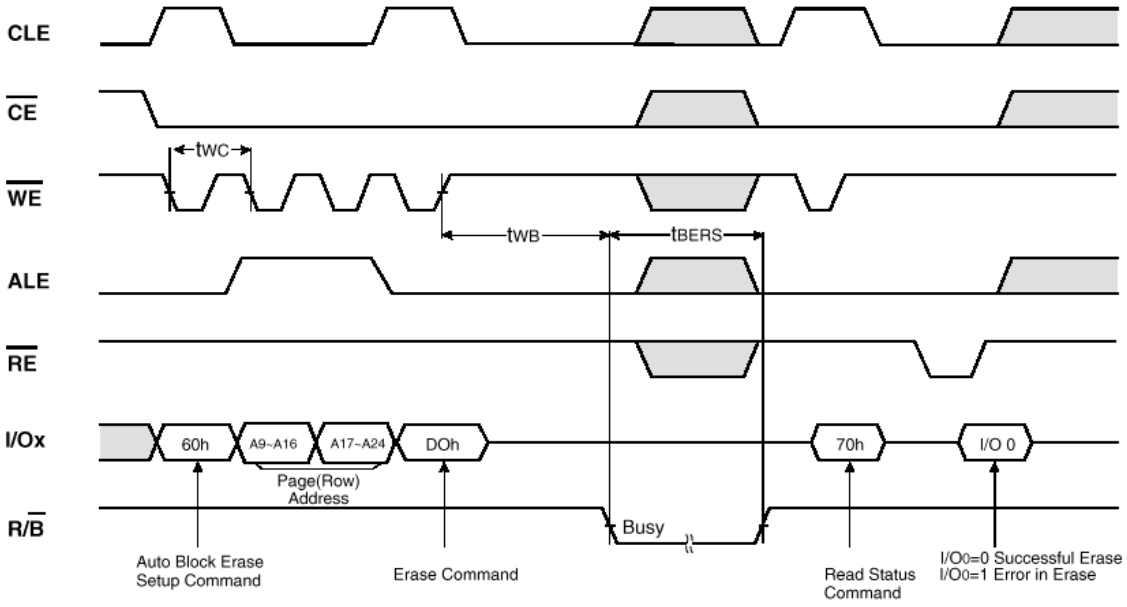


Figure 20-23. Erase Operation

20.8.4 ECC Control

The ECC (Error Code Correction) block is responsible for correcting one bit per read, and for detecting if there is more then one bit with an error.

When the NANDFC accesses the NAND Flash device for Program operation, it generates code (24 bits for Main area data and 10 bits for Spare area data). When it accesses the NAND Flash device for a Read operation, it generates ECC code, and indicates how many errors were detected, and their positions in addition to correcting 1 error bit. The ECC code is updated by the NANDFC automatically. After a Read operation, the AHB Host can know whether there is an error or not by reading the status register (see the ECC_Status_Result register). The indication in the status register is either:

- No error
- 1 bit error (correctable), or
- Greater than 2 bit errors (uncorrectable)

Because the generated ECC code at read/program operation is not updated to the internal RAM buffer, but is updated to the NAND Flash spare area upon program operation, the AHB Host can read generated ECC code only from NAND Flash spare area.

20.8.5 Address Control

This module is responsible for address control and generation. It defines the RAM buffer Address Generation (RAM buffer Address for Data In/ Data Out).

It generates and takes into account the Lock State Sequence (For more details, see Section 20.9.3, “Write Protection Operation”) and therefore contains the Flash Memory Lock Address Comparator, and RAM buffer Lock Address Comparator, which are used to determine if this area is protected or not. It also generates the RAM buffer Address for Boot Load and RAM buffer Address for Error Correction.

20.8.6 RAM Buffer (SRAM)

The internal RAM Buffer is a 2112-byte single Port RAM buffer which is a synchronous high performance design. This memory has 528 words of 32-bits each, from which 512 words are used for the main buffer and the remaining 16 Words are allotted to a spare area, which is used for ECC (Error Correction). This Memory is used as a Bootram during boot from the NAND Flash device, and as a buffer at normal operation.

20.8.7 Registers (Command, Address, Status, and Others.)

This module contains 15 registers of 16-bits each. Using these registers, the AHB Host can control the NANDFC, read status on various operations and perform a direct access of commands, and insert addresses to the NAND Flash device. For more details, refer to Section 20.6.1, “Memory Map.”

20.8.8 Read and Write Control

The Read and Write Control Block contains a connection to the Internal bus (which is connected to the Internal RAM buffer and the registers). This Internal bus is responsible for the Internal Synchronous read and Asynchronous write. It supports Burst Read Latency (3, 4, 5, 6, 7 cycle) and Synchronous Read Burst Length (4, 8, 16, 32, continuous word).

It is also responsible for RAM buffer Control and Register Control, RAM buffer Lock Control and Address and Data latches.

20.8.9 Data Output Control

This module defines Data output of 16-bits to the Internal bus which is driven to the AHB interface. It includes RAM buffer Data output, Register Data output and RAM buffer Synchronization for the read mode pipe line.

20.8.10 Host Control

This module defines Host control which is connected to the AHB Interface though the internal bus. It detects Chip Enable and controls the Reset and Output Enable, and generates the $\overline{\text{SRAM\_WE}}$ signal.

20.8.11 AHB Bus Interface

The AHB bus interface is an adapter between ABMA AHB bus and the internal bus.

On the AHB bus side it supports a 16-bit and 32-bit bus width, burst and non burst operations. On the internal bus side it supports 32-bit bus width with a Synchronous Burst Read and an Asynchronous Random Write. It also supports Programmable Read latency for the internal bus (this also effects the latency on the AHB bus).

20.8.11.1 Big/Little Endian

The AHB bus interface supports both Big and Little Endian data types. The nfc_endian pin controls the Endian mode. Only the AHB side is controlled by the nfc_endian pin; the NAND Flash device side is always in Little Endian mode.

20.8.11.2 Burst Access Support

When a data transaction from the AHB is a burst it will create a synchronous burst read on the internal bus for read cycles, and several asynchronous random writes for write cycles. Table 20-26 lists NANDFC supported access burst types.

Table 20-26. NAND Flash Burst Access Support

HBURST	Burst Type	Supported	Description
000	SINGLE	Yes	Single transfer
001	INCR	Yes	Incrementing burst
010	WRAP4	No	4-beat wrapping burst
011	INCR4	Yes	4-beat incrementing burst
100	WRAP8	No	8-beat wrapping burst
101	INCR8	Yes	8-beat incrementing burst
110	WRAP16	No	16-beat wrapping burst
111	INCR16	Yes	16-beat incrementing burst

NOTE

NANDFC supports bursts of 16/32-bit words only. Bursts of byte words (8-bits) is not supported.

20.8.12 I/O Pins Sharing

The NAND Flash Controller has logic that allows it to share I/O pins with pins of another memory controller. The NAND Flash Controller's state machine halts when a request to free the pins is asserted. The NAND Flash signals when it finishes the existing transfer allowing the other memory controller to be able to control them.

Because the NAND Flash accesses are long and relatively slow, the priority is given to the other memory controller and the NAND Flash Controller will have to wait till the pins are free before it can continue with its accesses.

One example for this pin muxing is sharing the 16 I/O pins of the NAND Flash Controller with the Data pins of the WEIM when interfacing to the PSRAM.

20.9 Initialization/Application Information

This section describes how to operate the NANDFC using its registers and its interrupts, and is divided into the following subjects:

- Normal operation—To operate a NAND Flash device using the NANDFC the user should use the instructions in Section 20.9.1, "Normal Operation."
- ECC operation—ECC operation is used when an error is detected.
- Write protection operation (both to the internal memory and the flash device)—Write protection is used when the programmer wishes to protect part of the NAND Flash device memory from being written except in certain cases. There are two levels of protection: software (for frequently-changed memory locations), and hardware (for memory locations whose contents are rarely changed).

20.9.1 Normal Operation

"Normal Operations" are composed of fundamental building block operations shown in Section 20.9.1.1, "Fundamental Building Block Operations," in addition to specific operations, as shown in the flowcharts in Section 20.9.1.2, "Read NAND Flash ID Operation" to Section 20.9.1.7, "HOT Reset (Controller and NAND Flash Reset)."

20.9.1.1 Fundamental Building Block Operations

20.9.1.1.1 Preset Operation

Figure 20-24 is a flowchart of the preset operation.

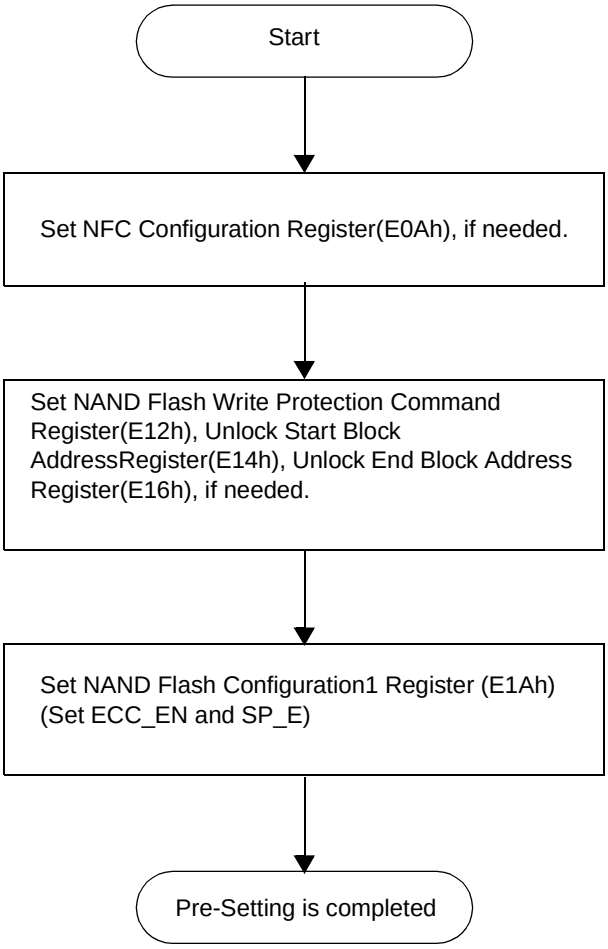


Figure 20-24. Flowchart of Preset Operation

20.9.1.1.2 NAND Flash Command Input Operation

Figure 20-25 is a flowchart of the NAND Flash Command Input.

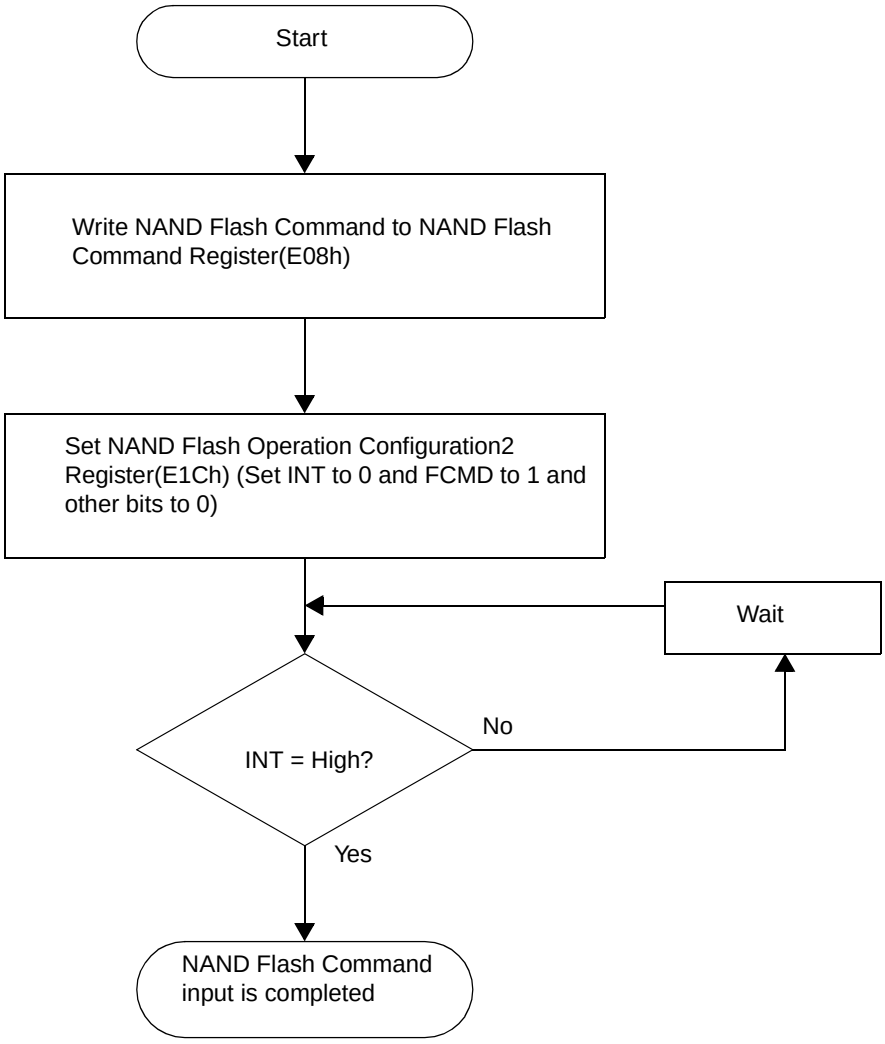


Figure 20-25. Flowchart of NAND Flash Command Input Operation

20.9.1.1.3 NAND Flash Address Input Operation

Figure 20-26 is a flowchart of the NAND Flash Address Input Operation.

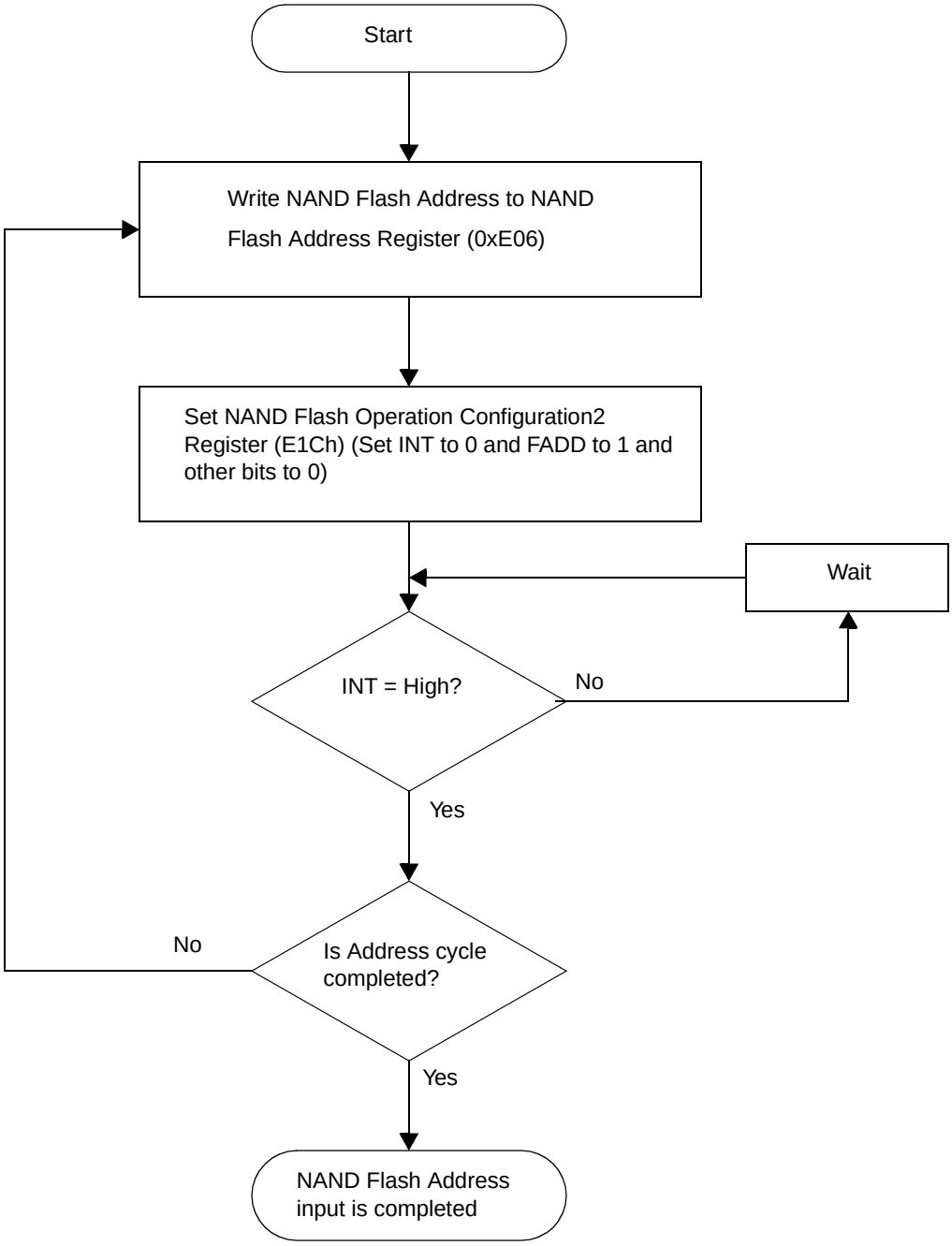


Figure 20-26. Flowchart of NAND Flash Address Input Operation

20.9.1.1.4 NAND Flash Data Input Operation

Figure 20-27 is a flowchart of the NAND Flash Data Input Operation.

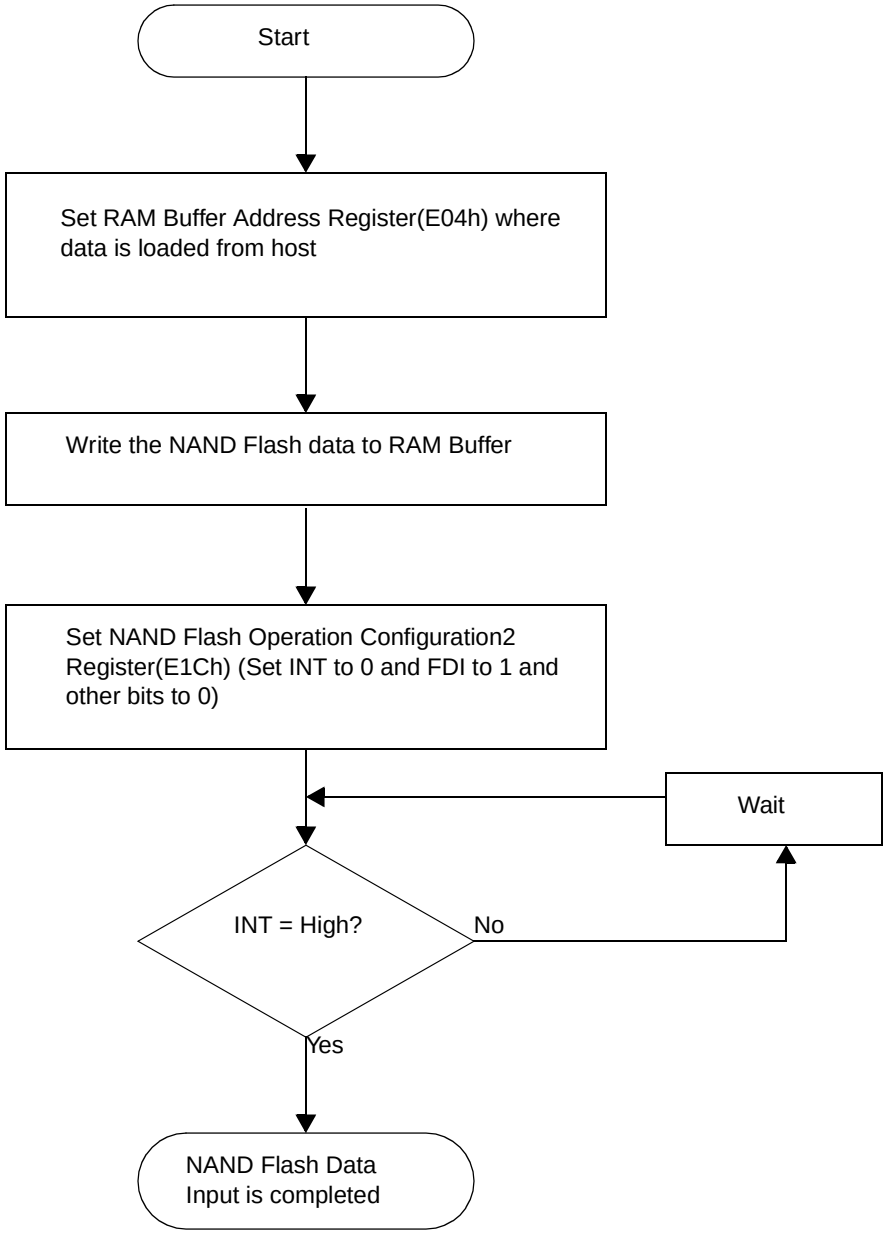


Figure 20-27. Flowchart of NAND Flash Data Input Operation

20.9.1.1.5 NAND Flash Data Output Operation

Figure 20-28 is a flowchart of the NAND Flash Data Output Operation.

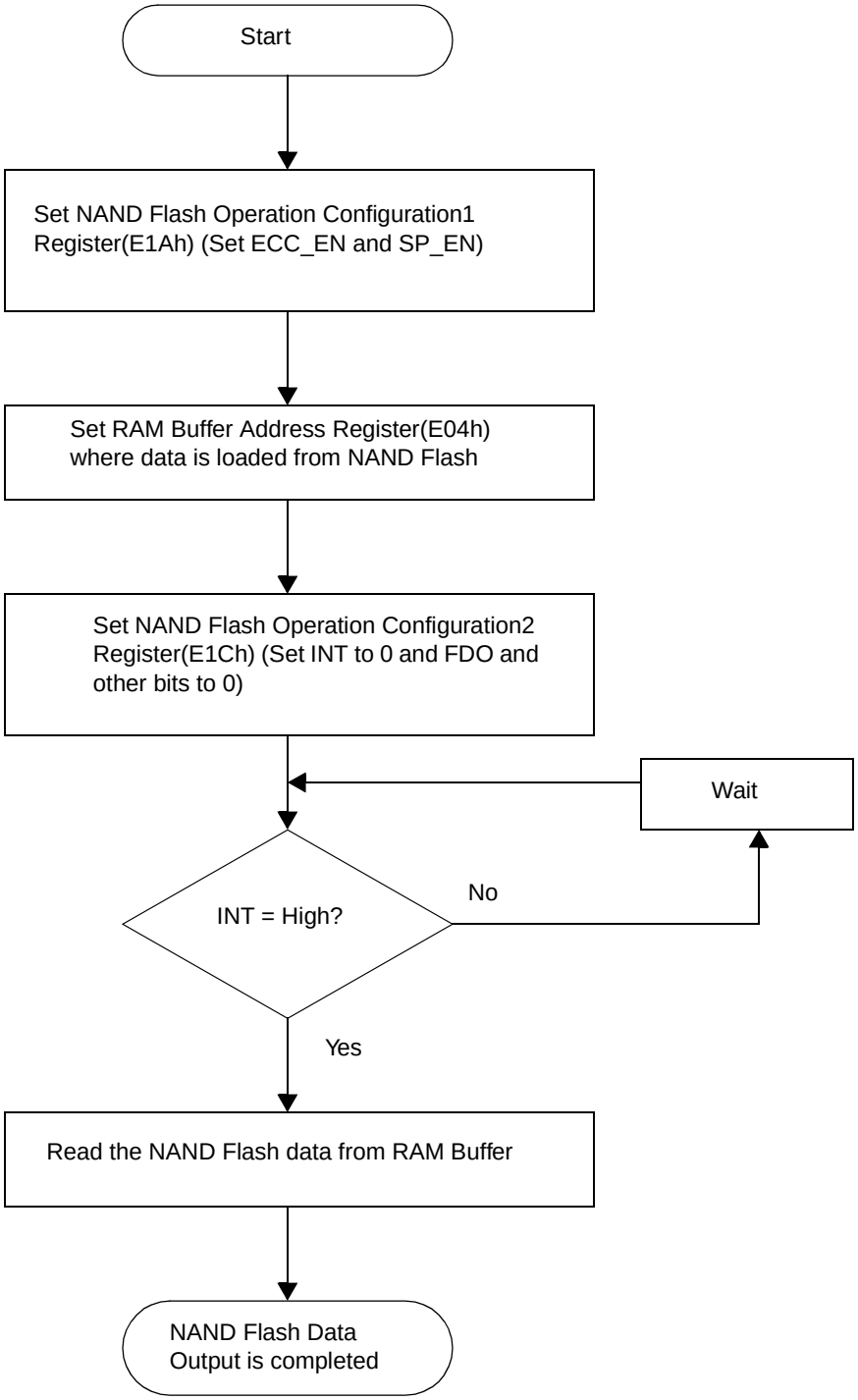


Figure 20-28. Flowchart of NAND Flash Data Output Operation

20.9.1.2 Read NAND Flash ID Operation

Figure 20-29 is a flowchart of the Read NAND Flash ID Operation.

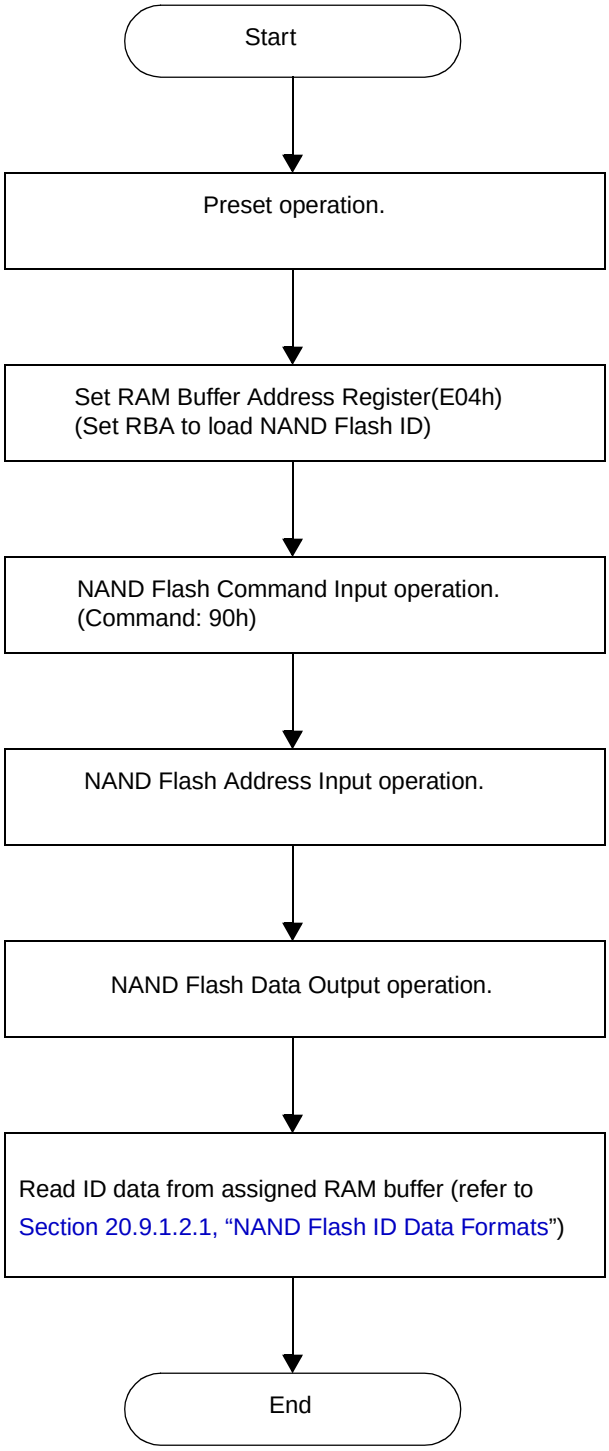


Figure 20-29. Flowchart of Read NAND Flash ID Operation

20.9.1.2.1 NAND Flash ID Data Formats

The format of NAND Flash ID data stored in the RAM buffer (for X8 org. NAND Flash) is shown in Figure 20-30.

RAM Buffer of RBA address	1st halfword				2nd halfword				3rd halfword							
	1st byte of ID		2nd byte of ID		3rd byte of ID		4th byte of ID		5th byte of ID		6th byte of ID					
	LSB			MSB												

Figure 20-30. NAND Flash ID Data Format (x8)

The format of NAND Flash ID data stored in the RAM buffer (for X16 org. NAND Flash) is shown in Figure 20-31.

RAM Buffer of RBA address	1st halfword				2nd halfword				3rd halfword							
	1st byte of ID		XXh		2nd byte of ID		XXh		3rd byte of ID		XXh					
	LSB			MSB												

RAM Buffer of RBA address	4th halfword				5th halfword				6th halfword							
	4th byte of ID		XXh		5th byte of ID		XXh		6th byte of ID		XXh					
	LSB			MSB												

Figure 20-31. NAND Flash ID Data Format (x16)

20.9.1.3 NAND Flash Status Read Operation

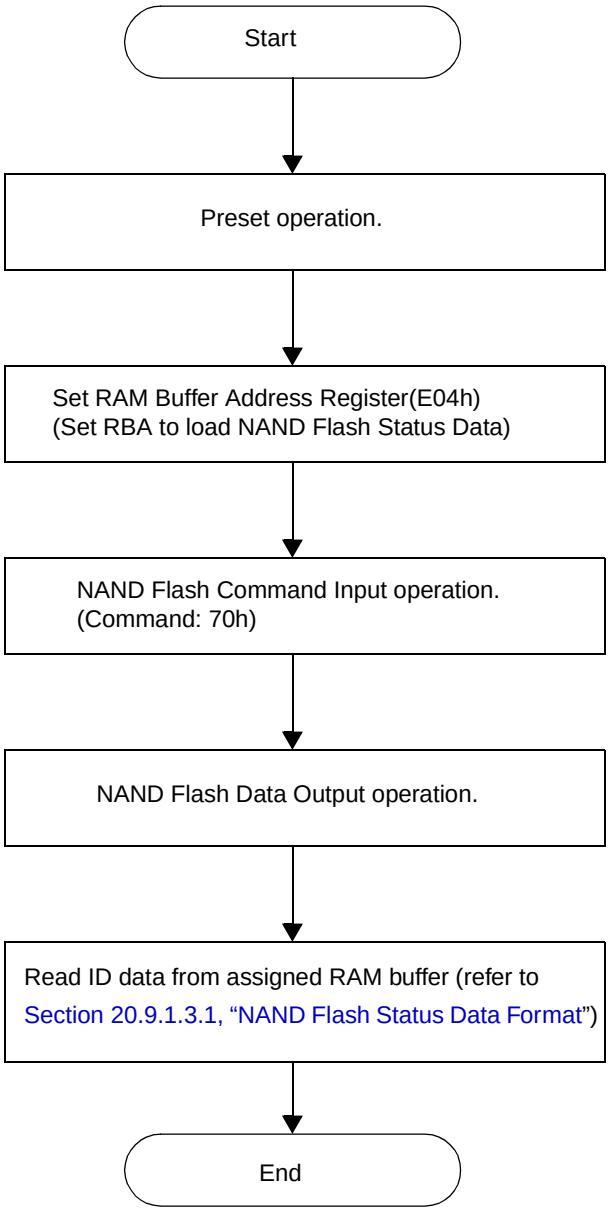


Figure 20-32. Flowchart of Read NAND Flash Status Operation

20.9.1.3.1 NAND Flash Status Data Format

The assignment of NAND Flash Status data stored in the RAM buffer (for both X8/X16 organization NAND Flash) is shown in Figure 20-33.

RAM Buffer of RBA address	1st halfword				- - - - - - - -
	1st Byte of Status		XXh		
	LSB			MSB	

Figure 20-33. NAND Flash Status Data Format

20.9.1.4 Read NAND Flash Data Operation

Figure 20-34 shows a flowchart of a read NAND Flash data operation.

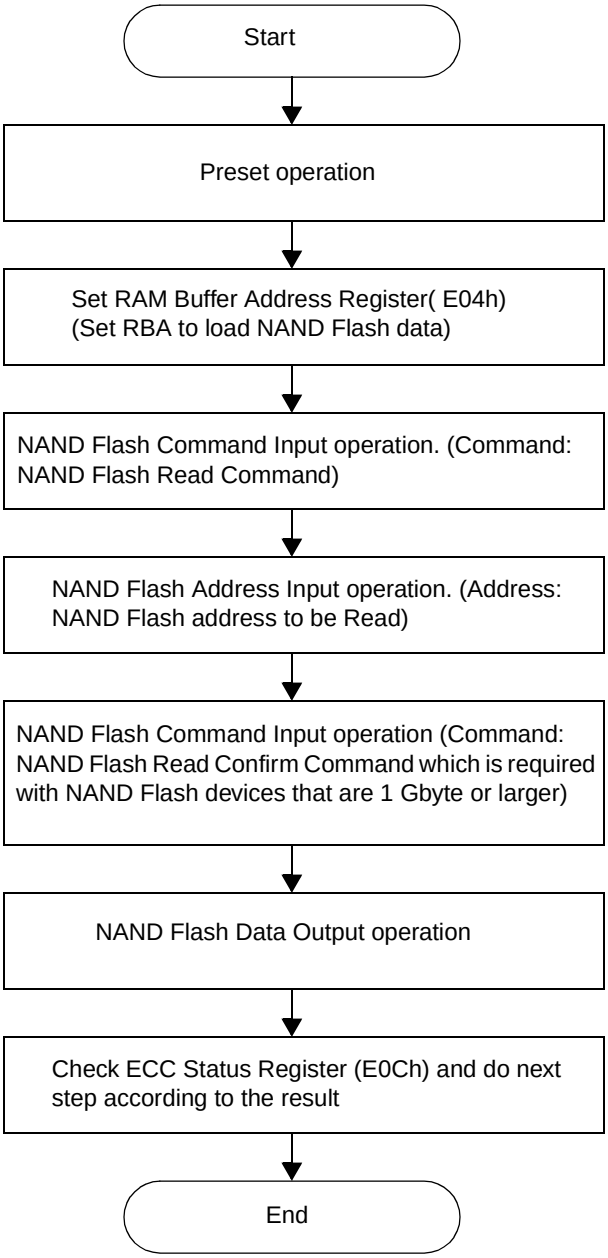


Figure 20-34. Flowchart of Read NAND Flash Data Operation

20.9.1.5 Program NAND Flash Data Operation

Figure 20-35 shows a flowchart of a program NAND Flash data operation.

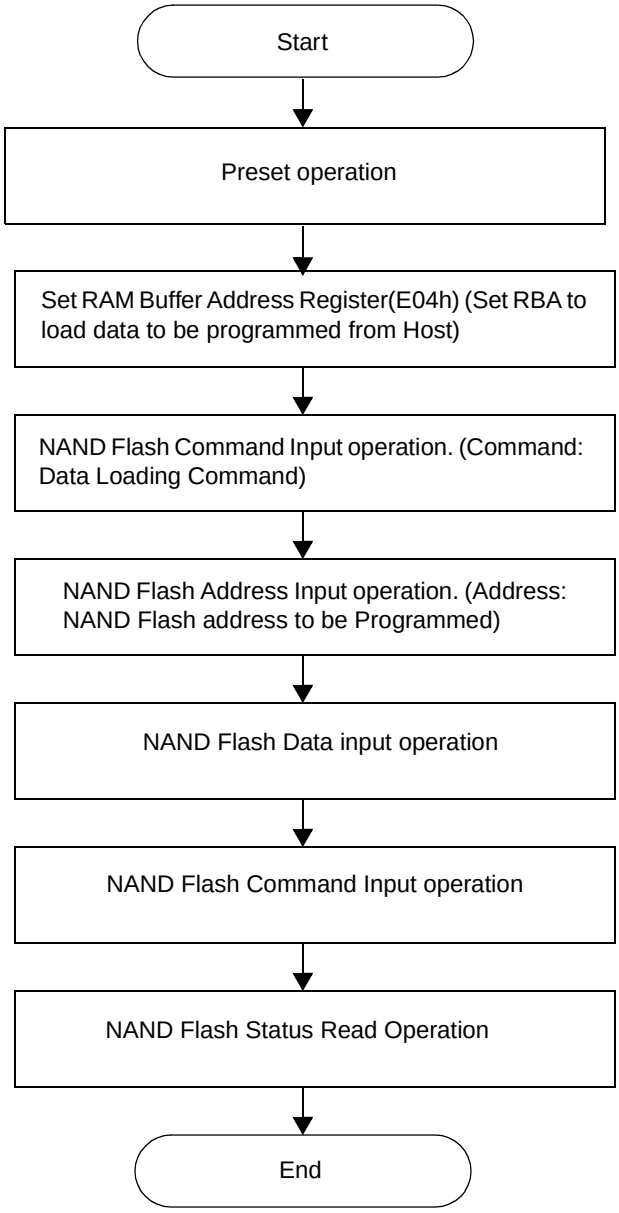


Figure 20-35. Flowchart of Program NAND Flash Data Operation

20.9.1.6 Erase NAND Flash Data Operation

Figure 20-36 shows a flowchart of an erase NAND Flash operation.

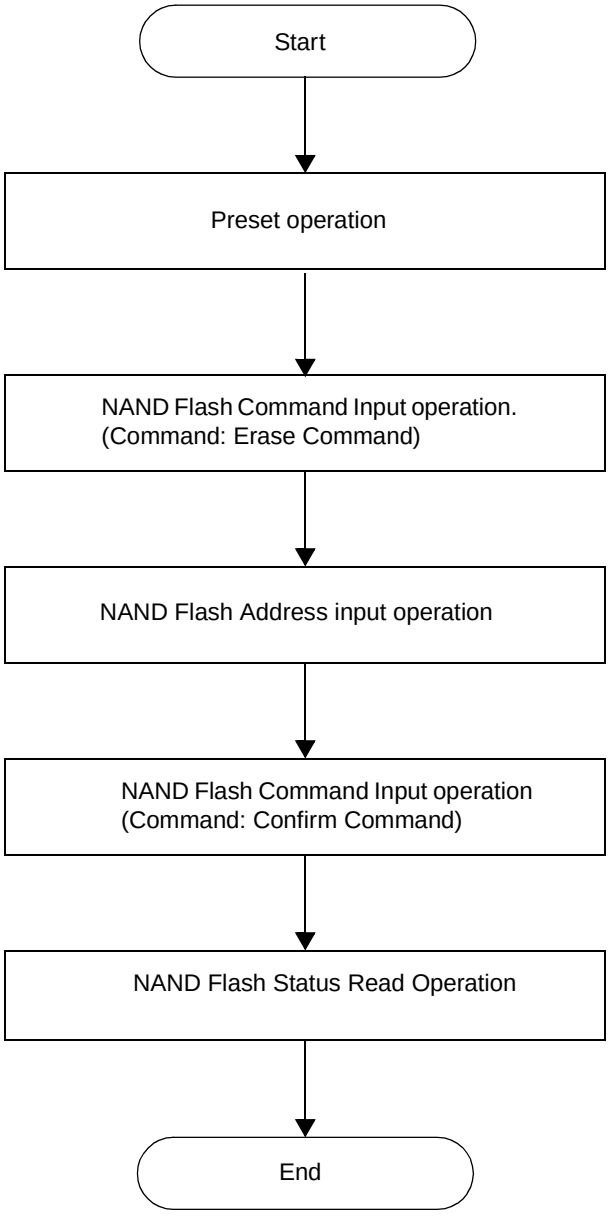


Figure 20-36. Flowchart of Erase NAND Flash Operation

20.9.1.7 HOT Reset (Controller and NAND Flash Reset)

A warm (or “hot”) reset causes the NANDFC and the NAND Flash device cease their current operation and causes the internal registers to revert to their default state. Figure 20-37 shows a flowchart of a hot reset operation.

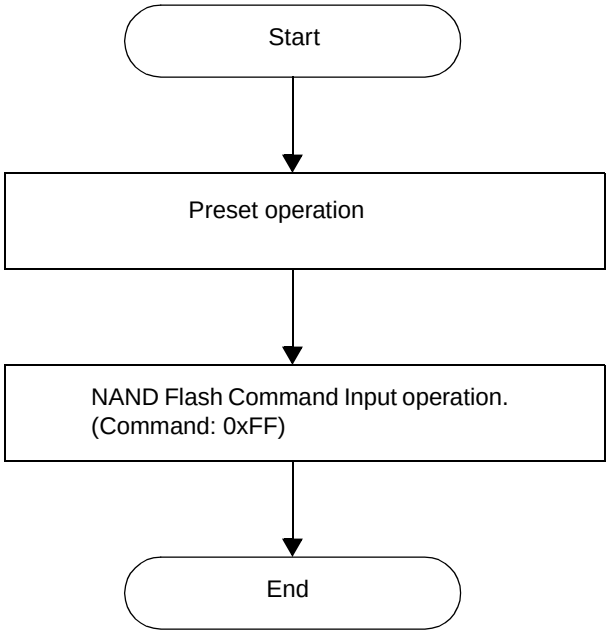


Figure 20-37. Flowchart of a Hot Reset Operation

20.9.2 ECC Operation

20.9.2.1 ECC Normal Operation

When the NANDFC accesses the NAND Flash device for Program operation, it generates ECC code (24bits for Main area data and 10 bits for spare area data).

When the NANDFC accesses the NAND Flash device for a Read operation, it generates ECC code, detects the error number and position and corrects a 1-bit error, if applicable. Table 20-27 shows the ECC code assignment of the NAND Flash spare area. This ECC code is updated by NANDFC automatically.

After the Read operation, the AHB Host can know whether there is error or not by reading the status register (see the ECC_Status_Result register). Error type can be: a) no error, b) 1-bit error (correctable), or c) 2- or more bit error (uncorrectable).

Because generated ECC code at read/program operation is not updated to the internal buffer RAM, but is updated to the NAND Flash spare area immediately upon program operation, the AHB Host can read generated ECC code only from NAND Flash spare area.

20.9.2.2 ECC Bypass Operation

In ECC bypass operation, NANDFC generates an ECC result which indicates error position (refer to Table 20-27), but does not correct the error. After a Read operation, the host can know whether there is an error or not by reading the status register (refer to Table 20-14).

Error type is divided into no error, 1bit error (correctable), or 2 bits error (uncorrectable). In 1-bit error case, the Host can correct the error by itself after reading the ECC Result register (see the ECC_Status_Result register).

Table 20-27. ECC Code/Result Readability

Operation	Read Operation		Program Operation	
	ECC Code from spare area buffer	ECC Result from register	ECC Code from spare area buffer	ECC Result from register
ECC operation	Invalid (Pre-written ECC code [1])	Valid	Invalid (old data [2])	—
ECC bypass	Invalid (Pre-written ECC code)	Valid	Invalid (old data)	—

1 Pre-written ECC code: ECC code which is previously written to NAND Flash spare area in program operation.

2 Old data: ECC code is not updated to spare buffer, so ECC code placement of spare buffer remains old data.

20.9.2.3 How to Operate the ECC

To generate ECC and carry out the correction by the NANDFC:

— Program with ECC operation/Read with ECC operation

To generate ECC by the NANDFC and carry out the correction by the AHB Host:

— Program with ECC operation/Read without ECC operation

NOTE

The AHB Host can read the ECC results from the ECC status register (see the ECC_Status_Result register) after a read operation in both ECC Normal operation and ECC Bypass cases. When NANDFC reads NAND Flash data, the ECC code for read data is not updated into RAM buffer.

20.9.3 Write Protection Operation

The NANDFC offers a software write protection feature, and a hardware write protection feature. Both are described in this section.

20.9.3.1 Write Protection for RAM Buffer (LSB 1 Kbyte)

The NANDFC offers a software write protection feature for the first 2pages (main + spare area data) of the RAM buffer, which protects RAM buffer data. This write protection is carried out by setting the WPC bit of the NF_WR_PROT register.

The default state is locked state, and the first 2 pages go to this state after a cold or warm reset.

Write protection availability for main/spare memory regions in the RAM buffer are described on Table 20-28. A state diagram of RAM buffer write protection is shown in Figure 20-38.

Table 20-28. Write Protection for Main/Spare RAM Buffer

Main Area	Spare Area	
1st page RAM buffer		Write Protection Available
2nd page RAM buffer		
3rd page RAM buffer		Write Protection not available
4th page RAM buffer		

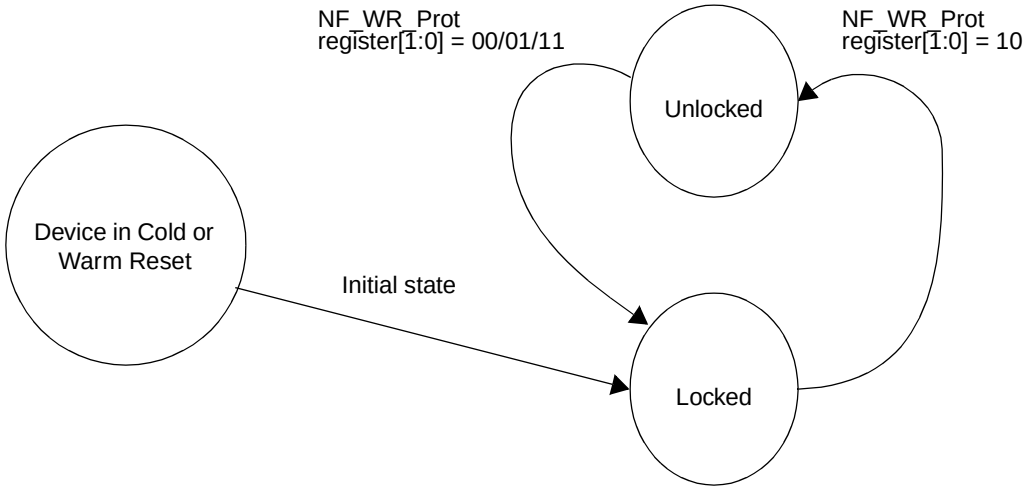


Figure 20-38. State Diagram of RAM Buffer Write Protection

20.9.3.2 Write Protection Modes

NANDFC offers both hardware and software Write Protection options for the NAND Flash device. The software Write Protection feature is used by executing the LOCK BLOCK command or LOCK-TIGHT BLOCK command, and the hardware Write Protection feature is used by executing a cold or warm reset. The WP signal is asserted only upon POR.

20.9.3.3 Write Protection Commands

There are two write protection states: Locked and Lock-Tight:

- Locked state means that memory block in question is write protected (it cannot be written to), but the Unlock command can “un”-lock it. Useful for frequently changed memory blocks.
- Lock-Tight state is a higher level of protection, and means that the memory block in question is write protected, but the UNLOCK command cannot unlock it. Useful for memory blocks whose contents are rarely changed.

The following summarizes the locking functionality:

— All blocks power-up in a locked state. The UNLOCK command can unlock these blocks.

— The LOCK-TIGHT BLOCK command locks blocks and prevents it (them) from being unlocked.
  – LOCK-TIGHT *state can be reverted to locked state only when Cold/Warm reset is executed.*
— Writing to the unlock start/end address registers (Unlock_Start_Blk_Add and Unlock_End_Blk_Add) while the NANDFC is in the Lock-Tight state does not affect the unlock address.

20.9.3.4 Write Protection Status

The current Write Protection status of the NANDFC can be read in NAND Flash Write Protection status register (NAND_FLASH_WR_PR_ST). There are three bits: US, LS, and LTS, which are not cleared by a hot reset. These Write Protection status bits are updated as soon as the Write Protection command is entered.

Figure 20-39 shows a state diagram for the write protection of the NANDFC.

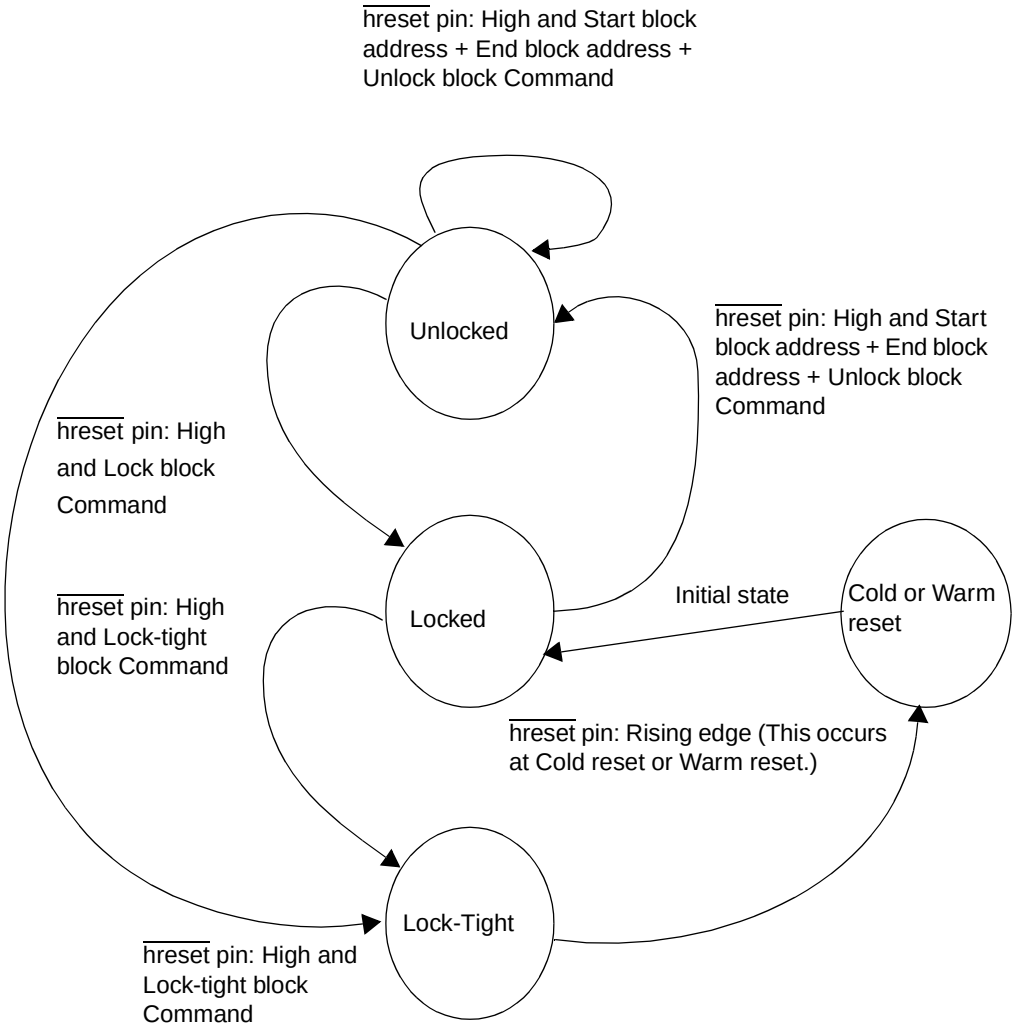


Figure 20-39. State Diagram of NAND Flash Write Protection

20.9.3.4.1 Lock Sequence

The following describes the “lock” sequence:

1. Command Sequence: LOCK BLOCK Command (0x02)
2. All blocks default to locked after initial Cold reset or Warm reset
3. Locking some of the blocks is not available; all memory blocks are locked upon reset.
4. Unlocked memory blocks can be locked by using the LOCK BLOCK command. The status of a locked memory block can be changed to “unlocked” or “lock-tight” using the appropriate software commands.

20.9.3.4.2 Unlock Sequence

The following describes the “unlock” sequence:

1. Command Sequence: Start block address + End block address + UNLOCK BLOCK Command (04h)
2. Unlocked blocks can be programmed or erased.
3. The status of an unlocked block can be changed to *locked* or *lock-tight* using the appropriate software commands.
4. Only one sequential area can be released to unlocked state from locked state; Unlocking multi areas is not available.

20.9.3.4.3 Lock-Tight Sequence

The following describes the “lock-tight” sequence:

1. Only locked blocks can be “locked-tight” by the LOCK-TIGHT BLOCK command.
2. Command Sequence: LOCK-TIGHT BLOCK Command (0x01)
3. Unlocking multi area is not available.
4. Lock-tight blocks revert to the locked state at Cold/Warm reset.

20.9.4 Memory Configuration Examples

The following figures show memory connection for various bit values: An 8-bit configuration example is shown in Figure 20-40, a 16-bit configuration example is shown in Figure 20-41, and some examples of NANDFC Pin configurations for various NAND Flash devices are shown in Table 20-29.

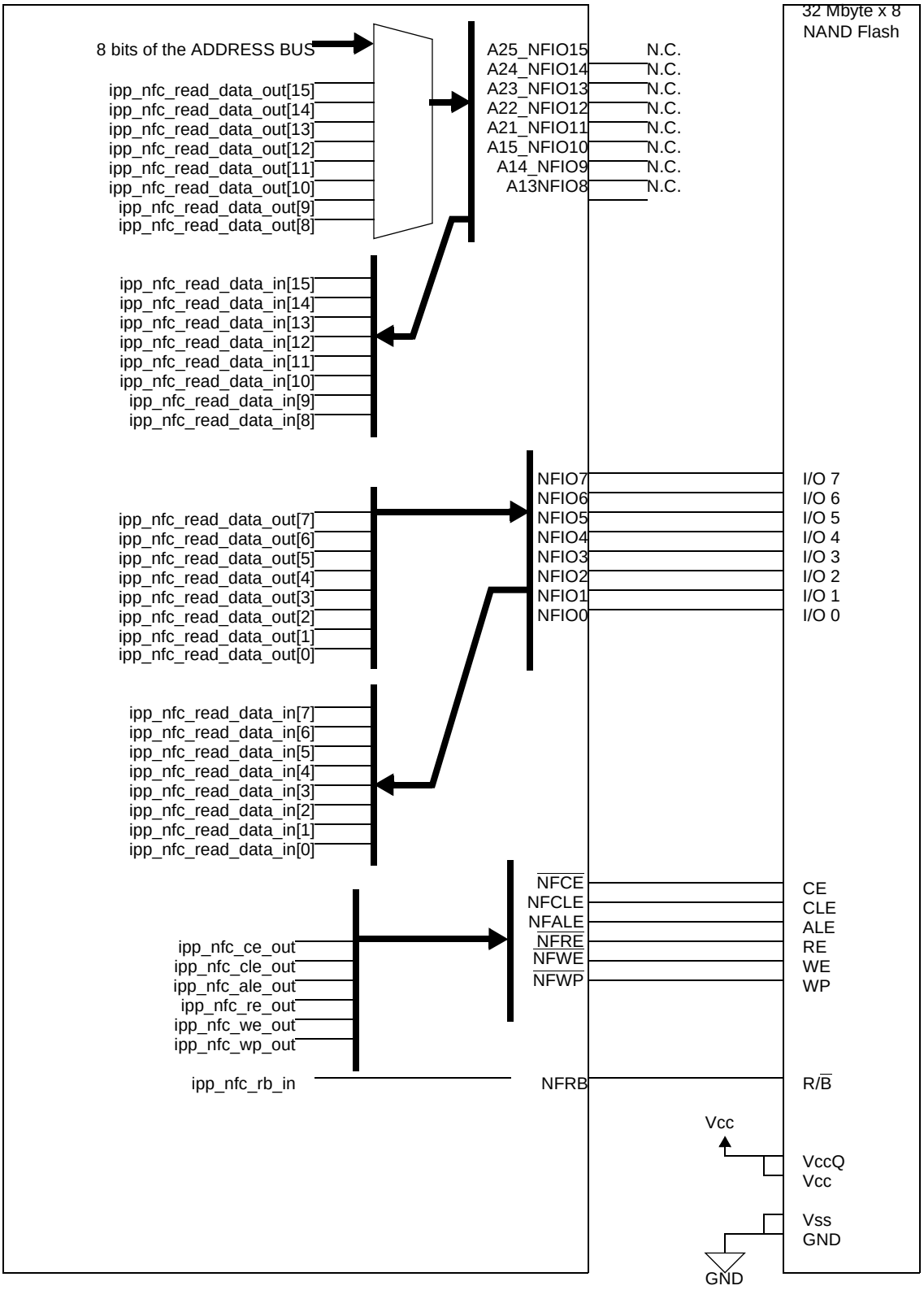


Figure 20-40. 256 Mbit (32 M x 8 Bit) NAND Flash Connection Diagram

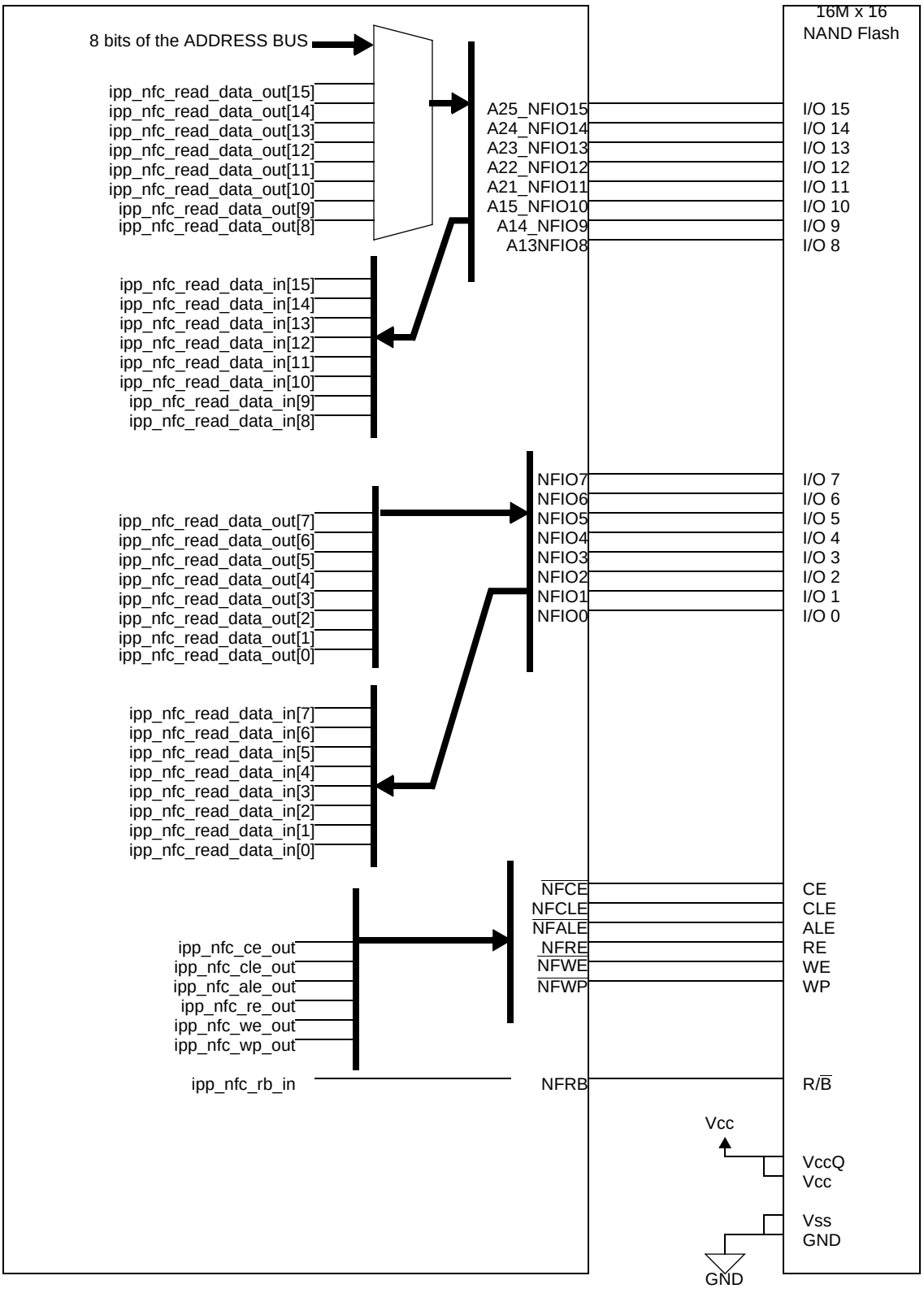


Figure 20-41. 256 Mbit (16 M x 16 Bit) NAND Flash Connection Diagram

Table 20-29. Examples for NANDFC Pin Configuration for Selected Memory Devices

Device	BOOT	NFC_FMS	$\overline{\text{NF8BOOT}}$	$\overline{\text{NF16BOOT}}$	NF_16BIT_SEL
SAMSUNG K9F5608 (32M x 8Bit) Page size is 528 Bytes	NO	0	1	1	0
	YES	0	0	1	X
SAMSUNG K9F5616 (16M x 16Bit) Page size is 528 Bytes	NO	0	1	1	1
	YES	0	1	0	X
SAMSUNG K9F1G08 (128M x 8Bit) Page size is 2112 Bytes	NO	1	1	1	0
	YES	1	0	1	X
SAMSUNG K9F1G16 (64M x 16Bit) Page size is 2112 Bytes	NO	1	1	1	1
	YES	1	1	0	X

NOTE

The NANDFC can support High Speed (HS) NAND Flash by supplying higher frequencies (up to 50 MHz) to the Flash Clock input.

Chapter 21
Personal Computer Memory Card International Association (PCMCIA) Controller

This chapter describes the Personal Computer Memory Card International Association (PCMCIA) controller for the i.MX31 and i.MX31L. The association standard is PCMCIA 2.1, which defines the use of memory and I/O devices as insertable and exchangeable peripherals for personal computers or PDAs. Examples of these types of devices include compact flash and WLAN adapters.

21.1 Overview

The PCMCIA host adapter module provides the control logic for PCMCIA socket interfaces, and requires some additional external analog power switching logic and buffering. The additional external buffers allow the PCMCIA host adapter module to support one PCMCIA socket.

Figure 21-1 shows the PCMCIA controller block diagram.

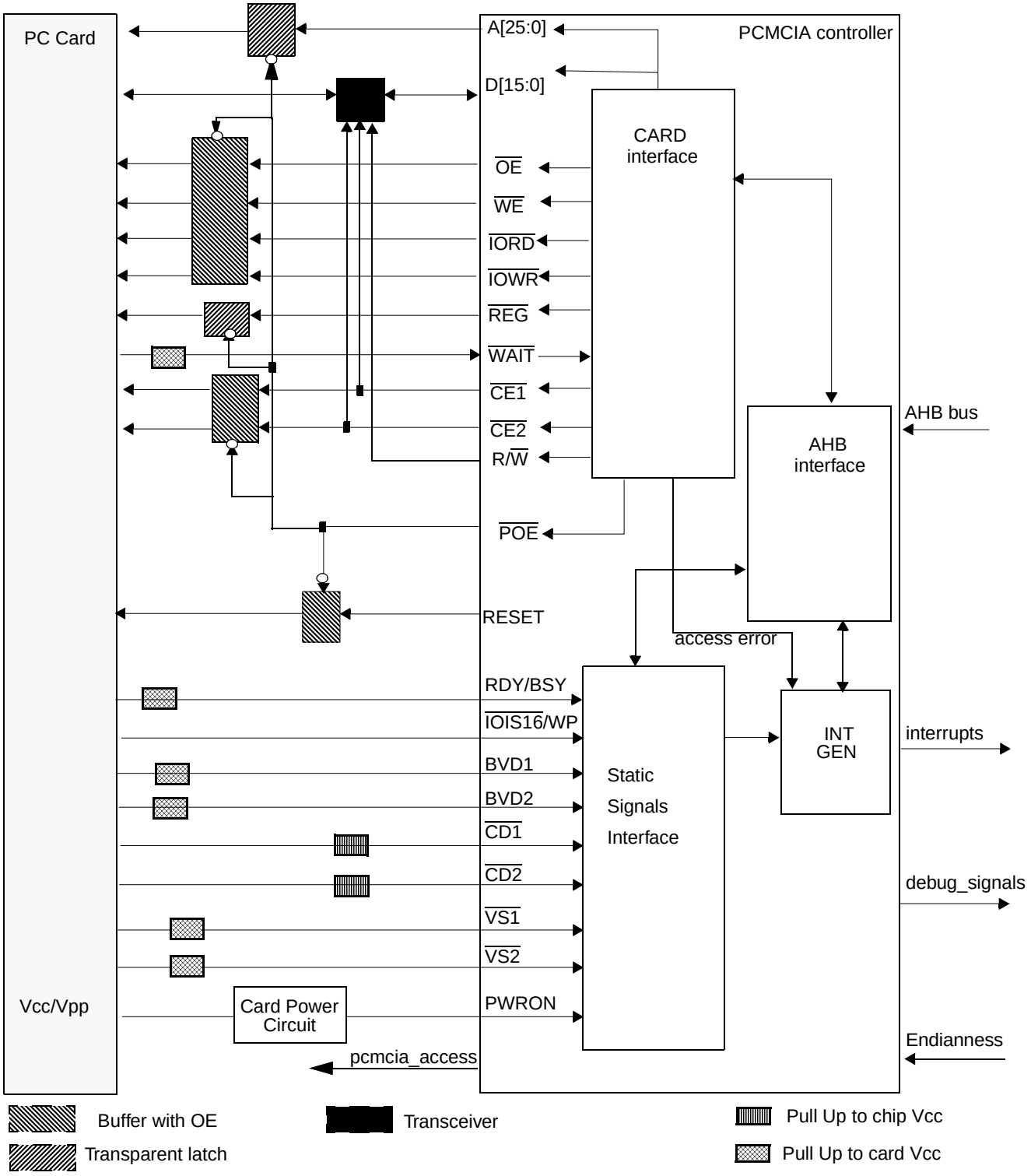


Figure 21-1. PCMCIA Controller Interface Block Diagram

21.2 Features

The PCMCIA controller includes the following features:

- A host adapter interface fully compliant with the PCMCIA standard release 2.1 (PC Card -16).
  - — Supports one PCMCIA socket
  - — Supports hot-insertion
  - — Supports card detection
  - — Provides mappings to common memory space, attribute memory space, and I/O space. Each space is up to 64 Mbyte in size.
  - — Supports 5 memory windows
  - — Generates a single interrupt to the ARM11 core
  - — Provides fully programmable PC card access timing
  - — Handles interrupts from the card
- The PCMCIA controller is part of the EMI complex and shares its pins with the EIM, SDRAMC, and NAND flash controller.
- Supports ATA disk emulation

21.3 External Signal Description

21.3.1 Detailed Signal Descriptions

Table 21-2 shows the PCMCIA signal descriptions for the pins that are used to control the PCMCIA interface.

Table 21-2. PCMCIA Signal Descriptions

Signal	In/Out	Description
Standard Pins		
A[25:0]	Output	Address Bus. These address bus output lines allows direct addressing of up to 64 Mbytes of linear memory on the PCMCIA card.
D[15:0]	I/O	Data Bus. Bidirectional. PCMCIA socket data I/O pins.
$\overline{CE1}$, $\overline{CE2}$	Output	Card Enable. When a PCMCIA access is performed, $\overline{CE1}$ enables even bytes, $\overline{CE2}$ enables odd bytes. See also Section 21.5.9, “Data and Control Signals Relations” and Table 21-20.
$\overline{OE}$	Output	Output Enable. During PCMCIA accesses, $\overline{OE}$ is used to drive memory read data from a PC card in a PCMCIA socket.
$\overline{WE}$	Output	Write Enable. Program during PCMCIA access, $\overline{WE}$ is used to latch memory write data to the PC card in a PCMCIA socket. Can also be used as the programming strobe for PC cards using programmable memory technologies.
$\overline{REG}$	Output	Register Accesses Attribute Memory Select. When $\overline{REG}$ is asserted during PCMCIA access, card access is limited to attribute memory when a memory access occurs ($\overline{WE}$ or $\overline{OE}$ are asserted) and to I/O ports when I/O access occurs ($\overline{IORD}$ or $\overline{IOWR}$ are asserted). if REG is asserted, accesses to common memory are blocked.

Table 21-2. PCMCIA Signal Descriptions (continued)

Signal	In/Out	Description
$\overline{\text{IORD}}$	Output	I/O Read. This output goes active (low) for I/O reads from the socket. This signal is asserted together with $\overline{\text{REG}}$ and it is used to read data from the PC card I/O space. $\overline{\text{IORD}}$ is valid only when $\overline{\text{REG}}$ and either $\overline{\text{CE1}}$ or $\overline{\text{CE2}}$ signals are also asserted.
$\overline{\text{IOWR}}$	Output	I/O Write. This output goes active (low) for I/O write to the socket. Asserted with $\overline{\text{REG}}$ during PCMCIA accesses, used to latch data into the PC cards I/O space. $\overline{\text{IOWR}}$ is valid only when $\overline{\text{REG}}$ and either $\overline{\text{CE1}}$ and $\overline{\text{CE2}}$ signals are also asserted.
$\overline{\text{WAIT}}$	Input	Extend bus cycle. Input. Asserted by the PC card to delay completion of the pending memory or I/O cycle.
$\overline{\text{IOIS16}}$/WP	Input	I/O port is 16-bits. When the card and its socket are programmed for I/O interface operation, this signal is used as $\overline{\text{IOIS16}}$ and must be asserted by the PC card when the address on the bus corresponds to an address on the PC card and the I/O port being addressed supports 16-bit accesses. If the I/O region in which the address resides is programmed as 8-bit wide $\overline{\text{IOIS16}}$ is ignored. Write Protect. When the card and socket are programmed for memory interface operation, this signal is used as WP. It reflects the state of the write protect of the PC card. The PC card must assert WP when the card switch is enabled. It must be negated when the switch is disabled. For a PC card that is writable without a switch, WP must be connected to ground.If the PC card is permanently write-protected, WP must be connected to Vcc.
$\overline{\text{VS1}}$, $\overline{\text{VS2}}$	Input	Voltage sense. Input. Generated by the card to notify the socket of the card's CIS VCC requirements.
$\overline{\text{CD1}}$, $\overline{\text{CD2}}$	Input	Card Detect. Provide proper detection of card insertion. They must be connected to ground internally on the PC card, thus, these signals are forced low when a card is placed on the socket. These signals must be pulled up to system Vcc to allow card detection to function when the card socket is powered down.
BVD1/$\overline{\text{STSCHG}}$, BVD2/$\overline{\text{SPKR}}$	Input	These two lines can be used for battery voltage detection or status change/speaker. Battery Voltage Detect. When the card and its socket are programmed for memory interface operation, these signals are generated by the PC card with on board battery to report the battery condition. See Table 21-3 for a description of the logical combinations representing battery condition. Status Change. When the card is in I/O interface operation, BVD1 is used as Status Change and is generated by the I/O PC card. Status Change must be held negated when the "signal on change" bit and the "change" bit in the card status register are either or both zero. STSCHG must be asserted when both bits = 1. Speaker. Input. When the card is in I/O interface operation BVD2 is used as Audio Digital Waveform. A card that does not this capability should drive $\overline{\text{SPKR}}$ high.
READY/$\overline{\text{IREQ}}$	Input	Ready. When the card and its socket are programmed for memory interface operation, this signal is used as RDY/BSY and must be asserted by a PC card to indicate that a PC card is busy processing a previous write command. When the card and its socket are programmed for I/O interface operation, this signal is used as $\overline{\text{IREQ}}$ and must be asserted by a PC card to indicate that a device on the PC card requires service by host software. Must be held negated when no interrupt is requested.
RESET	Output	Card Reset. Output. Provided to clear the card's configuration option register, thus placing the card in its default (memory only interface) state and beginning an additional card initialization. RESET signal has inverted polarity when in TrueIDE mode. See also Section 21.5.10, "True IDE Mode Access."

Table 21-2. PCMCIA Signal Descriptions (continued)

Signal	In/Out	Description
PCMCIA Controller Module Pins		
POWERON	Input	Power is On. The card supply circuitry can use this signal as an interrupt to notify when the card's power supply reaches the full required voltage.
R/$\overline{W}$	Output	External Transceiver Direction. Negated during read cycles and asserted during write accesses.
$\overline{POE}$	Output	PCMCIA buffers output enable. An output line reflecting the value of PGCR[POE] bit. Used to three-state control signals and to latch the address. See Figure 21-1 for a simplified block diagram.
SPKROUT	Output	Speaker Output. Provides a digital audio waveform to be driven to the system's speaker. This signal is connected directly to the SPKR input.
Endianness	Input	Endianness control. Input. This input pin defines the Endianness mode of the module. '1' is Big Endian. This module should be tied high or low. See Section 21.5.13, "Endianness Support."
pcmcia_access	Output	This output signal is used to indicate that a valid access to the card is performed. This signal is used by the EMI muxing to select the pcmcia_if port and drive it to the pins.
$\overline{ipi\_int\_pcmcia}$	Output	This is the interrupt line from the pcmcia_if module. This signal is a logical OR of all interrupts generated by the pcmcia_if.
$\overline{ipi\_int\_vs1}$	Output	This interrupt is generated if the voltage sense #1 input signal from the card has changed.
$\overline{ipi\_int\_vs2}$	Output	This interrupt is generated if the voltage sense #2 input signal from the card has changed.
$\overline{ipi\_int\_wp}$	Output	This interrupt is generated if the write protect input signal from the card has changed.
$\overline{ipi\_int\_cd1}$	Output	This interrupt is generated if the card detect #1 input signal from the card has changed.
$\overline{ipi\_int\_cd2}$	Output	This interrupt is generated if the card detect #2 input signal from the card has changed.
$\overline{ipi\_int\_bvd1}$	Output	This interrupt is generated if the battery voltage detect #1 input signal from the card has changed.
$\overline{ipi\_int\_bvd2}$	Output	This interrupt is generated if the battery voltage detect #2 input signal from the card has changed.
$\overline{ipi\_int\_rdy\_l}$	Output	This interrupt is generated if RDY/$\overline{IREQ}$ pin is low.
$\overline{ipi\_int\_rdy\_h}$	Output	This interrupt is generated if RDY/$\overline{IREQ}$ pin is high.
$\overline{ipi\_int\_rdy\_r}$	Output	This interrupt is generated if a rising edge was detected on the RDY/$\overline{IREQ}$ pin.
$\overline{ipi\_int\_rdy\_f}$	Output	This interrupt is generated if a falling edge was detected on the RDY/$\overline{IREQ}$ pin.
$\overline{ipi\_int\_poweron}$	Output	This interrupt is generated if the POWERON input signal from the card has changed
$\overline{ipi\_int\_sts}$	Output	This status change interrupt is a logic AND of the following: $\overline{ipi\_int\_vs1}$, $\overline{ipi\_int\_vs2}$, $\overline{ipi\_int\_wp}$, $\overline{ipi\_int\_cd1}$, $\overline{ipi\_int\_cd2}$, $\overline{ipi\_int\_bvd1}$, $\overline{ipi\_int\_bvd2}$, $\overline{ipi\_int\_poweron}$.
$\overline{ipi\_int\_ireq}$	Output	This interrupt line is a logic AND of the following: $\overline{ipi\_int\_rdy\_l}$, $\overline{ipi\_int\_rdy\_h}$, $\overline{ipi\_int\_rdy\_r}$, ipi_int_rdy_f.
$\overline{ipi\_int\_err}$	Output	This interrupt is generated following an error detected by the PCMCIA controller. See Section 21.5.4.1, "Error Interrupt Conditions" for detailed description of error cases.

Table 21-3 provides descriptions for the BVD1 and BVD2 signals.

Table 21-3. BVD1 and BVD2 Descriptions

BVD1	BVD2	Description
1	1	battery is in good condition
1	0	battery is in warning condition and should be replaced, although data integrity on the card is assured.
0	X	battery is in no longer serviceable and data is lost.

21.4 Memory Map and Register Definition

Table 21-4 shows the memory map of the PCMCIA controller.

Table 21-4. PCMCIA Controller Memory Map

Address	Register	Access	Reset Value	Section/Page
0xB800_4000 (PIPR)	PCMCIA input Pins Register	Read Only	0x0000_00 – –	21.4.1.1/21-9
0xB800_4004 (PSCR)	PCMCIA Status Changed Register	Read/Write	0x0000_0000	21.4.1.2/21-11
0xB800_4008 (PER)	PCMCIA Enable Register	Read/Write	0x0000_1018	21.4.1.3/21-12
0xB800_400C (PBR0)	PCMCIA Base Register 0	Read/Write	0x0000_0000	21.4.1.4/21-14
0xB800_4010 (PBR1)	PCMCIA Base Register 1	Read/Write	0x0000_0000	21.4.1.4/21-14
0xB800_4014 (PBR2)	PCMCIA Base Register 2	Read/Write	0x0000_0000	21.4.1.4/21-14
0xB800_4018 (PBR3)	PCMCIA Base Register 3	Read/Write	0x0000_0000	21.4.1.4/21-14
0xB800_401C (PBR4)	PCMCIA Base Register 4	Read/Write	0x0000_0000	21.4.1.4/21-14
0xB800_4028 (POR0)	PCMCIA Option Register 0	Read/Write	0x0000_0000	21.4.1.5/21-15
0xB800_402C (POR1)	PCMCIA Option Register 1	Read/Write	0x0000_0000	21.4.1.5/21-15
0xB800_4030 (POR2)	PCMCIA Option Register 2	Read/Write	0x0000_0000	21.4.1.5/21-15
0xB800_4034 (POR3)	PCMCIA Option Register 3	Read/Write	0x0000_0000	21.4.1.5/21-15
0xB800_4038 (POR4)	PCMCIA Option Register 4	Read/Write	0x0000_0000	21.4.1.5/21-15
0xB800_4044 (POFR0)	PCMCIA Offset Register 0	Read/Write	0x0000_0000	21.4.1.6/21-19

Table 21-4. PCMCIA Controller Memory Map (continued)

Address	Register	Access	Reset Value	Section/Page
0xB800_4048 (POFR1)	PCMCIA Offset Register 1	Read/Write	0x0000_0000	21.4.1.6/21-19
0xB800_404C (POFR2)	PCMCIA Offset Register 2	Read/Write	0x0000_0000	21.4.1.6/21-19
0xB800_4050 (POFR3)	PCMCIA Offset Register 3	Read/Write	0x0000_0000	21.4.1.6/21-19
0xB800_4054 (POFR4)	PCMCIA Offset Register 4	Read/Write	0x0000_0000	21.4.1.6/21-19
0xB800_4060 (PGCR)	PCMCIA General Control Register	Read/Write	0x0000_0008	21.4.1.7/21-20
0xB800_4064 (PGSR)	PCMCIA General Status Register	Read/Write	0x0000_0000	21.4.1.8/21-21

21.4.1 Register Summary

Table 21-2 shows the key to the register fields, and Table 21-5 shows the register figure conventions.

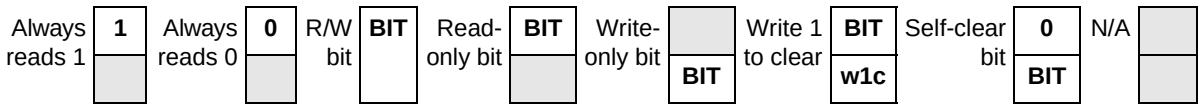


Figure 21-2. Key to Register Fields

Table 21-5. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.

Table 21-5. Register Figure Conventions (continued)

Convention	Description
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 21-6 shows the PCMCIA register summary.

Table 21-6. PCMCIA Controller Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xB800_4000 (PIPR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	POWERONt	RDY	BVD2	BVD1	$\overline{CD}$		WP	VS	
	W																
0xB800_4004 (PSCR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	POWC	RDYR	RDYF	RDYH	RDYL	BVDC2	BVDC1	CDC2	CDC1	WPC	VSC2	VSC1
	W					w1c	w1c	w1c			w1c	w1c	w1c	w1c	w1c	w1c	w1c
0xB800_4008 (PER)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	ERRINTEN	POWERONEN	RDYRE	RDYFE	RDYHE	RDYLE	BVDE2	BVDE1	CDE2	CDE1	WPE	VSE2	VSE1
	W																
0xB800_400C (PBR0) 0xB800_4010 (PBR1) 0xB800_4014 (PBR2) 0xB800_4018 (PBR3) 0xB800_401C (PBR4)	R	0	0	0	0	0	0	PBA[25:16]									
	W																
	R	PBA[15:0]															
	W																

Table 21-6. PCMCIA Controller Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xB800_4028 (POR0) 0xB800_402C (POR1) 0xB800_4030 (POR2) 0xB800_4034 (POR3) 0xB800_4038 (POR4)	R	0	0	PV	WPEN	WP	PRS		PPS	PSL[6:0]							PSST[5]
	W																
	R	PSST[4:0]					PSHT[5:0]						0 BSIZE				
	W																
0xB800_4044 (POFR0) 0xB800_4048 (POFR1) 0xB800_404C (POFR2) 0xB800_4050 (POFR3) 0xB800_4054 (POFR4)	R	0	0	0	0	0	0	POFA[25:16]									
	W																
	R	POFA[15:0]															
	W																
0xB800_4060 (PGCR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	LPMEN	SPKREN	POE	RESET
	W																
0xB800_4064 (PGSR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	NWINE	LPE	SE	CDE	WPE
	W												w1c	w1c	w1c	w1c	w1c

21.4.1.1 PCMCIA Input Pins Register (PIPR)

This register indicates the status of inputs from the PCMCIA card to the host: battery voltage detect, card detect, ready, voltage sense, and write protect status (BVD, CD, RDY, VS, WP). PIPR is a read-only register.

The register should be clocked with a gated clock. This clock is active only when trying to access the peripheral. When accessing this register, a 2-wait state is added by the PCMCIA controller. The reset values in Figure 21-3 are the reset values of the PIPR flip-flops. The actual data read reflects the value to the PCMCIA controller. Table 21-7 provides the register's field descriptions.

0xB800_4000 (PIPR) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	POWERON	RDY	BVD2	BVD1	CD		WP	VS	
W																
Reset	0	0	0	0	0	0	0	0	—	—	—	—	—	—	—	—

Figure 21-3. PCMCIA Input Pins Register (PIPR)

Table 21-7. PIPR Field Descriptions

Field	Description
31–9	Reserved.
8 POWERON	Power is on. This bit indicates the status of the power signal from the card. 0 Card indicates that it did not reach its power supply requirements. 1 Card indicates Power is on
7 RDY	RDY/$\overline{\text{BSY}}$/$\overline{\text{IREQ}}$. When the card and its socket are in memory interface operation, this bit functions as RDY/BSY indicating that the card is busy processing a previous write command. When the card and its socket are in I/O interface operation, this bit functions as IREQ indicating that a device on the PC card requires service by host software. This interrupt could be either level or pulse and can have either high or low polarity. This data can be read in the CIS of the card itself. 0 PC card is busy processing a previous command or performing initialization. 1 PC card is ready to accept a new data-transfer command.
6 BVD2	Battery Voltage detect 2/SPKR IN. When the card and its socket are in memory interface mode, this bit reflects the BVD2 signal. For details about settings, see Table 21-3. When the card and its socket are in I/O mode, this bit is used as SPKR IN (speaker in) for a Binary Audio signal, an optional signal which is available only when the card and the socket have been configured for the I/O interface.
5 BVD1	Battery Voltage detect 1/STSCHG IN. When the card and its socket are in memory interface mode, this bit reflects the BVD1 signal. For details, see Table 21-3. When the card and its socket are in I/O mode, this bit is used as STSCHG (status change) indicator. 0 Status has not changed. 1 Status has changed. **Note:** To find out the exact signals that changed value, the Status Change register of the card itself should be read.

Table 21-7. PIPR Field Descriptions (continued)

Field	Description
4–3 $\overline{CD}$	Card Detect 1 and Card Detect 2. Card Detect 1 and Card Detect 2 bits indicate a proper detection of card insertion. When both bits are '0', the card is inserted properly. 00 Card is inserted properly. 01 Card is inserted improperly. 11 Card is inserted improperly. 11 Card is not inserted. These bits are asynchronous.
2 WP	Write Protect. This bit reflects the state of the write protect switch on the PC card. 0 Write Protect Switch is disabled. 1 Write Protect switch is enabled.
1–0 VS	Voltage Sensor. VS bits notify the host of the card's Card Information Structure (CIS) Vcc requirements. This data can be used by the host to control external voltage transceiver. For details see the PCMICA PCCARD standard.

21.4.1.2 PCMCIA Status Change Register (PSCR)

Each bit in the PSCR register is set any time a change in the signal it monitors occurs. The status is cleared using a "write 1 to clear" operation on the register. The contents of PSCR, shown in Figure 21-4, are logically AND'ed with the PER register to generate a PCMCIA interrupt.

The register should be clocked with a gated clock. This clock is active only when trying to access the peripheral. When accessing this register, two wait states are added by the PCMCIA. The inputs to the module are sampled twice before being read, to avoid meta-stability. This is done in spite of the fact that interrupts are generated even when the clock is off.

The bit assignments for the PSCR register are shown in Figure 21-4. The bit field descriptions are provided in Table 21-8. Each of the bits in Table 21-8 (excluding RDYL and RDYHare) set (=1) when a change occurs in its corresponding parameter. It is zeroed on system reset. RDYL and RDYH are level sensitive and therefore, reflect the value of the RDY pin and have no reset value.

0xB800_4004 (PSCR) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	POWC	RDYR	RDYF	RDYH	RDYL	BVDC2	BVDC1	CDC2	CDC1	WPC	VSC2	VSC1
W					w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	—	—	0	0	0	0	0	0	0

Figure 21-4. PCMCIA Status Change Register (PSCR)

Table 21-8. PSCR Field Descriptions

Field	Description
31–12	Reserved.
11 POWC	POWERON signal changed. 0 No change has occurred in the POWERON signal since system reset. 1 A change has occurred in the POWERON signal since system reset.
10 RDYR	RDY/$\overline{\text{IREQ}}$ pin rising edge detect. Device and socket positive edge interrupt. 0 No rising edge has occurred in RDY since system reset. 1 A rising edge occurred in RDY since system reset.
9 RDYF	RDY/$\overline{\text{IREQ}}$ pin falling edge detect. Device and socket negative edge interrupt. 0 No falling edge has occurred in RDY since system reset. 1 A falling edge occurred in RDY since system reset.
8 RDYH	This bit reflects value of RDY signal. RDY/$\overline{\text{IREQ}}$ pin is high indicating a device and socket high level interrupt.
7 RDYL	RDY/$\overline{\text{IREQ}}$ pin is low. Device and socket loveless interrupt. This bit is the inverted value of RDY signal
6 BVDC2	Battery Voltage 2/SPKR IN Changed. 0 No change has occurred in Battery Voltage #2 or SPKR since system reset. 1 A change has occurred in Battery Voltage #2 or SPKR since system reset.
5 BVDC1	Battery Voltage 1/STSCHG Changed. 0 No change has occurred in Battery Voltage #1 since system reset. 1 A change has occurred in Battery Voltage #1 since system reset.
4 CDC2	Card Detect 2 hanged. 0 No change has occurred in card detect #2 since system reset. 1 A change has occurred in card detect #2 since system reset.
3 CDC1	Card Detect 1 Changed. 0 No change has occurred in card detect #1 since system reset. 1 A change has occurred in card detect #1 since system reset.
2 WPC	Write Protect Changed. 0 No change has occurred in the write protect status since system reset. 1 A change has occurred in the write protect status since system reset.
1 VSC2	Voltage Sense2 Changed. 0 No change has occurred in voltage sensor #2 since system reset. 1 A change has occurred in voltage sensor #2 since system reset.
0 VSC1	Voltage Sense1 Changed. 0 No change has occurred in voltage sensor #1 since system reset. 1 A change has occurred in voltage sensor #1 since system reset.

21.4.1.3 PCMCIA Enable Register (PER)

Setting a bit in PER, shown in Figure 21-5, enables the corresponding interrupt. When accessing this register, one wait state will be added by the PCMCIA. Table 21-9 shows the register's field descriptions.

0xB800_4008 (PER) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	ERRINTEN	POWERONEN	RDYRE	RDYFE	RDYHE	RDYLE	BVDE2	BVDE1	CDE2	CDE1	WPE	VSE2	VSE1
W																
Reset	0	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0

Figure 21-5. PCMCIA Enable Register (PER)

Table 21-9. PER Field Descriptions

Field	Description
31–13	Reserved.
12 ERRINTEN	Error Interrupt Enable. Setting this bit enables the interrupt as a result of an error signal. 0 Interrupt is disabled. 1 Interrupt is enabled. Note that the default is to enable the interrupt.
11 POWERONEN	Power is On Interrupt Enable. Setting this bit enables the interrupt as a result of any Power On signal change. 0 Interrupt is disabled. 1 Interrupt is enabled.
10 RDYRE	RDY/$\overline{\text{IREQ}}$ pin rising edge interrupt enable. 0 Interrupt is disabled. 1 Interrupt is enabled.
9 RDYFE	RDY/$\overline{\text{IREQ}}$ pin falling edge interrupt enable. 0 Interrupt is disabled. 1 Interrupt is enabled.
8 RDYHE	RDY/$\overline{\text{IREQ}}$ pin is high level interrupt enable. 0 Interrupt is disabled. 1 Interrupt is enabled.
7 RDYLE	RDY/$\overline{\text{IREQ}}$ pin is low level interrupt enable. 0 Interrupt is disabled. 1 Interrupt is enabled.
6 BVDE2	Battery Voltage 2/SPKR IN interrupt enable. Setting this bit enables the interrupt as a result of any signal change from battery voltage sensor #2 OR from the SPKR IN signal. 0 Interrupt is disabled. 1 Interrupt is enabled.
5 BVDE1	Battery Voltage 1/STSCHG interrupt enable. Setting this bit enables the interrupt as a result of any signal change from the battery voltage sensor #1. 0 Interrupt is disabled. 1 Interrupt is enabled.

Table 21-9. PER Field Descriptions (continued)

Field	Description
4 CDE2	Card Detect 2 interrupt enable. Setting this bit enables the interrupt as a result of any signal change from card detect #2. **Note:** The default setting enables the interrupt. 0 Interrupt is disabled. 1 Interrupt is enabled.
3 CDE1	Card Detect 1 interrupt enable. Setting this bit enables the interrupt as a result of any signal change from card detect #1 **Note:** The default setting enables the interrupt. 0 Interrupt is disabled. 1 Interrupt is enabled.
2 WPE	Write Protect interrupt enable. Setting this bit enables the interrupt as a result of any signal change in the write protect status. 0 Interrupt is disabled. 1 Interrupt is enabled.
1 VSE2	Voltage sense2 interrupt enable. Setting this bit enables the interrupt as a result of any signal change from voltage sensor #2. 0 Interrupt is disabled. 1 Interrupt is enabled.
0 VSE1	Voltage sense1 interrupt enable. Setting this bit enables the interrupt as a result of any signal change from voltage sensor #1. 0 Interrupt is disabled. 1 Interrupt is enabled.

21.4.1.4 PCMCIA Base Registers 0–4 (PBR0–PBR4)

This is compared to the address bus to determine if a PCMCIA window is being accessed by an internal bus master. PBA is used in conjunction with POR[BSIZE]. When accessing this register, 1-wait state will be added by the PCMCIA. The field assignments for this register are shown in Figure 21-6. Table 21-10 shows the register's field descriptions.

0xB800_400C (PBR0)
0xB800_4010 (PBR1)
0xB800_4014 (PBR2)
0xB800_4018 (PBR3)
0xB800_401C (PBR4)

Access: User Read/Write

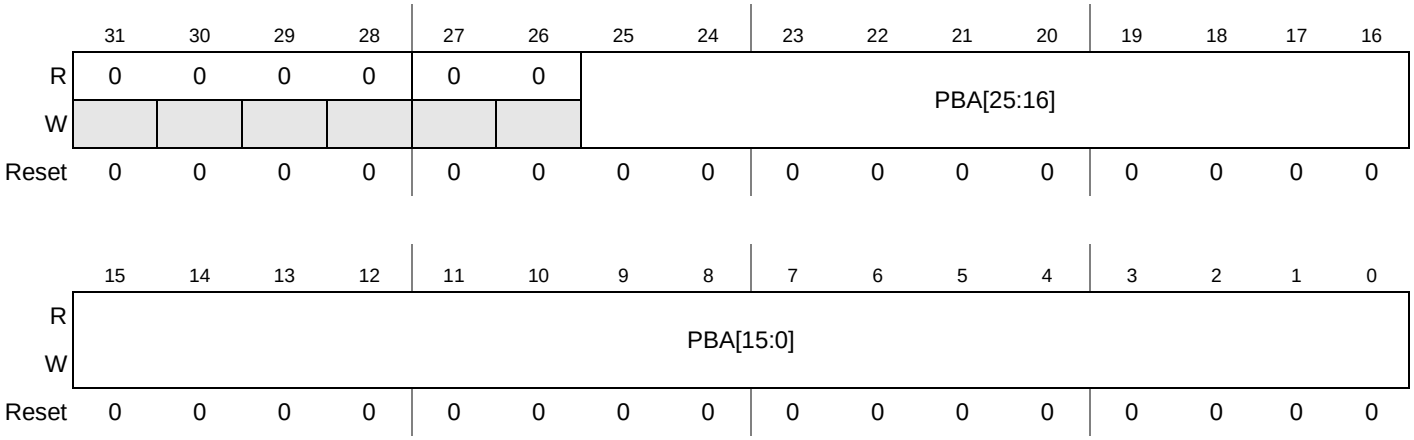


Figure 21-6. PCMCIA Base Registers 0–4 (PBR0–PBR4)

Table 21-10. PBR0–PBR4 Field Descriptions

Field	Description
31–26	Reserved.
25–0 PBA	PCMCIA Base Address.

21.4.1.5 PCMCIA Option Registers 0–4 (POR0–POR4)

The POR registers shown in Figure 21-7 handle time manipulation, provide the address mask for the bank size, and define the region, write protection, and validation. When accessing this register 1-wait state will be added by the PCMCIA. Table 21-11 shows the register's field descriptions.

0xB800_4028 (POR0)
0xB800_402C (POR1)
0xB800_4030 (POR2)
0xB800_4034 (POR3)
0xB800_4038 (POR4)

Access: User Read/Write

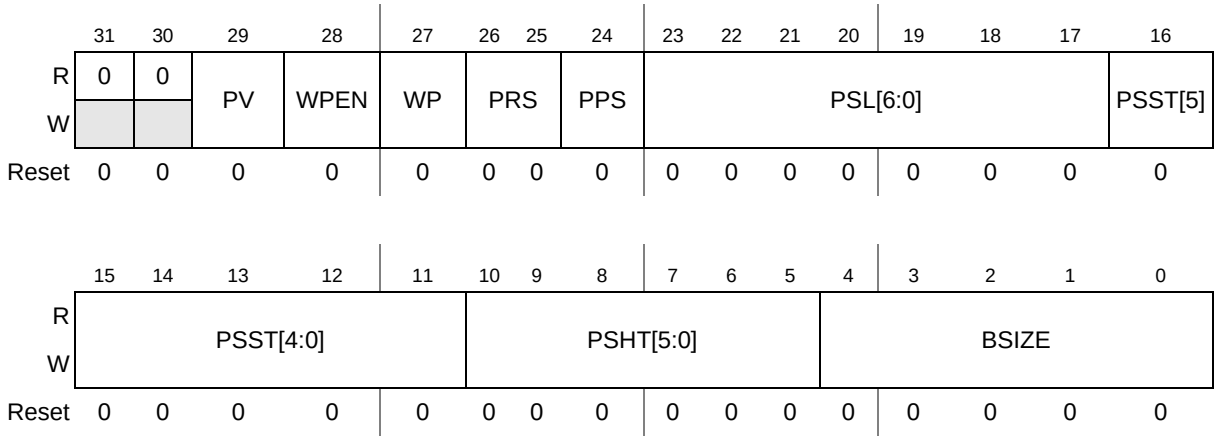


Figure 21-7. PCMCIA Option Registers 0–4 (POR0–POR4)

Table 21-11. POR0–POR4 Field Descriptions

Field	Description
31–30	Reserved.
29 PV	PCMCIA Valid. Defines whether the contents of the PBR and POR pair are valid (window enable). See also Table 21-13. 0 This bank is invalid. 1 This bank is valid.
28 WPEN	PCMCIA Write Protect Input Enable. This bit is the write protect input signal enable bit, controlled by software. When this bit is cleared, the WP input to the PCMCIA module is ignored. To see the relationship between this bit, the WP bit, and the WP signal coming from the card refer to Section 21.5.7, “Write Protect.” 0 Write Protect input (WP) signal is ignored. 1 Write Protect (WP) input signal is enabled.
27 WP	PCMCIA Write Protect Enable This bit is the write protect enable bit controlled by software. To see the relationship between this bit, the WPEN bit, and the WP signal coming from the card refer to Section 21.5.7, “Write Protect.” 0 Not write protected. 1 Write Protected. Attempting to write to this window causes an interrupt.
26–25 PRS	PCMCIA Region Select. 00 Common memory space. 01 TrueIDE mode. 10 Attribute memory space. 11 I/O space.
24 PPS	PCMCIA Port Size. Specifies the port size of the PCMCIA window. Refer to Section 21.5.8, “16-Bit/8-Bit Support” for more details. 0 16-bit port size 1 8-bit port size

Table 21-11. POR0–POR4 Field Descriptions (continued)

Field	Description
23–17 PSL	PCMCIA Strobe Length. Determines the number of cycles the strobe is asserted during a PCMCIA access for this window and, thus, it is the main parameter for determining cycle length. The cycle may be lengthened by asserting $\overline{\text{WAIT}}$. **Note:** To sample the $\overline{\text{WAIT}}$ signal it must be synchronized by two FF. This means that if the system must rely on the $\overline{\text{WAIT}}$ signal, the PSL should be calculated as the maximum valid time on WAIT plus two. For example, suppose we have a 100 MHz clock (one period is therefore 10 ns) and the card specification says that the time from WE/OE low to $\overline{\text{WAIT}}$ valid is 70 ns. The PSL value should be at least: (70 ns/10 ns)+2=9. 0000000 Strobe asserted 128 clocks cycles. 0000001 Strobe asserted 1 clocks cycles. 0000010 Strobe asserted 2 clocks cycles. ... 1111111 Strobe asserted 127 clocks cycles.
16–11 PSST	PCMCIA Strobe Set Up Time (address to strobe assertion). Specifies when $\overline{\text{IOWR}}$ or $\overline{\text{WE}}$ is asserted during a PCMCIA write access or when $\overline{\text{IORD}}$ or $\overline{\text{OE}}$ are asserted during a PCMCIA read access handled by the PCMCIA controller. This helps meet address/setup time requirements for slow memories and peripherals. 000000 Reserved. 000001 Address to strobe assertion 1 clock. 000010 Address to strobe assertion 2 clock. ... 111111 Address to strobe assertion 63 clock. **Note:** Using PSST=000001 is not allowed when WPEN bit is set since the synchronization of WP signal takes 2 clocks.
10–5 PSHT	PCMCIA Strobe Hold Time (strobe negation to address negation).Specifies when $\overline{\text{IOWR}}$ or $\overline{\text{WE}}$ are negated during a PCMCIA write or when $\overline{\text{IORD}}$ or $\overline{\text{OE}}$ are negated during a PCMCIA read. Used to meet address/data hold time requirements for slow memories and peripherals. 000000 Strobe negation to address change 0 clock. 000001 Strobe negation to address change 1 clock. ... 111111 Strobe negation to address change 63 clock.
4–0 BSIZE	PCMCIA Bank Size. Determines the address mask field of each POR and provides masking for any of the corresponding bits in the associated PBR. The bank size corresponds to values of this bit field as indicated in Table 21-12. BSIZE determines not only the bank size, but also how the address is compared with PBR[PBA]. If BSIZE is a virtual field, the MASK is defined as shown in Table 21-13. Addr, AND MASK PBA, AND MASK for a valid PCMCIA access; otherwise, it is not a valid PCMCIA access.

Table 21-12. BSIZE Values

Value	Meaning	Value	Meaning	Value	Meaning
00000	1 byte	01111	1 Kbyte	11110	1 Mbyte
00001	2 byte	01110	2 Kbyte	11111	2 Mbyte
00011	4 byte	01010	4Kbyte	11101	4 Mbyte
00010	8 byte	01011	8 Kbyte	11100	8 Mbyte
00110	16 byte	01001	16 Kbyte	10100	16 Mbyte
00111	32 byte	01000	32 Kbyte	10101	32 Mbyte

Table 21-12. BSIZE Values (continued)

Value	Meaning	Value	Meaning	Value	Meaning
00101	64 byte	11000	64 Kbyte	10111	64 Mbyte
00100	128 byte	11001	128 Kbyte		
01100	256 byte	11011	256 Kbyte		
01101	512 byte	11010	512 Kbyte		

NOTE

BSIZE determines not only the bank size, but also how the address is compared with PBR[PBA]. According to the virtual field, MASK as defined as shown on Table 21-13.

Table 21-13. BSIZE Mask

BSIZE	MASK																													
00000	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
00001	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
00011	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
00010	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	
00110	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	
00111	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	
00101	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	
00100	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	
01100	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	
01101	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	
01111	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	
01110	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	
01010	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	
01011	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	
01001	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	
01000	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
11000	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
11001	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
11011	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
11010	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
11110	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
11111	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Table 21-13. BSIZE Mask (continued)

BSIZE	MASK
11101	1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
11100	1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
10100	1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
10101	1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
10111	1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

21.4.1.6 PCMCIA Offset Registers 0–4 (POFR0–POFR4)

The offset address of the window. PBA is used in conjunction with POR[BSIZE]. The external address is ext_addr= POFA + haddr and MASK. When accessing this register 1-wait state is added by the PCMCIA/CF controller. The field definition of the POFRx registers is shown in Figure 21-8. Table 21-14 shows the field descriptions.

0xB800_4044 (POFR0)
0xB800_4048 (POFR1)
0xB800_404C (POFR2)
0xB800_4050 (POFR3)
0xB800_4054 (POFR4)

Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	POFA[25:16]									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	POFA[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 21-8. PCMCIA Offset Registers 0–4 (POFR0–POFR4)

Table 21-14. POFR0–POFR4 Field Descriptions

Field	Description
31–26	Reserved.
25–0 POFA	PCMCIA Offset Address. The offset address of the window. POFA is used in conjunction with POR[BSIZE]. The external address is ext_addr= POFA + haddr and $\overline{\text{MASK}}$.

Example 21-1. Calculating Offset Address

```
If:
haddr[25:0] = 0x0000263, MASK = 0x3FFFFC0 POFA = 0x0000161
then:
 ext_addr = 0x0000161 + 0x0000263 & (0x3FFFFC0) = 0x0000161 + 0x0000023 = 0x0000184
```

21.4.1.7 PCMCIA General Control Register (PGCR)

This is the general control register for the PCMCIA controller. When accessing this register, 1-wait state is added by the PCMCIA controller. Field definitions of the POFRx registers are shown in Figure 21-9. Table 21-15 shows the field descriptions.

0xB800_4060 (PGCR) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	LPMEN	SPKREN	POE	RESET
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Figure 21-9. PCMCIA General Control Register (PGCR)

Table 21-15. PGCR Field Descriptions

Field	Description
31–4	Reserved.
3 LPMEN	Low Power Mode Enable. This bit puts the module into low power mode. In this case external memory accesses are disabled. The reset value is “1” (Low power mode). 0 Normal power mode. 1 Low power mode.
2 SPKREN	SPKROUT Routing Enable. This bit enables the routing of SPKRIN to SPKROUT. 0 Routing disabled. 1 Routing enabled.
1 POE	Card Output Enable. This bit enables the POE signal, used to three-state the external buffers. The POE signal will toggle as follows: POE (signal) PGCR[POE]&PCMCIA_ACCESS (signal). 0 POE signal is disabled. 1 POE signal is enabled.
0 RESET	Card Reset. This bit provides a software reset to the card. This bit is not self clearing, software must modify this bit to take the card out of reset. 0 Card is not in reset. 1 Card is in reset.

21.4.1.8 PCMCIA General Status Register (PGSR)

This is a general status register. If an error interrupt was generated to the host, the host can access this register to find out what caused the error interrupt. All the bits in this register are cleared by writing ‘1’ to the appropriate bit. When accessing this register, 1-wait state is added by the PCMCIA controller. Field definition of the POFRx registers are shown in Figure 21-10. Table 21-16 shows the field descriptions.

0xB800_4064 (PGSR) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	NWINE	LPE	SE	CDE	WPE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 21-10. PCMCIA General Status Register (PGSR)

Table 21-16. PGSR Field Descriptions

Field	Description
31–5	Reserved.
4 NWINE	No Window Error. Attempt to access a card address to an address that is not mapped by any window. 0 No attempt to access a card address to an address that is not mapped by any window since the last system reset was made. 1 An attempt to access a card address to an address that is not mapped by any window since the last system reset was made.
3 LPE	Low Power Error. Attempt to access a card when in low power mode. 0 No attempt to access a card when in low power mode was made since the last system reset 1 An attempt to access a card when in low power mode was made since the last system reset
2 SE	Size Error. A 16-bit access to an 8-bit card was made. 0 No 16-bit access to an 8-bit card was made since the last system reset. 1 A 16-bit access to an 8-bit card was made since the last system reset
1 CDE	Card Detect Error. Attempt to access a card when the card is not inserted. 0 No attempt to access a card when the card was not inserted has been made since last system reset. 1 An attempt to access a card when the card was not inserted has been made since last system reset. In addition, once set no accesses are enabled until it is cleared (even if a card was inserted).
0 WPE	Write Protect Error. Attempt to write to a write protected address. 0 No attempt was made to write to a protected address since the last system reset. 1 An attempt was made to write to a protected address since the last system reset.

21.5 Functional Description

This section describes the operation of memory and I/O cards, interrupt detection and handling, power control, and reset.

21.5.1 Modes of Operation

The following are the PCMCIA modes of operation:

- Memory-only card mode
- I/O card mode
- TrueIDE mode
- Low power mode

21.5.2 Windowing Capabilities

The PCMCIA I/F provides five memory windows. The user can define each memory window as a common memory space, I/O space or attribute memory space. This is done by programming the region select bits (PRS) bits in each POR register for each window.

Configuring a window is done by programming the window's base address (PBA bits in the corresponding PBR register), and by programming the bank size (BSIZE bits in the corresponding POR register).

21.5.2.1 Window Overlapping

Window overlapping is not allowed. The PCMCIA I/F does not indicate to the CPU about window overlapping, software responsible for this. Having overlapping windows will cause unexpected results.

21.5.3 $\overline{\text{WAIT}}$ Signal

The $\overline{\text{WAIT}}$ signal is asserted by the PC card to delay completion of the pending cycle. The access must terminate before the bus time out monitor generates a bus time error.

21.5.4 Interrupts

There are 14 interrupt sources in the PCMCIA controller. The PCMCIA controller generates an interrupt signal for each interrupt source. In addition, the PCMCIA generates a signal which is a locator of all the possible interrupts. It is up to the system's integrator to decide which signal(s) to connect to the system's interrupt controller module. The PCMCIA's interrupt sources are described in Table 21-17.

Table 21-17. PCMCIA I/F Interrupt Sources

Interrupt	Enabled With	Comments
VS1	PER.VSE1	See ipi_int_vs1 on page 21-5.
VS2	PER.VSE2	
WP	PER.WPE	See ipi_int_wp on page 21-5.

Table 21-17. PCMCIA I/F Interrupt Sources (continued)

Interrupt	Enabled With	Comments
CD1	PER.CDE1	See ipi_int_cd1 on page 21-5.
CD2	PER.CDE2	See ipi_int_cd2 on page 21-5.
BVD1	PER.BVDE1	See ipi_int_bvd1 on page 21-5.
BVD2	PER.BVDE2	See ipi_int_bvd2 on page 21-5.
POWERON	PGCR.POWERONEN	This signal is not part of the PCMCIA standard. See ipi_int_poweron on page 21-5.
STATUS CHANGE		“OR” of all the above interrupts. See ipi_int_sts on page 21-5.
RDY_L	PER.RDYLE	
RDY_H	PER.RDYHE	
RDY_R	PER.RDRE	
RDY_F	PER.RDYFE	
IREQ		This interrupt is an “OR” of RDY_L, RDY_H, RDY_R, and RDY_F.
ERR	PER.ERRINTEN	

The PCMCIA input pins register (PIPR) reports any change of inputs from the PCMCIA card to the host (BVD,CD,RDY,VS). The content of the PCMCIA controller status changed register (PSCR) are logically AND’ed with the PCMCIA controller enable register (PER) to generate a PCMCIA controller interrupt. The interrupt level is user programmable and the PCMCIA controller can generate an additional interrupt for RDY/IREQ that can trigger upon a level (low or high) change or edge (fall or rise) of the input signal.

21.5.4.1 Error Interrupt Conditions

Any of the following error conditions can cause an error interrupt:

- Attempt to access a card when the card is in low power mode (LPM).
- Attempt to write to a write protected area, see Section 21.5.7, “Write Protect.”
- Attempt to access a card when the card is not inserted.
- Attempt to do a 16-bit access to an 8-bit card violating the PPS settings or 32-bit access. See Section 21.5.8, “16-Bit/8-Bit Support.”
- Attempt to access card with an address which does not match any window.

An access to the card which yields an error will take 0-3 wait states to complete according to the following conditions:

- Card detect error when PGSR[CDE] is cleared and write protect error which results from the WP signal from the card will take 2 wait states. In case these signals (WP CD) change during access, the access will end two cycles after.
- Size error, no window error low power mode and write protect error which comes from POR[WP] will take one wait state.

Due to the synchronization mechanism on CD signals and WP signal, a change in these signals during access less then two cycles before it finishes would not yield an error. That should not be a problem since these signals typically do not change too much.

21.5.5 Power Control

When LPMEN is set in the PGCR, the PCMCIA I/F internal clocks should be gated off. The module is in “listening mode.” It waits for an indication that a card has been inserted. All the static signals are synchronous in both cases and status change interrupts are generated as necessary (to wake up the core from stop on card detect, for example). In the first case (LPMEN) read from PIPR and read/write from/to PSCR should take 2 wait states to complete because of the synchronizations of the static signals to the core’s clock.

21.5.6 Reset and Three-Score Control

The card can be reset by software. This is done by writing to the RESET bit in the PGCR register. Output of external latches can be disabled by writing to the POE bit in PGCR register.

21.5.7 Write Protect

Write protect is handled in two ways:

- Card’s write protect—the WP signal comes from the card.
- Window’s write protect—the WP bit in the POR register.

When the WPEN is cleared and the card is in memory interface mode, the WP signal coming from the card is ignored. This way it is possible to enable write protection on selected memory regions, even if the card’s WP pin is asserted. The settings for the Write Protect bits are shown in Table 21-18. When the card is in IO mode (POR.PRS = 11),WPEN bit is ignored.

Table 21-18. Write Protect

POR.PRS	POR.WP bit	WP Signal	POR.WPEN	Protect Mode
X0 (memory I/F)	X	0	1	Write enabled
		1	1	Write protected
	0	X	0	Write enabled
	1	X	0	Write protected
11(I/O mode)	0	X	X	Write enabled
	1			Write protected

An interrupt is generated by the PCMCIA controller at any attempt to access a write protected area.

21.5.8 16-Bit/8-Bit Support

The PCMCIA controller supports 16-bit/8-bit accesses. The access size is defined by the $\overline{\text{IOIS16}}$ signal and the PPS bit in the corresponding POR register. The settings for the $\overline{\text{IOIS16}}$ and PPS bits are shown in Table 21-19.

Table 21-19. $\overline{\text{IOIS16}}$ and PPS Bit Relations

$\overline{\text{IOIS16}}$	PPS	Access Size
0	0	16-bit access to a 16-bit card, the host can generate 8-bit accesses and 16-bit access.
0	1	8-bit access although the card is 16-bit. The host should generate 8-bit accesses only. If the host tries to do a 16-bit access, an interrupt is generated
1	0	8-bit access although 16-bit access is selected by PPS. If the host attempts to do a 16-bit access, the PCMCIA I/F writes the lower part of the data to the card.
1	1	8-bit access to 8-bit card. The host should generate 8-bit accesses only. If the host tries to do a 16-bit access, an interrupt is generated.

21.5.9 Data and Control Signals Relations

Table 21-20 describes data and control signal relations in different access modes. Data bus (D) is the data bus of the PC-card.

Table 21-20. Data and Control Signal Relations

Function Mode	REG	CE2	CE1	A0	OE	WE	IORD	IOWR	D[15:8]	D[7:0]
Standby mode	x	1	1	x	x	x	x	x	High-z	High-z
8-bit read from common memory	1 1 1	1 1 0	0 0 1	0 1 x	0 0 0	1 1 1	1 1 1	1 1 1	High-z High-z Odd-Byte[1]	Even-Byte Odd-Byte High-z[1]
16-bit read from common memory	1	0	0	x	0	1	1	1	Odd-Byte	Even-Byte
8-bit write to common memory	1 1 1	1 1 0	0 0 1	0 1 x	1 1 1	0 0 0	1 1 1	1 1 1	xxx xxx Odd-Byte[1]	Even-Byte Odd-Byte xxx[1]
16-bit write to common memory	1	0	0	x	1	0	1	1	Odd-Byte	Even-Byte
8-bit read from attribute memory	0 0 0	1 1 0	0 0 0	0 1 x	0 0 0	1 1 1	1 1 1	1 1 1	High-z High-z Not-valid	Even-Byte Not-valid Even-Byte
16-bit read from attribute memory	0	0	0	x	0	1	1	1	Not-valid	Even-Byte
8-bit write to attribute memory	0 0	1 0	0 0	0 x	1 1	0 0	1 1	1 1	xxx xxx	Even-Byte Even-Byte

Table 21-20. Data and Control Signal Relations (continued)

Function Mode	REG	CE2	CE1	A0	OE	WE	IORD	IOWR	D[15:8]	D[7:0]
8-bit read from I/O	0 0 0	1 1 0	0 0 1	0 1 x	1 1 1	1 1 1	0 0 0	1 1 1	High-z High-z Odd-Byte[1]	Even-Byte Odd-Byte High-z[1]
16-bit read from I/O	0	0	0	x	1	1	0	1	Odd-Byte	Even-Byte
8-bit write to I/O	0 0 0	1 1 0	0 0 1	0 1 x	1 1 1	1 1 1	1 1 1	0 0 0	xxx xxx Odd-Byte	Even-Byte Odd-Byte xxx
16-bit write to I/O	0	0	0	x	1	1	1	0	Odd-Byte	Even-Byte
I/O inhibit	1	x	x	x	x	x	0	1	High-z	High-z

[1] Note these are all the access modes which are supported by the standard. In the PCMCIA controller, the 8-bit access to odd byte is done by CE1 and A0 only, that is, the data will be always driven on D[7:0].

21.5.10 True IDE Mode Access

In True IDE mode windows, the selection of either task file or alt reg is made by haddr[3]

Haddr[3] = 0—will yield an access to task file or data register

Haddr[3] = 1—will yield a write to control register or read of Alt. status register

Table 21-21 shows data, control and address relations in TrueIDE mode.

Table 21-21. Data, Control and Address Relations in TrueIDE Mode

Function mode	CE2	CE1	A0-3	IORD	IOWR	D[15:8]	D[7:0]
Standby mode	1	1	xx	x	x	High-z	High-z
Task file Write	1	0	1-7h	1	0	xxx	Data In
Task file Read	1	0	1-7h	0	1	High-z	Data Out
Data Register Write	1	0	0	1	0	Odd-Byte	Even-Byte
Data Register Read	1	0	0	0	1	Odd-Byte	Even-Byte
Control Register Write	0	1	6h	1	0	xxx	Data In
Alt Status Read	0	1	6h	0	1	High-z	Data out
Invalid Mode	0	0	x	x	x	High-z	High-z

21.5.11 Card Extraction

When the card is extracted the PCMCIA controller’s registers are not reset. The registers settings remain the same as before the card’s extraction. This allows the host software to quickly activate the card once the CIS indicates that it is the same card.

21.5.12 TrueIDE Support

The ATA standard specifies the AT attachment interface between host systems and storage devices. The PCMCIA controller can be dynamically configured to support a PCMCIA-compatible ATA disk interface (commonly known as IDE) instead of the standard PCMCIA card interface. Using the TrueIDE interface on the PCMCIA controller changes the function of some card socket signals to support the needs of the ATA disk interface. The TrueIDE signals assignment on the PCMCIA connector is described in Table 21-22.

Table 21-22. PCMCIA Card TrueIDE Signal Names and Assignments

PC Card Signal	TrueIDE	Comment
D[15:0]	D[15:0]	
$\overline{CE1}$	$\overline{CS0}$[1]	Task file register select in TrueIDE mode.
$\overline{CE2}$	$\overline{CS1}$[1]	Alternate status register select in TrueIDE mode.
$\overline{OE}$	ATASEL[2]	The CF card samples this bit on power-on sequence. If low, the card will enter TrueIDE mode, else it will enter PC CARD mode.
A[2:0]	A[2:0]	Address
A[10:3]	not used	These bits should be connected to ‘0’ on TrueIDE
$\overline{WE}$	$\overline{WE}$	
READY/$\overline{IREQ}$	INTRQ	Interrupt request—INTRQ is the ATA notation and is asserted HIGH.
WP/$\overline{IOIS16}$	$\overline{IOCS16}$	
$\overline{CD1}$	$\overline{CD1}$	
$\overline{CD2}$	$\overline{CD2}$	
$\overline{VS1}$	$\overline{VS1}$	
$\overline{VS2}$	$\overline{VS2}$	
RESET	$\overline{RESET}$	This signal is asserted LOW in ATA mode, HIGH in other modes.
$\overline{WAIT}$	IOCHRDY	IO Channel Ready—asserted HIGH—polarity inversion of $\overline{WAIT}$.
$\overline{REG}$	not used	this bit should be connected to ‘1’ on TrueIDE
BVD1/$\overline{STSCHG}$	$\overline{PDIAG}$	Diagnostics complete signal.
BVD2/$\overline{SPKR}$	$\overline{DASP}$	Disk Active
$\overline{IORD}$	$\overline{IORD}$	
$\overline{IOWR}$	$\overline{IOWR}$	

1 In TrueIDE mode, #CS0 and #CS1 (task file chip select) behave differently from #CE1 and #CE2 (byte lane chip selects)

2 Dynamic change of ATASEL is not supported since it requires power up of the card. Therefore the ATASEL pin of the card will grounded in the socket.

21.5.13 Endianness Support

The PCMCIA controller supports Big and Little Endian. The Endianness is defined according to the Endianness input pin to the module. This input should be tied high or low by the chip integrator. Connecting the input to “1” means Big Endian.Connecting the input to “0” means Little Endian. Dynamic Endianness is not supported.

21.6 Timing Diagrams

Figure 21-11 and Figure 21-12 show PCMCIA typical accesses.

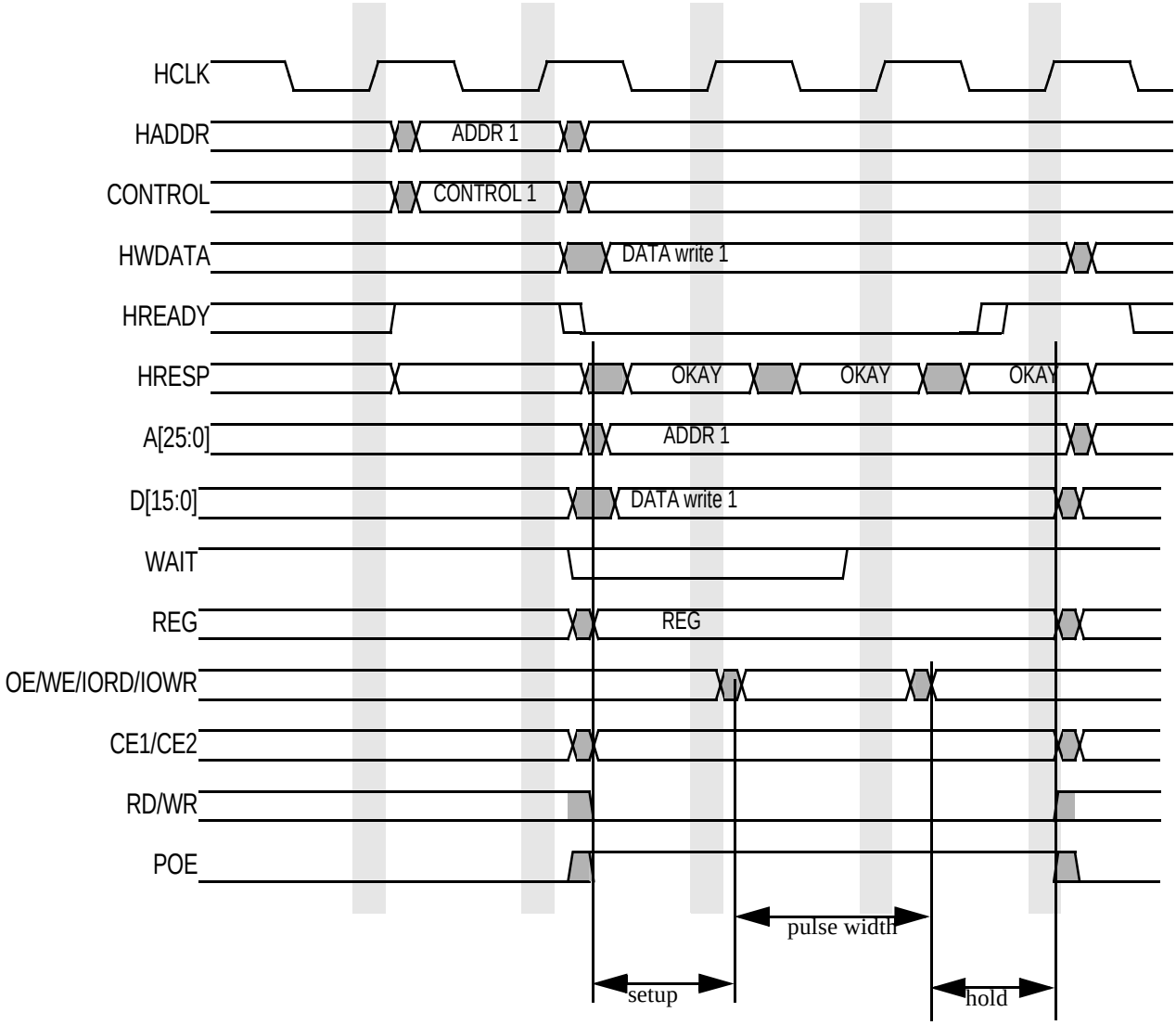


Figure 21-11. Write Accesses PSHT=1, PSST =1

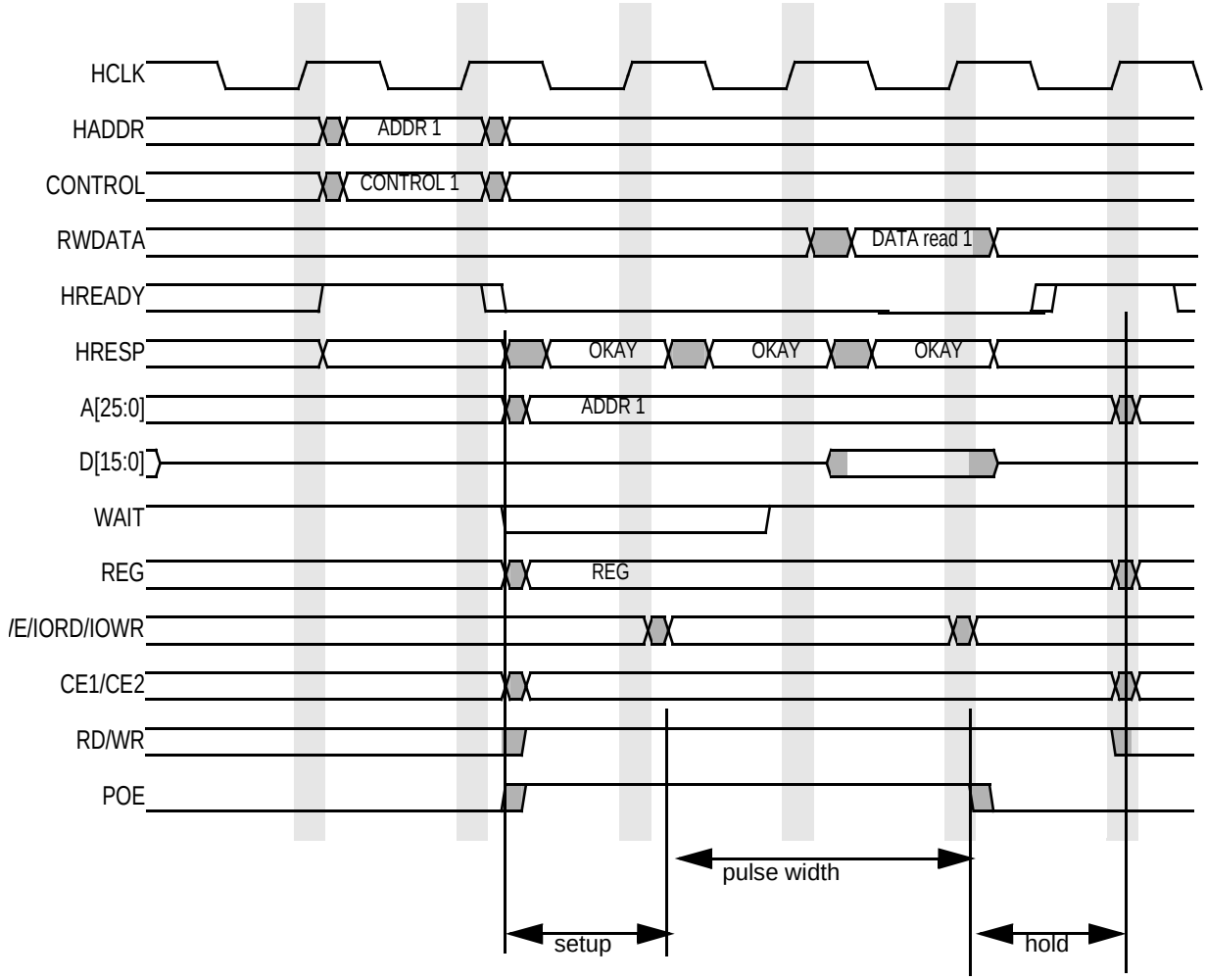


Figure 21-12. Read Cycle PSHT=1, PSST =1

Book II, Part 5: Connectivity Peripherals

Introduction

The i.MX31 and i.MX31L contain the following modules that provide communication with a variety of different peripheral using several different interfaces:

1-Wire Module

The 1-Wire module provides bidirectional communication between the ARM11 core and the Add-Only-Memory EPROM (DS2502). The 1-kilobit EPROM is used to hold battery information and communicate with the ARM11 Platform using the IP interface. The ARM11 (through the 1-Wire interface) acts as the bus master and the DS2502 device is the slave. The 1-Wire peripheral does not trigger interrupts; hence, it is necessary for the ARM11 to poll of the 1-Wire to manage the module. The 1-Wire uses an external pin (to connect to the DS2502). Timing requirements are met in hardware with the help of a 1-MHz clock. The clock divider generates a 1-MHz clock that is used as time reference by the state machine. Timing requirements are crucial for proper operation, and the 1-Wire state machine and the internal clock provide the necessary signal.

Advanced Technology Attachment (ATA)

The Advanced Technology Attachment (ATA) module provides an AT attachment host interface for the MX31. Its main use is to provide an interface with IDE hard disc drives and ATAPI optical disc drives. It

interfaces with the ATA device using industry standard ATA signals. The ATA interface is compliant to the ATA-6 standard, and supports following ATA standard protocols:

- PIO mode 0, 1, 2, 3, and 4
- Multiword DMA mode 0, 1, and 2
- Ultra DMA modes 0, 1, 2, 3, and 4 with a bus clock of 50 MHz or higher
- Ultra DMA modes 5 with bus clock of 80 MHz or higher

The ATA interface has 2 buses connected to it. The CPU bus provides communication with the ARM11 host processor and the DMA bus provides communication between the ATA module and the host DMA unit. All internal ATA registers are visible from both buses, allowing Smart Direct Memory Access (SDMA) access to program the interface.

There are basically two protocols that can be active at the same time on the ATA bus. The first and simplest protocol (PIO mode access) can be started at any time by either the ARM11 or the host SDMA to the ATA bus. The PIO mode is a slow protocol, mainly intended to be used to program an ATA disc drive, but also possible to use to transfer data to/from the disc drive.

The second protocol is the DMA mode access. DMA mode is started by the ATA interface after receiving a DMA request from the drive, and only if the ATA interface has been programmed to accept the DMA request. In DMA mode, either multiword DMA or ultra DMA protocol is used on the ATA bus. All transfers between FIFO and host IP or DMA IP bus are zero wait states transfer, so high speed transfer between FIFO and DMA/host bus is possible.

Configurable Serial Peripheral Interface (CSPI)

There are three identical CSPI modules in the i.MX31 and i.MX31L ICs. Each CSPI is equipped with data FIFOs, and is a master/slave configurable serial peripheral interface module, capable of interfacing to both SPI master and slave devices. The CSPI Ready ($\overline{\text{SPI\_RDY}}$) and Chip Select ($\overline{\text{SS}}$) control signals enable fast data communication with fewer software interrupts.

The CSPI is used for fast data communication with fewer software interrupts. It includes the following features:

- Full-duplex synchronous serial interface
- Master/Slave configurable
- Four chip selects to support multiple peripherals
- Transfer continuation function allows unlimited length data transfers
- 32-bit wide by 8-entry FIFO for both transmit and receive data
- Polarity and phase of the Chip Select ($\overline{\text{SS}}$) and SPI Clock (SCLK) are configurable
- DMA support

Fast Infrared Interface (FIR) Module

The Fast Infrared Interface (FIR) module provides support for infrared communication. The FIR is capable of establishing a 0.576 Mbit/s, 1.152 Mbit/s or 4 Mbit/s half duplex link via a LED and IR detector. It supports 0.576 Mbit/s, 1.152 Mbit/s Medium Infrared (MIR) physical layer protocol and 4Mbit/s Fast

Infrared (FIR) physical layer protocol defined by IrDA, version 1.4. In addition, the Serial Infrared (SIR) protocol, which supports data rate 115.2kbps or lower, is implemented in UART module. The FIR interface signals are multiplexed with the UART counterpart signals via GPIO configuration for a complete Infrared Interface supporting SIR, MIR, and FIR modes.

Inter-Integrated Circuit Bus (I^2C)

The Inter-Integrated Circuit Bus (I^2C) module provides a serial interface for controlling the Sensor Interface and other external devices. Data rates of up to 100 kbps are supported.

The I^2C module provides functionality of a standard I^2C slave and master. The I^2C module is designed to be compatible with the standard Phillips I^2C bus protocol. The I^2C is a two-wire, bidirectional serial bus that provides a simple, efficient method of data exchange, minimizing the interconnection between devices. This bus is suitable for applications requiring occasional communications over a short distance between many devices. The flexible I^2C allows additional devices to be connected to the bus for expansion and system development.

Keypad Port (KPP)

The Keypad Port (KPP) is designed to interface with keypad matrix with 2-contact or 3-point contact keys. The KPP is designed to simplify the software task of scanning a keypad matrix. With appropriate software support, the KPP is capable of detecting, debouncing, and decoding one or multiple keys pressed simultaneously in the keypad. The KPP supports up to 8 x 8 external key pad matrices. Its port pins can be used as general purpose I/O. Using an open drain design, the KPP includes glitch suppression circuit design, multiple keys, long key, and standby key detection.

Secure Digital Host Controller (SDHC)

The Security Digital Host Controller (SDHC) integrates both MultiMediaCard (MMC) support along with Secure Digital (SD) memory and I/O functions, including SD memory and I/O combo card.

The Multi Media Card (MMC), is a universal low cost data storage and communication media that is designed to cover a wide area of applications as, among others, electronic toys, organizers, PDAs, and smart phones.

The Secure Digital Card (SD), is an evolution of MMC technology, with two additional pins in the form factor. It is specifically designed to meet the security, capacity, performance, and environment requirement inherent in newly emerging audio and video consumer electronic devices.

Subscriber Identification Module (SIM)

The SIM Interface Module (SIM) is designed to facilitate communication to SIM cards or Eurochip pre-paid phone cards. The SIM module has two ports that can be used to interface with the various cards. The interface with the MCU is via a 16-bit connection, The SIM module I/O interface can be operated in one of three modes of operation.

Two-wire interface

In this mode, both the IC pin RX and IC pin TX are used to interface to the SmartCard. This is activated by resetting the 3VOLT bit in the port control register to a “0”.

External 1-Wire interface

In this mode the IC pins RX and TX are tied together external to the IC and routed to the smartcard. The 3volt bit in the port control register is reset to a “0” and the OD bit in the OD_CONFIG register is set to a “1”. For this interface to work properly the IC pin (RX-TX) must be pulled high by a resistor. The value should be selected small enough to give a fast enough rise time.

Internal 1-Wire interface

In this mode the IC pin TX is routed to the SmartCard. The receive pin RX is connected to the TX pin internal to the IC. The 3VOLT bit in the port control register is reset to a “1” and the OD bit in the OD_CONFIG register is set to a “1”. For this interface to work properly the IC pin TX must be pulled high by a resistor. The value should be selected small enough to give a fast enough rise time.

Universal Asynchronous Receiver Transmitter (UART)

The Universal Asynchronous Receiver Transmitter (UART) provides serial communication capability with external devices through an RS-232 cable or through use of external circuitry that converts infrared signals to electrical signals (for reception), or transforms electrical signals to signals that drive an infrared LED (for transmission) to provide low speed IrDA compatibility.

The i.MX31 and i.MX31L contain five UART modules. Each UART module is capable of standard RS-232 non-return-to-zero (NRZ) encoding format and IrDA-compatible infrared modes.

The UART transmits and receives characters containing either 7 or 8 bits (program selectable). To transmit, data is written from the IP data bus (SkyBlue line interface) to a 32-byte transmitter FIFO (TxFIFO). This data is passed to the shift register and shifted serially out on the transmitter pin (TXD). To receive, data is received serially from the receiver pin (RXD) and stored in a 32-halfwords-deep receiver FIFO (RxFIFO). The received data is retrieved from the RxFIFO on the IP data bus. The RxFIFO and TxFIFO generate maskable interrupts as well as DMA Requests when the data level in each of the FIFO reaches a programmed threshold level.

Universal Serial Bus, On-The-Go (USBOTG), High-Speed

The i.MX31 and i.MX31L use a Universal Serial Bus, On-The-Go (USBOTG) module that provides all of the functionality required to support three independent USB ports, compatible with the USB 2.0 specification. In addition to the normal USB functionality, the module also provides support for direct connections to on-board USB peripherals and supports multiple interface types for serial transceivers. The USB module provides high performance USB On-The-Go (OTG) functionality, compliant with the USB 2.0 specification, the OTG supplement, and the ULPI 1.0 Low Pin Count specification.

Chapter 22
1-Wire Interface (1-Wire)

The 1-Wire interface provides the communication line to most of 1-Wire devices working at standard speed. The interface sends or receives one bit at a time. The required protocol as well as standard timings for accessing the device are defined by Maxim-Dallas Semiconductor.

22.1 Overview

The 1-Wire module is a peripheral device to the core and communicates with it using the IP interface. This chapter describes the 1-Wire module, as well as 1-Wire mechanism and its timing diagrams

22.2 Features

Figure 22-1 shows a block-level description of the 1-Wire module.

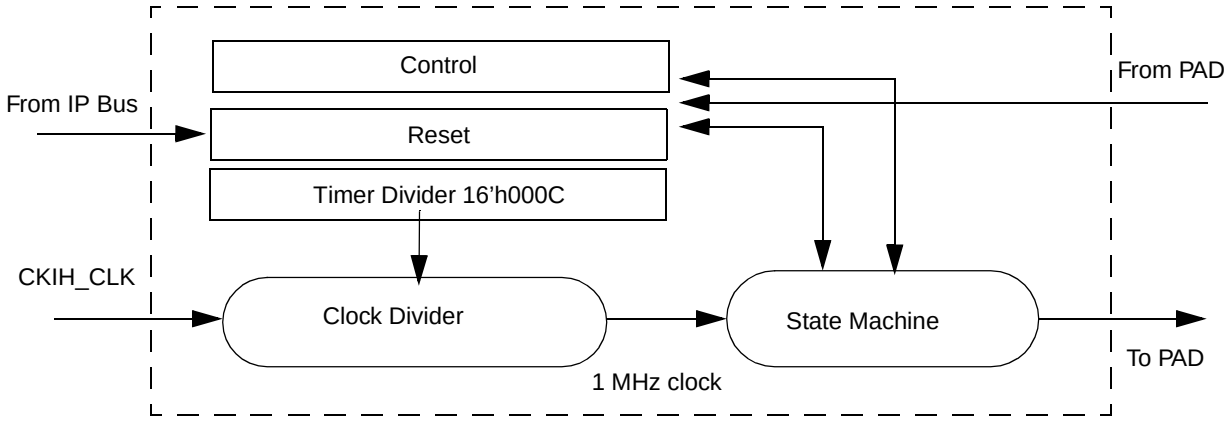


Figure 22-1. 1-Wire Block Diagram

The clock divider generates a 1-MHz clock that is used as a time reference by the state machine. Transitions between the states of the state machine as well as actions triggered at precise time deadlines are expressed using the 1-MHz clock. The state machine performs all required actions to dialog with the external device.

See Figure 22-2 for an overview of the connections for the 1-Wire module.

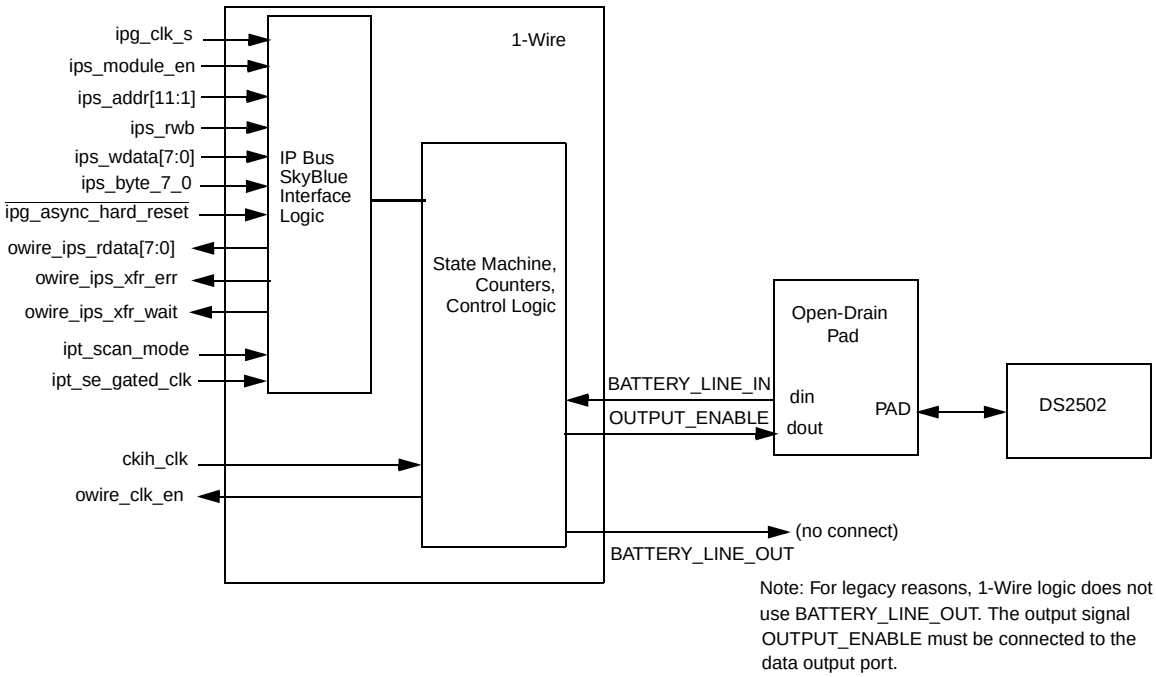


Figure 22-2. 1-Wire Connections

22.3 External Signal Descriptions

The 1-Wire needs an external pin to connect to the 1-Wire device. This pin, named BATT_LINE, has an open drain type, and can be used as a GPIO.

22.4 Memory Map and Register Definition

The 1-Wire includes three 16-bit registers. Section 22.4.3, “Register Descriptions” provides the detailed descriptions for all of the 1-Wire registers.

22.4.1 Memory Map

See Table 22-2 for the 1-Wire memory map.

Table 22-2. 1-Wire Memory Map

Address	Register	Reset Value	Access	Section/Page
0x43F9_C000 (CONTROL)	Control register	0x0000	R/W	22.4.3.1/22-4
0x43F9_C002 (TIME_DIVIDER)	Time divider register	0x0000	R/W	22.4.3.2/22-5
0x43F9_C004 (RESET)	Reset register	0x0000	R/W	22.4.3.3/22-7

22.4.2 Register Summary

Figure 22-3 shows the key to the register fields, and Table 22-3 shows the register figure conventions.

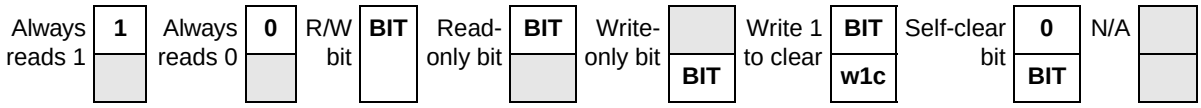


Figure 22-3. Key to Register Fields

Table 22-3. Register Figure Conventions Key to Register Fields

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 22-4 shows the 1-Wire register summary.

Table 22-4. 1-Wire Register Summary

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x43F9_C000 (CONTROL)	R	0	0	0	0	0	0	0	0	RPP	PST	WR 0	WR 1	RDST	0	0	0
	W																
0x43F9_C002 (TIME_DIVIDER)	R	0	0	0	0	0	0	0	0	DVDR							
	W																
0x43F9_C004 (RESET)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RESET
	W																

NOTE

All registers are byte-writable.

22.4.3 Register Descriptions

22.4.3.1 Control Register (CONTROL)

The control register updates the status of the reset, presence, write0, write1, and read bits. when read, this register lets the user know whether the device (1-Wire) is connected.

Figure 22-4 shows the CONTROL register, and Table 22-5 shows the register's field descriptions.

0x43F9_C000 (CONTROL) Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	RPP	PST	WR0	WR1	RDST	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 22-4. Control Register

Table 22-5. Control Register Field Descriptions

Field	Description
15–8	Reserved
7 RPP	Reset presence pulse. This bit is self-clearing and is cleared after the presence is determined. 0 Do nothing./Reset pulse complete. 1 Generate reset pulse and sample for DS2502 presence pulse.
6 PST	Presence status. This bit is valid after the RPP bit is self-cleared. 0 Device is not present. 1 Device is present.
5 WR0	Write 0. This bit is self-clearing and is cleared when the write of the bit is complete. 0 Do nothing./ Write sequence complete. 1 Write a 0 bit to the interface.
4 WR1	Write 1/ Read. This bit is self-clearing and is cleared when the write of the bit is complete. This reads a bit, since the Write 1 and Read timing are identical. The value of the read bit is stored in RDST, and is valid after WR1/RD is self-cleared. 0 Do nothing./Write sequence complete. 1 Write a 1 bit to the interface.
3 RDST	Read status. This bit is valid after the WR1/RD bit is self cleared. 0 A 0 was sampled during a read. 1 A 1 was sampled during a read.
2–0	Reserved

22.4.3.2 Time Divider Register (TIME_DIVIDER)

The time divider register is used to program the pre-divider factors for dividing the ckih_clk down to 1 MHz.

Figure 22-5 shows the TIME_DIVIDER register, and Table 22-6 shows the register's field descriptions. Table 22-7 shows the system timing requirements.

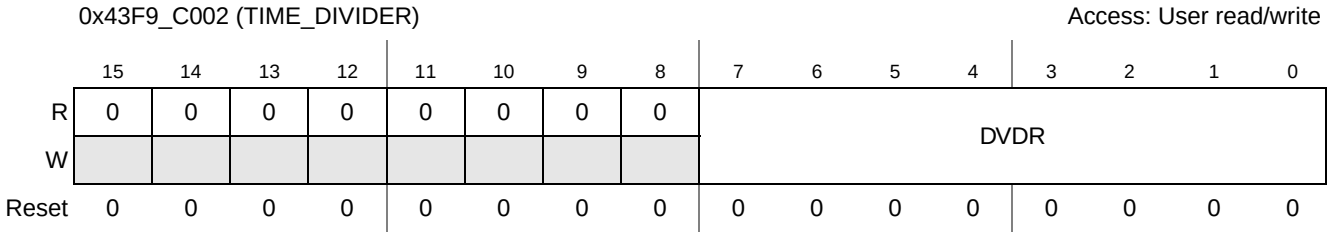


Figure 22-5. Time Divider Register

Table 22-6. Time Divider Register Field Descriptions

Field	Description
15–8	Reserved
7–0 DVDR	Pre-divider factor. The 1-Wire also contains a clock divider register used to generate the internal time base within the module. Internal time generation is made up by a clock divider. The purpose of this internal time generation is to make a 1 MHz clock from the main clock. It is the user's responsibility to program this register so that the binary rate frequency is as close as possible to 1 MHz (1 /(divider+1). If the clock frequency is 30 MHz, then the proper value to write to the divider register is 29. $00 1 (default) $01 2 - - - - - - $FF 256

NOTE

The precision of the generated clock is very important to get a proper behavior of the 1-Wire module. This module is based on a state machine that undertakes actions at defined times.

Table 22-7. System Timing Requirements

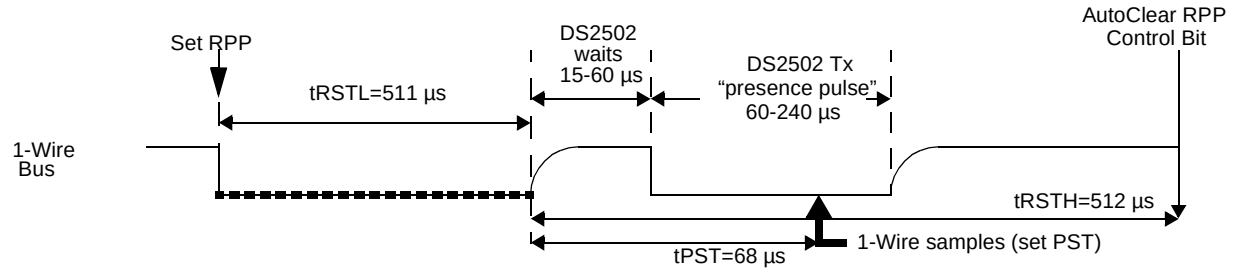
Timing	Values (µs)	Min. (µs)	Max. (µs)	Absolute Precision	Relative Precision
t_{RSTL}	511	480	—	31	0.0645
t_{PST}	68	60	75	7	0.1
t_{RSTH}	512	480	—	32	0.0645
t_{LOW0}	100	60	120	20	0.2
t_{LOWR}	5	1	15	4	0.8
t_{READ_sample}	13		15	2	0.15

The most stringent constraint is 0.0645 as a relative time imprecision. The time relative precision is directly derived from the frequency of the source clock (f).

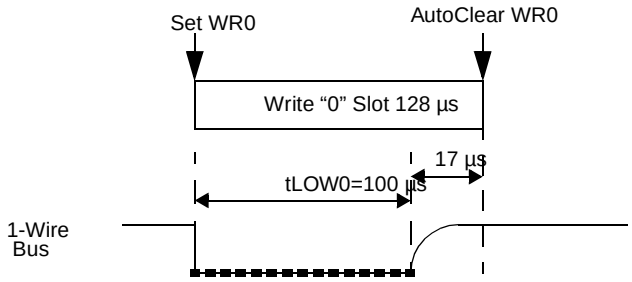
Time Relative Precision = 1/f -1 = divider/clock (MHz) - 1 ***Eqn. 22-1***

See Figure 22-6 and Table 22-8 for examples of relative time precision for different main clock frequencies.

RESET and PRESENCE



WRITE 0



READ

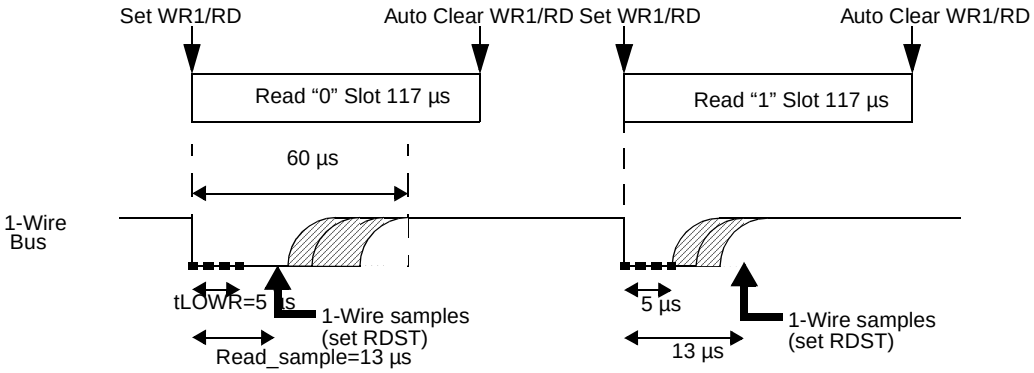


Figure 22-6. Time Precision for Reset and Presence, Write0, and Read Pulses

When using the 1-Wire module, note examples of relative time imprecision listed in Table 22-8. If the main clock is an exact integer multiple of 1 MHz, then the generated frequency is exactly 1 MHz.

Table 22-8. Examples of Relative Time Imprecision

Main Clock (CKIH_CLK) Frequency (MHz)	13	16.8	19.44
Clock Divide Ratio	13	17	19
Generated Frequency (MHz)	1	0.9882	1.023
Relative Time Imprecision	0	0.0117	0.023

NOTE

A main clock frequency at 10 MHz causes improper function of the module.

22.4.3.3 Reset Register (RESET)

The reset register is used to do a software reset of the 1-Wire module.

Figure 22-7 shows the RESET register, and Table 22-9 shows the register's field descriptions.

0x43F9_C004 (RESET) Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RST
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 22-7. Reset Register

Table 22-9. Reset Register Field Descriptions

Field	Description
15–1	Reserved
0 RST	Software reset. The reset register is used to reset the module using the software. This register is not self-clearing; therefore, the programmer must write a '1' to reset the registers and then write a '0' to release the reset signal. 0 1-Wire is out of reset. 1 1-Wire is in reset.

22.5 Functional Description

The following sections describe the functional operation and modes of the 1-Wire module.

22.5.1 Low Power Modes

The 1-Wire goes into low power mode whenever it is not in use. It goes into low power mode by gating off the clock. The 1-Wire is considered to not be in use whenever the 1-Wire is not communicating with the DS2502. In other words, the 1-Wire goes into low power mode when the RPP, WR0, and WR1 bits of the control register are all zero. As soon as you write to any of those bits, the 1-Wire exits low power mode.

22.5.2 Reset Sequence with Reset Pulse Presence Pulse

To begin any communications with the DS2502, an initialization procedure must be issued. A reset pulse must be generated, and then a presence pulse must be detected. The minimum reset pulse length is 480 µs. The bus master (1-Wire) generates this pulse. After the DS2502 detects a rising edge on the 1-Wire bus, the DS2502 waits 15-60 µs before it transmits back a presence pulse. The presence pulse lasts 60-240 µs. See Figure 22-8 for the timing diagram for this sequence.

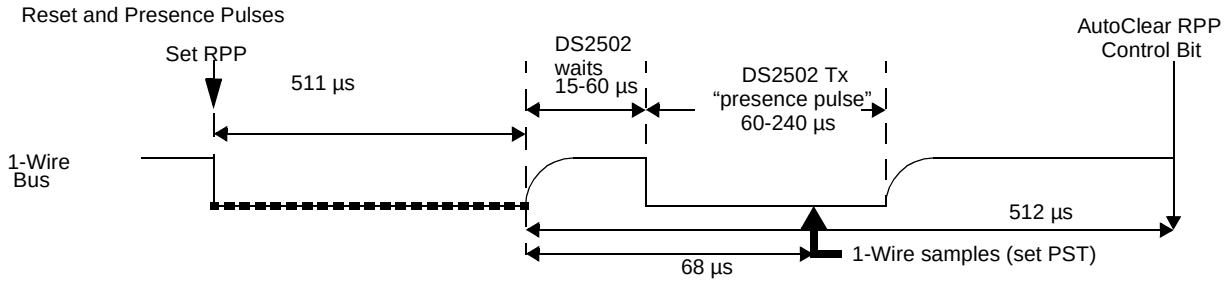


Figure 22-8. 1-Wire Initialization

The reset pulse begins the initialization sequence. It is initiated when the RPP bit in the control register (Control[RPP]) is set. When the presence bit is detected, this bit is cleared. The presence pulse is used by the bus master to determine if at least one DS2502 is connected. The software determines if more than one DS2502 exists. The 1-Wire samples for the DS2502 presence bit. The presence bit is latched in the 1-Wire control register PST bit (Control[PST]). When the PST bit is set to a ‘1’, it means that a DS2502 is present. If the bit is set to a ‘0’, then no device was found.

22.5.3 Write 0

The Write 0 function simply writes a zero bit to the DS2502. See Figure 22-9 for the timing diagram for the sequence. The sequence takes 117 µs. The 1-Wire bus is held low for 100 µs.

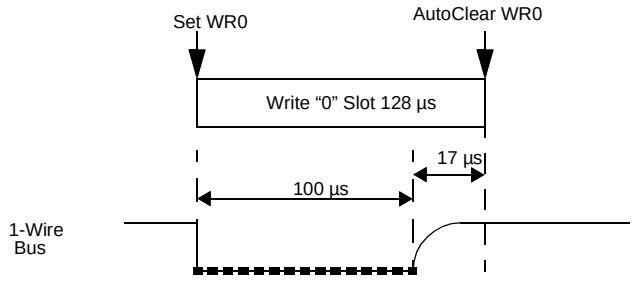


Figure 22-9. Write 0 Timing

The Write 0 pulse sequence is initiated when the WR0 control bit is set in the control register (Control[WR0]). When the write is complete, the WR0 bit is automatically cleared.

22.5.4 Write 1/Read Data

The Write 1 and Read timing is identical. The 1-Wire bus is first driven low from 1 µs to 15 µs. According to the DS2502 documentation, the DS2502 has a delay circuit which is used to synchronize the DS2502 with the bus master (1-Wire). This delay circuit is triggered off the falling edge of the data line and is used to decide when the DS2502 should sample the line. In the case of a Write 1, the 1-Wire bus is released after 60 µs, allowing the DS2502 to sample the 1-Wire bus. For Write 1, the sampling window for 1-Wire bus is 15 µs to 60 µs. The timing is the same for Reads, except when a read 0 slot is issued, the delay circuit holds the data line low to override the 1 generated by the bus master (1-Wire). For Reads, the 1-Wire bus is sampled nominally 13 µs after the read operation started.

See Figure 22-10 and Figure 22-11 for timing diagrams for Write 1 and Read sequences.

For Write 1 or Read, the write1/read bit (Control[WR1/RD]) is set and auto-cleared when the sequence has been completed. For a Read, the read status bit (Control[RDST]) is set to the value of the read.

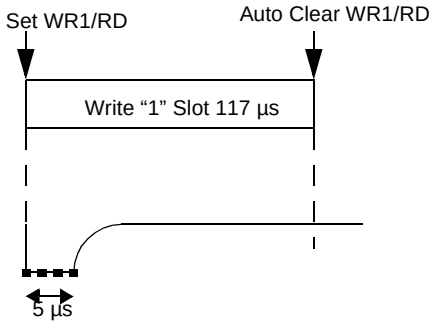


Figure 22-10. Write 1 Timing

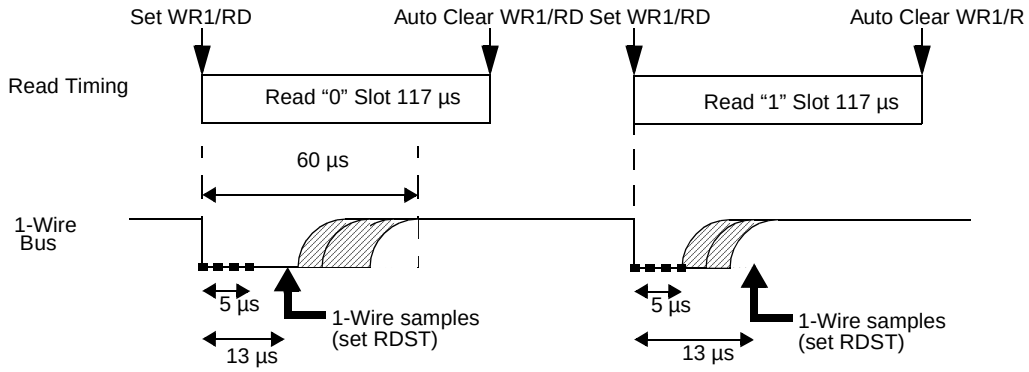


Figure 22-11. Read Timing

22.5.5 1-Wire Clock Path

The ckih_clk signal in the 1-Wire module is actually the ipg_clk coming from the Clock Controller Module (CCM). This clock uses mcu_main_clk as source clock through hclk_divider and ipg_divider. Please refer to Chapter 3, “Clocks, Power Management and Reset (AP Clock Controller Module)” for mode details.

In a typical case the MCU_PLL is used to generate mcu_main_clk through MCU clock switch unit, and deliver a 399 MHz clock.

The hclk_divider is set at 0x2 to divide by 3 => hclk = 133 MHz.

And, ipg_divider is set at 0x1 to divide by 2 => ipg_clk = ckih_clk = 66.5 MHz.

Finally, it is necessary to set TIMER_DIVIDER.DVDR bits at 65 to divide by 66 => 1 MHz_clk = 1.0075 MHz. The error on the 1 MHz clock is acceptable based on the values shown in Table 22-7.

Chapter 23
Advanced Technology Attachment (ATA)

The Advanced Technology Attachment (ATA) module is an AT attachment host interface. Its main use is to interface with hard disc drives and optical disc drives. It interfaces with the ATA device over a number of ATA signals. It is possible to connect a bus buffer between the host side and the device side. In this case, the ata_buffer_en signal should be used to control the direction the buffer is driving. If ata_buffer_en is high, it drives outward to the device. If ata_buffer_en is low, it drives inward to the host. Figure 23-1 shows the block diagram of the ATA interface.

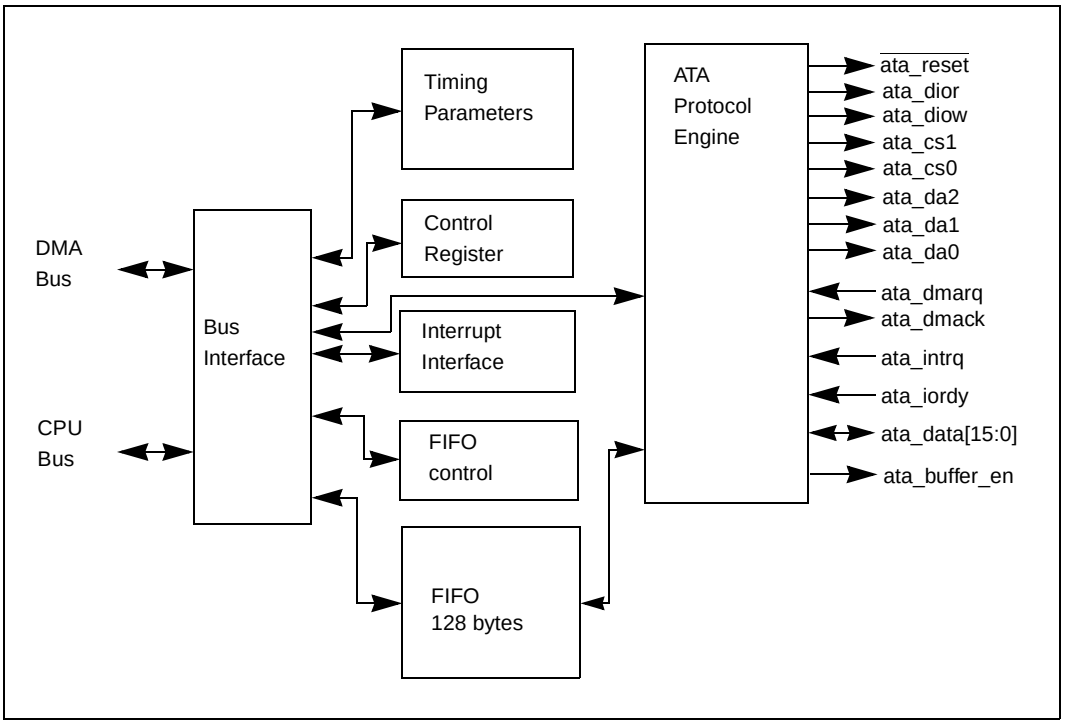


Figure 23-1. ATA interface Block Diagram.

23.1 Overview

The ATA module is an AT attachment host interface. Its main use is to interface with IDE hard disc drives and ATAPI optical disc drives. It interfaces with the ATA device over a number of ATA signals.

The ATA interface is compliant to the ATA-6 standard and supports the following protocols:

- PIO mode 0, 1, 2, 3, and 4
- Multiword DMA mode 0, 1, and 2

- Ultra DMA modes 0, 1, 2, 3, and 4 with bus clock of 50 MHz or higher
- Ultra DMA mode 5 with bus clock of 80 MHz or higher

The ATA interface has two buses connected:

- One CPU bus for communication with the host processor
- One DMA bus for communication with the host DMA unit

All internal registers are visible from both buses, allowing SDMA access to program the interface.

Before accessing the ATA bus, the host must program the timing parameters to be used on the ATA bus. The timing parameters control the timing on the ATA bus. Most timing parameters are programmable as a number of clock cycles (1 to 255). Some are implied.

After programming the timing parameters, there are two protocols that can be active at the same time on the ATA bus, as follows:

- First protocol:

  This protocol is a PIO mode access that can be performed at any time by the host CPU or the host SDMA to the ATA bus. During PIO mode access, the incoming IP bus cycle is translated to an ATA bus cycle by the ATA protocol engine. The IP bus cycle is stalled until completion of the ATA bus cycle on read, or until putting the write data on the ATA bus on write. The PIO mode is a slow protocol, mainly intended to program the ATA disc drive, but also possibly to use to transfer data to/from the disc drive. During PIO mode, the FIFO is not active.
- Second protocol:

  This protocol is the DMA mode access. DMA mode is started by the ATA interface it receives a DMA request from the drive, and only if the ATA interface has been programmed to accept the DMA request. In DMA mode, either multiword DMA or ultra DMA protocol is used on the ATA bus. Once started, data transfer is organized between the ATA bus and the FIFO. Data transfer pauses to prevent FIFO overflow/underflow. Data transfer resumes when there is space again in the FIFO, or when the FIFO has been refilled.

  During DMA transfer, there is no direct interface between the ATA bus and the host IP or host DMA IP bus. Instead, the transfer takes place between the ATA bus and the FIFO; the FIFO informs the host DMA unit when it needs to be refilled or emptied. In this case, it sends an ALARM flag to the host DMA. When the host DMA receives the fifo_tx_alarm, it should write some data to the FIFO. (typically 32 bytes). When the host DMA receives the fifo_rcv_alarm, it should read some data from the FIFO (typically 32 bytes).

  The FIFO filling level at which the alarms are produced is programmable. For completeness, there is a third alarm associated with the host DMA operation fifo_txfer_end_alarm. This alarm signals the end of the transfer, and requests the SDMA to take steps to complete the transfer: transfer the bytes remaining in the FIFO to the host memory, and inform the host CPU the transfer is completed.

All transfers between FIFO and host IP or DMA IP bus are zero wait states transfer, so high speed transfer between FIFO and DMA/host bus is possible.

When a PIO access is performed during a running DMA transfer, the DMA transfer is paused, the PIO access done, and the DMA transfer resumes again.

23.1.1 Features

The ATA interface includes the following features:

- Has programmable timing on the ATA bus. Works with wide range of bus frequencies.
- Is compliant with ATA-6 specification
  - Supports PIO modes 0, 1, 2, 3, and 4
  - Supports multiword DMA modes 0, 1, and 2.
  - Supports ultra DMA modes 0, 1, 2, 3, and 4 with bus clock of at least 50 MHz.
  - Supports ultra DMA mode 5 with bus clock of at least 80 MHz.
- Can be used with off-chip bus transceiver if pads are not compliant with ATA voltage levels.
- Is 128-byte FIFO part of interface
- Has FIFO receive alarm, FIFO transmit alarm, and FIFO end of transmission alarm to DMA unit
- Performs zero-wait cycles transfer between DMA bus and FIFO allows fast FIFO reading/writing

23.1.2 Modes of Operation

The interface offers two operation modes that can be used together, as follows:

- PIO:

  An access to the ATA bus in PIO mode occurs whenever an ATA PIO register is read or written to by the host CPU or the host (smart) DMA unit. During a PIO transfer, the incoming IP bus cycle is translated to an ATA PIO bus cycle by the ATA protocol engine. No buffering of data occurs, so the host CPU or host DMA cycle is stalled until the ATA bus read data is available on read, or is stalled until the IP bus data can be put on the ATA bus during write.

  PIO accesses can be done to the bus at any time, even during a running ATA DMA transfer. In this case, the DMA transfer is paused, the PIO cycle is completed, and the DMA transfer is resumed.

- DMA:

  In DMA mode, data is transferred between the ATA bus and the FIFO. Two different DMA protocols are supported on the ATA bus: ultra DMA mode and multiword DMA mode. Selection is made using a control register bit.

  A DMA transfer is started when DMA mode transfer has been enabled by writing some control bit, and when the drive connected to the ATA bus pulls its DMARQ line high.

  During an ATA bus DMA transfer, data is transferred between the ATA bus and the FIFO. The transfer pauses to avoid FIFO overflow/underflow.

  It is the task of the host CPU or the host SDMA unit to read data or write data to the FIFO to keep the transfer going. Normal set-up is that the host (smart) DMA unit does this task. For this purpose, the fifo_rcv_alarm and fifo_tx_alarm signals are sent to the host DMA unit.

  The signal—fifo_rcv_alarm—informs the host DMA unit that there is at least one packet of data waiting in the FIFO to be read by the host DMA. Whenever this signal is high, the host DMA should transfer one packet of data from the FIFO to the main memory. Typical packet size is 32 bytes (8 long words), but other packet sizes can be handled, too.

The signal—fifo_tx_alarm—informs the host DMA unit that there is space for at least one packet to be written by the host DMA. Whenever this signal is high, the host DMA should transfer one packet of data from main memory to the FIFO.The typical packet size is 32 bytes (8 long words), but other packet sizes can be handled, too.

23.2 External Signal Description

Table 23-1 provides the list of signals that enter and exit this module to peripherals within the i.MX31 and i.MX31L chips.

Table 23-1. Signal Properties

Name	Port	Function	Reset State	Type
$\overline{\text{ata\_reset}}$		ATA bus reset signal. Active low. If active, ATA device is reset[1]	0	out
ata_dior		ATA bus read strobe	1	out
ata_diow		ATA bus write strobe	1	out
ata_cs1		ATA bus chip select 1	1	out
ata_cs0		ATA bus chip select 0	1	out
ata_da2		ATA bus address line 2	0	out
ata_da1		ATA bus address line 1	0	out
ata_da0		ATA bus address line 0	0	out
ata_dmarq		ATA bus DMA request	-	in
ata_dmack		ATA bus DMA acknowledge	1	out
ata_intrq		ATA bus interrupt request	-	in
ata_iordy		ATA bus iordy	-	out
ata_data[15:0]		ATA data bus (Little-Endian)	Hi-z	tri-state in-out
ata_buffer_en		Buffer enable for external bus transceiver[2]	0	out

1 This signal is a standard ATA bus signal. It conforms with the ATA specification.

2 It is optional to put a 74xxx245 bus transceiver between the host side of the data bus and the device side of the data bus. If the transceiver is used, its enable should be tied low (always enable), and its direction pin should be tied to ata_buffer_en in such a way that it drives from host to device when ata_buffer_en is high, and drives from device to host when ata_buffer_en is low.

23.2.1 Detailed Signal Descriptions

This section provides detailed signal descriptions.

23.2.1.1 $\overline{\text{ata\_reset}}$ (out)

This signal is the ATA reset signal. When low, the ATA bus is in reset state. When high, no reset. The ATA bus is in reset whenever the appropriate bit in the control register is cleared. After system reset, the ATA bus is in reset.

23.2.1.2 ata_dior (out)

This signal corresponds to the ATA signal DIOR. During PIO and multiword DMA transfer, the function is read strobe. During ultra DMA in burst, the function is HDMARDY. During ultra DMA out burst, the function is host strobe.

23.2.1.3 ata_diow (out)

This signal corresponds to the ATA signal DIOW. During PIO and multiword DMA transfer, the function is write strobe. During ultra DMA burst, the function is STOP, signalling whenever host wants to terminate running ultra DMA transfer.

23.2.1.4 ata_cs0, ata_cs1, ata_da2, ata_da1, ata_da0 (out)

These signals are the address group of the ATA bus—ata_cs0 and ata_cs1 are the chip selects; ata_da2, ata_da1, and ata_da0 are the 3 address lines. All five lines follow the same timing.

23.2.1.5 ata_dmarq (in)

This signal is the ATA bus device DMA request. It is pulled high by the device if it is to transfer data using multiword DMA or ultra DMA mode

23.2.1.6 ata_dmack (out)

This signal is the ATA bus host DMA acknowledge. It is pulled low by the host when it grants the DMA request.

23.2.1.7 ata_intrq (in)

This signal is the ATA bus interrupt request. It is pulled high by the device whenever it is to interrupt the host CPU.

23.2.1.8 ata_iordy (in)

This signal is the ATA bus IORDY line. It has three functions:

- IORDY—active low wait during PIO cycles
- DDMARDY—active low device ready during ultra DMA out transfers
- DSTROBE—device strobe during ultra DMA in transfers

23.2.1.9 ata_data[15:0] (in/out-tristate)

This is the ATA data bus.

23.2.1.10 ata_buffer_en

This is a buffer direction control signal.

23.2.2 Timing on ATA Bus

This section explains timing on the ATA bus and how to ensure the ATA interface meets timing. Timing figures and equations that need to be fulfilled for the host to meet timing are also explained.

23.2.2.1 Timing Parameters

In the timing equations, some timing parameters are used. These parameters depend on the implementation of the ATA interface on silicon, the bus buffer used, the cable delay, and cable skew. Table 23-2 lists parameters used to specify the ATA timing.

Table 23-2. Timing Parameters

Name	Meaning	Controlled By
T	Bus clock period	Clock Generator
ti_ds	Set-up time ata_data to ata_iordy edge (UDMA-in only)	Top Level Design
ti_dh	Hold time ata_iordy edge to ata_data (UDMA-in only)	Top Level Design
tco	Propagation delay bus clock L-to-H to ata_cs0, ata_cs1, ata_da2, ata_da1, ata_da0, ata_dior, ata_diow, ata_dmack, ata_data, ata_buffer_en	Top Level Design
tsu	Set-up time ata_data to bus clock L-to-H	Top Level Design
tsui	Set-up time ata_iordy to bus clock H-to-L	Top Level Design
thi	Hold time ata_iordy to bus clock H to L	Top Level Design
tskew1	MaxImum difference in propagation delay bus clock L-to-H to any of following signals: ata_cs0, ata_cs1, ata_da2, ata_da1, ata_da0, ata_dior, ata_diow, ata_dmack, ata_data (write), ata_buffer_en	Top Level Design
tskew2	Maximum difference in buffer propagation delay for any of following signals: ata_cs0, ata_cs1, ata_da2, ata_da1, ata_da0, ata_dior, ata_diow, ata_dmack, ata_data (write), ata_buffer_en	Transceiver
tskew3	Maximum difference in buffer propagation delay for any of following signals ata_iordy, ata_data (read)	Transceiver
tbuf	Maximum buffer propagation delay	Transceiver
tcable1	Cable propagation delay for ata_data	Cable
tcable2	Cable propagation delay for control signals ata_dior, ata_diow, ata_iordy, ata_dmack	Cable
tskew4	Maximum difference in cable propagation delay between ata_iordy and ata_data (read)	Cable
tskew5	Maximum difference in cable propagation delay between (ata_dior, ata_diow, ata_dmack) and ata_cs0, ata_cs1, ata_da2, ata_da1, ata_da0, ata_data(write)	Cable
tskew6	Maximum difference in cable propagation delay without accounting for ground bounce	Cable

23.2.2.2 PIO Mode Timing

Figure 23-2 shows a timing diagram for PIO read mode.

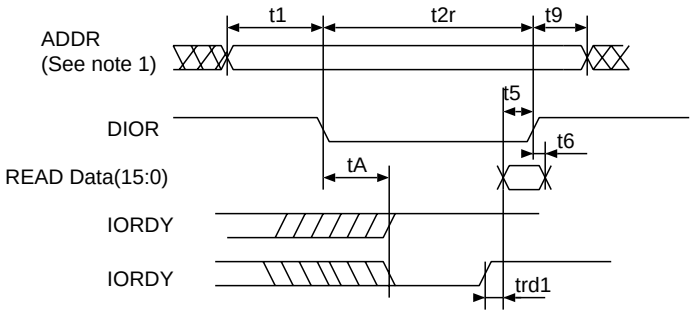


Figure 23-2. PIO Read Mode Timing

To fulfill PIO read mode timing, the different timing parameters given in Table 23-3 must be observed.

Table 23-3. Timing Parameters PIO Read

ATA Parameter	PIO Read Mode Timing Parameter[1]	Value	How To Meet?
t1	t1	t1(min) = time_1 * T–(tskew1 + tskew2 + tskew5)	time_1
t2	t2r	t2(min) = time_2r * T–(tskew1 + tskew2 + tskew5)	time_2r
t9	t9	t9(min) = time_9 * T–(tskew1 + tskew2 + tskew6)	time_9
t5	t5	t5(min) = tco + tsu + tbuf + tbuf + tcable1 + tcable2	if not met, increase time_2
t6	t6	0	
tA	tA	tA(min) = (1.5 + time_ax) * T–(tco + tsui + tcable2 + tcable2 + 2*tbuf)	time_ax
trd	trd1	trd1(max) = (-trd) + (tskew3 + tskew4) trd1(min) = (time_pio_rdx–0.5)*T–(tsu + thi) (time_pio_rdx–0.5) * T > tsu + thi + tskew3 + tskew4	time_pio_rdx
t0	—	t0(min) = (time_1 + time_2 + time_9) * T	time_1, time_2r, time_9

1 See Figure 23-2.

Figure 23-3 shows a timing diagram for PIO write mode.

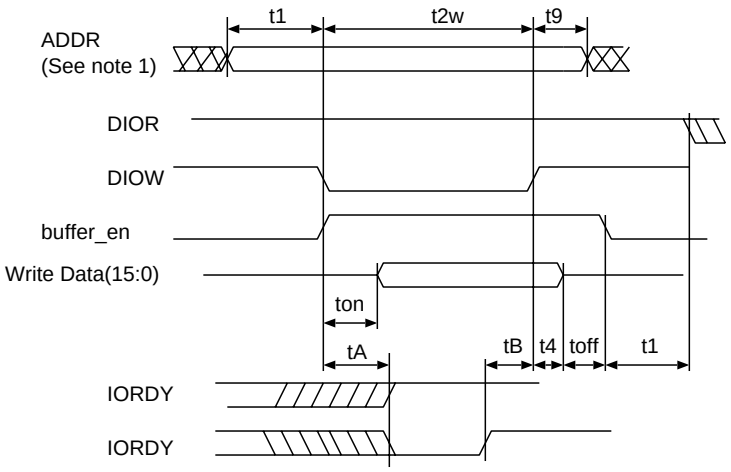


Figure 23-3. PIO Write Mode Timing

To fulfill PIO write mode timing, the different timing parameters given in Table 23-4 must be observed.

Table 23-4. Timing Parameters PIO Write

ATA Parameter	PIO Write Mode Timing Parameter[1]	Value	How To Meet?
t1	t1	t1(min) = time_1 * T–(tskew1 + tskew2 + tskew5)	time_1
t2	t2w	t2(min) = time_2w * T–(tskew1 + tskew2 + tskew5)	time_2w
t9	t9	t9(min) = time_9 * T–(tskew1 + tskew2 + tskew6)	time_9
t3	—	t3(min) = (time_2w–time_on)* T–(tskew1 + tskew2 +tskew5)	if not met, increase time_2w
t4	t4	t4(min) = time_4 * T–tskew1	time_4
tA	tA	tA = (1.5 + time_ax) * T–(tco + tsui + tcable2 + tcable2 + 2*tbuf)	time_ax
t0	—	t0(min) = (time_1 + time_2 + time_9) * T	time_1, time_2r, time_9
—	—	Avoid bus contention when switching buffer on by making ton long enough	
		Avoid bus contention when switching buffer off by making toff long enough	

1 See Figure 23-3.

23.2.2.3 Timing in Multiword DMA Mode

Figure 23-4 shows a read timing diagram; Figure 23-5 shows a write timing diagram for MDMA mode.

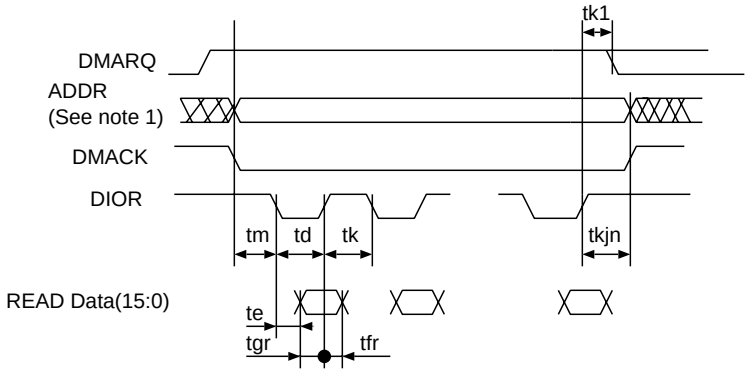


Figure 23-4. MDMA Read Timing

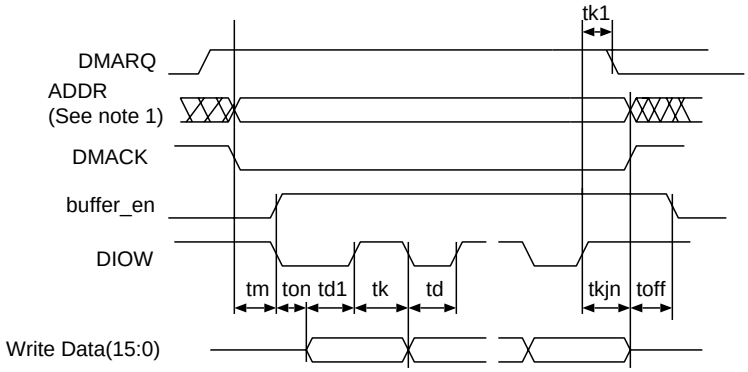


Figure 23-5. MDMA Write Timing

To fulfill MDMA read and write mode timing, the different timing parameters given in Table 23-5 must be observed.

Table 23-5. Timing Parameters MDMA Read and Write

ATA Parameter	MDMA Read TIming[1] and MDMA Write TIming[2]	Value	How To Meet?
tm, ti	tm	tm(min) = ti(min) = time_m * T–(tskew1 + tskew2 + tskew5)	time_m
td	td, td1	td1(min) = td(min) = time_d * T–(tskew1 + tskew2 + tskew6)	time_d
tk	tk	tk(min) = time_k * T–(tskew1 + tskew2 + tskew6)	time_k
t0	—	t0(min) = (time_d + time_k) * T	time_d, time_k
tg(read)	tgr	tgr(min-read) = tco + tsu + tbuf + tbuf + tcable1 + tcable2 tgr(min-drive) = td–te(drive)	time_d
tf(read)	tfr	tfr(min-drive) =0k	—
tg(write)	—	tg(min-write) = time_d * T -(tskew1 + tskew2 + tskew5)	time_d
tf(write)		tf(min-write) = time_k * T–(tskew1 + tskew2 + tskew6)	time_k
tL		tL(max) = (time_d + time_k-2)*T–(tsu + tco + 2*tbuf + 2*tcable2)	time_d, time_k[3]

Table 23-5. Timing Parameters MDMA Read and Write (continued)

ATA Parameter	MDMA Read TIming[1] and MDMA Write TIming[2]	Value	How To Meet?
tn, tj	tkjn	tn= tj= tkjn = (max(time_k, . . . , time_jn) * T–(tskew1 + tskew2 + tskew6)	time_jn
	ton toff	ton = time_on * T–tskew1 toff = time_off * T–tskew1	

1 See Figure 23-4.

2 See Figure 23-5.

3 tk1 in the UDMA figures equals (tk -2*T)

23.2.2.4 UDMA In Timing Diagrams

UDMA mode timing is more complicated than PIO mode or MDMA mode. Figure 23-6 shows timing for UDMA in transfer start.

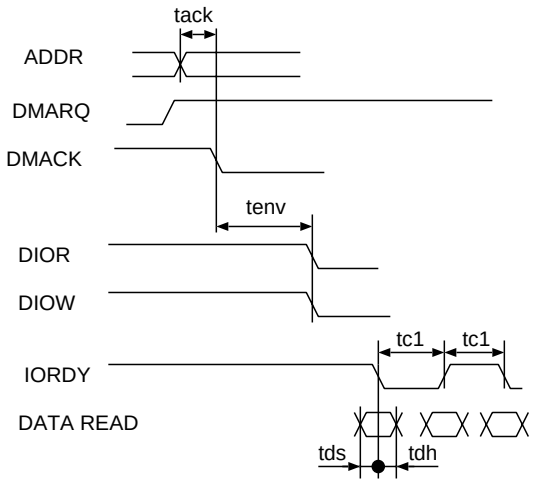


Figure 23-6. UDMA in Transfer Start

Figure 23-7 shows timing for UDMA in host terminating transfer.

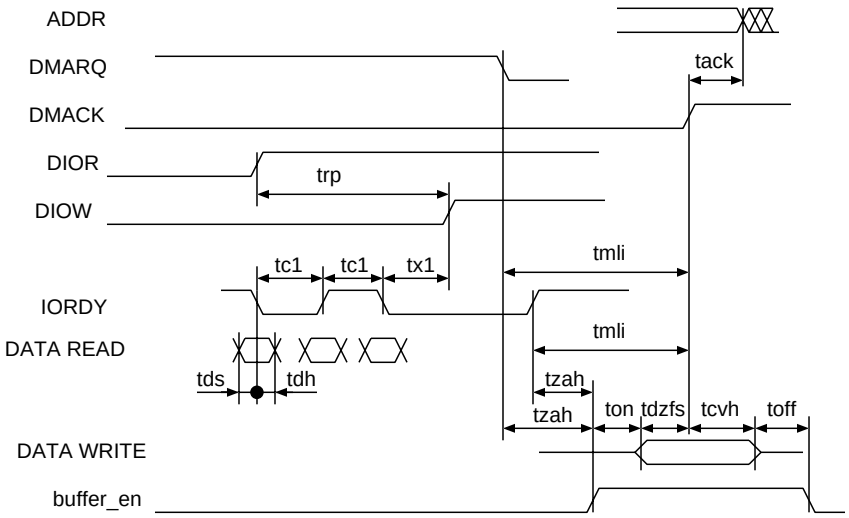


Figure 23-7. UDMA in Host Terminates Transfer

Figure 23-8 shows timing for UDMA in device terminating transfer.

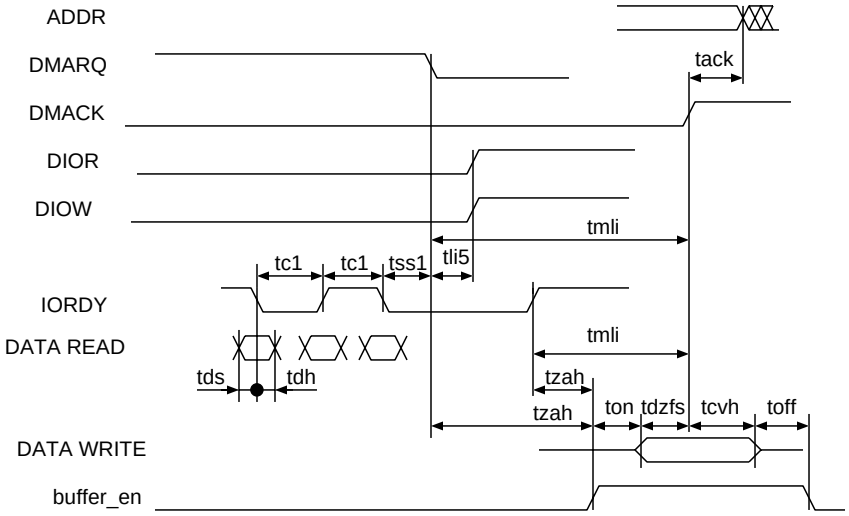


Figure 23-8. UDMA in Device Terminates Transfer

Table 23-6 shows timing parameters for UDMA in burst.

Table 23-6. Timing Parameters for UDMA in Burst

ATA Parameter	Specification Parameter	Value	Required Conditions
tack	tack	tack(min) = (time_ack * T)–(tskew1 + tskew2)	time_ack
tenv	tenv	tenv(min) = (time_env * T)–(tskew1 + tskew2) tenv(max) = (time_env * T) + (tskew1 + tskew2)	time_env
tds	tds1	tds–(tskew3)–ti_ds > 0	tskew3, ti_ds, ti_dh should be low enough
tdh	tdh1	tdh–(tskew3) -ti_dh > 0	

Table 23-6. Timing Parameters for UDMA in Burst (continued)

ATA Parameter	Specification Parameter	Value	Required Conditions
tcyc	tc1	(tcyc–tskew + TBD) > T	T big enough
trp	trp	trp(min) = time_rp * T–(tskew1 + tskew2 + tskew6)	time_rp
	tx1[1]	(time_rp * T)–(tco + tsu + 3T + 2 *tbuf + 2*tcable2) > trfs (drive)	time_rp
tmli	tmli1	tmli1(min) = (time_mlix + 0.4) * T	time_mlix
tzah	tzah	tzah(min) = (time_zah + 0.4) * T	time_zah
tdzfs	tdzfs	tdzfs = (time_dzfs * T)–(tskew1 + tskew2)	time_dzfs
tcvh	tcvh	tcvh = (time_cvh *T)–(tskew1 + tskew2)	time_cvh
	ton toff	ton = time_on * T–tskew1 toff = time_off * T–tskew1	

1 There is a special timing requirement in the ATA host that requires the internal DIOW to go only high three clocks after the last active edge on the DSTROBE signal. The equation given on this line tries to capture this constraint.

2. Make ton and toff big enough to avoid bus contention.

23.2.2.5 UDMA Out Timing Diagrams

UDMA mode timing is more complicated than PIO mode or MDMA mode. Figure 23-9 shows timing for UDMA out transfer start.

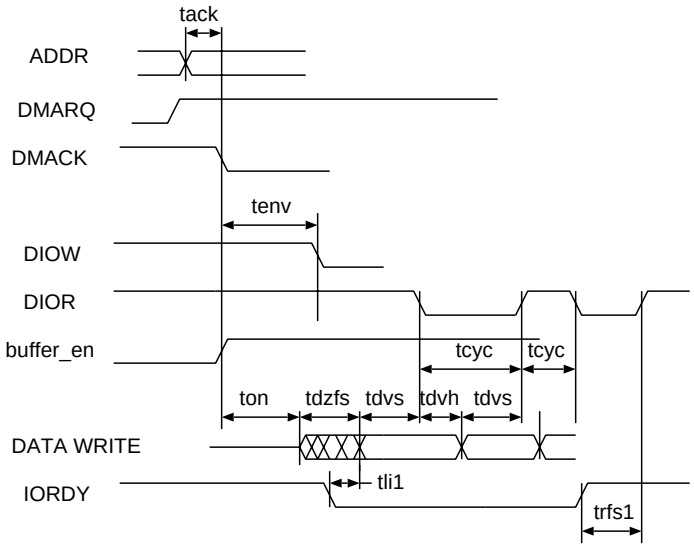


Figure 23-9. UDMA Out Transfer Start

Figure 23-10 shows timing for UDMA out host terminating transfer.

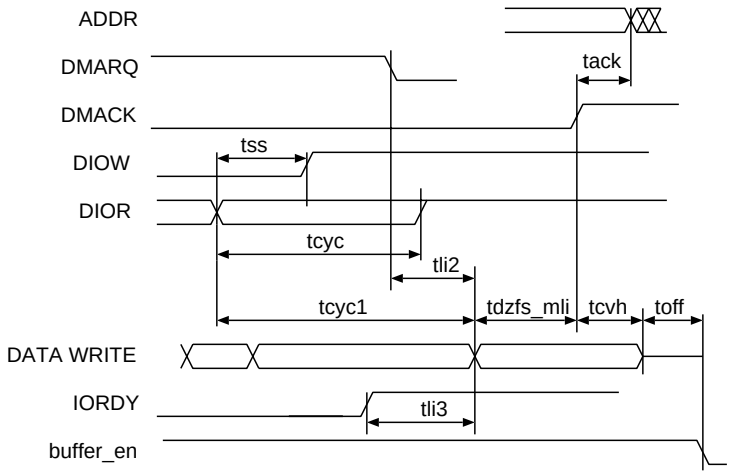


Figure 23-10. UDMA Out Host Terminates Transfer

Figure 23-11 shows timing for UDMA out device terminating transfer.

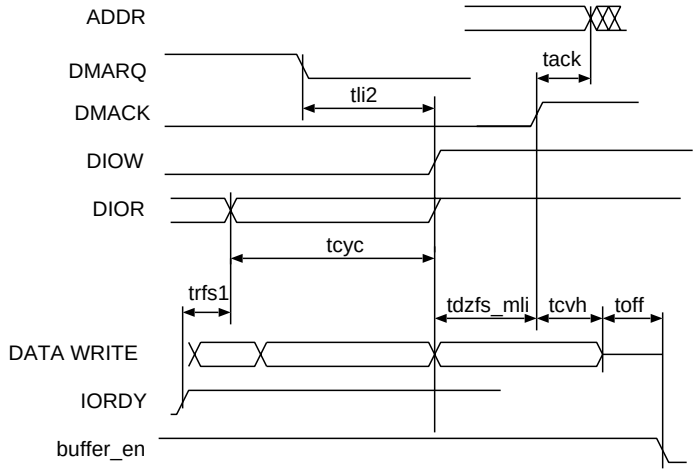


Figure 23-11. UDMA Out Device Terminates Transfer

Table 23-7 shows timing parameters for UDMA out burst.

Table 23-7. Timing Parameters UDMA Out Burst

ATA Parameter	Specification Parameter	Value	How To Meet?
tack	tack	tack(min) = (time_ack * T)–(tskew1 + tskew2)	time_ack
tenv	tenv	tenv(min) = (time_env * T)–(tskew1 + tskew2) tenv(max) = (time_env * T) + (tskew1 + tskew2)	time_env
tdvs	tdvs	tdvs = (time_dvs * T)–(tskew1 + tskew2)	time_dvs
tdvh	tdvh	tdvs = (time_dvh * T)–(tskew1 + tskew2)	time_dvh
tcyc	tcyc	tcyc = time_cyc * T–(tskew1 + tskew2)	time_cyc

Table 23-7. Timing Parameters UDMA Out Burst (continued)

ATA Parameter	Specification Parameter	Value	How To Meet?
t2cyc		t2cyc = time_cyc * 2 * T	time_cyc
trfs1	trfs	trfs = 1.6 * T + tsui + tco + tbuf + tbuf	-
-	tdzfs	tdzfs = time_dzfs * T–(tskew1)	time_dzfs
tss	tss	tss = time_ss * T–(tskew1 + tskew2)	time_ss
tmli	tdzfs_mli	tdzfs_mli =max(time_dzfs, time_mli) * T–(tskew1 + tskew2)	
tli	tli1	tli1 > 0	
tli	tli2	tli2 > 0	
tli	tli3	tli3 > 0	
tcvh	tcvh	tcvh = (time_cvh *T)–(tskew1 + tskew2)	time_cvh
	ton toff	ton = time_on * T–tskew1 toff = time_off * T–tskew1	

23.3 Memory Map and Register Definition

Section 23.3.3, “Register Descriptions” provides the detailed descriptions for all of the ATA registers.

23.3.1 Memory Map

Table 23-8 shows the ATA memory map.

Table 23-8. ATA Memory Map

Address	Register	Description	Access	Reset Value	Section/Page
0x43F8_C000 (TIME_OFF)	TIME_OFF	Transceiver timing parameter. Controls toff	R/W	0x01	23.3.3.2.1/23-20
0x43F8_C001 (TIME_ON)	TIME_ON	Transceiver timing parameter. Controls ton	R/W	0x01	23.3.3.2.2/23-20
0x43F8_C002 (TIME_1)	TIME_1	PIO timing parameter. Controls t1	R/W	0x01	23.3.3.2.3/23-21
0x43F8_C003 (TIME_2W)	TIME_2W	PIO timing parameter. Controls t2 during write cycles	R/W	0x01	23.3.3.2.4/23-21
0x43F8_C004 (TIME_2R)	TIME_2R	PIO timing parameter. Controls t2 during read cycles	R/W	0x01	23.3.3.2.5/23-21
0x43F8_C005 (TIME_AX)	TIME_AX	PIO timing parameter. Controls tA	R/W	0x01	23.3.3.2.6/23-21
0x43F8_C00F (TIME_PIO_RDX)	TIME_PIO_RDX	PIO timing parameter. Controls trd	R/W	0x01	23.3.3.2.7/23-22

Table 23-8. ATA Memory Map (continued)

Address	Register	Description	Access	Reset Value	Section/Page
0x43F8_C007 (TIME_4)	TIME_4	PIO timing parameter. Controls t4	R/W	0x01	23.3.3.2.8/23-22
0x43F8_C008 (TIME_9)	TIME_9	PIO timing parameter. Controls t9	R/W	0x01	23.3.3.2.9/23-22
0x43F8_C009 (TIME_M)	TIME_M	MDMA timing parameter. Controls tm	R/W	0x01	23.3.3.2.10/23-22
0x43F8_C00A (TIME_JN)	TIME_JN	MDMA timing parameter. Controls tn and tj	R/W	0x01	23.3.3.2.11/23-23
0x43F8_C00B (TIME_D)	TIME_D	MDMA timing parameter. Controls td	R/W	0x01	23.3.3.2.12/23-23
0x43F8_C00C (TIME_K)	TIME_K	MDMA timing parameter. Controls tk	R/W	0x01	23.3.3.2.13/23-23
0x43F8_C00D (TIME_ACK)	TIME_ACK	UDMA timing parameter. Controls tack	R/W	0x01	23.3.3.2.14/23-23
0x43F8_C00E (TIME_ENV)	TIME_ENV	UDMA timing parameter. Controls tenv.	R/W	0x01	23.3.3.2.15/23-24
0x43F8_C00F (TIME_PIO_RDX)	TIME_RPX	UDMA timing parameter. Controls trp.	R/W	0x01	23.3.3.2.16/23-24
0x43F8_C010 (TIME_ZAH)	TIME_ZAH	UDMA timing parameter. Controls tzah.	R/W	0x01	23.3.3.2.17/23-24
0x43F8_C011 (TIME_MLIX)	TIME_MLIX	UDMA timing parameter. Controls tmli.	R/W	0x01	23.3.3.2.18/23-24
0x43F8_C012 (TIME_DVH)	TIME_DVH	UDMA timing parameter. Controls tdvh.	R/W	0x01	23.3.3.2.19/23-25
0x43F8_C013 (TIME_DZFS)	TIME_DZFS	UDMA timing parameter. Controls tdzfs.	R/W	0x01	23.3.3.2.20/23-25
0x43F8_C014 (TIME_DVS)	TIME_DVS	UDMA timing parameter. Controls tdvs.	R/W	0x01	23.3.3.2.21/23-25
0x43F8_C015 (TIME_CVH)	TIME_CVH	UDMA timing parameter. Controls tcvh.	R/W	0x01	23.3.3.2.22/23-25
0x43F8_C016 (TIME_SS)	TIME_SS	UDMA timing parameter. Controls tss	R/W	0x01	23.3.3.2.23/23-26
0x43F8_C017 (TIME_CYC)	TIME_CYC	UDMA timing parameter. Controls tcyc and t2cyc.	R/W	0x01	23.3.3.2.24/23-26
0x43F8_C01C (FIFO_DATA_16)	FIFO_DATA_16	16-bit wide or 32-bit wide data port to/from FIFO.	R/W	0x— — — —	23.3.3.3.1/23-26
0x43F8_C018 (FIFO_DATA_32)	FIFO_DATA_32		R/W	0x— — — —_ — — — —	23.3.3.3.2/23-27
0x43F8_C020 (FIFO_FILL)	FIFO_FILL	Fifo Filling In Halfwords	R	0x00	23.3.3.3.3/23-27

Table 23-8. ATA Memory Map (continued)

Address	Register	Description	Access	Reset Value	Section/Page
0x43F8_C024 (ATA_CONTROL)	ATA_CONTROL	Ata Interface Control Register	R/W	0x00	23.3.3.5.1/23-29
0x43F8_C028 (INTERRUPT_PENDING)	INTERRUPT_ PENDING	Interrupt Pending Register	R	0x1 —	23.3.3.5.1/23-29
0x43F8_C02C (INTERRUPT_ENABLE)	INTERRUPT_ ENABLE	Interrupt Enable Register	R/W	0x0 —	23.3.3.5.2/23-30
0x43F8_C030 (INTERRUPT_CLEAR)	INTERRUPT_ CLEAR	Interrupt Clear Register	W	0x— —	23.3.3.5.3/23-31
0x43F8_C034 (FIFO_ALARM)	FIFO_ALARM	Fifo Alarm Threshold	R/W	0x00	23.3.3.7/23-32
0x43F8_C0A0 (DRIVE_DATA)	DRIVE_DATA	Drive Data Register	16-bit R/W		23.3.3.7/23-32
0x43F8_C0A4 (DRIVE_FEATURES)	DRIVE_ FEATURES	Drive Features Register	R/W		23.3.3.7/23-32
0x43F8_C0A8 (DRIVE_SECTOR_COUNT)	DRIVE_SECTO R_COUNT	Drive Sector Count Register	R/W		23.3.3.7/23-32
0x43F8_C0AC (DRIVE_SECTOR_NUM)	DRIVE_SECTO R_NUM	Drive Sector Number Register	R/W		23.3.3.7/23-32
0x43F8_C0B0 (DRIVE_CYL_LOW)	DRIVE_CYL_ LOW	Drive Cylinder Low Register	R/W	—	23.3.3.7/23-32
0x43F8_C0B4 (DRIVE_CYL_HIGH)	DRIVE_CYL_ HIGH	Drive Cylinder High Register	R/W	—	23.3.3.7/23-32
0x43F8_C0B8 (DRIVE_DEV_HEAD)	DRIVE_DEV_ HEAD	Drive Device Head Register	R/W	—	23.3.3.7/23-32
0x43F8_C0BC (DRIVE_STATUS[rd]/DRIVE _COMMAND[wr])	DRIVE_STATUS /DRIVE_ COMMAND	Drive Status Register/Drive Command Register	R/W	—	23.3.3.7/23-32
0x43F8_C0D8 (DRIVE_ALT_STATUS[rd]/D RIVE_CONTROL[wr])	DRIVE_ALT_ STATUS/DRIVE _CONTROL	Drive Alternate Status Register/Drive Control Register	R/W	—	23.3.3.7/23-32

23.3.2 Register Summary

Figure 23-12 shows the key to the register fields, and Table 23-9 shows the register figure conventions.

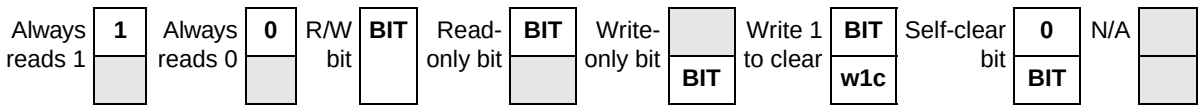


Figure 23-12. Key to Register Fields

Table 23-9. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 23-10 shows the ATA register summary.

Table 23-10. ATA Register Summary

Name		7	6	5	4	3	2	1	0
0x43F8_C000 (TIME_OFF)	R	TIME_OFF[7:0]							
	W								
0x43F8_C001 (TIME_ON)	R	TIME_ON[7:0]							
	W								
0x43F8_C002 (TIME_1)	R	TIME_1[7:0]							
	W								
0x43F8_C003 (TIME_2W)	R	TIME_2W[7:0]							
	W								
0x43F8_C004 (TIME_2R)	R	TIME_2R[7:0]							
	W								
0x43F8_C005 (TIME_AX)	R	TIME_AX[7:0]							
	W								

Table 23-10. ATA Register Summary (continued)

Name		7	6	5	4	3	2	1	0
0x43F8_C00F (TIME_PIO_RDX)	R	TIME_RDX[7:0]							
	W								
0x43F8_C007 (TIME_4)	R	TIME_4[7:0]							
	W								
0x43F8_C008 (TIME_9)	R	TIME_9[7:0]							
	W								
0x43F8_C009 (TIME_M)	R	TIME_M[7:0]							
	W								
0x43F8_C00A (TIME_JN)	R	TIME_JN[7:0]							
	W								
0x43F8_C00B (TIME_D)	R	TIME_D[7:0]							
	W								
0x43F8_C00C (TIME_K)	R	TIME_K[7:0]							
	W								
0x43F8_C00D (TIME_ACK)	R	TIME_ACK[7:0]							
	W								
0x43F8_C00E (TIME_ENV)	R	TIME_ENV[7:0]							
	W								
0x43F8_C00F (TIME_RPX)	R	TIME_RPX[7:0]							
	W								
0x43F8_C010 (TIME_ZAH)	R	TIME_ZAH[7:0]							
	W								
0x43F8_C011 (TIME_MLIX)	R	TIME_MLIX[7:0]							
	W								
0x43F8_C012 (TIME_DVH)	R	TIME_DVH[7:0]							
	W								
0x43F8_C013 (TIME_DZFS)	R	TIME_DZFS[7:0]							
	W								
0x43F8_C014 (TIME_DVS)	R	TIME_DVS[7:0]							
	W								
0x43F8_C015 (TIME_CVH)	R	TIME_CVH[7:0]							
	W								

Table 23-10. ATA Register Summary (continued)

Name		7	6	5	4	3	2	1	0
0x43F8_C016 (TIME_SS)	R	TIME_SS[7:0]							
	W								
0x43F8_C017 (TIME_CYC)	R	TIME_CYC[7:0]fifo_data[15:0]							
	W								
0x43F8_C018 (FIFO_DATA_32)	R	fifo_data[23:16]							
	W								
0x43F8_C01C (FIFO_DATA_16)	R	fifo_data[15:0]							
	W								
0x43F8_C020 (FIFO_FILL)	R	FIGO_FILL[7:0]							
	W								
0x43F8_C024 (ATA_CONTROL)	R	$\overline{\text{fifo\_rst}}$	$\overline{\text{ata\_rst}}$	fifo_tx_en	fifo_rcv_en	dma_pending	dma_ultra_selected	dma_write	iordy_en
	W								
0x43F8_C028 (INTERRUPT_PENDING)	R	ata_intrq1	fifo_underflow	fifo_overflow	controller_idle	ata_irtrq2	0	0	0
	W								
0x43F8_C02C (INTERRUPT_ENABLE)	R	ata_intrq1	fifo_underflow	fifo_overflow	controller_idle	ata_irtrq2	0	0	0
	W								
0x43F8_C030 (INTERRUPT_CLEAR)	R	0			0	0	0	0	0
	W		fifo_underflow	fifo_overflow					
0x43F8_C034 (FIFO_ALARM)	R	FIFO_ALARM[7:0]							
	W								

23.3.3 Register Descriptions

This section contains the detailed register descriptions for the ATA registers.

23.3.3.1 Endianness in ATA

The ATA interface works both in Little Endian and Big Endian mode. The addresses of all registers are independent of Endianness mode. (They remain same.) The few16-bit and 32-bit registers represent strings of 2 or 4 bytes. The byte order in the 16-bit or 32-bit register is dependent on Endianness selection.

- Little Endian, 16-bit or 32-bit register:
  — bits [7:0]: byte 0

  - bits [15:8]: byte 1
  - bits [23:8]: byte 2
  - bits [31:24]: byte 3
- Big Endian, 32-bit register
  - bits [31:24]: byte 0
  - bits [23:16]: byte 1
  - bits [15:8]: byte 2
  - bits [7:0]: byte 3
- Big Endian, 16-bit register
  - bits [15:8]: byte 0
  - bits [7:0]: byte 1

23.3.3.2 Timing Registers

Registers (ata_base +$0) to (ata_base + $17) contain timing parameters. These timing parameters control the timing on the ATA bus as shown in details in the following illustrations:

- Section 23.2.2.2, “PIO Mode Timing”
- Section 23.2.2.3, “Timing in Multiword DMA Mode”
- Section 23.2.2.4, “UDMA In Timing Diagrams”
- Section 23.2.2.5, “UDMA Out Timing Diagrams”

Every timing parameter is 8-bit wide and can assume valid values between 1 and 255. The reset value is always 1.

All figures in this section show timing registers.

23.3.3.2.1 TIME_OFF Register

Figure 23-13 shows the TIME_OFF register.

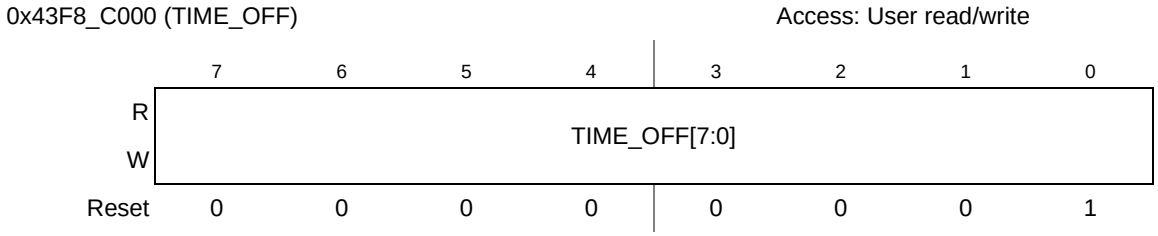


Figure 23-13. TIME_OFF Register

23.3.3.2.2 TIME_ON Register

Figure 23-14 shows the TIME_ON register.

0x43F8_C001 (TIME_ON) Access: User read/write

	7	6	5	4	3	2	1	0
R W	TIME_ON[7:0]							
Reset	0	0	0	0	0	0	0	1

Figure 23-14. TIME_ON Register

23.3.3.2.3 TIME_1 Register

Figure 23-15 shows the TIME_1 register.

0x43F8_C002 (TIME_1) Access: User read/write

	7	6	5	4	3	2	1	0
R W	TIME_1[7:0]							
Reset	0	0	0	0	0	0	0	1

Figure 23-15. TIME_1 Register

23.3.3.2.4 TIME_2W Register

Figure 23-16 shows the TIME_2W register.

0x43F8_C003 (TIME_2W) Access: User read/write

	7	6	5	4	3	2	1	0
R W	TIME_2W[7:0]							
Reset	0	0	0	0	0	0	0	1

Figure 23-16. TIME_2W Register

23.3.3.2.5 TIME_2R Register

Figure 23-17 shows the TIME_2R register.

0x43F8_C004 (TIME_2R) Access: User read/write

	7	6	5	4	3	2	1	0
R W	TIME_2R[7:0]							
Reset	0	0	0	0	0	0	0	1

Figure 23-17. TIME_2R Register

23.3.3.2.6 TIME_AX Register

Figure 23-18 shows the TIME_AX register.

0x43F8_C005 (TIME_AX) Access: User read/write

	7	6	5	4	3	2	1	0
R W	TIME_AX[7:0]							
Reset	0	0	0	0	0	0	0	1

Figure 23-18. TIME_AX Register

23.3.3.2.7 TIME_PIO_RDX Register

Figure 23-19 shows the TIME_PIO_RDX register.

0x43F8_C00F (TIME_PIO_RDX) Access: User read/write

	7	6	5	4	3	2	1	0
R W	TIME_RDX[7:0]							
Reset	0	0	0	0	0	0	0	1

Figure 23-19. TIME_PIO_RDX Register

23.3.3.2.8 TIME_4 Register

Figure 23-20 shows the TIME_4 register.

0x43F8_C007 (TIME_4) Access: User read/write

	7	6	5	4	3	2	1	0
R W	TIME_4[7:0]							
Reset	0	0	0	0	0	0	0	1

Figure 23-20. TIME_4 Register

23.3.3.2.9 TIME_9 Register

Figure 23-21 shows the TIME_9 register.

0x43F8_C008 (TIME_9) Access: User read/write

	7	6	5	4	3	2	1	0
R W	TIME_9[7:0]							
Reset	0	0	0	0	0	0	0	1

Figure 23-21. TIME_9 Register

23.3.3.2.10 TIME_M Register

Figure 23-22 shows the TIME_M register.

0x43F8_C009 (TIME_M) Access: User read/write

	7	6	5	4	3	2	1	0
R W	TIME_M[7:0]							
Reset	0	0	0	0	0	0	0	1

Figure 23-22. TIME_M Register

23.3.3.2.11 TIME_JN Register

Figure 23-23 shows the TIME_JN register.

0x43F8_C00A (TIME_JN) Access: User read/write

	7	6	5	4	3	2	1	0
R W	TIME_JN[7:0]							
Reset	0	0	0	0	0	0	0	1

Figure 23-23. TIME_JN Register

23.3.3.2.12 TIME_D Register

Figure 23-24 shows the TIME_D register.

0x43F8_C00B (TIME_D) Access: User read/write

	7	6	5	4	3	2	1	0
R W	TIME_D[7:0]							
Reset	0	0	0	0	0	0	0	1

Figure 23-24. TIME_D Register

23.3.3.2.13 TIME_K Register

Figure 23-25 shows the TIME_K register.

0x43F8_C00C (TIME_K) Access: User read/write

	7	6	5	4	3	2	1	0
R W	TIME_K[7:0]							
RESET	0	0	0	0	0	0	0	1

Figure 23-25. TIME_K Register

23.3.3.2.14 TIME_ACK Register

Figure 23-26 shows the TIME_ACK register.

0x43F8_C00D (TIME_ACK) Access: User read/write

	7	6	5	4	3	2	1	0
R W	TIME_ACK[7:0]							
Reset	0	0	0	0	0	0	0	1

Figure 23-26. TIME_ACK Register

23.3.3.2.15 TIME_ENV Register

Figure 23-27 shows the TIME_ENV register.

0x43F8_C00E (TIME_ENV) Access: User read/write

	7	6	5	4	3	2	1	0
R W	TIME_ENV[7:0]							
Reset	0	0	0	0	0	0	0	1

Figure 23-27. TIME_ENV Register

23.3.3.2.16 TIME_RPX Register

Figure 23-28 shows the TIME_RPX register.

0x43F8_C00F (TIME_RPX) Access: User read/write

	7	6	5	4	3	2	1	0
R W	TIME_RPX[7:0]							
Reset	0	0	0	0	0	0	0	1

Figure 23-28. TIME_RPX Register

23.3.3.2.17 TIME_ZAH Register

Figure 23-29 shows the TIME_ZAH register.

0x43F8_C010 (TIME_ZAH) Access: User read/write

	7	6	5	4	3	2	1	0
R W	TIME_ZAH[7:0]							
Reset	0	0	0	0	0	0	0	1

Figure 23-29. TIME_ZAH Register

23.3.3.2.18 TIME_MLIX Register

Figure 23-30 shows the TIME_MLIX register.

0x43F8_C011 (TIME_MLIX) Access: User read/write

	7	6	5	4	3	2	1	0
R W	TIME_MLIX[7:0]							
Reset	0	0	0	0	0	0	0	1

Figure 23-30. TIME_MLIX Register

23.3.3.2.19 TIME_DVH Register

Figure 23-31 shows the TIME_DVH register.

0x43F8_C012 (TIME_DVH) Access: User read/write

	7	6	5	4	3	2	1	0
R W	TIME_DVH[7:0]							
Reset	0	0	0	0	0	0	0	1

Figure 23-31. TIME_DVH Register

23.3.3.2.20 TIME_DZFS Register

Figure 23-32 shows the TIME_DZFS register.

0x43F8_C013 (TIME_DZFS) Access: User read/write

	7	6	5	4	3	2	1	0
R W	TIME_DZFS[7:0]							
Reset	0	0	0	0	0	0	0	1

Figure 23-32. TIME_DZFS Register

23.3.3.2.21 TIME_DVS Register

Figure 23-33 shows the TIME_DVS register.

0x43F8_C014 (TIME_DVS) Access: User read/write

	7	6	5	4	3	2	1	0
R W	TIME_D[7:0]							
Reset	0	0	0	0	0	0	0	1

Figure 23-33. TIME_DVS Register

23.3.3.2.22 Time_CVH Register

Figure 23-34 shows the TIME_CVH register.

0x43F8_C015 (TIME_CVH) Access: User read/write

	7	6	5	4	3	2	1	0
R W	TIME_CVH[7:0]							
Reset	0	0	0	0	0	0	0	1

Figure 23-34. TIME_CVH Register

23.3.3.2.23 TIME_SS Register

Figure 23-35 shows the TIME_SS register.

0x43F8_C016 (TIME_SS) Access: User read/write

	7	6	5	4	3	2	1	0
R W	TIME_SS[7:0]							
Reset	0	0	0	0	0	0	0	1

Figure 23-35. TIME_SS Register

23.3.3.2.24 TIME_CYC Register

Figure 23-36 shows the TIME_CYC register.

0x43F8_C017 (TIME_CYC) Access: User read/write

	7	6	5	4	3	2	1	0
R W	TIME_CYC[7:0]							
Reset	0	0	0	0	0	0	0	1

Figure 23-36. TIME_CYC Register

23.3.3.3 FIFO Data Registers

23.3.3.3.1 FIFO_DATA Register in 16-Bit Mode

Figure 23-37 shows the FIFO_DATA (16-bit mode) register.

0x43F8_C01C (FIFO_DATA_16) Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	FIFO_DATA[15:0]															
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Figure 23-37. FIFO_DATA Register In 16-bit Mode

23.3.3.3.2 FIFO_DATA Register in 32-Bit Mode

Figure 23-38 shows the FIFO_DATA (32-bit mode) register.

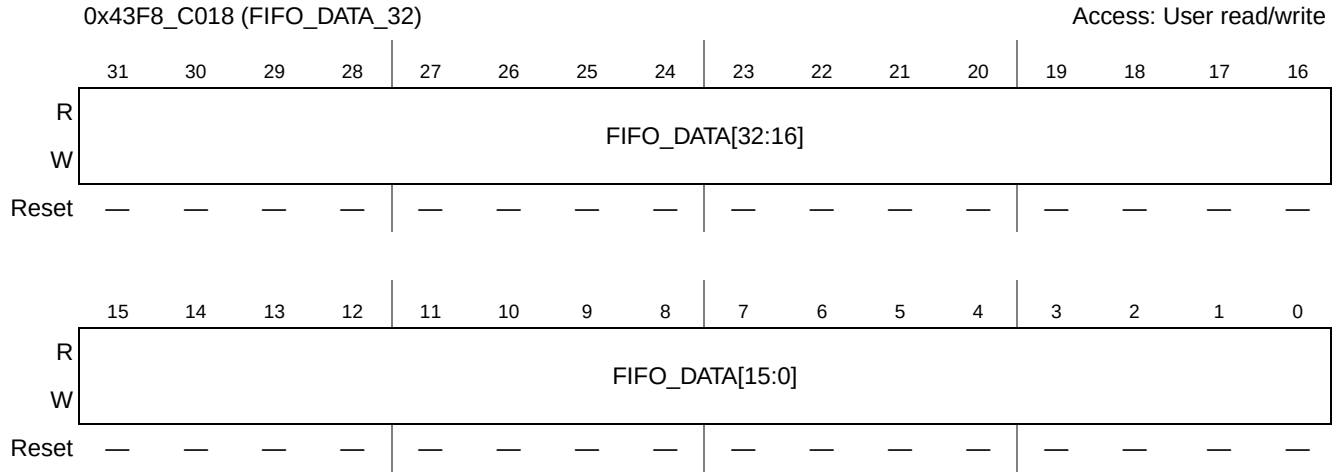


Figure 23-38. FIFO_DATA Register in 32-Bit Mode

The FIFO_Data register is used to read or write data to the internal FIFO. It can be accessed as a 16-bit register or as a 32-bit register. Any long write to the register puts the four bytes written into the FIFO. Any word write puts the two bytes written into the FIFO. Any long read reads four bytes from the FIFO. Any word read reads two bytes from the FIFO.

23.3.3.3.3 FIFO_FILL Register

Figure 23-39 shows the FIFO_FILL register.

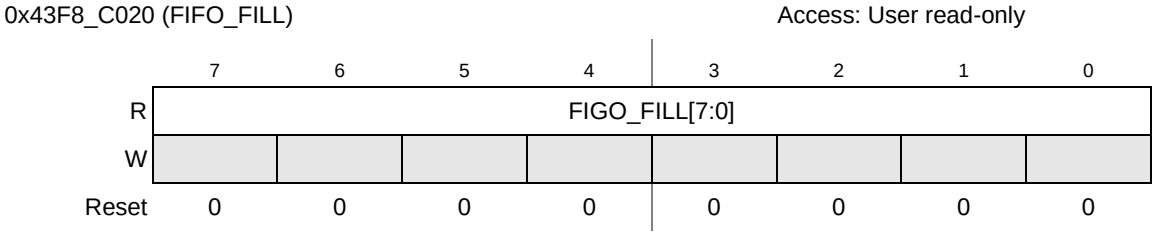


Figure 23-39. FIFO_FILL Register

FIFO_FILL is a read-only register. Any read to it returns the current number of halfwords present in the fifo.

23.3.3.4 ATA_CONTROL Register

Figure 23-40 shows the ATA_CONTROL register, and Table 23-11 shows the register's field descriptions.

0x43F8_C024 (ATA_CONTROL) Access: User read/write

	7	6	5	4	3	2	1	0
R W	$\overline{\text{fifo\_rst}}$	$\overline{\text{ata\_rst}}$	fifo_tx_en	fifo_rcv_en	dma_pending	dma_ultra_selected	dma_write	iordy_en
Reset	0	0	0	0	0	0	0	0

Figure 23-40. ATA_CONTROL Register

Table 23-11. ATA_CONTROL Register Field Descriptions

Field	Description
7 $\overline{\text{fifo\_rst}}$	This field controls if the internal FIFO is in reset or enabled. 0 FIFO reset 1 FIFO normal operation
6 $\overline{\text{ata\_rst}}$	This bit controls the level on the $\overline{\text{ata\_reset}}$ pin, and controls the reset of the internal ATA protocol engine. 0 $\overline{\text{ata\_reset}}$ = 0. ATA drive is reset, and internal protocol engine reset. 1 $\overline{\text{ata\_reset}}$ = 1. ATA drive is not reset, and internal protocol engine normal operation.
5 fifo_tx_en	FIFO transmit enable. This bit controls if the FIFO makes transmit data requests to the DMA. If enabled, the FIFO requests the DMA to refill it whenever FIFO filling drops below the alarm level. 0 FIFO refill by DMA disabled. 1 FIFO refill by DMA enabled.
4 fifo_rcv_en	FIFO receive enable. This bit controls if the FIFO makes receive data requests to the DMA. If enabled, the FIFO requests the DMA to empty it whenever FIFO filling becomes greater or equal to the alarm level. 0 FIFO empty by DMA disabled. 1 FIFO empty by DMA enabled.
3 dma_pending	DMA pending bit. This bit controls if the ATA interface responds to a DMA request originating in the drive. If this bit is asserted, the ATA interface started a multiword DMA or ultra DMA burst whenever the drive asserts ata_dmarq. 0 ATA interface does not start DMA burst. 1 ATA interface starts multiword DMA or ultra DMA burst whenever drive asserts dmarq.
2 dma_ultra_selected	This bit indicates if a DMA burst started, the UDMA or MDMA protocol is used. 0 Multiword DMA protocol will be used 1 Ultra DMA protocol will be used
1 dma_write	This bit indicates the data direction on any DMA burst started. 0 DMA in burst, ATA interface reads from drive 1 DMA out burst, ATA interface writes to drive
0 iordy_en	This bit indicates if the ata_iordy handshake is used during PIO mode. 0 IORDY is disregarded. 1 IORDY handshake is used.

23.3.3.5 Interrupt Registers

A group of three registers control the interrupt interface from the ATA module and going to the CPU and DMA. There are two interrupts controlled by these registers:

- ipbus_int. This interrupt is controlled by bits 3,4, 5, and 6 of the interrupt registers. It is asserted if one of the 4 bits is set in the interrupt_pending register, while the same bit is set in the interrupt_enable register. This interrupt goes to the CPU.
- fifo_txfer_end_alarm. This interrupt is controlled by bit 7 of the interrupt registers. If ata_intrq1 is set in both the interrupt enable and interrupt pending register, fifo_txfer_end_alarm is asserted. The goal of this interrupt is to inform the DMA that the running data transfer has ended. This interrupt goes to the SDMA.

These three registers have mostly the same bits. If a bit is set in the interrupt pending register, its interrupt is pending, and produces an interrupt if the same bit is set in the interrupt enable register. Some bits in the interrupt pending register are sticky bits. Writing a ‘1’ to the corresponding bit in the interrupt clear bit resets them.

23.3.3.5.1 INTERRUPT_PENDING Register

Figure 23-41 shows the INTERRUPT_PENDING register, and Table 23-12 shows the register’s field descriptions.

0x43F8_C028 (INTERRUPT_PENDING) Access: User read-only

	7	6	5	4	3	2	1	0
R	ata_intrq1	fifo_ underflow	fifo_ overflow	controller_ idle	ata_irtrq2	0	0	0
W								
Reset	0[1]	0	0	1	0	—	—	—

Figure 23-41. INTERRUPT_PENDING Register

[1] Interrupts ata_intrq1 and ata_intrq2 only reset to 0 if during reset the interrupt input is low.

Table 23-12. INTERRUPT_PENDING Register Field Descriptions

Field	Description
7 ata_intrq1	ATA interrupt request 1. This bit reflects the value of the ata_intrq interrupt input. It is set in the interrupt pending register when the drive interrupt is pending, cleared otherwise. When the bit is set in the interrupt pending register, and the same bit is set in the interrupt enable register, fifo_txfer_end_alarm is asserted, signalling the end of the transfer to the DMA. The interrupt clear register has no influence on this bit.
6 fifo_underflow	FIFO underflow. This bit reports FIFO underflow. Sticky bit. It is set in the interrupt pending register when there is a FIFO underflow condition. It is cleared by writing a ‘1’ to this bit in the interrupt clear register. When the bit is set in the interrupt pending register, and the same bit is set in the interrupt enable register, ipbus_int IS active, signalling interrupt to the CPU.
5 fifo_overflow	FIFO overflow. This bit reports FIFO overflow. Sticky bit. It is set in the interrupt pending register when there is a FIFO overflow condition. It is cleared by writing a ‘1’ to this bit in the interrupt clear register. When the bit is set in the interrupt pending register, and the same bit is set in the interrupt enable register, ipbus_int will be active, signalling interrupt to the CPU.
4 controller_idle	Controller idle. This bit reports controller idle. It is set when the ATA protocol engine is idle; there is no activity on the ATA bus. It is cleared when there is activity on the ATA bus. When the bit is set in the interrupt pending register, and the same bit is set in the interrupt enable register, ipbus_int is active, signalling interrupt to the CPU. The interrupt clear register has no influence on this bit.

Table 23-12. INTERRUPT_PENDING Register Field Descriptions (continued)

Field	Description
3 ata_intrq2	ATA interrupt request 2. This bit reflects the value of the ata_intrq interrupt input. It is set in the interrupt pending register when the drive interrupt is pending, cleared otherwise. It has exactly the same functioning as ata_intrq1, but this bit affects ipbus_int, while the other affects interrupt to the DMA. When the bit is set in the interrupt pending register, and the same bit is set in the interrupt enable register, ipbus_int is asserted, signalling the CPU the drive is requesting attention. The interrupt clear register has no influence on this bit.
2–0 Uncommitted	Reserved

23.3.3.5.2 INTERRUPT_ENABLE Register

Figure 23-42 shows the INTERRUPT_ENABLE register, and Table 23-13 shows the register's field descriptions.

0x43F8_C02C (INTERRUPT_ENABLE) Access: User read/write

	7	6	5	4	3	2	1	0
R	ata_intrq1	fifo_ underflow	fifo_ overflow	controller_ idle	ata_irtrq2	0	0	0
W								
Reset	0	0	0	0	0	—	—	—

Figure 23-42. INTERRUPT_ENABLE Register

Table 23-13. INTERRUPT_ENABLE Register Field Descriptions

Field	Description
7 ata_intrq1	ATA interrupt request 1. This bit reflects the value of the ata_intrq interrupt input. It is set in the interrupt pending register when the drive interrupt is pending, cleared otherwise. When the bit is set in the interrupt pending register, and the same bit is set in the interrupt enable register, fifo_txfer_end_alarm is asserted, signalling the end of the transfer to the DMA. The interrupt clear register has no influence on this bit.
6 fifo_underflow	FIFO underflow. This bit reports FIFO underflow. Sticky bit. It is set in the interrupt pending register when there is a FIFO underflow condition. It is cleared by writing a '1' to this bit in the interrupt clear register. When the bit is set in the interrupt pending register, and the same bit is set in the interrupt enable register, ipbus_int is active, signalling interrupt to the CPU.
5 fifo_overflow	FIFO overflow. This bit reports FIFO overflow. Sticky bit. It is set in the interrupt pending register when there is a FIFO overflow condition. It is cleared by writing a '1' to this bit in the interrupt clear register. When the bit is set in the interrupt pending register, and the same bit is set in the interrupt enable register, ipbus_int is active, signalling interrupt to the CPU.
4 controller_idle	Controller idle. This bit reports controller idle. It is set when the ATA protocol engine is idle; there is no activity on the ATA bus. It is cleared when there is activity on the ATA bus. When the bit is set in the interrupt pending register, and the same bit is set in the interrupt enable register, ipbus_int is active, signalling interrupt to the CPU. The interrupt clear register has no influence on this bit.

Table 23-13. INTERRUPT_ENABLE Register Field Descriptions (continued)

Field	Description
3 ata_intrq2	ATA interrupt request 2. This bit reflects the value of the ata_intrq interrupt input. It is set in the interrupt pending register when the drive interrupt is pending, cleared otherwise. It has exactly the same functioning as ata_intrq1, but this bit affects ipbus_int, while the other affects interrupt to the DMA. When the bit is set in the interrupt pending register, and the same bit is set in the interrupt enable register, ipbus_int is asserted, signalling the drive is requesting attention to the CPU. The interrupt clear register has no influence on this bit.
2–0 Uncommitted	Reserved

23.3.3.5.3 INTERRUPT_CLEAR Register

Figure 23-43 shows the INTERRUPT_CLEAR register, and Table 23-14 shows the register's field descriptions.

0x43F8_C030 (INTERRUPT_CLEAR) Access: User write-only

	7	6	5	4	3	2	1	0
R	0			0	0	0	0	0
W		fifo_ underflow	fifo_ overflow					
Reset	—	—	—	—	—	—	—	—

Figure 23-43. INTERRUPT_CLEAR Register

Table 23-14. INTERRUPT_CLEAR Register Field Descriptions

Field	Description
7 Uncommitted	Reserved
6 fifo_underflow	FIFO underflow. This bit reports FIFO underflow. Sticky bit. It is set in the interrupt pending register when there is a FIFO underflow condition. It is cleared by writing a '1' to this bit in the interrupt clear register. When the bit is set in the interrupt pending register, and the same bit is set in the interrupt enable register, ipbus_int is active, signalling interrupt to the CPU.
5 fifo_overflow	FIFO overflow. This bit reports FIFO overflow. Sticky bit. It is set in the interrupt pending register when there is a FIFO overflow condition. It is cleared by writing a '1' to this bit in the interrupt clear register. When the bit is set in the interrupt pending register, and the same bit is set in the interrupt enable register, ipbus_int is active, signalling interrupt to the CPU.
4–0 Uncommitted	Reserved

23.3.3.6 FIFO_ALARM Register

Figure 23-44 shows the FIFO_ALARM register.

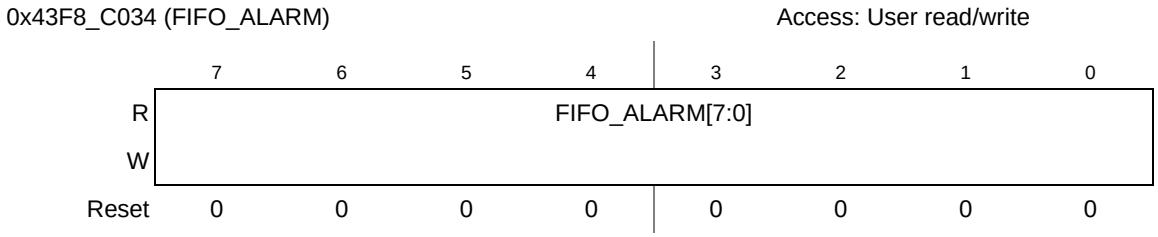


Figure 23-44. FIFO_ALARM Register

This register contains the threshold to generate fifo_rcv_alarm and fifo_tx_alarm to the DMA interface.

- If (fifo_tx_enable == 1 && fifo_fill < fifo_alarm): fifo_tx_alarm is set 1, request is made to DMA to refill fifo.
- If (fifo_rcv_alarm == 1 && fifo_fill >= fifo_alarm): fifo_rcv_alarm is set 1, request is made to DMA to empty fifo.

23.3.3.7 Drive Registers Connected to ATA Bus

Table 23-15. Drive Registers Connected to ATA Bus

Address	Name	Description	Access
0x43F8_C0A0 (DRIVE_DATA)	drive_data	drive data register	R/W
0x43F8_C0A4 (DRIVE_FEATURES)	drive_features	drive features register	R/W
0x43F8_C0A8 (DRIVE_SECTOR_COUNT)	drive_sector_count	drive sector count register	R/W
0x43F8_C0AC (DRIVE_SECTOR_NUM)	drive_sector_num	drive sector number register	R/W
0x43F8_C0B0 (DRIVE_CYL_LOW)	drive_cyl_low	drive cylinder low register	R/W
0x43F8_C0B4 (DRIVE_CYL_HIGH)	drive_cyl_high	drive cylinder high register	R/W
0x43F8_C0B8 (DRIVE_DEV_HEAD)	drive_dev_head	drive device head register	R/W
0x43F8_C0BC (DRIVE_STATUS[rd]/DRIVE_COMMAND[wr])	drive_status/drive_command	drive status register/drive command register	R/W
0x43F8_C0D8 (DRIVE_ALT_STATUS[rd]/DRIVE_CONTROL[wr])	drive_alt_status/drive_control	Drive alternate status register/Drive control register	R/W

Some registers are addressable, but are not present in the ATA interface module. See the list in Table 23-15. If a read or write access is made to one of these registers, the read or write is mapped to a PIO read or write cycle on the ATA bus, and the corresponding register in the device attached to the ATA bus is accessed.

If the drive_data register is accessed while the ATA interface operates in Big Endian mode, the bytes to/from the ATA bus are swapped. No swaps occur in Little Endian mode, nor for any other register.

23.4 Functional Description

The ATA interface provides two ways to communicate with the ATA peripherals connected to the ATA bus.

- PIO mode read/write operation to the ATA bus
- DMA transfers with the ATA bus

The operation of the peripheral is described in detail in the following sections.

23.4.1 Resetting ATA Bus

The ATA bus reset $\overline{\text{ata\_reset}}$ is asserted whenever bit 6 $\overline{\text{ata\_rst}}$ of register ata_control is cleared to 0. At the same time, the ATA protocol engine is reset. When this bit is set to 1, the reset is released.

23.4.2 Programming ATA Bus Timing and iordy_en

The timing that the ATA interface operates with on the ATA bus is programmable. The 24 timing registers at 0x43F8_C00 to 0x43F8_C017 are used for this. Section 23.2, “External Signal Description” details how these registers affect the timing parameters on the ATA bus. It is allowed to reprogram these registers at any time when the ATA bus is idle, so before reprogramming, ensure the following:

- Bit dma_pending in ata_control register is cleared.
- Bit controller_idle in interrupt_pending register is set.

These two conditions can be accomplished by first writing dma_pending to ‘0’, then waiting until controller_idle is set, then reprogramming the timing parameters. If dma_pending was ‘1’ before the reprogramming started, it should be set again after new timing is in effect to allow the drive to finish the current DMA transfer.

It is necessary to wait for controller_idle because a PIO read or write to the ATA bus terminates after the bus cycle with the CPU has been terminated. If the wait for controller_idle does not occur, the new timing values can affect a bus cycle that is still running and cause error.

The bit iordy_en in register ata_control influences whether the ATA interface responds to the iordy signal coming from the drive. To reprogram it, same rules as for the timing registers apply: Reprogramming is allowed only when dma_pending is cleared, while controller_idle is set.

23.4.3 Access to ATA Bus in PIO Mode

Access to the ATA bus in PIO mode is possible after the following conditions:

- The $\overline{\text{ata\_rst}}$ bit in register ata_control is set.
- Timing parameters have been programmed.

To access the drive in PIO mode, simply read or write to the correct drive register. The bus cycle is translated to an ATA cycle, and the drive is accessed.

When drive registers are accessed while the ATA bus is in reset, the read or write is discarded, not done.

23.4.4 Using DMA Mode to Receive Data from ATA Bus

Apart from PIO mode, the ATA interface supports also MDMA and UDMA mode to transfer data. DMA mode can be used to receive data from the drive (DMA in transfer). In DMA receive mode, the protocol engine transfers data from the drive to the FIFO using multiword DMA or ultra DMA protocol. The transfer pauses when one of following occurs:

- The FIFO is full.
- The drive deasserts its dma request signal ata_dmarq.
- The bit dma_pending in the ata_contol register is cleared.

When the cause of the transfer pausing is removed, the transfer restarts. The end of the transfer is signalled by the drive to the host by asserting the ata_intrq signal. Alternatively, the host can read the device status register. In this register, the drive also indicates if the transfer has ended.

The transfer of data from the FIFO into the memory is handled by the host system DMA. Whenever the FIFO filling is above the alarm threshold, the DMA should read one packet of data from the FIFO and store this in main memory. In doing so, the DMA prevents the FIFO from getting full, and keeps the transfer from drive to FIFO running.

The steps for setting up a DMA data transfer from device to host are the following:

1. Ensure the ATA bus is not in reset, and that all timing registers are programmed.
2. Ensure the FIFO is empty by reading it until it is empty or by resetting it.
3. Initialize the DMA channel connected to fifo_rcv_alarm. Every time fifo_rcv_alarm is high, the DMA should read <packetsize> long interrupts from the FIFO and store them to main memory. (Typical packetsize is 8 longs.)
4. Write 2 * <packetsize> to fifo_alarm register. This way, the FIFO requests attention from the DMA when there is at least one packet ready for transfer.
5. To make the ATA ready for a DMA transfer from device to host, perform the following steps:
   a) Ensure the FIFO is out of reset by setting bit $\overline{\text{fifo\_rst}}$ to ‘1’ in the ATA control register.
   b) Program fifo_rcv_en=1 in ata_control register. This enables the FIFO to by emptied by the DMA.
   c) Program dma_pending =1, dma_write=0, ultra_mode_selected=0/1 in ata_control register. The ultra_mode_selected bit should be 1 if you want to transfer data using the UDMA mode. It should be 0 if you want to transfer data using the MDMA mode.
6. The host side of the DMA is ready. Send commands to the drive in PIO mode that cause it to request DMA transfer on the ATA bus. The nature of these commands is beyond the scope of this document. Refer to the ATA specification for details about communicating with the drive.
7. When the drive now requests DMA transfer by pulling ata_dmarq high, the ATA interface acknowledges with ata_dmack, and the transfer starts. Data is transferred automatically to the FIFO, and from there, on to the host memory.
8. During the transfer, the host can monitor for end of transfer by reading some device ATA registers. These reads caused the running DMA to pause. After the read is completed, the DMA resumes. The host can also wait unit the drive asserts ata_intrq. This also indicates end of transfer.

9. At the end of transfer, the host or host DMA should wait until controller_idle is set, and next read the remaining halfwords from the FIFO, and transfer these to memory.

NOTE

There can be less than <packetsize> remaining bytes, so transfer is not automatic by the DMA.

23.4.5 Using DMA Mode to Transmit Data to ATA Bus

Apart from PIO mode, the ATA interface also supports MDMA and UDMA mode to transfer data. DMA mode can be used to transmit data to the drive (DMA out transfer). In DMA transmit mode, the protocol engine transfers data from the FIFO to the drive using metalwork DMA or ultra DMA protocol. The transfer pauses when one of following occurs:

- The FIFO is empty.
- The drive deasserts its dma request signal ata_dmarq.
- The bit dma_pending in the ata_contol register is cleared.

When the cause of the transfer pausing is removed, the transfer restarts. The end of the transfer is signalled by the drive to the host by asserting the ata_intrq signal. Alternatively, the host can read the device status register. In this register, the drive also indicates if the transfer has ended.

The transfer of data from the memory to the FIFO is handled by the host system DMA. Whenever the FIFO filling is below the alarm threshold, the DMA should read one packet of data from the main memory, and store this in the FIFO. In doing so, the DMA prevents the FIFO from getting empty, and keeps the transfer from FIFO to drive running.

The steps for setting up a DMA data transfer from device to host are the following:

1. Ensure the ATA bus is not in reset and all timing registers are programmed.
2. Ensure the FIFO is empty by reading it until it is empty, or by resetting it.
3. Initialize the DMA channel connected to fifo_tx_alarm. Every time fifo_tx_alarm is high, the DMA should read <packetsize> long interrupts from the main memory, and write them to the FIFO. (Typical packetsize is 8 longs.) Program the DMA so that it does not transfer more than <sectorsize> longwords in total.
4. Write FIFO_SIZE–2 * <packetsize> to fifo_alarm register. This way, the FIFO requests attention from the DMA when there is room for at least one extra packet. FIFO_SIZE should be given in halfwords. (typical 64 halfwords)
5. To make the ATA ready for a DMA transfer from host to device, perform the following steps:
   a) Ensure the FIFO is out of reset by setting bit $\overline{\text{fifo\_rst}}$ to 1 in ata_control register.
   b) Program fifo_tx_en=1 in ata_control register. This enables the FIFO to be filled by DMA.
   c) Program dma_pending =1, dma_write=1, ultra_mode_selected=0/1 in ata_control register. The ultra_mode_selected bit should be 1 if you want to transfer data using UDMA mode. It should be 0 if you want to transfer data using MDMA mode.

6. The host side of the DMA is ready. Send commands to the drive in PIO mode that cause it to request DMA transfer on the ATA bus. The nature of these commands is beyond the scope of this document. Refer to the ATA specification for details about communicating with the drive.
7. When the drive now requests DMA transfer by pulling ata_dmarq high, the ATA interface acknowledges with ata_dmack, and the transfer starts. Data is transferred automatically from the FIFO, and also from host memory to the FIFO.
8. During the transfer, the host can monitor for end of transfer by reading some device ATA registers. These reads cause the running DMA to pause. After the read is completed, the DMA resumes. The host can also wait unit the drive asserts ata_intrq. This also indicates end of transfer.

On end of transfer, no extra FIFO manipulations are needed.

Chapter 24
Configurable Serial Peripheral Interface (CSPI)

The Configurable Serial Peripheral Interface (CSPI) module allows rapid data communication with fewer software interrupts than conventional serial communications. The CSPI module contains one 8 × 32 receive buffer (RXFIFO) and one 8 × 32 transmit buffer (TXFIFO). Figure 24-1 shows the CSPI block diagram.

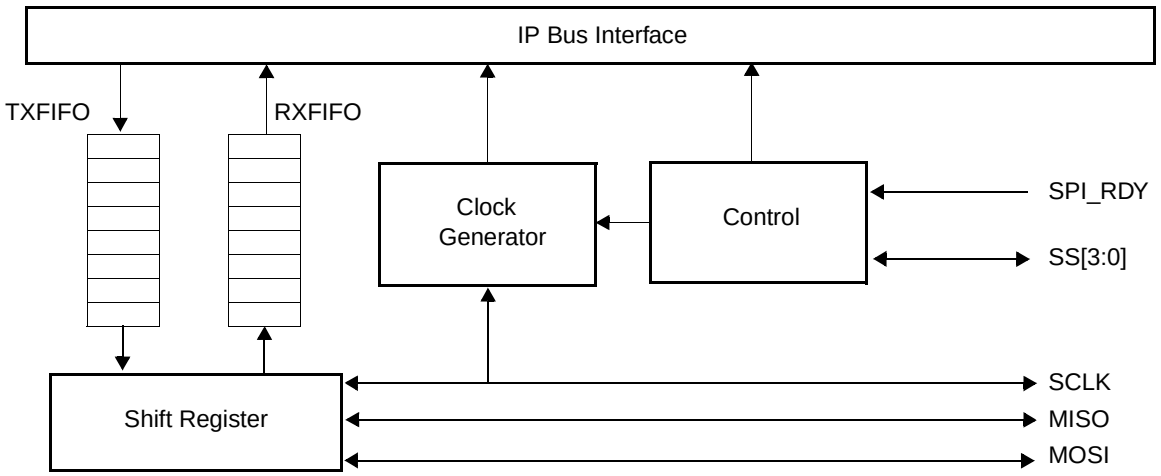


Figure 24-1. CSPI Block Diagram

24.1 Features

The CSPI is used for fast data communication with fewer software interrupts. It includes the following features:

- Full-duplex synchronous serial interface
- Master/Slave configurable
- Four chip selects to support multiple peripherals
- Transfer continuation function allows unlimited length data transfers
- 32-bit wide by 8-entry FIFO for both transmit and receive data
- Polarity and phase of the Chip Select ($\overline{SS}$) and SPI Clock (SCLK) are configurable
- DMA support

24.1.1 Modes of Operation

The following are two CSPI modes of operation:

- **Master Mode**—When the CSPI module is configured as a master, it uses a serial link to transfer data between the CSPI and an external device. A chip-enable signal and a clock signal are used to transfer data between these two devices. If the external device is a transmit-only device, the CSPI master's output port can be ignored and used for other purposes. To use the internal TXFIFO and RXFIFO, two auxiliary output signals, $\overline{SS}$ and $\overline{SPI\_RDY}$, are used for data transfer rate control. The user can also program the sample period control register to a fixed data transfer rate.
- **Slave Mode**–When the CSPI module is configured as a slave, the user can configure the CSPI Control register to match the external SPI master's timing. In this configuration, $\overline{SS}$ becomes an input signal, and is used to control data transfers through the Shift register, as well as to load/store the data FIFO.

24.2 External Signal Description

Table 24-1 provides a detailed description of the CSPI signals.

Table 24-1. CSPI–Detailed Signal Descriptions

Signal	I/O	Reset	Description
MOSI	I/O	O	Master Out Slave In. In Master mode, this bidirectional signal is a TX output signal from the Data Shift register. In Slave mode, it is an RX input from external SPI device.
$\overline{MISO}$	I/O	O	Master In Slave Out—In Master mode, this bidirectional signal is an RX input signal to the Data Shift register. In Slave mode, it is a TX output to external SPI device.
$\overline{SCLK}$	I/O	O	SPI Clock—In Master mode, this bidirectional signal is a SPI clock output. In Slave mode, it is a SPI clock input.
$\overline{SS[3:0]}$	I/O	I	Chip selects—In Master mode, these bidirectional signals are outputs. In Slave mode, these are inputs. These signals are selected in/out by Chip Select bits in the CSPI Control register.
SPI_RDY	I	I	SPI Ready—It is an input from an external SPI slave device. This signal triggers the CSPI to start a burst. Edge-trigger or level-trigger can be configured in the CSPI Control register.

24.3 Memory Map and Register Definition

The CSPI includes eight 32-bit registers. Section 24.3.3, "Register Descriptions" provides the detailed descriptions for the CSPI registers.

24.3.1 Memory Map

Table 24-2 shows the CSPI memory map.

Table 24-2. CSPI Memory Map

Address	Register	Access	Reset Value	Section/Page
0x43FA_4000 (RXDATA1) 0x5001_0000 (RXDATA2) 0x53F8_4000 (RXDATA3)	Receive Data Register (RXDATA)	R	0x0000_0000	24.3.3.1/24-5
0x43FA_4004 (TXDATA1) 0x5001_0004 (TXDATA2) 0x53F8_4004 (TXDATA3)	Transmit Data Register (TXDATA)	W	0x0000_0000	24.3.3.2/24-6
0x43FA_4008 (CONREG1) 0x5001_0008 (CONREG2) 0x53F8_4008 (CONREG3)	Control Register (CONREG)	R/W	0x0000_0000	24.3.3.3/24-7
0x43FA_400C (INTREG1) 0x5001_000C (INTREG2) 0x53F8_400C (INTREG3)	Interrupt Control Register (INTREG)	R/W	0x0000_0000	24.3.3.4/24-10
0x43FA_4010 (DMAREG1) 0x5001_0010 (DMAREG2) 0x53F8_4010 (DMAREG3)	DMA Control Register (DMAREG)	R/W	0x0000_0000	24.3.3.5/24-12
0x43FA_4014 (STATREG1) 0x5001_0014 (STATREG2) 0x53F8_4014 (STATREG3)	Status Register (STATREG)	R/W	0x0000_0003	24.3.3.6/24-13
0x43FA_4018 (PERIODREG1) 0x5001_0018 (PERIODREG2) 0x53F8_4018 (PERIODREG3)	Sample Period Control Register (PERIODREG)	R/W	0x0000_0000	24.3.3.7/24-14

24.3.2 Register Summary

Figure 24-2 shows the key to the register fields, and Table 24-3 shows the register figure conventions.

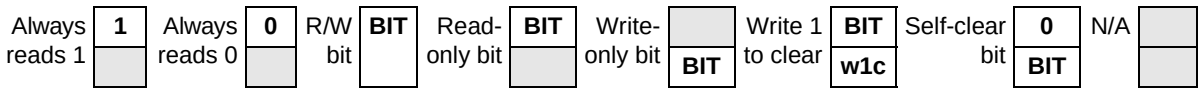


Figure 24-2. Key to Register Fields

Table 24-3. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.

Table 24-3. Register Figure Conventions (continued)

Convention	Description
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 24-4 shows the CSPI register summary.

Table 24-4. CSPI Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x43FA_4000 (RXDATA1) 0x5001_0000 (RXDATA2) 0x53F8_4000 (RXDATA3)	R	RXDATA[31:16]															
	W																
	R	RXDATA[15:0]															
	W																
0x43FA_4004 (TXDATA1) 0x5001_0004 (TXDATA2) 0x53F8_4004 (TXDATA3)	R																
	W	TXDATA[31:16]															
	R																
	W	TXDATA[15:0]															
0x43FA_4008 (CONREG1) 0x5001_0008 (CONREG2) 0x53F8_4008 (CONREG3)	R	0	0	0	0	0	0	CHIP SELECT		0	0	DRCTL		0	DATA RATE		
	W																
	R	0	0		BIT COUNT					SSPOL	SSCTL	PHA	POL	SMC	XCH	MODE	EN
	W																
0x43FA_400C (INTREG1) 0x5001_000C (INTREG2) 0x53F8_400C (INTREG3)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	TCEN	BOEN	ROEN	RFEN	RHEN	RREN	TFEN	THEN	TEEN
	W																

Table 24-4. CSPI Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x43FA_4010 (DMAREG1) 0x5001_0010 (DMAREG2) 0x53F8_4010 (DMAREG3)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	RF DEN	RH DEN	0	0	TH DEN	TE DEN
	W																
0x43FA_4014 (STATREG1) 0x5001_0014 (STATREG2) 0x53F8_4014 (STATREG3)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	TC	BO	RO	RF	RH	RR	TF	TH	TE
	W								w1c	w1c							
0x43FA_4018 (PERIODREG1) 0x5001_0018 (PERIODREG2) 0x53F8_4018 (PERIODREG3)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	CSRC	SAMPLE PERIOD[14:0]														
	W																
0x43FA_41C0 (TESTREG1) 0x5001_01C0 (TESTREG2) 0x53F8_41C0 (TESTREG3)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	SWAP	LBC	0	0	SMSTATUS				RXCNT				TXCNT			
	W																

24.3.3 Register Descriptions

The following section describes the detailed register descriptions for the CSPI registers.

24.3.3.1 Receive Data Register (RXDATA)

The Receive Data register (RXDATA) is a read-only register that forms the top word of the 8×32 receive FIFO. This register holds the data received from an external SPI device during a data transaction. Only word-sized read operations are allowed.

Figure 24-3 shows the RXDATA register, and Table 24-5 shows the register's field descriptions.

0x43FA_4000 (RXDATA1)
0x5001_0000 (RXDATA2)
0x53F8_4000 (RXDATA3)

Access: User read-only

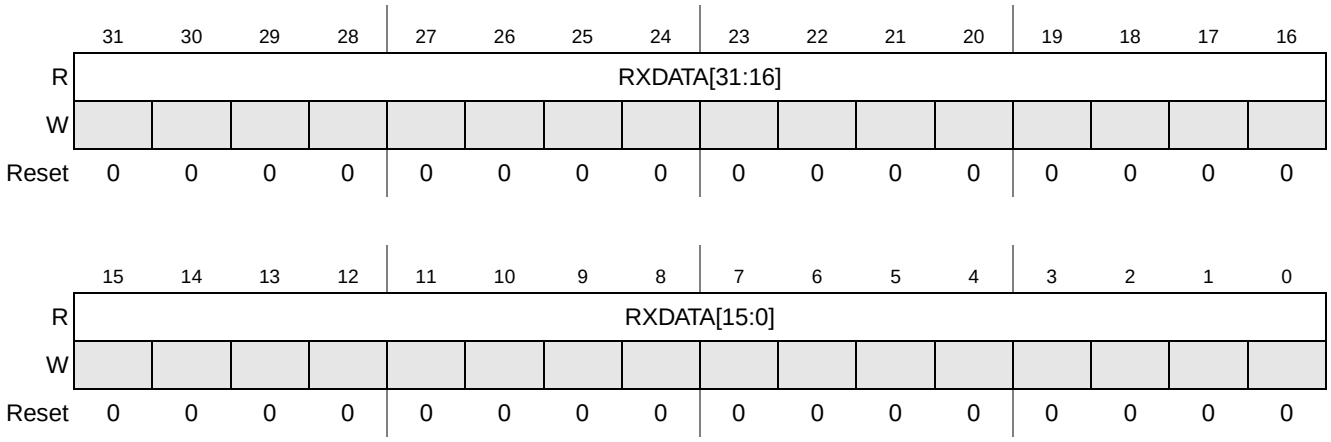


Figure 24-3. RXDATA Register Diagram

Table 24-5. RXDATA Register Field Descriptions

Field	Description
31–0 RXDATA	Receive Data. This register holds the top word of the receive data FIFO. The FIFO is advanced for each read of this register. The data read is undefined when the Receive Data Ready (RR) bit in the Interrupt Control/Status register is cleared. Zeros are read when CSPI is disabled.

24.3.3.2 Transmit Data Register (TXDATA)

The Transmit Data (TXDATA) register is a write-only data register that forms the top word of the 8×32 TXFIFO. The TXFIFO can be written to as long as it is not full, even when the XCH bit in CONREG is set. This allows the user write access to the TXFIFO during a SPI data exchange process. Writes to this register are ignored when the CSPI module is disabled (EN bit of CSPI CONREG is cleared).

Figure 24-4 shows the CNTRL register, and Table 24-6 shows the register's field descriptions.

0x43FA_4004 (TXDATA1)
0x5001_0004 (TXDATA2)
0x53F8_4004 (TXDATA3)

Access: User write-only

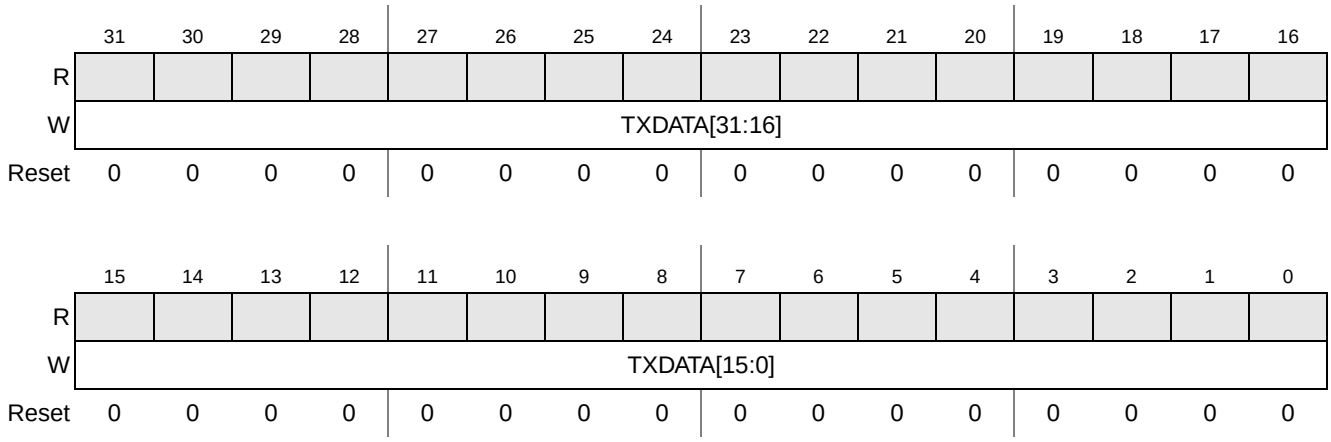


Figure 24-4. TXDATA Register Diagram

Table 24-6. TXDATA Register Field Descriptions

Field	Description
31–0 TXDATA	Transmit Data. This register holds the top word of data loaded into the FIFO. Data written to this register must be a word operation. The number of bits actually transmitted is determined by the BIT_COUNT field of the corresponding SPI Control register. If this field contains more bits than the number specified by BIT_COUNT, the extra bits are ignored. For example, to transfer 10 bits of data, a 32-bit word must be written to this register. Bits 9-0 are shifted out and bits 31-10 are ignored. When the CSPI module is operating in Slave mode, zeros are shifted out when the FIFO is empty. Zeros are read when CSPI is disabled.

24.3.3.3 Control Register (CONREG)

The Control Register (CONREG) allows the user to enable the CSPI module, configure its operating modes, specify the divider value, phase, and polarity of the clock, configure the $\overline{SS}$ and $\overline{SPI\_RDY}$ control signal, and define the transfer length. The reserved bits are always read as 0.

Figure 24-5 shows the CNTRL register, and Table 24-7 shows the register's field descriptions.

0x43FA_4008 (CONREG1)
0x5001_0008 (CONREG2)
0x53F8_4008 (CONREG3)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	CHIP SELECT		0	0	DRCTL		0	DATA RATE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	BIT COUNT					SSPOL	SSCTL	PHA	POL	SMC	XCH	MODE	EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 24-5. CSPI Control Register

Table 24-7. CONREG Register Field Descriptions

Field	Description
31–26	Reserved, all bits should read zero.
25–24 CHIP SELECT	CHIP SELECT. Select one of four external SPI Master/Slave Devices. In master mode, these two bits select the external slave devices by asserting the $\overline{SS}$n outputs.Only the selected $\overline{SS}$n signal will be active while the remaining 3 signals will be negated. Chip Select 00 $\overline{SS}$0 will be asserted. 01 $\overline{SS}$1 will be asserted. 10 $\overline{SS}$2 will be asserted. 11 $\overline{SS}$3 will be asserted.
23-22	Reserved, all bits should read zero.
21–20 DRCTL	SPI Data Ready Control. This 2-bit field selects the utilization of the $\overline{SPI\_RDY}$ in master mode. CSPI will check this fields before it start a SPI burst. 00 Don't Care $\overline{SPI\_RDY}$ 01 Burst will be triggered by failing edge of $\overline{SPI\_RDY}$. 10 Burst will be triggered by low level of $\overline{SPI\_RDY}$. 11 $\overline{RSV}$.
19	Reserved, all bits should read zero.

Table 24-7. CONREG Register Field Descriptions (continued)

Field	Description
18–16 DATA RATE	SPI Data Rate Control. This three-bit field selects the baud rate of the SCLK based on a division of the ipg_clk. These bits allow CSPI to synchronize with different external SPI devices. The max frequency is one quarter of ipg_clk. The divide ratio is determined according to the following table using the equation: $2^{(n+2)}$. SPI Data Rate Control (Master Mode only) 000 Divide by 4. 001 Divide by 8. 010 Divide by 16. 011 Divide by 32. 100 Divide by 64. 101 Divide by 128. 110 Divide by 256. 111 Divide by 512.
15–13	Reserved, all bits should read zero.
12–8 BIT COUNT	This field selects the length of a word to be transferred. A maximum of 32 bits can be transferred in a single SPI transfer. Multiple transfers may be chained together to form unlimited length messages using the SSCTL bit to keep the $\overline{SS}$ asserted between transfers. In master mode, BITCOUNT controls the number of bits per serial transfer. The transmit FIFO transfers a 32-bit data word to the shift register, however only the n least-significant (n=BIT COUNT + 1) are shifted out. The remaining bits are ignored. In slave mode, provided SSCTL= 0, this field controls the number of bits (BIT COUNT + 1) received in each data word. After BITCOUNT + 1 bits have been shifted into the shift register, the contents are transferred to the receive FIFO regardless of the state of the $\overline{SS}$ input. When SSCTL bit is 1, data transfer to/from the FIFO are controlled by the $\overline{SS}$ input and this field is ignored. SPI Data Rate Control (Master Mode only) 00000 Least 1 bit of a word to be transferred. 00001 Least 2 bits of a word to be transferred. ..... 01111 Least 16 bits of a word to be transferred. 10000 Least 17 bits of a word to be transferred. ..... 11110 Least 31 bits of a word to be transferred. 11111 All 32 bits of a word to be transferred.
11–10	Reserved, all bits should read zero.
7 SSPOL	SPI SS Polarity Select. In both Master and Slave mode, this bit selects the polarity of the $\overline{SS}$ signal. 0 Active low 1 Active high
6 SSCTL	In master mode, this bit selects the output wave form for the SS signal. 0 SS remains asserted between SPI bursts. 1 Negate SS between SPI bursts. In slave mode, this bit controls the timing of data transfer from the shift register to the receive FIFO. 0 RXFIFO advanced by BIT COUNT. 1 RXFIFO advanced by SS edge. (SSPOL = 0: rising edge; SSPOL = 1: falling edge)
5 PHA	SPI Clock/Data Phase Control. This bit controls the clock/data phase relationship. Refer to Figure 24-18, SPI Burst with Different POL and PHA Configuration, for a description. 0 Phase 0 operation. 1 Phase 1 operation.

Table 24-7. CONREG Register Field Descriptions (continued)

Field	Description
4 POL	SPI Clock Polarity Control. This bit controls the polarity of the SCLK signal. Refer to Figure 24-18, SPI Burst with Different POL and PHA Configuration, for a description. 0 Active high polarity (0 = Idle) 1 Active low polarity (1 = Idle)
3 SMC	Start Mode Control. This bit is used in master mode only and it controls how CSPI start a SPI burst. 0 XCH bit controls when a SPI burst can start. Write a 1 to XCH bit will start a SPI burst or multiple bursts. (controlled by SSCTL) 1 Immediately start a SPI burst when data is written in TXFIFO.
2 XCH	SPI Exchange Bit. If the SMC bit is cleared, writing a 1 to this bit starts one SPI bursts/multiple SPI bursts according to SSCTL bit. This bit remains set while either the exchange is in progress, or the CSPI is waiting for active input if $\overline{\text{SPIRDY}}$ is enabled through DRCTL. This bit is cleared automatically when all data in the TXFIFO and Shift register have been shifted out. In Slave mode, this bit is ignored. 0 Idle 1 Initiates exchange (write) or busy (read)
1 MODE	SPI Function Mode Select. This bit selects the operating mode of the CSPI. 0 Slave Mode 1 Master Mode
0 EN	SPI Module Enable Control. This bit enables the CSPI. This bit must be asserted before writing to other registers or initiating an exchange. Writing zero to this bit disables the module and resets the internal logic with the exception of the CONREG. The module's internal clocks are gated off whenever the module is disabled. 0 CSPI is disabled. 1 CSPI is enabled.

24.3.3.4 Interrupt Control Register (INTREG)

The 32-bit Interrupt Control Register (INTREG) enables the generation of interrupts to the MCU. The reserved bits cannot be written and always read as 0. If CSPI is disabled, this register reads zero.

Figure 24-6 shows the CNTRL register, and Table 24-8 shows the register's field descriptions.

0x43FA_400C (INTREG1)
0x5001_000C (INTREG2)
0x53F8_400C (INTREG3)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	TCEN	BOEN	ROEN	RFEN	RHEN	RREN	TFEN	THEN	TEEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 24-6. Interrupt Control Register Diagram

Table 24-8. INTREG Register Field Descriptions

Field	Description
31–9	Reserved, all bits should read zero.
8 TCEN	Transfer Completed Interrupt enable. This bit enables the Transfer Completed Interrupt. 0 Disable 1 Enable
7 BOEN	Bit Counter Overflow Interrupt enable. This bit enables the Bit Counter overflow Interrupt (More than 32 bits are received in a word). 0 Disable 1 Enable
6 ROEN	RXFIFO Overflow Interrupt enable. The bit enables the RXFIFO Overflow Interrupt. 0 Disable 1 Enable
5 RFEN	RXFIFO Full Interrupt enable. The bit enables the RXFIFO Full Interrupt. 0 Disable 1 Enable
4 RHEN	RXFIFO Half Full Interrupt enable. The bit enables the RXFIFO Half Full Interrupt. 0 Disable 1 Enable
3 RREN	RXFIFO Ready Interrupt enable. The bit enables the RXFIFO Ready Interrupt. 0 Disable 1 Enable
2 TFEN	TXFIFO Full Interrupt enable. The bit enables the TXFIFO Full Interrupt. 0 Disable 1 Enable

Table 24-8. INTREG Register Field Descriptions (continued)

Field	Description
1 THEN	TXFIFO Half Empty Interrupt enable. The bit enables the TXFIFO Half Empty Interrupt. 0 Disable 1 Enable
0 TEEN	TXFIFO Empty Interrupt enable. The bit enables the TXFIFO Empty Interrupt. 0 Disable 1 Enable

24.3.3.5 DMA Control Register (DMAREG)

The DMA Control Register (DMAREG) provides the user a way to use the CSPI in DMA. Direct Memory Access (DMA) allows transfer of data between device and memory. Peripherals such as the CSPI supporting DMA use DMA request and acknowledge signals. The CSPI sends out DMA requests when the appropriate FIFO conditions are matched. The reserved bits cannot be written to and are always read as 0. If the CSPI is disabled, this register is also read as 0.

Figure 24-7 shows the CNTRL register, and Table 24-9 shows the register's field descriptions.

0x43FA_4010 (DMAREG1)
0x5001_0010 (DMAREG2)
0x53F8_4010 (DMAREG3)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	RF DEN	RH DEN	0	0	TH DEN	TE DEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 24-7. DMA Control Register Diagram

Table 24-9. DMAREG Register Field Descriptions

Field	Description
31–6	Reserved, all bits should read zero.
5 RFDEN	RXFIFO Full DMA Request Enable. This bit enables/disables the RXFIFO Full DMA Request. 0 Disable 1 Enable
4 RHDEN	RXFIFO Half Full DMA Request Enable. This bit enables/disables the RXFIFO Half Full DMA Request. 0 Disable 1 Enable

Table 24-9. DMAREG Register Field Descriptions (continued)

Field	Description
3–2	Reserved, should be cleared.
1 THDEN	TXFIFO Half Empty DMA Request Enable. This bit enables/disables the TXFIFO Half Empty DMA Request. 0 Disable 1 Enable
0 TEDEN	TXFIFO Empty DMA Request Enable. This bit enables/disables the TXFIFO Empty DMA Request. 0 Disable 1 Enable

24.3.3.6 Status Register (STATREG)

The CSPI Status Register (STATREG) reflects the status of the CSPI module operating condition. The reserved bits cannot be written and always read as 0. If the CSPI is disabled, this register reads 0x0000_0003.

Figure 24-8 shows the CNTRL register, and Table 24-10 shows the register's field descriptions.

0x43FA_4014 (STATREG1)
0x5001_0014 (STATREG2)
0x53F8_4014 (STATREG3)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	TC	BO	RO	RF	RH	RR	TF	TH	TE
W								w1c	w1c							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Figure 24-8. Status Register Diagram

Table 24-10. STATREG Register Field Descriptions

Field	Description
31–9	Reserved, should be cleared.
8 TC	Transfer Completed. When set, this bit indicates that all the data in TXFIFO has been loaded in the Shift register, and the Shift register has shifted out all the bits. Writing 1 to this bit clears it. 0 Busy 1 Transfer Completed

Table 24-10. STATREG Register Field Descriptions (continued)

Field	Description
7 BO	Bit Counter Overflow. When set, this bit indicates that Bit Counter is overflows while the Configurable Serial Peripheral Interface is in slave mode (MODE = 0) with SSCTL = 1. Writing 1 to this bit clears it. 0 Bit Counter is not overflowed. 1 Bit Counter is overflowed.
6 RO	RXFIFO Overflow. When set, this bit indicates that RXFIFO has overflowed. 0 RXFIFO is available. 1 RXFIFO has overflowed.
5 RF	RXFIFO Full. This bit is set when the RXFIFO is full (8 words). 0 Not Full 1 Full
4 RH	RXFIFO Half Full. This bit is set if the RXFIFO is half full (≥ 4 words in RXFIFO). 0 Less than 4 words are stored in RXFIFO. 1 Four or more words are available in RXFIFO.
3 RR	RXFIFO Ready. This bit is set any time there is one or more words stored in RXFIFO (≥ 1 words). 0 No valid data in RXFIFO 1 More than 1 word in RXFIFO
2 TF	TXFIFO Full. This bit is set when if the TXFIFO is full (8 words). 0 TXFIFO is not Full. 1 TXFIFO is Full.
1 TH	TXFIFO Half empty. This bit is set if the TXFIFO is more than half empty (≤ 4 words in TXFIFO). 0 TXFIFO holds more than 4 words. 1 TXFIFO holds 4 or fewer words.
0 TE	TXFIFO Empty. This bit is set if the TXFIFO is empty. 0 TXFIFO contains one or more words. 1 TXFIFO is empty.

24.3.3.7 Sample Period Control Register (PERIODREG)

The Sample Period Control Register (PERIODREG) provides the user a way to insert delays (wait states) between consecutive SPI transfers. Control bits in this register select the clock source for the sample period counter and the delay count indicating the number of wait states to be inserted between data transfers. Delay counts are only applicable when the CSPI module is operating in master mode.

Figure 24-9 shows the CNTRL register, and Table 24-11 shows the register's field descriptions.

0x43FA_4018 (PERIODREG1)
0x5001_0018 (PERIODREG2)
0x53F8_4018 (PERIODREG3)

Access: User read/write

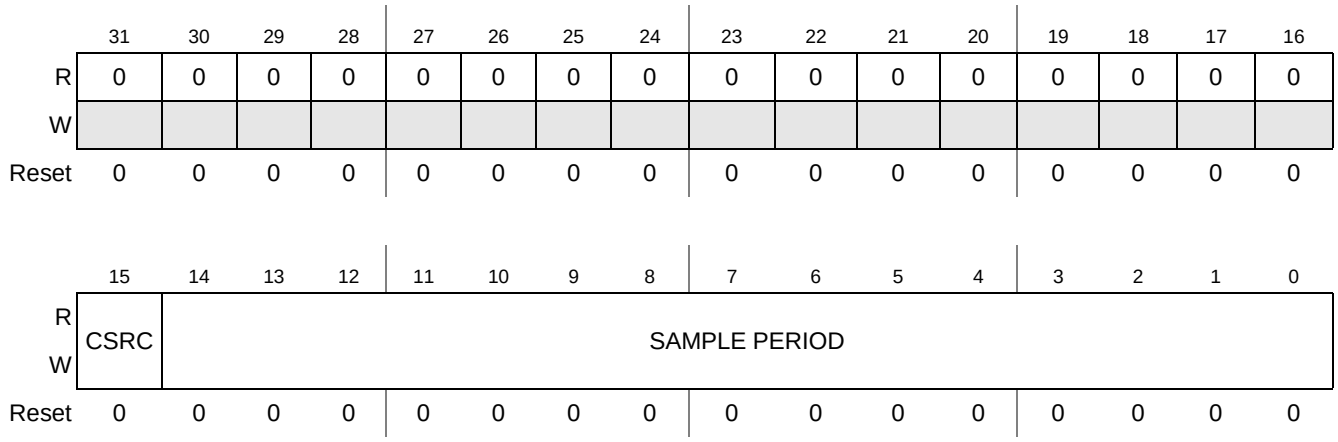


Figure 24-9. Sample Period Control Register Diagram

Table 24-11. PERIODREG Register Field Descriptions

Field	Description
31–16	Reserved, all bits should read zero.
15 CSRC	Clock Source Control. This bit selects the clock source for the sample period counter. 0 SPI Clock (SCLK) 1 CKIL (32.768 KHz)
14–0 SAMPLE PERIOD	Sample Period Control. These bits control the number of wait states to be inserted in data transfers. During the idle clocks, the state of the $\overline{SS}$ output will operate according to the SSCTL control field in CONREG. 0x0000 0 wait states inserted 0x0001 1 wait state inserted .... 0x7FFE 32766 wait states inserted 0x7FFF 32767 wait states inserted

24.3.3.8 Test Control Register (TESTREG)

The Test Control Register (TESTREG) provides the user a mechanism to internally connect the receive and transmit devices of the CSPI module, display the status of the state machine, monitor the contents of the receive and transmit FIFO, and debug the CSPI.

Figure 24-10 shows the CNTRL register, and Table 24-12 shows the register's field descriptions.

0x43FA_41C0 (TESTREG1)
0x5001_01C0 (TESTREG2)
0x53F8_41C0 (TESTREG3)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SWAP	LBC	0	0	SMSTATUS				RXCNT				TXCNT			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 24-10. Test Control Register Diagram

Table 24-12. TESTREG Register Field Descriptions

Field	Description
31–16	Reserved, All bits should be read as zero.
15 SWAP	Data Swap. This bit is used to swap data as it is read from the RXFIFO. When this bit is set, data read from RXFIFO is swapped. RXDATA[31:0] is swapped as follows: {RXDATA[7:0],RXDATA[15:8], RXDATA[23:16],RXDATA[31:24]} 0 Data read from RXFIFO is unchanged. 1 Data read from RXFIFO is swapped.
14 LBC	Loopback Control. This bit is used in Master mode only. When this bit is set, the CSPI module connects the transmitter and receiver sections internally, and the data shifted out from the most-significant bit of the Shift register is looped back into the least-significant bit of the Shift register. In this way, a self-test of the complete transmit/receive path can be made. The output pins are not affected, and the input pins are ignored. 0 Not connected. 1 Internally connected.
13–12	Reserved, all bits should read zero.
11–8 SMSTATUS	State Machine Status. These bits indicate status of the state machine for test purpose.

Table 24-12. TESTREG Register Field Descriptions (continued)

Field	Description
7–4 RXCNT	RXFIFO Counter. These bits indicate the number of words in RXFIFO. 0000 0 word in RXFIFO 0001 1 word in RXFIFO .... 0111 7 words in RXFIFO 1000 8 words in RXFIFO
3–0 TXCNT	TXFIFO Counter. These bits indicate the number of words in TXFIFO. 0000 0 word in TXFIFO 0001 1 word in TXFIFO .... 0111 7 words in TXFIFO 1000 8 words in TXFIFO

24.4 Functional Description

This section describes the timings for the CSPI. Figure 24-11 shows the generic CSPI timing.

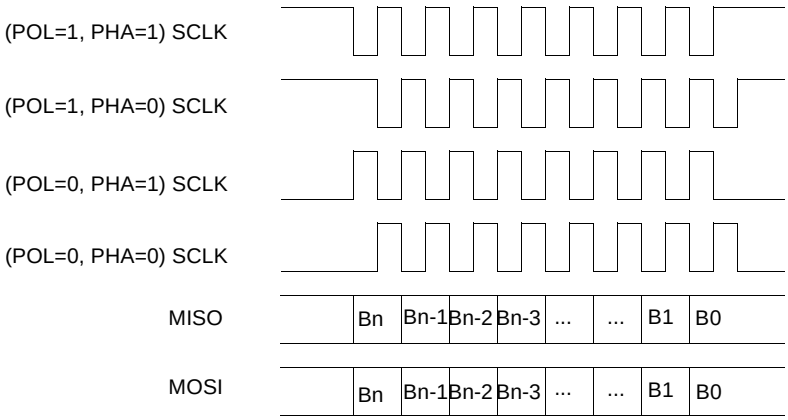


Figure 24-11. CSPI Generic Timing

24.4.1 Phase and Polarity Configurations

The serial peripheral interface master uses the SCLK signal to transfer data in and out of the shift register. Data is clocked using any one of four programmable clock phase and polarity combinations.

During Phase 0, Polarity 0 and Phase 1, Polarity 1 operations, output data changes on the falling clock edge and input data is shifted in on the rising edge. The most-significant bit is output when the CPU loads the transmit data.

During Phase 1, Polarity 0 and Phase 0, Polarity 1 operations, output data changes on the rising edges of the clock and is shifted in on falling edges. The most-significant bit is output on the first rising SCLK edge.

Polarity inverts SCLK, but does not change the edge-triggered events that are internal to the serial peripheral interface master. This flexibility allows it to operate with most serial peripheral devices.

24.4.2 Master Mode

The CSPI master uses the $\overline{SS}$ signal to enable an external SPI device and uses SPICLK to transfer data in and out of the Shift register. The $\overline{SPI\_RDY}$ enables fast data communication with fewer software interrupts. By using PERIODREG, the CSPI can be used for a fixed data transfer rate.

When CSPI is in Master mode the $\overline{SS}$, SCLK, and MOSI are output signals and the MISO is an input. Figure 24-12 shows a typical SPI burst.

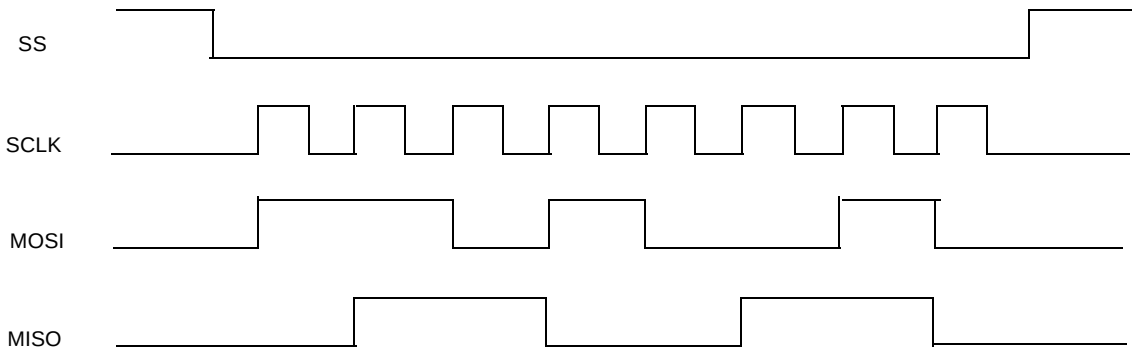


Figure 24-12. Typical SPI Burst (8-bit transfer)

In Figure 24-12, the $\overline{SS}$ signal enables the selected external SPI device and the SCLK synchronizes data transfer. MOSI and MISO change on rising edge of SCLK and the MISO is latched on the falling edge of the SLCK clock. The data shifted out is 0xD2, and the data shifted in is 0x66.

24.4.2.1 Master Mode with $\overline{SPI\_RDY}$

By default, the CSPI does not use $\overline{SPI\_RDY}$ in master mode. a SPI burst begins when the following events happen: the CSPI is enabled, TXFIFO has data in it, and CONREG[XCH] or CONREG[SMC] is set. When CONREG[DRCTL] contains either 01 or 10, the $\overline{SPI\_RDY}$ controls when a SPI burst starts.

If CONREG[DRCTL] is set to 01, the SPI burst can be triggered only if a falling edge of $\overline{SPI\_RDY}$ has been detected. Figure 24-13 shows the relationship between a SPI burst and the falling edge of $\overline{SPI\_RDY}$.

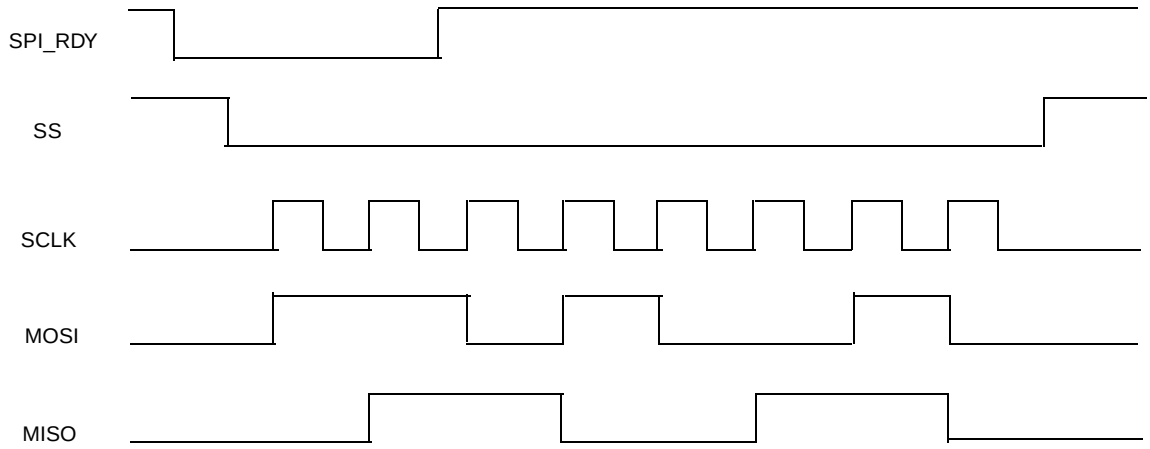


Figure 24-13. Relationship Between a SPI Burst and the Falling Edge of $\overline{\text{SPI\_RDY}}$

An SPI burst does not start until the falling edge of $\overline{\text{SPI\_RDY}}$ is detected. The next SPI burst starts when the next $\overline{\text{SPI\_RDY}}$ falling edge is detected, after the last burst has finished.

If CONREG[DRCTL] is set to 10, the SPI burst can be triggered only if $\overline{\text{SPI\_RDY}}$ is low. Figure 24-14 shows the relationship between a SPI burst and $\overline{\text{SPI\_RDY}}$. The SPI burst does not begin until $\overline{\text{SPI\_RDY}}$ goes low. The next SPI burst begins after the last burst has finished if $\overline{\text{SPI\_RDY}}$ remains low.

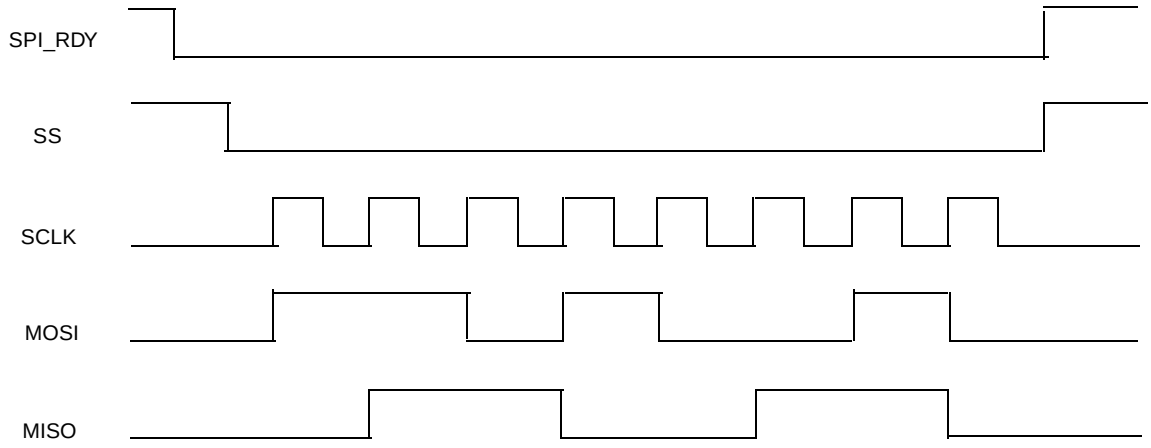


Figure 24-14. Relationship Between a SPI Burst and $\overline{\text{SPI\_RDY}}$

24.4.2.2 Master Mode with Wait States

Wait states can be inserted between SPI bursts. This provides a way for the user to slow down the SPI burst to meet the timing requirements of a slower SPI device. Figure 24-15 shows wait states inserted between SPI bursts.

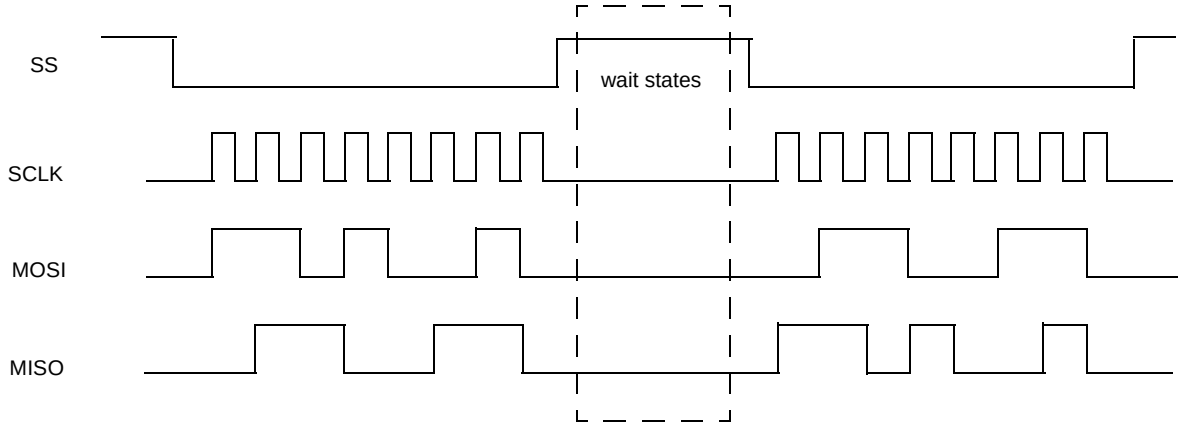


Figure 24-15. SPI Bursts with Wait States

In this case, the number of wait states is controlled by PERIODREG[SAMPLE PERIOD] and the wait states' clock source is selected by PERIODREG[CSRC].

24.4.2.3 Master Mode with SSCTL Control

SSCTL controls whether current operation is single burst or multiple bursts. When SSCTL is set, current operation is multiple bursts transfer. When SSCTL is cleared, current operation is single burst transfer. A SPI burst can contains multiple words as defined in BURST LENGTH.

Figure 24-16 shows one SPI burst while SSCTL is clear.

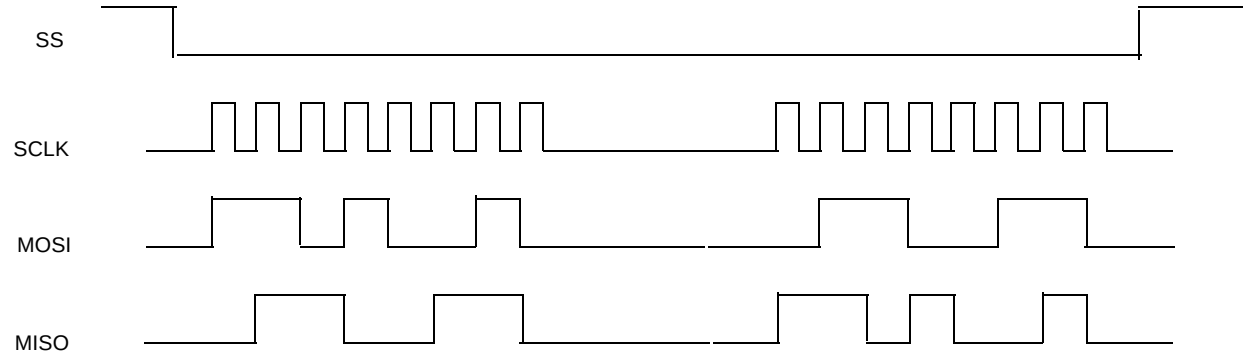


Figure 24-16. SPI Burst While SSCTL is Clear

In this case, two words in RXFIFO have been combined and transmitted in one SPI burst. The maximum length of a single SPI burst is defined in BURST LENGTH field in CONREG. This provides a way for transferring a longer SPI burst by writing data into TXFIFO while CSPI is transmitting.

Figure 24-17 shows two SPI bursts are transmitted while SSCTL is set.

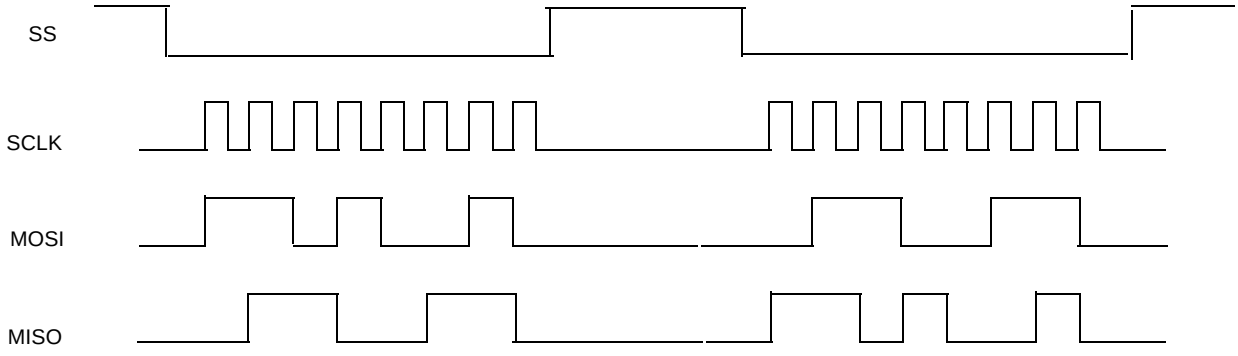


Figure 24-17. SPI Bursts while SSCTL is Set

In this case, two words are transmitted, one word each SPI burst. The CSPI will continue to transmit SPI bursts until the TXFIFO is empty. When wait states can be inserted between SPI bursts, the $\overline{SS}$ will negate between SPI bursts until the wait states finish.

24.4.2.4 Master Mode with PHA Control

CONREG[PHA] controls how the transmit data shifts out and the receive data shifts in.

When CONREG[PHA] is set, the transmit data will shift out on the rising edge of SCLK, and the receive data is latched on the falling edge of SCLK. The most-significant bit is output on the first rising SPICLK edge.

When CONREG[PHA] is cleared, the transmit data is shifted out on the falling edge of SCLK and the receive data is latched on the rising edge of SCLK. The most-significant bit is output when the CPU loads the transmitted data.

Inverting the SPICLK polarity does not impact the edge-triggered operations because they are internal to the serial peripheral interface master. Figure 24-18 shows a SPI burst using different POL and PHA configurations.

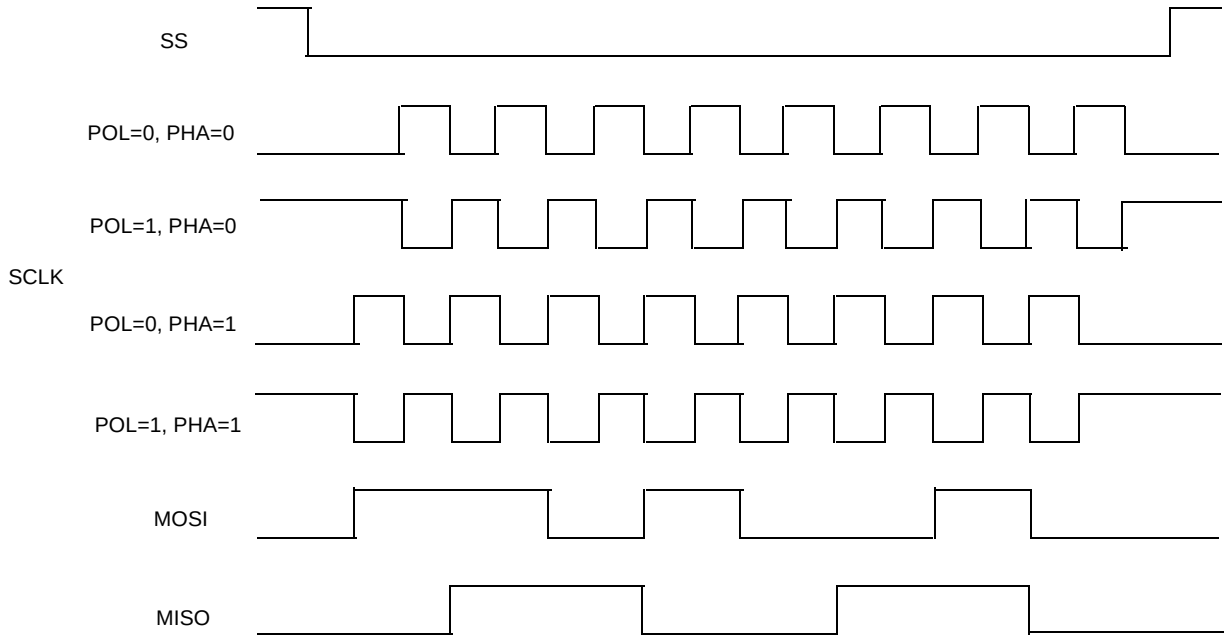


Figure 24-18. SPI Burst with Different POL and PHA Configuration

24.4.3 Slave Mode

When the CSPI module is configured as a slave, the user can configure the CSPI Control register to match the external SPI master's timing. In this configuration, $\overline{SS}$ becomes an input signal, and is used to latch data into or load data out to the internal data Shift registers, as well as to increment the data FIFO.

The $\overline{SS}$, SCLK, and MOSI are inputs and MISO is output. Most of their timing diagrams are the same as in Master mode, because the inputs come from a SPI master device.

However, it is different when $\overline{SS}$ is used to increment data FIFO. When the SSCTL is set while CSPI is in Slave mode, the data FIFO will increment at $\overline{SS}$ rising edge (when SSPOL = 1, should be falling edge).

Figure 24-19 shows a SPI burst in which data FIFO is incremented by $\overline{SS}$ rising edge.

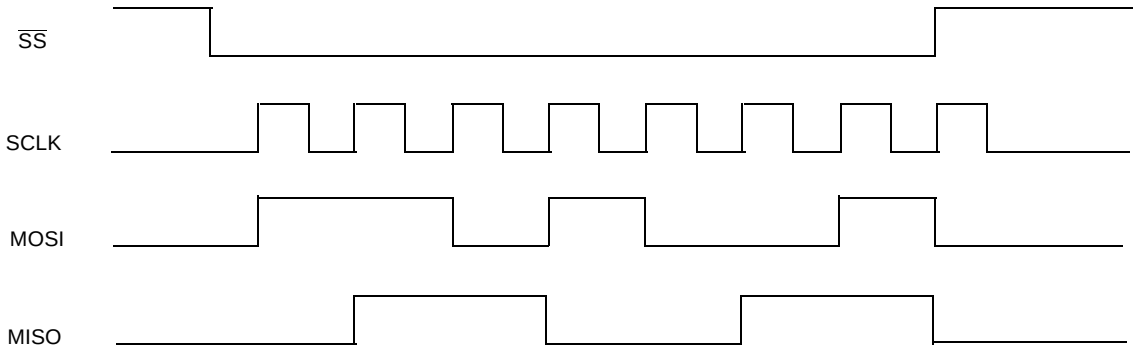


Figure 24-19. Increment Data FIFO by SS Rising Edge

In this case, the data received is not 0xD2 but 0x69. Only the most significant 7 bits are loaded to RXFIFO.

24.4.4 Interrupt Control

Interrupt control is not a specific mode of operation; however, it provides a basic method to utilize the CSPI FIFOs.

You can program the CSPI to enable the TXFIFO empty, TXFIFO half, and TXFIFO full interrupt. You can also use the interrupt service routine to fill the TXFIFO with data to be transferred. Furthermore, you can also enable RXFIFO ready, RXFIFO half, and RXFIFO full to retrieve data from RXFIFO by using the interrupt service routine.

Three other interrupt sources can be used to control/debug the SPI bursts. The transfer-completed interrupt tells the user that there is not data left in TXFIFO and the data in the Shift register is shifted out. The bit counter overflow interrupt tells the user that the CSPI received more than 32 bits in a SPI burst and the remaining bits will be lost. (Only in Slave mode will using SS increment the data FIFO.) The RXFIFO overflow interrupt tells the user that the RXFIFO received more than 8 words and will not accept any other word. Figure 24-20 shows a program sequence of SPI bursts using interrupt.

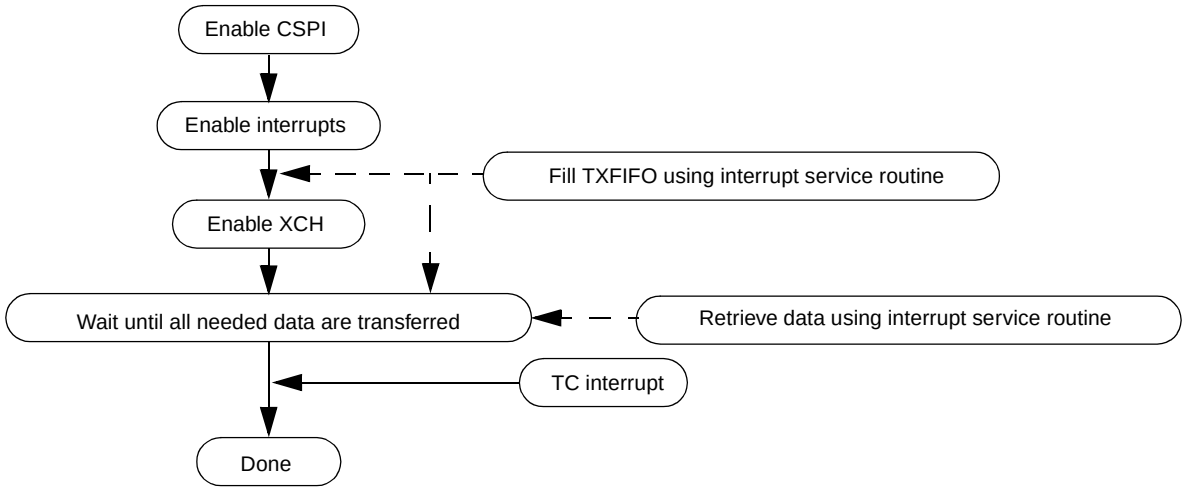


Figure 24-20. Program Sequence of SPI Burst Using Interrupt

24.4.5 DMA Control

DMA control provides another way to utilize the FIFOs in the CSPI module. Peripherals such as the CSPI which support DMA, use DMA request and acknowledge signals. Larger amounts of data can be transferred using DMA control, thereby reducing interrupts and CPU loading. When the appropriate conditions are matched, the module will send out a DMA request, and the DMA will deal with the following cases: TXFIFO empty, TXFIFO half, RXFIFO half, and RXFIFO full.

Figure 24-21 shows a program sequence of SPI bursts using the DMA.

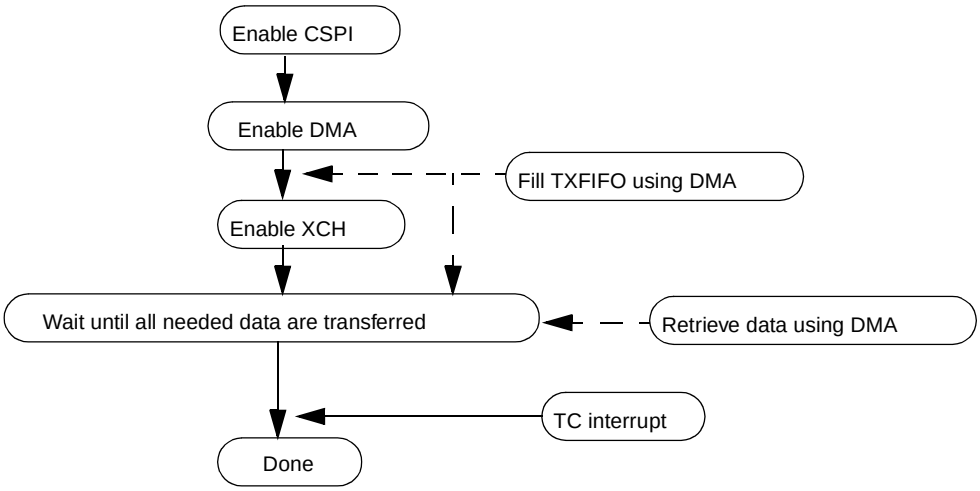


Figure 24-21. Program Sequence of SPI Burst Using DMA

24.5 Initialization/Application Information

This section provides initialization and application information for CSPI.

Figure 24-22 shows two Flowcharts for the master and slave mode of operations supported by the CSPI.

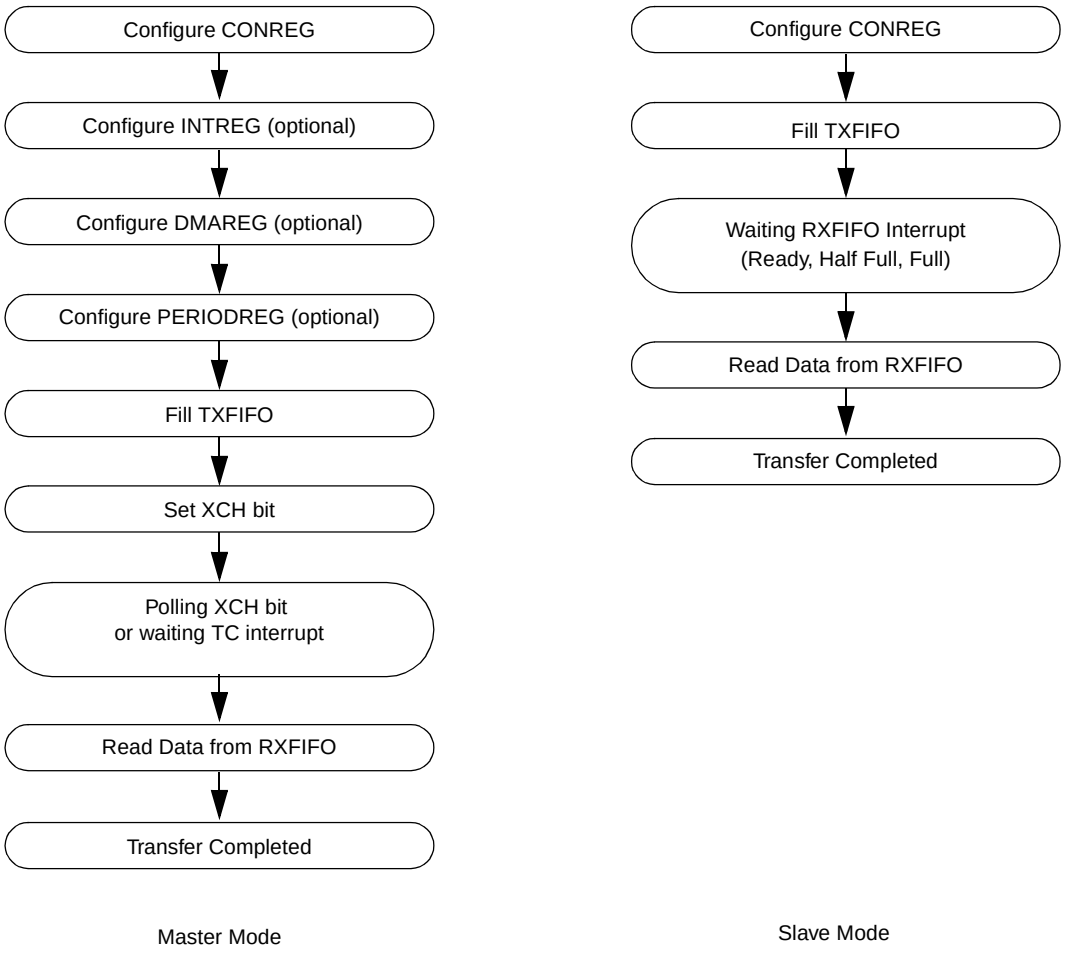


Figure 24-22. Flowchart of CSPI Operation

24.6 CSPI Signal Multiplexing

Table 24-13 provides the primary and alternative pin assignments for the CSPI signals.

Table 24-13. CSPI Signal Primary and Alternative Pin Assignments

Module	Signal Name	Primary Signal Location	Alternate Signal Location	Alternate Mode	Primary Signal Muxed	BGA Pin
		Primary Signal?	BGA Pin			
CSPI1	CSPI1_SS0	Yes	P3	Hardware Mode 1	RI_DTE1	G12
	CSPI1_SS1	Yes	P1	Hardware Mode 1	DCD_DTE1	B13
	CSPI1_SS2	Yes	P6	Hardware Mode 1	DTR_DCE2	F12
	CSPI1_SS3	No	-	Alternative Mode 2	CSPI2_SS1	C6
				Alternative Mode 1	DCD_DCE1	B12
	CSPI1_SCLK	Yes	N2	Alternative Mode 1	DSR_DCE1	A11
	CSPI1_MISO	Yes	P7	Hardware Mode 1	DSR_DTE1	A12
	CSPI1_MOSI	Yes	P2	Hardware Mode 1	DTR_DTE1	C12
	CSPI1_SPI_RDY	Yes	N3	Alternative Mode 1	RI_DCE1	F11
CSPI2	CSPI2_SS0	Yes	B5	-	-	-
	CSPI2_SS1	Yes	C6	-	-	-
	CSPI2_SS2	Yes	A5	-	-	-
	CSPI2_SS3	No	-	Alternative Mode 2	CSPI1_SS1	P1
	CSPI2_SCLK	Yes	C7	-	-	-
	CSPI2_MISO	Yes	A4	-	-	-
	CSPI2_MOSI	Yes	E3	-	-	-
	CSPI2_SPI_RDY	Yes	B6	-	-	-

Table 24-13. CSPI Signal Primary and Alternative Pin Assignments (continued)

Module	Signal Name	Primary Signal Location	Alternate Signal Location	Alternate Mode	Primary Signal Muxed	BGA Pin
		Primary Signal?	BGA Pin			
CSPI3	CSPI3_SS0	No	-	Alternative Mode 1	CSPI2_SS0	B5
	CSPI3_SS1	No	-	Alternative Mode 1	CSPI2_SS1	C6
	CSPI3_SS2	No	-	Alternative Mode 2	CSPI1_SS0	P3
	CSPI3_SS3	No	-	Alternative Mode 2	CSPI1_SS2	P6
	CSPI3_SCLK	Yes	E1	-	-	-
	CSPI3_MISO	Yes	G3	-	-	-
	CSPI3_MOSI	Yes	D2	-	-	-
	CSPI3_SPI_RDY	Yes	G6	-	-	-

24.7 CSPI Pin Configuration

Table 24-14 shows how CSPI signals must be configured for CSPI operation

Table 24-14. CSPI Pin Configuration

Module	Signal	Mode	Pin Configuration Procedure
CSPI1[1]	CSPI1_SS02[2]	Functional	Set sw_mux_ctl_cspi1_ss0 = 0x12
		HW1	Set GPR[2]=1
	CSPI1_SS12[2]	Functional	Set sw_mux_ctl_cspi1_ss1 = 0x12
		HW1	Set GPR[2]=1
	CSPI1_SS22[2]	Functional	Set sw_mux_ctl_cspi1_ss2 = 0x12
		HW1	Set GPR[2]=1
	CSPI1_SS32[2]	Alt 2	Set sw_mux_ctl_cspi2_ss1 = 0x38
		Alt 1	Set sw_mux_ctl_dcd_dce1 = 0x24
	CSPI1_SCLK2[2]	Functional	Set sw_mux_ctl_cspi1_sclk = 0x12
		Alt 1	Set sw_mux_ctl_dsr_dce1 = 0x24
	CSPI1_MISO2[2]	Functional	Set sw_mux_ctl_cspi1_miso = 0x12
		HW1	Set GPR[2]=1
	CSPI1_MOSI2[2]	Functional	Set sw_mux_ctl_cspi1_mosi = 0x12
		HW1	Set GPR[2]=1

Table 24-14. CSPI Pin Configuration (continued)

Module	Signal	Mode	Pin Configuration Procedure
	CSPI1_SPI_RDY2[2]	Functional	Set sw_mux_ctl_cspi1_spi_rdy = 0x12
		Alt 1	Set sw_mux_ctl_ri_dce1 = 0x24
CSPI21[1]	CSPI2_SS0	Functional	Set sw_mux_ctl_cspi2_ss0 = 0x12
	CSPI2_SS1	Functional	Set sw_mux_ctl_cspi2_ss1 = 0x12
	CSPI2_SS2	Functional	Set sw_mux_ctl_cspi2_ss2 = 0x12
	CSPI2_SS3	Alt 2	Set sw_mux_ctl_cspi1_ss1 = 0x38
	CSPI2_SCLK	Functional	Set sw_mux_ctl_cspi2_sclk = 0x12
	CSPI2_MISO	Functional	Set sw_mux_ctl_cspi2_miso = 0x12
	CSPI2_MOSI	Functional	Set sw_mux_ctl_cspi2_mosi = 0x12
	CSPI2_SPI_RDY	Functional	Set sw_mux_ctl_cspi2_spi_rdy = 0x12
CSPI31[1]	CSPI3_SS0	Alt 1	Set sw_mux_ctl_cspi2_SS0 = 0x24
	CSPI3_SS1	Alt 1	Set sw_mux_ctl_cspi2_SS1 = 0x24
	CSPI3_SS2	Alt 2	Set sw_mux_ctl_cspi1_ss0 = 0x38
	CSPI3_SS3	Alt 2	Set sw_mux_ctl_cspi1_ss2 = 0x38
	CSPI3_SCLK	Functional	Set sw_mux_ctl_cspi3_sclk = 0x12
	CSPI3_MISO	Functional	Set sw_mux_ctl_cspi3_miso = 0x12
	CSPI3_MOSI	Functional	Set sw_mux_ctl_cspi3_mosi = 0x12
	CSPI3_SPI_RDY	Functional	Set sw_mux_ctl_spi_rdy = 0x12

1 CSPIx pins must only be configured if CSPIx is being used

2 Only one of the modes should be configured.

24.8 Recommended Signal Pad Configuration

Table 24-15 shows the recommended pad settings for CSPI operation and how to configure these settings.

Table 24-15. Recommended CSPI Signal Pad Configuration

Module	Signal	Mode	Pad Configuration Procedure

Table 24-15. Recommended CSPI Signal Pad Configuration

CSPI1[1]	CSPI1_SS02[2]	Functional	Set sw_pad_ctl_cspi1_ss0 = 00 0000 0000
		HW1	Set sw_pad_ctl_ri_dte1 = 00 0000 0000
	CSPI1_SS12[2]	Functional	Set sw_pad_ctl_cspi1_ss1 = 00 0000 0000
		HW1	Set sw_pad_ctl_dcd_dte1 = 00 0000 0000
	CSPI1_SS22[2]	Functional	Set sw_pad_ctl_cspi1_ss2 = 00 0000 0000
		HW1	Set sw_pad_ctl_dtr_dce2 = 00 0000 0000
	CSPI1_SS32[2]	Alt 2	Set sw_pad_ctl_cspi2_ss1 = 00 0000 0000
		Alt 1	Set sw_pad_ctl_dcd_dce1 = 00 0000 0000
	CSPI1_SCLK2[2]	Functional	Set sw_pad_ctl_cspi1_sclk = 00 0000 0000
		Alt 1	Set sw_pad_ctl_dsr_dce1 = 00 0000 0000
	CSPI1_MISO2[2]	Functional	Set sw_pad_ctl_cspi1_miso = 00 0000 0000
		HW1	Set sw_pad_ctl_dsr_dte1 = 00 0000 0000
	CSPI1_MOSI2[2]	Functional	Set sw_pad_ctl_cspi1_mosi = 00 0000 0000
		HW1	Set sw_pad_ctl_dtr_dte1 = 00 0000 0000
	CSPI1_SPI_RDY2[2]	Functional	Set sw_pad_ctl_cspi1_spi_rdy = 00 0000 0000
		Alt 1	Set sw_pad_ctl_ri_dce1 = 00 0000 0000
CSPI21[1]	CSPI2_SS0	Functional	Set sw_pad_ctl_cspi2_ss0 = 00 0000 0000
	CSPI2_SS1	Functional	Set sw_pad_ctl_cspi2_ss1 = 00 0000 0000
	CSPI2_SS2	Functional	Set sw_pad_ctl_cspi2_ss2 = 00 0000 0000
	CSPI2_SS3	Alt 2	Set sw_pad_ctl_cspi1_ss1 = 00 0000 0000
	CSPI2_SCLK	Functional	Set sw_pad_ctl_cspi2_sclk = 00 0000 0000
	CSPI2_MISO	Functional	Set sw_pad_ctl_cspi2_miso = 00 0000 0000
	CSPI2_MOSI	Functional	Set sw_pad_ctl_cspi2_mosi = 00 0000 0000
	CSPI2_SPI_RDY	Functional	Set sw_pad_ctl_cspi2_spi_rdy = 00 0000 0000
CSPI31[1]	CSPI3_SS0	Alt 1	Set sw_pad_ctl_cspi2_SS0 = 00 0000 0000
	CSPI3_SS1	Alt 1	Set sw_pad_ctl_cspi2_SS1 = 00 0000 0000
	CSPI3_SS2	Alt 2	Set sw_pad_ctl_cspi1_ss0 = 00 0000 0000
	CSPI3_SS3	Alt 2	Set sw_pad_ctl_cspi1_ss2 = 00 0000 0000
	CSPI3_SCLK	Functional	Set sw_pad_ctl_cspi3_sclk = 00 0000 0000
	CSPI3_MISO	Functional	Set sw_pad_ctl_cspi3_miso = 00 0000 0000
	CSPI3_MOSI	Functional	Set sw_pad_ctl_cspi3_mosi = 00 0000 0000
	CSPI3_SPI_RDY	Functional	Set sw_pad_ctl_spi_rdy = 00 0000 0000

1 CSPIx pads must only be configured if CSPIx is being used.

2 Only one of the modes should be configured.

Chapter 25
Fast Infrared Interface (FIR)

Fast Infrared Interface (FIR) signals are multiplexed with UART2 counterpart signals using a GPIO configuration for a complete infrared interface that supports SIR, MIR, and FIR modes, as illustrated in Figure 25-1. In addition, the Serial Infrared (SIR) protocol, which supports data rate which supports data rates up to 1.875 Mbit/s is implemented in each UART module.

The FIR is capable of establishing any of the following links:

- 0.576 Mbit/s
- 1.152 Mbit/s
- 4 Mbit/s half duplex (using an LED and IR detector)
- 4 Mbit/s Fast Infrared (FIR) physical layer protocol (defined by IrDA, version 1.4)

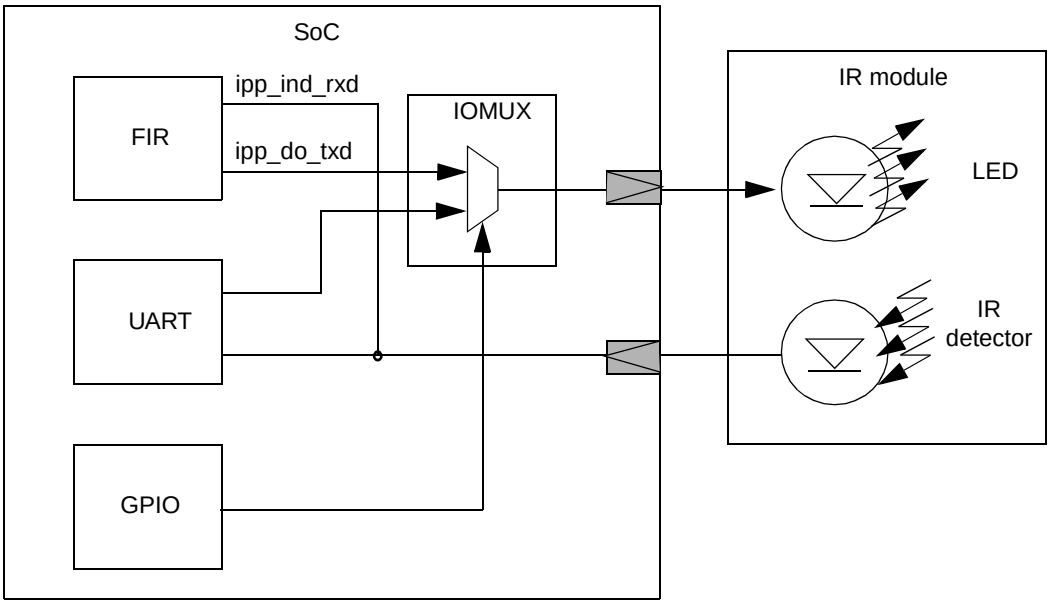


Figure 25-1. FIR Integration Diagram

25.1 Overview

See Figure 25-2 for an illustration of the hardware components that comprise the FIR controller.

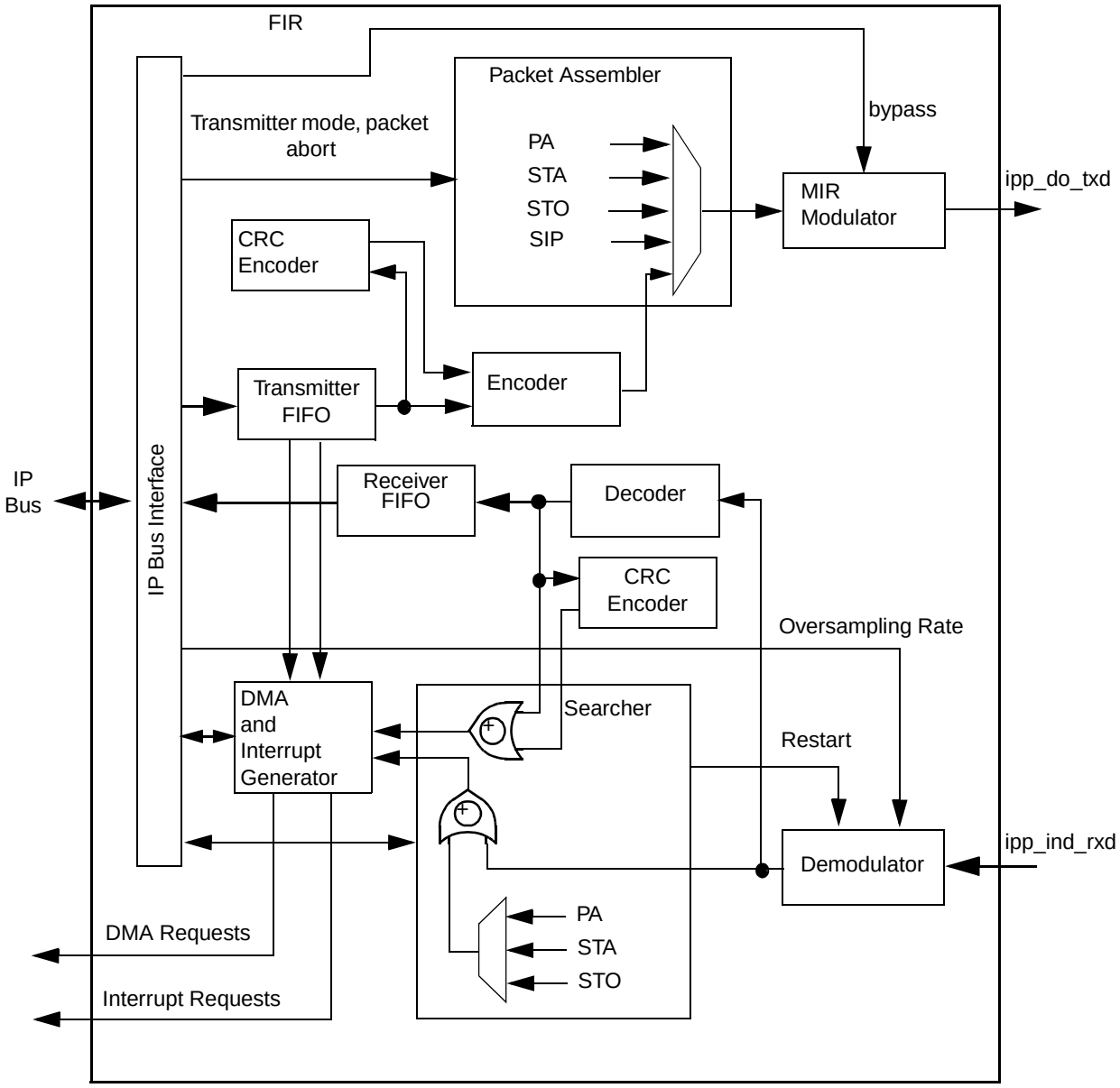


Figure 25-2. FIR Block Diagram

The FIR is divided into these four major functional components:

- Transmitter
- Receiver
- FIFO
- IP interface

Each of the four components that make up the FIR is further made up of individual blocks. The transmitter consists of these blocks:

- Packet Assembler

- CRC Encoder
- MIR Modulator

The receiver consists of these blocks:

- Searcher
- CRC Encoder
- Decoder
- Demodulator

25.1.1 Overview of IrDA Medium Infrared and Fast Infrared Standards

The FIR supports:

- MIR (0.576 Mbit/s, 1.152 Mbit/s)
- FIR (4 Mbit/s) physical layer protocols

This subsection provides a brief overview of the MIR and FIR standards.

25.1.1.1 MIR Packet Structure

The MIR packet format follows the standard High-Level Data Link Control (HDLC) format, except that it requires two beginning flags.

The MIR packet consists of the following:

- Two beginning flags (STA)
- An address
- Control fields
- Data fields
- A frame check sequence (CRC) field
- A minimum of one ending flag (STO)

See Table 25-1 for an illustration of the MIR packet structure, where STA (start flag) and STO (stop flag) are equal, predefined sequences (the left symbol is transmitted/received first).

Table 25-1. MIR Packet Structure

STA	STA	Address	Control and Data	CRC	STO

The fields in Table 25-1 are defined as follows:

- STA, STO

  The MIR links use the same physical layer flag, b’0111,1110, for both STA and STO.
- 8-bit Address Field
- 8-bit Control Field plus up to 2045 bytes in the information field
- CRC field

The MIR links use a 16-bit CRC-CCITT to check received frames for errors. The CRC is computed from the ADDR and Data fields.

The address, control, data, and CRC fields are not transmitted in original form: They are first converted according to the MIR standards described in the next sections.

25.1.1.2 FIR Packet Structure

See Table 25-2 for an illustration of the 4 Mbit/s FIR data packet format. The chip patterns and symbols for PA, STA, CRC field, and STO are listed in Table 25-2, where PA (preamble), STA, and STO are a predefined sequence (left symbol transmitted/received first).

Table 25-2. FIR Packet Structure

PA	STA	Address and Control and Data	CRC32	STO

The fields in Table 25-2 are defined as follows:

- PA—The preamble field is used by the receiver to establish phase lock. The preamble field consists of exactly sixteen repeated transmissions of the following stream of symbols:
  b’1000,0000,1010,1000
- STA—The STA consists of exactly one transmission of the following stream of symbols:
  b’0000,1100,0000,1100,0110,0000,0110,0000
- STO—The STO consists of exactly one transmission of a stream of symbols:
  b’0000,1100,0000,1100,0000,0110,0000,0110
- ACD—The payload data is encoded as described in the 4 PPM encoding in Table 25-2. The encoded symbols reside in the ACD field and can be up to 2048 bytes long.
- CRC32—The CRC field consists of the 4 PPM encoded data, resulting from the IEEE 802 CRC32 algorithm for cyclic redundancy check as applied to the payload data contained in the packet.

25.1.1.3 MIR CRC

The CRC field is 16 bits long and generated according to the CRC-CCITT algorithm. The CRC polynomial is defined as follows:

Eqn. 25-1

$$CRC(x) = x^{16} + x^{12} + x^5 + 1$$

The CRC(x) value is inverted prior to transmission.

25.1.1.4 FIR CRC

The CRC32 field is 32 bits long and generated according to IEEE 802 CRC32 algorithm. The CRC32 polynomial is defined as follows:

Eqn. 25-2

$$CRC32(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

The CRC(x) value is inverted prior to transmission.

25.1.1.5 MIR Modulation

For MIR data rates, the NZR modulation scheme is used. A “0” is represented by a light pulse. The optical pulse duration is nominally 1/4 of a bit duration. The LED is off when a “1” is transmitted.

Table 25-3. MIR Modulation

Data Bit	Data Symbol (Address, Control, and Data)
0	1000
1	0000

NOTE

Tolerance of bit rate according to the standard must not exceed $\pm$0.1%.

25.1.1.6 FIR Modulation

For 4.0 Mbit/s, the modulation scheme is 4 PPM. In the modulation scheme, a pair of bits is taken together and called a data symbol. A data symbol is divided into four chips, only one of which contains an optical pulse. The nominal pulse duration is 125 ns. A “1” is represented by a light pulse.

Table 25-4. 4 PPM Mapping

Data Bit Pair	4 PPM Data Symbol
00	1000
01	0100
10	0010
11	0001

Verify that the tolerance of chip rate according to the standard does not exceed $\pm$0.01%. Pulse width tolerance of an electrical signal driven by an IR detector (photodiode) can be greater than the tolerance of the optical signal defined by IrDA, and can depend on the LED and IR devices used with the FIR.

25.1.2 Features

The FIR includes the following distinctive features:

- 0.576 Mbit/s, MIR protocol
- 1.152 Mbit/s, MIR protocol
- 4 Mbit/s, 4 PPM, FIR protocol
- Device Destination Detection Hardware Support
- Interrupt generation
- DMA capability
- SIP generation for collision avoidance

25.1.3 Modes of Operation

The FIR supports the following modes:

- Hardware packet assembly:
  - FIR mode
  - 0.576 Mbps MIR mode
  - 1.152 Mbps MIR mode
  - Serial Infrared Interaction Pulse (SIP) generation
- Hardware packet search:
  - FIR mode
  - 0.576 Mbps MIR mode
  - 1.152 Mbps MIR mode
- Software packet assembling
- Software packet search

25.2 External Signal Description

See Table 25-5 for the list of signals used to control the FIR.

Table 25-5. Signal Properties

Name	Direction	Port	Function	Reset State	Pull-Up
Signals Connecting To IC IO					
LED and IR Detector Interface					
IPP_DO_TXD	Output	ipp_do_txd	Data transmit signal. Active high.	1	—
IPP_IND_RXD	Input	ipp_ind_rxd	Data receive signal. Active high.	NA	At top level IO ring, if necessary

Table 25-5. Signal Properties (continued)

Name	Direction	Port	Function	Reset State	Pull-Up
Interrupts					
$\overline{\text{IPI\_INT\_FIFO}}$	Output	$\overline{\text{ipi\_int\_fifo}}$	FIFO overrun/underrun interrupt. Active low.	1	NA
$\overline{\text{IPI\_INT\_PAI}}$	Output	$\overline{\text{ipi\_int\_pai}}$	Packet abort interrupt. Active low.	1	NA
$\overline{\text{PI\_INT\_PEI}}$	Output	$\overline{\text{ipi\_int\_pei}}$	Packet end interrupt. Active low.	1	NA
$\overline{\text{IPI\_INT\_TCI}}$	Output	$\overline{\text{ipi\_int\_tci}}$	Transmit complete interrupt. Active low.	1	NA
$\overline{\text{IPI\_INT}}$	Output	$\overline{\text{ipi\_int}}$	“And” of all interrupts described above. Active low.	1	NA

25.2.1 Detailed Signal Descriptions

25.2.1.1 IPP_DO_FIRI_TXD

This is the Data Transmit Signal, which is the output signal from the Packet Assembler module. It must be connected to an LED through the IOMUX, as shown in Figure 25-1. Add a DC-level translator off-chip, if necessary.

25.2.1.2 IPP_IND_FIRI_RXD

This is the Data Receive Signal, which is the signal driven by the IR detector. Add a DC-level translator off-chip, if necessary. This signal is input to the Demodulator module.

25.3 Memory Map and Register Definition

The FIR includes six 32-bit registers, a transmit FIFO, and a receive FIFO. Section 25.3.3, “Register Descriptions” provides detailed descriptions for all of the FIR registers.

25.3.1 FIR Memory Map

Table 25-6 shows the FIR memory map.

Table 25-6. FIR Memory Map

Address	Register	Reset Value	Access	Section/Page
0x53F8_C000 (FIRITCR)	FIR Transmit Control Register	0x0000_0000	R/W	25.3.3.1/25-9
0x53F8_C004 (FIRITCTR)	FIR Transmit Count Register	0x0000_0000	R/W	25.3.3.2/25-11
0x53F8_C008 (FIRIRCR)	FIR Receive Control Register	0x0000_0000	R/W	25.3.3.3/25-12
0x53F8_C00C (FIRITSR)	FIR Transmit Status Register	0x0000_0000	R/Write one to clear	25.3.3.4/25-14

Table 25-6. FIR Memory Map (continued)

Address	Register	Reset Value	Access	Section/Page
0x53F8_C010 (FIRIRSR)	FIR Receive Status Register	0x0000_0000	R/Write one to clear	25.3.3.5/25-15
0x53F8_C014	Transmitter FIFO	—	W	25.4.2/25-18
0x53F8_C018	Receiver FIFO	—	R	25.4.4/25-20
0x53F8_C01C (FIRICR)	FIR Control Register	0x0000_0000	R/W	25.3.3.6/25-16

25.3.2 Register Summary

Figure 25-3 shows the key to the register fields, and Table 25-7 shows the register figure conventions.

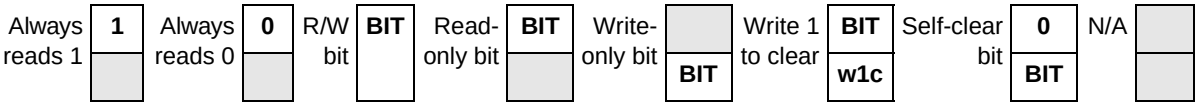


Figure 25-3. Key to Register Fields

Table 25-7. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 25-8 shows the FIR register summary.

Table 25-8. FIR Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x53F8_C000 (FIRITCR)	R	0	0	0	0	0	0	0	HAG	TPA							
	W																
	R	0	SRF			TDT			TCIE	TPEIE	TFUIE	PCF	PC	SIP	TPP	TM		TE
	W																
0x53F8_C004 (FIRITCTR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	TPL										
	W																
0x53F8_C008 (FIRIRCR)	R	0	0	0	0	0	0	RAM		RA							
	W																
	R	0	0	0	0	RPEDE	RDT			RPA	RPEIE	PAIE	RFOIE	RPP	RM		RE
	W																
0x53F8_C00C (FIRITSR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	TFP								0	0	0	0	TC	SIPE	TPE	TFU
	W													w1c	w1c	w1c	w1c
0x53F8_C010 (FIRIRSR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	RFP								0	0	PAS	RPE	RFO	BAM	CRCE	DDE
	W												w1c	w1c	w1c	w1c	w1c
0x53F8_C014 Transmitter FIFO	R																
	W	Transmitter FIFO															
	R																
	W	Transmitter FIFO															
0x53F8_C018 Receiver FIFO	R	Receiver FIFO															
	W																
	R	Receiver FIFO															
	W																
0x53F8_C01C (FIRICR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	BL							0	OSF			
	W																

25.3.3 Register Descriptions

This section contains the detailed register descriptions for the FIR registers.

25.3.3.1 FIR Transmitter Control Register (FIRITCR)

The Transmitter Control Register is a 32-bit, Read/Write register that controls transmitter operation. Figure 25-4 shows the register; Table 25-9 provides its field descriptions.

0x53F8_C000 (FIRITCR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	HAG	TPA							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	SRF		TDT			TCIE	TPEIE	TFUIE	PCF	PC		TPP	TM		TE
W												SIP				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-4. FIR Transmitter Control Register

Table 25-9. FIR Transmitter Control Register Field Descriptions

Field	Description
31–25	Reserved
24 HAG	Hardware Address Generator. When this bit is set, the content of the TPA bits is transmitted as a packet address. When the bit is cleared, the packet address is read from TX FIFO. 0 Read packet address from TX FIFO 1 Use TPA bits as packet address
23–16 TPA	Transmit Packet Address. This field contains the 8-bit Transmit Packet Address. If the HAG bit is cleared, the TPA bits have no effect.
15	Reserved
14–13 SRF	Start Field Repeat Factor. This field contains the number of PA or STA fields transmitted in the beginning of the packet for FIR and MIR mode, respectively. In FIR mode, only the 00 value should be used. 00 16 PA or 2 STA fields is transmitted. 01 32 PA or 4 STA fields is transmitted. 10 64 PA or 8 STA fields is transmitted. 11 128 PA or 16 STA fields is transmitted.
12–10 TDT	This field controls the TX FIFO depth that triggers a DMA request. Refer to Section 25.4.2, “Transmitter FIFO,” for details. **Note:** To avoid TX FIFO overflow, verify that the sum of TDT level and burst length (set by BL bits in the FIRICR register) does not exceed TX FIFO size, which is 128 bytes. 0 DMA request generated when TX FIFO is empty 1 DMA request generated when TX FIFO contain 16 bytes of data 2 DMA request generated when TX FIFO contain 32 bytes of data 3 DMA request generated when TX FIFO contain 48 bytes of data —— 7 DMA request generated when TX FIFO contains 112 bytes of data
9 TCIE	Transmit Complete Interrupt Enable. This bit enables the TC status bits of the FIRITSR register to generate a Transmit Complete Interrupt. 0 Transmit Complete Interrupt is not triggered by TC bit. 1 Transmit Complete Interrupt is triggered by TC bit.

Table 25-9. FIR Transmitter Control Register Field Descriptions (continued)

Field	Description
8 TPEIE	Transmitter Packet End Interrupt Enable. This bit enables the TPE and SIPE status bits of the FIRITSR register to generate a Packet End Interrupt. 0 PEI interrupt is not triggered by TPE or SIPE bits. 1 PEI interrupt is triggered by TPE or SIPE bits.
7 TFUIE	Transmitter FIFO Underrun Interrupt Enable. This bit enables the TFU status bit of the FIRITSR register to generate a TFU interrupt. 0 TFU interrupt disabled 1 TFU interrupt enabled
6 PCF	Packet Complete by FIFO. This bit determines how a packet is completed if a TX FIFO underrun event occurs. Do not write software intentionally to cause underrun events. However, if due to erroneous conditions, the value of this bit selects between two recovery modes. Set the PCF based on system and upper layer IrDA protocol requirements. 0 Send CRC and STO fields 1 Send packet abort symbol
5 PC	Packet Complete. This bit determines how a packet is completed when the TE bit is cleared or when the SIP bit is set in the middle of the transfer. Do not write software intentionally to clear the TE bit, or to set the SIP bit in the middle of the transfer. However, if it is done due to erroneous conditions, the value of this PC bit selects between two recovery modes. Set PC based on system and upper layer IrDA protocols requirements. 0 Send CRC and STO fields 1 Send packet abort symbol
4 SIP	Transmit Enable of SIP. Writing “1” to this bit produces a “Serial Infrared Interaction Pulse” transmission. Writing a “0” to this bit is ignored. This bit is always read as “0”. If this bit is set while in the middle of the transfer, the packet will be completed according to the setting of the PC bit.
3 TPP	Transmitter Pulse Polarity bit. 0 Transmitted pulse is not inverted. 1 Transmitted pulse is inverted.
2–1 TM	Transmitter Mode. This bits controls the transmission mode. 00 4 Mbps FIR Mode 01 0.576 Mbps MIR Mode 10 1.152 Mbps MIR Mode 11 Software Packet Assembling
0 TE	Transmitter Enable. This bit controls the FIR transmitter. If this bit is cleared in the middle of the transfer, the packet will be completed according to the setting of the PC bit. When the packet is completed, the transmitter clocks are then gated OFF. 0 Transmitter Disabled 1 Transmitter Enabled

Software packet assembling mode is not recommended, and should be used for experimental purposes. See25.4.1.4/25-18

25.3.3.2 FIR Transmitter Count Register (FIRITCTR)

The Transmitter Count Register is 32-bit, Read/Write register that controls the packet size. Figure 25-5 shows the register; Table 25-10 provides its field descriptions.

0x53F8_C004 (FIRITCTR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R						TPL										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-5. FIR Transmitter Count Register

Table 25-10. FIR Transmitter Count Register Field Descriptions

Field	Description
31–11	Reserved
10–0 TPL	Transmit Packet Length bits. The length of the address, control, and data fields (see Table 25-1 and Table 25-2). 0 Send 1 byte 1 Send 2 bytes —— 2047 Send 2048 bytes

25.3.3.3 FIR Receiver Control Register (FIRIRCR)

The Receiver Control Register is 32-bit, Read/Write register that controls receiver operation. Figure 25-6 shows the register; Table 25-11 provides its field descriptions.

0x53F8_C008 (FIRIRCR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	RAM		RA							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	RPEDE	RDT			RPA	RPEIE	PAIE	RFOIE	RPP	RM		RE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-6. FIR Receiver Control Register

Table 25-11. FIR Receiver Control Register Field Descriptions

Field	Description
31–26	Reserved
25–24 RAM	Address Match bit. The value of this bit can be changed when the RE bit is cleared. 00 Does not match address 01 Match packet address to RA bits 10 Match packet address to Broadcast address 11 Match packet address to RA bits and to broadcast address
23–16 RA	Receiver Address. Determines Receiver Packet Address. If the RAM bit value is 00 or 10, the RA bit has no effect. The value of this bit can be changed when the RE bit is cleared.
15–12	Reserved
11 RPEDE	Receiver Packet End DMA Request Enable bit; Enable DMA request generation at the end of the packet. If this bit is set, verify that the BL value in the FIRICR register is not greater than the sum of the address, control, data, and CRC field length. 0 DMA request is not affected by the end of packet. 1 DMA request is generated at the end of packet.
10–8 RDT	Receiver DMA Request Trigger level. Sets minimum RX FIFO depth, which triggers a DMA request. For more details, refer to Section 25.4.4, “Receiver FIFO.” **Note:** To avoid RX FIFO underflow, set the RDT level to less than or equal value of the burst length (set by the BL bit in the FIRICR register). 0 Reserved 1 DMA request generated when RX FIFO contains 16 bytes of data 2 DMA request generated when RX FIFO contains 32 bytes of data 3 DMA request generated when RX FIFO contains 48 bytes of data —— 7 DMA request generated when RX FIFO contains 112 bytes of data
7 RPA	Receiver Packet Abort bit. Determines behavior of the RX FIFO upon detection of an illegal symbol. When an illegal symbol is detected, the DDE or CRCE bit in the FIRIRSR register is set. If the RPA bit is set, the RX FIFO pointers are cleared and the receiver starts to search for the PA or STA fields for FIR and MIR mode, respectively. If RPA is cleared, the receiver continues to write to the RX FIFO. 0 Does not clear the RX FIFO upon detection of an illegal symbol 1 Clears the RX FIFO upon detection of illegal symbol
6 RPEIE	Receiver Packet End Interrupt Enable bit. Enables the RPE status bit in the FIRIRSR register to assert the Packet End Interrupt. 0 PEI interrupt is not triggered by RPE bit. 1 PEI interrupt is triggered by RPE bit.
5 PAIE	Packet Abort Interrupt Enable bit. Enables the DDE, CRCE status bits in the FIRIRSR register to cause a PAI interrupt. 0 PAI interrupt is disabled. 1 PAI interrupt is enabled.
4 RFOIE	Receiver FIFO Overrun Interrupt Enable bit. Enables the RFO status bit in the FIRIRSR register to cause a RFOI interrupt. 0 RFOI interrupt is disabled. 1 RFOI interrupt is enabled.
3 RPP	Receiver Pulse Polarity bit. 0 Receiver signal is not inverted. 1 Received signal is inverted.

Table 25-11. FIR Receiver Control Register Field Descriptions (continued)

Field	Description
2–1 RM	Receiver Mode bits. 00 FIR Mode 01 0.576 Mbps MIR Mode 10 1.152 Mbps MIR Mode 11 Software Packet Assembling
0 RE	Receiver enable bit. When this bit is cleared, the receiver clocks are gated OFF. 0 Receiver is disabled. 1 Receiver is enabled.

Software packet assembling mode is not recommended, and should be used for experimental purposes. See25.4.1.4/25-18

25.3.3.4 FIR Transmit Status Register (FIRITSR)

The Transmit Status Register is 32-bit register that reflects the status of the transmitter and the TX FIFO. The FIRITSR contains read-only and write-one-to-clear bits. Figure 25-7 shows the register; Table 25-12 provides its field descriptions.

0x53F8_C00C (FIRITSR) Access: User read/write one to clear

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TFP								0	0	0	0	TC	SIPE	TPE	TFU
W													w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-7. FIR Transmit Status Register

Table 25-12. FIR Transmit Status Register Descriptions

Field	Description
31–16	Reserved
15–8 TFP	Transmitter FIFO Pointer bits. The value of the TFP bits represent the number of available bytes in TX FIFO (read-only bits).
7–4	Reserved
3 TC	Transmit Complete bit. Indicates that the transmission is completed (including the CRC and STO fields), and the transmitter FIFO is empty. This bit is cleared by writing a “1”. 0 Transmitting is not completed. 1 Transmitting is completed.

Table 25-12. FIR Transmit Status Register Descriptions (continued)

Field	Description
2 SIPE	SIP End bit. Indicates that “Serial Infrared Interaction Pulse” has been transmitted. This bit is cleared by writing a “1”. 0 SIP transmission is not completed. 1 SIP had been transmitted.
1 TPE	Transmitter Packet End bit. Indicates that the TPS bytes of data (address, control, and data fields) have been transmitted. This bit is cleared by writing a “1”. 0 Transmissions of address, control, and data fields are not completed. 1 Address, control, and data fields had been transmitted.
0 TXU	Transmitter FIFO Underrun bit. Indicates the occurrence of a TX FIFO underrun. A TX FIFO underrun occurred if the transmitter attempted to read from the TX FIFO when the TX FIFO is empty. This bit is cleared by writing a “1”. 0 No transmitter FIFO underrun 1 Transmitter FIFO is underrun

25.3.3.5 FIR Receive Status Register (FIRIRSR)

The Receive Status Register is 32-bit register that reflects the status of the receiver and the FIFO. It contains read-only and write-one-to-clear bits. Figure 25-8 shows the register; Table 25-13 provides its field descriptions.

0x53F8_C010 (FIRIRSR) Access: User read/write one to clear

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RFP								0	0	PAS	RPE	RFO	BAM	CRCE	DDE
W												w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-8. FIR Receive Status Register

Table 25-13. FIR Receive Status Register Field Descriptions

Field	Description
31–16	Reserved
15–8 RFP	Receiver FIFO Pointer bits. The value of the RFP bits represent the number of available bytes in the RX FIFO (read-only bits).
7–6	Reserved

Table 25-13. FIR Receive Status Register Field Descriptions (continued)

Field	Description
5 PAS	Preamble Search bit. Indicates that the receiver is searching for PA+STA, or STA fields for FIR and MIR modes, respectively (read-only bit). 0 Receiver does not search for the PA or STA field. 1 Receiver searches for the PA or STA field.
4 RPE	Receiver Packet End bit. Indicates that the STO field or packet abort symbol (b'0000,000 and b'0000,0000 for MIR and FIR, respectively) is detected. This bit is cleared by writing a "1". 0 STO was not detected. 1 STO field is detected.
3 RFO	Receiver FIFO Overrun bit. Indicates that the receiver attempted to write to the RX FIFO when the RX FIFO is full. This bit is cleared by writing a "1". 0 Receiver FIFO is not overrun. 1 Receiver FIFO is overrun.
2 BAM	Broadcast Address Match bit. Indicates that the address field of the packet matches the Broadcast Address 0xFF. This bit is cleared by writing a "1". 0 No Broadcast 1 Broadcast packet
1 CRCE	CRC Error bit. Indicates that the CRC check failed. This bit is cleared by writing a "1". 0 No CRC check failure 1 CRC check failure
0 DDE	Address, Control, or Data Field Error bit. Indicates that an illegal symbol has been detected in the address, control, data, or CRC32/CRC fields. This bit is cleared by writing a "1". 0 No illegal symbols in address, control, data, or CRC field 1 Illegal symbol in address, control, data, or CRC field

25.3.3.6 FIR Control Register (FIRICR)

The Control Register is a 32-bit, Read/Write register that is shared by receiver and transmitter. Figure 25-9 shows the register; Table 25-14 provides its field descriptions.

0x53F8_C01C (FIRICR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	BL							0	OSF			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-9. FIR Control Register

Table 25-14. FIR Control Register Field Descriptions

Field	Description
31–12	Reserved
11–5 BL	Burst Length. Used to prevent an overrun of the TX FIFO and an underrun of the RX FIFO. The DMA request of the transmitter deasserts when the number of vacant bytes in the TX FIFO is less than the burst length settings specified by the BL bits. The DMA request of the receiver deasserts when the number of available bytes in the RX FIFO is less than the burst length settings specified by the BL bits. 1 1 byte 2 2 bytes 3 3 bytes —— 127 127 bytes 0 128 bytes
4	Reserved
3–0 OSF	Over Sampling Factor. This field controls the oversampling factor of the “chip” on the IPP_IND_RXD signal if the RE bit in the FIRIRCR register is set, or the prescaling factor of the IPG_CLK_FIRI_BAUD signal if the TE bit in the FIRITCR register is set. **Note:** The “chip” rate is 4 and 2 times greater than the bit rate for MIR and FIR, respectively. 0 Does not oversample 1 Oversamples by 2 2 Oversamples by 3 —— 15 Oversamples by 16

To enable proper receive operations, the period of the divided clock must be four or more times shorter than the pulse width of the IPP_IND_RXD signal, and one or more times shorter than the gap between two consecutive pulses. In other words, the OSF should be higher than 4.

NOTE

The pulse width of the IPP_IND_RXD signal is dependent upon the characteristics of the LED and the Photo Diode used with the FIR, and is different from the pulse width defined by IrDA for the optical signal. The value of the OSF bits must be adjusted accordingly. A greater OSF value leads to a better SNR on the expense of an increase in the power consumption.

25.4 Functional Description

This section details the FIR functional description.

25.4.1 Transmitter Overview

The transmitter has three modes of operation:

- MIR
- FIR
- Software Packet Assembly

25.4.1.1 MIR Mode

In MIR mode, the following sequence occurs:

1. The Packet Assembler block sends the STA sequences (two or more times) to the MIR Modulator block.
2. The Encoder block accesses the TX FIFO, aligns the data, and sends the ACD (Address, Control, and Data) fields to the Packet Assembler block.
3. The CRC field calculated by CRC block follows the ACD field; after which, the STO bits are sent to the Modulator.

25.4.1.2 FIR Mode

In FIR mode, the following sequence occurs:

1. The Packet Assembler block sends the predefined PA (16 or more times) and STA sequences to the LED.
2. The Encoder block accesses the TX FIFO, encodes the data (4 PPM), and sends the ACD (Address, Control and Data) to the Packet Assembler block.
3. The CRC32 field calculated by CRC block follows the ACD field; after which, the STO bits are sent to the Modulator.

NOTE

The MIR Modulator block is not operable in FIR mode.

If the DMA request is not served during MIR or FIR modes, and there is no valid data to transmit, the Assembler Module terminates the packet using the packet-abort symbol (with CRC/CRC32 and STO fields [see the PCF bit description]), and a TFUI interrupt is generated. The packet can also be aborted by the packet length counter or by the software (see PC bit description).

25.4.1.3 Serial Infrared Interaction Pulse

The transmitter must send the “Serial Infrared Interaction Pulse” (SIP) at 500 ms to guarantee that low speed IR devices will not interfere with its pulses. The pulse must be 8.7 µs wide; and positive phase must be 1.6 µs wide.

25.4.1.4 Software Packet Assembly Mode

In Software Packet Assembly mode, the entire packet is read from the TX FIFO, including the PA, STA, and STO bits. The MIR modulator and Encoder blocks are disabled. This mode is used for debug purposes and can be used for testing compatibility with future IrDA protocols, if appropriate.

25.4.2 Transmitter FIFO

The transmitter FIFO is a 128-byte FIFO used for holding transmit data. The TX FIFO gets filled by the accesses to the Transmitter FIFO address (by core or DMA) and is emptied by the transmitter. When the DMA is used, a pre-programmed FIFO trigger level controls the triggering of the DMA request. When the TX FIFO is emptied below the trigger-level, a DMA request signal is asserted. The DMA request gets

deasserted when the number of vacant bytes in the TX FIFO is less than the burst length settings specified by the BL bits in the FIRICR register. If the DMA request was not serviced and, as a result, there are no more data in the TX FIFO, the TFU bit in the FIRITSR register sets, and the $\overline{\text{IPI\_INT\_FIFO}}$ interrupt signal asserts depending on TFOIE bit in the FIRITCR register. During the transmit operation, the TFP bit value in the FIRITSR register reflects the amount of available bytes in the TX FIFO.

25.4.3 Receiver Overview

The receiver has three modes of operation:

- MIR
- FIR
- Software Packet Disassembly Modes

In MIR and FIR modes, each incoming pulse is oversampled by the programmable factor in the Demodulator block. Next, the Demodulator block detects the edges of the pulses and synchronizes to the phase of the transmitted pulse. Phase correction is performed until the end of the packet because the bit rate tolerance accumulates to very large values for long packets.

25.4.3.1 MIR Mode

In MIR mode, the Searcher block searches for the STA field. If the sequence is matched, the data stream following the STA field is sent to the Decoder block. The Decoder block searches for five consecutive “1”s and skips next the “zero” bit inserted by the transceiver for phase synchronization. The data is then written to the RX FIFO. In parallel, the Searcher Module searches for the STO field. When the STO field is detected, a PEI interrupt is generated together with a DMA request, and the corresponding flag is set in the status register. The Searcher module continuously searches for illegal symbols (seven or more consecutive “1”s). If an illegal symbol is detected, the PAI interrupt is generated and the corresponding flag is set in the status register.

25.4.3.2 FIR Mode

In FIR mode, the Searcher Module searches for the PA field and then for the STA field. If found, the “chip” stream is converted to a data stream by the Decoder block and then it is written to the RX FIFO. In a parallel operation, the Searcher Module searches for the STO field. When the STO field is detected, a PEI interrupt can be generated together with a DMA request. While receiving, the Searcher module continuously searches the PA, STA, ACD, CRC32, and STO fields for illegal symbols. In addition, the Searcher module looks for a packet abort symbol. On detection of an illegal symbol, a PAI interrupt is generated and the corresponding flag is set in the status register.

In MIR and FIR modes, the “Address” bits can be compared to a predefined value or/and to the broadcast value 0xFF (see the description of RAM bits). If the CRC field value does not match an expected value calculated by the CRC Encoder block, a PAI interrupt is generated. The CRC field sends the RX FIFO, irrespective of a comparison result.

25.4.3.3 Software Packet Disassembly Mode

In Software Packet Disassembly mode, the entire packet is sent to the RX FIFO, even if illegal symbols have been detected.

25.4.4 Receiver FIFO

The receiver FIFO is a 128-byte FIFO, used for holding received data. RX FIFO gets filled by incoming data and emptied by excess RX FIFO (by the core or DMA). When DMA is used, a pre-programmed FIFO trigger level controls the triggering of the DMA request. When RX FIFO is filled beyond the trigger level, a DMA request signal is asserted. The DMA request gets deasserted when the number of available bytes in RX FIFO is less than the burst length settings specified by the BL bits in the FIRICR register. If the DMA request was not serviced and, as a result, there are no vacant bytes in the RX FIFO for receive data, the RFO bit in the FIRIRSR register is set and the $\overline{\text{IPI\_INT\_FIFO}}$ interrupt signal is asserted depending on RFOIE bit in the FIRIRCR register. During a receive operation, the RFP bits' value in the FIRIRSR register reflects the amount of available bytes in the RX FIFO.

25.5 Initialization/Application Information

25.5.1 FIRI Clock Path

The IPG_CLK_FIRI_BAUD signal is the clock coming from the Clock Controller Module (CCM) and called FIRI_CLK. This clock has three possible sources: mcu_main_clk, serial_main_clk, or usb_main_clk. The field FIRS of the CCM's control register is used to select the clock source.

Once the source is selected, there are two dividers in the CCM: the FIRI pre-divider and FIRI divider. Depending on the source frequency and the desired baud rate, these dividers can be used to globally pre-divide the source clock and generate the IPG_CLK_FIRI_BAUD.

Finally, the FIRICR.OSF bits are used to precisely adjust the final baud rate inside the FIRI module. As suggested before, it is not advised to use a value of OSF lower than 4 for transmit and receive operation.

25.5.2 Examples of FIR Programming

25.5.2.1 Transmitter Programming Scenario

The purpose of this scenario is to transmit 1024 bytes in FIR mode using an infrared link. Verify that the packet length is 512 bytes.

- Configure Clock Controller. For this example, the frequency of the IPG_CLK_FIRI_BAUD clock is 48 MHz.
- Configure the DMA module for DMA-service FIR transactions. A DMA transaction will be requested by the negative edge of $\overline{\text{DMA\_REQ}}$[17] triggered by transmitter FIFO. The DMA access size is 4 bytes. For this example, the burst length is 4 accesses.
- Configure the FIR to work with a 32-byte burst (FIRICR.BL bits).

NOTE

Observe the burst length of the DMA, and the burst length selected by FIRICR. The BL bits are not necessarily equal.

- Set the FIR Oversampling Factor (FIRICR.OSF bits) to 6 to achieve a 4-Mbit rate with the IPG_CLK_FIRI_BAUD clock.
- Configure the FIR to transmit 512-byte long packets (FIRITCTR.TPL bits).
- Set the DMA trigger level to 16 (FIRITCR.TDT bits). A DMA request will be generated when there are only 16 bytes remaining in the TX FIFO. Select FIR mode (FIRITCR.TM bits).
- Finally, enable the transmitter (FIRITCT.TE bit). The FIR immediately asserts a DMA request because the TX FIFO is empty. The DMA controller delivers a first burst to the FIR. The FIR starts the transmission.

25.5.2.2 Receiver Programming Scenario

Fort the receiver programming scenario, the data is transmitted in FIR mode. An interrupt should be generated at the end of each packet.

- Configure Clock Controller. For this example, the frequency of the IPG_CLK_FIRI_BAUD clock is 48 MHz.
- Configure the DMA module for the FIR to DMA transactions. The DMA transaction will be requested by the negative edge of $\overline{\text{DMA\_REQ}}$[16] triggered by receiver FIFO. DMA access size is 4 bytes. For this example, burst length is 4 accesses.
- Configure the FIR to work with 32-byte burst (FIRICR.BL bits). Set the FIR Oversampling Factor (FIRICR.OSF bits) to 6 to achieve a 4-Mbit rate with the IPG_CLK_FIRI_BAUD clock.
- Set the DMA trigger level to 16 (FIRIRCR.RDT bits). A DMA request is generated when there are 16 bytes in RX FIFO. Select FIR mode (FIRIRCR.RM bits). Enable Packet End Interrupt.
- Finally, enable the receiver (FIRIRCR.RE bit). The receiver searches for the PA and STA fields. When a PEI interrupt is generated, verify that the software clears the FIRIRSR.RPE bit.

Chapter 26
Inter-Integrated Circuit (I^2C)

The Inter-Integrated Circuit (I^2C) module provides the functionality of a standard I^2C slave and master. The I^2C module is designed to be compatible with the standard Philips I^2C bus protocol. See the I^2C block diagram in Figure 26-1.

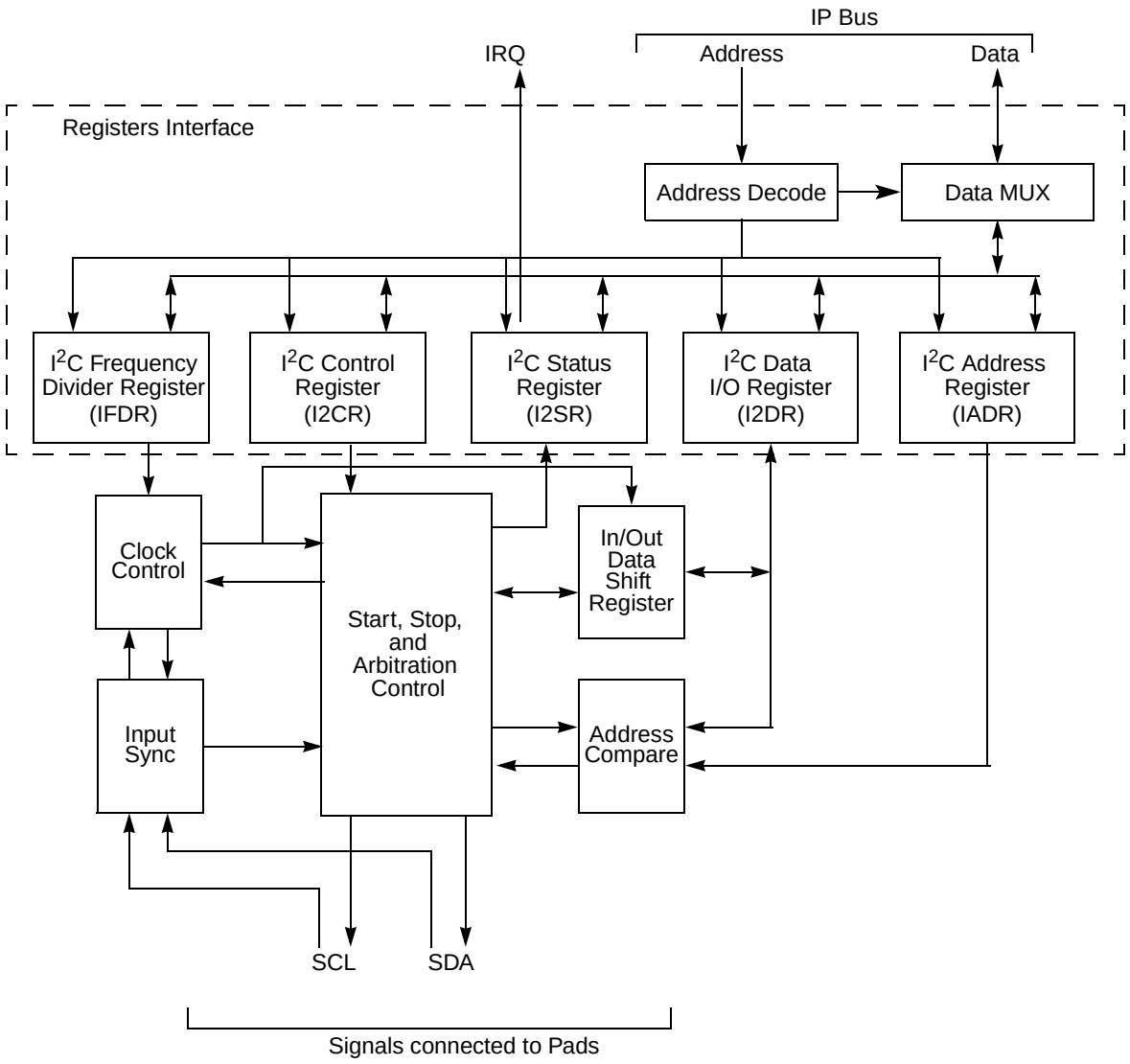


Figure 26-1. I^2C Block Diagram

26.1 Overview

The I2C is a two-wire, bidirectional serial bus that provides a simple, efficient method of data exchange, minimizing the interconnection between devices. This bus is suitable for applications requiring occasional communications over a short distance between many devices. The flexible I2C allows additional devices to be connected to the bus for expansion and system development. See the connection diagram in Figure 26-2.

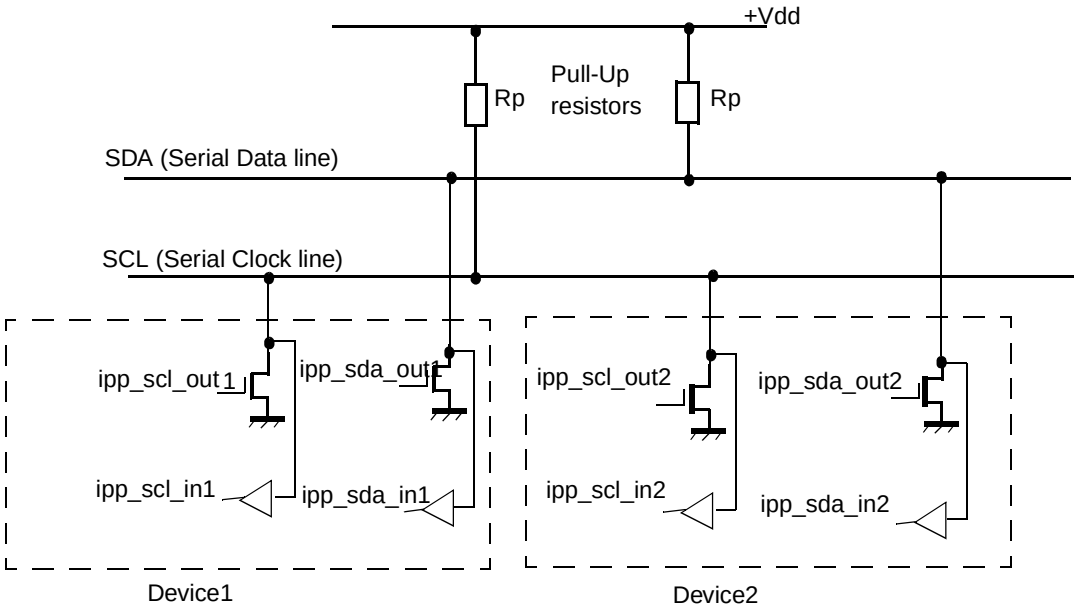


Figure 26-2. Connection of Devices to I2C Bus

The I2C operates up to 400 kbps, but it depends on the pad loading and timing. For pad requirement details, refer to Philips I2C Bus Specification, Version 2.1. The I2C system is a true multiple-master bus including arbitration and collision detection that prevents data corruption if multiple devices attempt to control the bus simultaneously. This feature supports complex applications with multiprocessor control and can be used for rapid testing and alignment of end products through external connections to an assembly-line computer.

26.1.1 Features

The I2C module has the following key features:

- Compatibility with I2C bus standard
- Multiple-master operation
- Software-programmable for one of 64 different serial clock frequencies
- Software-selectable acknowledge bit
- Interrupt-driven, byte-by-byte data transfer
- Arbitration-lost interrupt with automatic mode switching from master to slave
- Calling address identification interrupt

- Start and stop signal generation/detection
- Repeated START signal generation
- Acknowledge bit generation/detection
- Bus-busy detection

26.2 External Signal Description

Two pins are required for I^2C:

- I2C_SCL Bidirectional Clock Pin
- I2C_SDA Bidirectional Data Pin

For I^2C compliance, all devices connected to the SCL and SDA signals must have open-drain or open-collector outputs. The logic AND function is exercised on both lines with external pull-up resistors.

The module port signals going to the pad are tabulated in Table 26-2:

Table 26-2. Signal Properties

Name	Port	Function	Reset State	Pull-Up
IPP_SCL_IN	—	Serial Clock input	1	—
IPP_SCL_OUT	—	Serial Clock output	1	Active
IPP_SCL_OUT_EN	—	Serial Clock output enable	0	—
IPP_SDA_IN	—	Serial Data input	1	—
IPP_SDA_OUT	—	Serial Data output	1	Active
IPP_SDA_OUT_EN	—	Serial Data output enable	0	—

26.2.1 Detailed External Signal Descriptions

The following three signals are connected to the bidirectional driver for the SCL I/O Pad (through the IOMUX module):

- IPP_SCL_IN—Serial Input Clock
- IPP_SCL_OUT—Serial Output Clock
- IPP_SCL_OUT_EN—Serial Output Clock Enable

The pad needs to have open-drain connectivity. IPP_SCL_IN will be the input signal from the pad. IPP_SCL_OUT will be the output to the pad from the module. IPP_SCL_OUT_EN will act as the output enable.

The following three signals are connected to the bidirectional driver for the SDA I/O Pad (through the IOMUX module):

- IPP_SDA_IN—Serial Input Data
- IPP_SDA_OUT—Serial Output Data
- IPP_SDA_OUT _EN—Serial Output Data Enable

The pad should have open-drain connectivity. IPP_SDA_IN will be the input signal from the pad. IPP_SCL_OUT will be the output to the pad from module. IPP_SDA_OUT_EN will act as the output enable.

26.3 Memory Map and Register Definition

The I2C module contains five 16-bit registers. Section 26.3.3, "Register Descriptions" provides the detailed descriptions for all of the I2C registers.

26.3.1 I2C Memory Map

Table 26-3 shows the I2C memory map.

Table 26-3. I2C Memory Map

Address	Register	Access	Reset Value	Section/Page
0x43F8_0000 (IADR1) 0x43F9_8000 (IADR2) 0x43F8_4000 (IADR3)	I2C Address Register	R/W	0x0000	26.3.3.1/26-6
0x43F8_0004 (IFDR1) 0x43F9_8004 (IFDR2) 0x43F8_4004 (IFDR3)	I2C Frequency Divider Register	R/W	0x0000	26.3.3.2/26-7
0x43F8_0008 (I2CR1) 0x43F9_8008 (I2CR2) 0x43F8_4008 (I2CR3)	I2C Control Register	R/W	0x0000	26.3.3.3/26-8
0x43F8_000C (I2SR1) 0x43F9_800C (I2SR2) 0x43F8_400C (I2SR3)	I2C Status Register	R/W	0x0081	26.3.3.4/26-10
0x43F8_0010 (I2DR1) 0x43F9_8010 (I2DR2) 0x43F8_4010 (I2DR3)	I2C Data I/O Register	R/W	0x0000	26.3.3.5/26-11

NOTE

There are registers at addresses 0x43F8_02, 0x43F8_06, 0x43F8_a, 0x43F8_0e, which are reserved for future additions.

26.3.2 Register Summary

Figure 26-3 shows the key to the register fields, and Table 26-5 shows the register figure conventions.

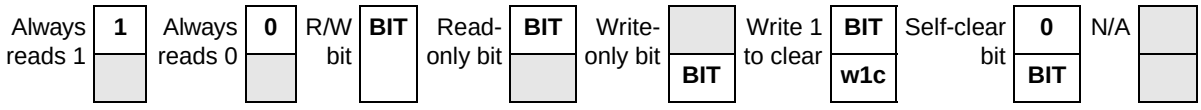


Figure 26-3. Key to Register Fields

Table 26-5. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 26-4 shows the I^2C register summary.

Table 26-4. I2C Register Summary

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x43F8_0010 (I2DR1) 0x43F9_8000 (IADR2) 0x43F8_4000 (IADR3)	R	0	0	0	0	0	0	0	0	ADR							0
	W																
0x43F8_0004 (IFDR1) 0x43F9_8004 (IFDR2) 0x43F8_4004 (IFDR3)	R	0	0	0	0	0	0	0	0	0	0	IC					
	W																
0x43F8_0008 (I2CR1) 0x43F9_8008 (I2CR2) 0x43F8_4008 (I2CR3)	R	0	0	0	0	0	0	0	0	IEN	IIEN	MSTA	MTX	TXAK	0	0	0
	W														RSTA		
0x43F8_000C (I2SR1) 0x43F9_800C (I2SR2) 0x43F8_400C (I2SR3)	R	0	0	0	0	0	0	0	0	ICF	IAAS	IBB	IAL	0	SRW	IIF	RXAK
	W																
0x43F8_0010 (I2DR1) 0x43F9_8010 (I2DR2) 0x43F8_4010 (I2DR3)	R	0	0	0	0	0	0	0	0	DATA							
	W																

26.3.3 Register Descriptions

This section contains the detailed register descriptions for the I2C registers in address order.

26.3.3.1 I2C Address Register (IADR)

Figure 26-6 shows the I2C Address Register; Table 26-5 provides its field descriptions.

0x43F8_0000 (IADR1)
0x43F9_8000 (IADR2)
0x43F8_4000 (IADR3)

Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	ADR							0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-6. I2C Address Register

The IADR holds the address the I2C responds to when addressed as a slave.

NOTE

The slave address is not the address sent on the bus during the address transfer. The register is not reset by a software reset.

Table 26-5. I2C Address Register Field Descriptions

Field	Description
15–8	Reserved
7–1 ADR	Slave address. Contains the specific slave address to be used by the I2C module. Slave mode is the default I2C mode for an address match on the bus.
0	Reserved

26.3.3.2 I2C Frequency Register (IFDR)

The IFDR provides a programmable prescaler to configure the clock for bit-rate selection. The register does not get reset by software reset. Figure 26-7 shows the I2C Frequency Register; Table 26-6 provides its field descriptions.

0x43F8_0004 (IFDR1)
0x43F9_8004 (IFDR2)
0x43F8_4004 (IFDR3)

Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	IC					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-7. I2C Frequency Register

Table 26-6. I2C Frequency Register Field Descriptions

Field	Description
15–6	Reserved
5–0 IC	I2C clock rate. Prescales the clock for bit-rate selection. Due to potentially slow SCL and SDA rise and fall times, bus signals are sampled at the prescaler frequency. The serial bit clock frequency is equal to IPG_CLK_PATREF divided by the divider shown in Table 26-7. **Note:** The IC can be changed anywhere in a program. I2C protocol supports bit rates up to 400 kbps. The IC bits need to be programmed in accordance with this constraint.

Table 26-7. IFDR Register Field Values

IC	Divider	IC	Divider	IC	Divider	IC	Divider
0x00	30	0x10	288	0x20	22	0x30	160
0x01	32	0x11	320	0x21	24	0x31	192
0x02	36	0x12	384	0x22	26	0x32	224
0x03	42	0x13	480	0x23	28	0x33	256
0x04	48	0x14	576	0x24	32	0x34	320
0x05	52	0x15	640	0x25	36	0x35	384
0x06	60	0x16	768	0x26	40	0x36	448
0x07	72	0x17	960	0x27	44	0x37	512
0x08	80	0x18	1152	0x28	48	0x38	640
0x09	88	0x19	1280	0x29	56	0x39	768
0x0A	104	0x1A	1536	0x2A	64	0x3A	896
0x0B	128	0x1B	1920	0x2B	72	0x3B	1024
0x0C	144	0x1C	2304	0x2C	80	0x3C	1280
0x0D	160	0x1D	2560	0x2D	96	0x3D	1536
0x0E	192	0x1E	3072	0x2E	112	0x3E	1792
0x0F	240	0x1F	3840	0x2F	128	0x3F	2048

26.3.3.3 I2C Control Register (I2CR)

The I2CR is used to enable the I2C module and the I2C interrupt. It also contains bits that govern operation as a slave or a master. Figure 26-8 shows the I2C Control Register; Table 26-8 provides its field descriptions.

0x43F8_0008 (I2CR1)
0x43F9_8008 (I2CR2)
0x43F8_4008 (I2CR3)

Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	IEN	IIEN	MSTA	MTX	TXAK	0	0	0
W														RSTA		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-8. I^2C Control Register

Table 26-8. I^2C Control Register Field Descriptions

Field	Description
15–8	Reserved
7 IEN	I^2C enable. Also controls the software reset of the entire I^2C module. Resetting the bit generates an internal reset to the module. If the module is enabled in the middle of a byte transfer, slave mode ignores the current bus transfer and starts operating when the next start condition is detected. Master mode is not aware that the bus is busy; so initiating a start cycle may corrupt the current bus cycle, ultimately causing either the current master or the I^2C module to lose arbitration. After which, bus operation returns to normal. 0 The module is disabled, but registers can still be accessed. 1 The I^2C module is enabled. This bit must be set before any other I2CR bits have any effect.
6 IIEN	I^2C interrupt enable. 0 I^2C module interrupts are disabled, but the status flag I2SR[IIF] continues to be set when an interrupt condition occurs. 1 I^2C module interrupts are enabled. An I2C interrupt occurs if I2SR[IIF] is also set.
5 MSTA	Master/slave mode select bit. If the master loses arbitration, MSTA is cleared without generating a STOP signal. **Note:** Module clock should be on for writing to the MSTA bit. 0 Slave mode. Changing MSTA from 1 to 0 generates a STOP and selects slave mode. 1 Master mode. Changing MSTA from 0 to 1 signals a START on the bus and selects master mode.
4 MTX	Transmit/receive mode select bit. Selects the direction of master and slave transfers. 0 Receive. When a slave is addressed, the software should set MTX according to the slave read/write bit in the I^2C status register (I2SR[SRW]). 1 Transmit. In master mode, MTX should be set according to the type of transfer required. Therefore, for address cycles, MTX is always 1.
3 TXAK	Transmit acknowledge enable. Specifies the value driven onto SDA during acknowledge cycles for both master and slave receivers. **Note:** Writing TXAK applies only when the I^2C bus is a receiver. 0 An acknowledge signal is sent to the bus at the ninth clock bit after receiving one byte of data. 1 No acknowledge signal response is sent (that is, the acknowledge bit = 1).
2 RSATA	Repeat start. Always reads as 0. Attempting a repeat start without bus mastership causes loss of arbitration. 0 No repeat start 1 Generates a repeated START condition
1–0	Reserved

26.3.3.4 I2C Status Register (I2SR)

The I2SR contains bits that indicate transaction direction and status. Figure 26-9 shows the I2C Address Register; and Table 26-9 provides its field descriptions.

0x43F8_000C (I2SR1)
0x43F9_800C (I2SR2)
0x43F8_400C (I2SR3)

Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	ICF	IAAS	IBB	IAL	0	SRW	IIF	RXAK
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1

Figure 26-9. I2C Status Register

Table 26-9. I2C Status Register Field Descriptions

Field	Description
15–8	Reserved
7 ICF	Data transferring bit. While one byte of data is transferred, ICF is cleared. 0 Transfer is in progress. 1 Transfer is complete, and set by the falling edge of the ninth clock of a byte transfer.
6 IAAS	I2C addressed as a slave bit. The CPU is interrupted if the interrupt enable (I2CR[IIEN]) is set. The CPU must check the slave read/write bit (SRW) and set its TX/RX mode accordingly. Writing to I2CR clears this bit. 0 Not addressed 1 Addressed as a slave. Set when its own address (IADR) matches the calling address.
5 IBB	I2C bus busy bit. Indicates the status of the bus. 0 Bus is idle. If a STOP signal is detected, IBB is cleared. 1 Bus is busy. When START is detected, IBB is set.
4 IAL	Arbitration lost. Set by hardware in the following circumstances (IAL must be cleared by software by writing a “0” to it): • SDA input sampled low when the master drives high during an address or data-transmit cycle. • SDA input sampled low when the master drives high during the acknowledge bit of a data-receive cycle. For the above two cases, the bit is set at the falling edge of 9th SCL clock during the ACK cycle. • A start cycle is attempted when the bus is busy. • A repeated start cycle is requested in slave mode. • A stop condition is detected when the master did not request it. **Note:** Software cannot set the bit. 0 No arbitration lost. 1 Arbitration is lost.
3	Reserved

Table 26-9. I2C Status Register Field Descriptions (continued)

Field	Description
2 SRW	Slave read/write. When the I2C is addressed as a slave, IAAS is set, and the slave read/write bit (SRW) indicates the value of the R/W command bit of the calling address sent from the master. SRW is valid only when a complete transfer has occurred, no other transfers have been initiated, and the I2C module is a slave and has an address match. 0 Slave receive, master writing to slave 1 Slave transmit, master reading from slave
1 IIF	I2C interrupt. Must be cleared by the software by writing a “0” to it in the interrupt routine. **Note:** The software cannot set the bit. 0 No I2C interrupt pending. 1 An interrupt is pending. This causes a processor interrupt request (if the interrupt enable is asserted [IIEN = 1]). The interrupt is set when one of the following occurs: • One byte transfer is completed (the interrupt is set at the falling edge of the ninth clock). • An address is received that matches its own specific address in slave-receive mode. • Arbitration is lost.
0 RXAK	Received acknowledge. This is the value received of the SDA input for the acknowledge bit during a bus cycle. 0 An “acknowledge” signal was received after the completion of an 8-bit data transmission on the bus. 1 A “No acknowledge” signal was detected at the ninth clock.

26.3.3.5 I2C Data Register (I2DR)

In master-receive mode, reading the data register (I2DR) allows a read to occur and initiates the next byte to be received. In slave mode, the same function is available after it is addressed. Figure 26-10 shows the I2C Data Register; Table 26-10 provides its field descriptions.

0x43F8_0010 (I2DR1)
0x43F9_8010 (I2DR2)
0x43F8_4010 (I2DR3)

Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	DATA							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-10. I2C Data Register

Table 26-10. I2C Data Register Field Descriptions

Field	Description
15–8	Reserved
7–0 DATA	Data Byte. Holds the last data byte received or the next data byte to be transferred. Software writes the next data byte to be transmitted or reads the data byte received.

NOTE

The core-written value in I2DR cannot be read back by the core: Only data written by the I2C bus side can be read.

26.4 Functional Description

26.4.1 I2C System Configuration

Out of a reset, the I2C module defaults to slave receive operations. Thus, when not operating as a master or responding to a slave transmit address, the I2C module will default to the slave receiver state. For exceptions, see Section 26.5.1, "Initialization Sequence."

NOTE

The I2C module is designed to be compatible with the Philips I2C bus protocol. For information on system configuration, protocol, and restrictions, refer to the I2C Bus Specification, Version 2.1. The I2C module supports Standard and Fast modes only.

26.4.2 I2C Protocol

The I2C communication protocol consists of six components, as follows:

- START
- Data Source/Recipient
- Data Direction
- Slave Acknowledge
- Data Acknowledge
- STOP

See Figure 26-11 for the I2C standard communication protocol, as defined in the following sections.

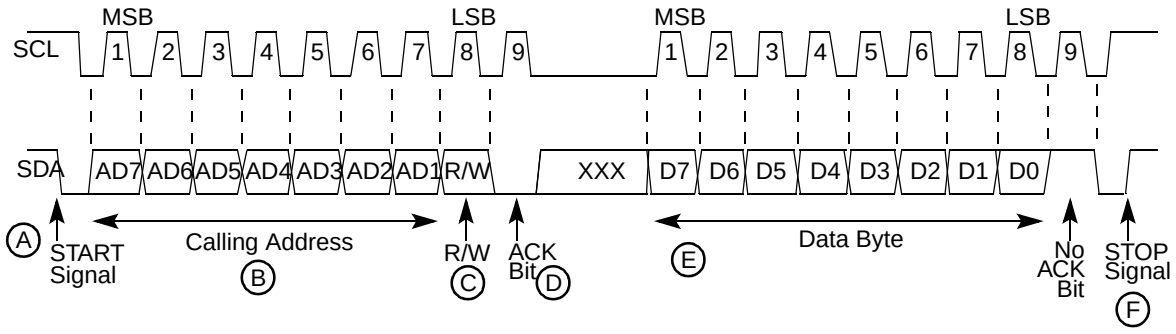


Figure 26-11. I2C Standard Communication Protocol

26.4.2.1 START Signal

When no other device is a bus master (both SCL and SDA lines are at logic high), a device can initiate communication by sending a START signal (see A in Figure 26-11). A START signal is defined as a high-to-low transition of SDA while SCL is high. This signal denotes the beginning of a data transfer (each data transfer can be several bytes long) and awakens all slaves.

26.4.2.2 Slave Address Transmission

The master sends the slave address in the first byte after the START signal (B). After the seven-bit calling address, it sends the R/W bit (C), which tells the slave data transfer direction.

Each slave must have a unique address. An I^2C master must not transmit an address that is the same as its slave address; it cannot be master and slave at the same time.

The slave whose address matches that sent by the master pulls SDA low at the ninth clock (D) to return an acknowledge bit.

26.4.2.3 Data Transfer

When successful slave addressing is achieved, the data transfer can proceed (E) on a byte-by-byte basis in the direction specified by the R/W bit sent by the calling master.

Data can be changed only while SCL is low and must be held stable while SCL is high, as shown in Figure 26-11. SCL is pulsed once for each data bit, with the mishap being sent first. The receiving device must acknowledge each byte by pulling SDA low at the ninth clock; therefore, a data byte transfer takes nine clock pulses.

If it does not acknowledge the master, the slave receiver must leave SDA high. The master can then generate a STOP signal to abort the data transfer or generate a START signal (a repeated start, as shown in Figure 26-12) to start a new calling sequence.

If the master receiver does not acknowledge the slave transmitter after a byte transmission, it means end-of-data to the slave. The slave releases SDA for the master to generate a STOP or START signal.

26.4.2.4 STOP Signal

The master can terminate communication by generating a STOP signal to free the bus. A STOP signal is defined as a low-to-high transition of SDA while SCL is at logical high (F).

NOTE

A master can generate a STOP even if the slave has made an acknowledgment; at which point, the slave must release the bus.

26.4.2.5 Repeat Start

Instead of signalling a STOP, the master can repeat the START signal, followed by a calling command (see A in Figure 26-12). A repeated START occurs when a START signal is generated without first generating

a STOP signal to end the communication. The master uses a repeated START to communicate with another slave or with the same slave in a different mode (transmit/receive mode) without releasing the bus.

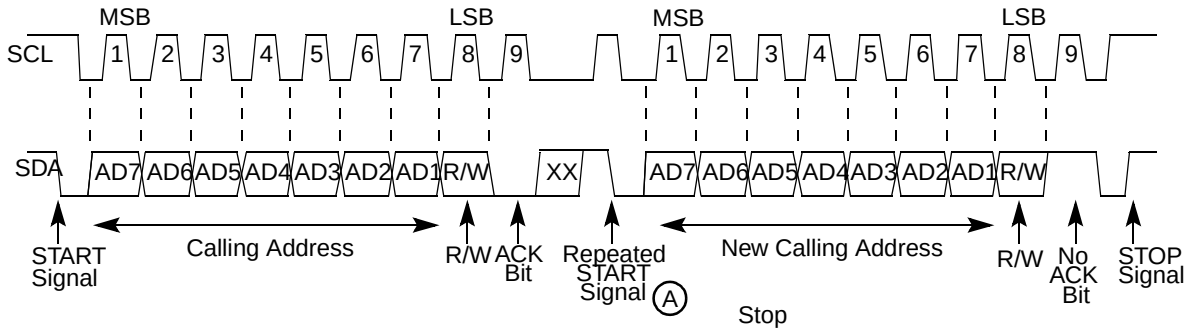


Figure 26-12. Repeated START

26.4.3 Arbitration Procedure

If multiple devices simultaneously request the bus, the bus clock is determined by a synchronization procedure in which the low period equals the longest clock-low period among the devices, and the high period equals the shortest. A data arbitration procedure determines the relative priority of competing devices. A device loses arbitration if it sends logic high while another sends logic low; it immediately switches to slave-receive mode and stops driving SDA. In this case, the transition from master to slave mode does not generate a STOP condition. Meanwhile, hardware sets the arbitration lost bit in the I^2C Status register (I2SR[IAL] to indicate loss of arbitration.

26.4.4 Clock Synchronization

Because wire-AND logic is used, a high-to-low transition on SCL affects devices connected to the bus. Devices start counting their low period when the master drives SCL low. When a device clock goes low, it holds SCL low until the clock high state is reached. However, the low-to-high change in this device clock may not change the state of SCL if another device clock is still in its low period. Therefore, the device with the longest low period holds the synchronized clock SCL low. Devices with shorter low periods enter a high wait state during this time (see Figure 26-13). When all devices involved have counted off their low period, the synchronized clock SCL is released and pulled high. There is then no difference between device clocks and the state of SCL, so all of the devices start counting their high periods. The first device to complete its high period pulls SCL low again.

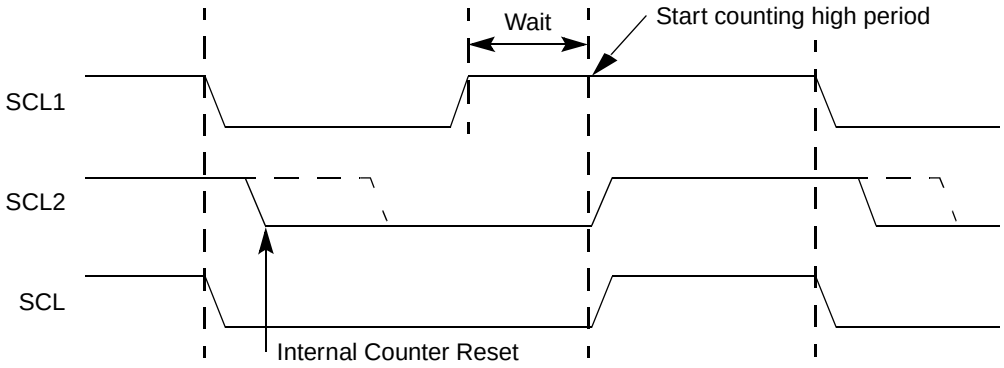


Figure 26-13. Synchronized Clock SCL

26.4.5 Handshaking

The clock synchronization mechanism can be used as a handshake in data transfers. Slave devices can hold SCL low after completing one byte transfer (9 bits). In such a case, the clock mechanism halts the bus clock and forces the master clock into a wait state until the slave releases SCL.

26.4.6 Clock Stretching

Slaves can use the clock synchronization mechanism to slow down the transfer bit rate. After the master has driven SCL low, the slave can drive SCL low for the required period and then release it. If the slave SCL low period is longer than the master SCL low period, the resulting SCL bus signal low period is stretched.

26.4.7 IP Bus Accesses

I^2C is a 16-bit IP module. Only halfword accesses should be performed to the module.

26.4.8 Generation of Transfer Error on IP Bus

If an address is received on the IP slave bus interface that is not implemented, an access error is generated (IPS_XFR_ERR is asserted). The input pin resp_sel provides the configuration capability to generate this response. The resp_sel pin must be asserted to enable the IPS_XFR_ERR signal.

26.5 Initialization/Application Information

26.5.1 Initialization Sequence

Before the interface can transfer serial data, registers must be initialized, as follows:

1. Set the data sampling rate (IFDR[IC] to obtain SCL frequency from the system bus clock.
2. Update the address in the (IADR) to define its slave address (address can range from 0 to 0x7f).
3. Set the I^2C enable bit (I2CR[IEN]) to enable the I^2C bus interface system.

4. Modify the bits in the I^2CR to select master/slave mode, transmit/receive mode, and interrupt-enable or not.

26.5.2 Generation of START

After completion of the initialization procedure, serial data can be transmitted by selecting the master transmitter mode. On a multiple-master bus system, the busy bus (I2SR[IBB]) must be tested to determine whether the serial bus is free. If the bus is free (IBB = 0), the START signal and the first byte (the slave address) can be sent. The data written to the data register comprises the address of the desired slave and the LSB indicates the transfer direction.

The free time between a STOP and the next START condition is built into the hardware that generates the START cycle. Depending on the relative frequencies of the system clock and the SCL period, it may be necessary to wait until the I^2C is busy after writing the calling address to the data register (I2DR) before proceeding to load data into the data register (I2DR).

26.5.3 Post-Transfer Software Response

Sending or receiving a byte sets the data transferring bit (I2SR[ICF]), which indicates one byte communication is finished. Upon completion, the interrupt status (I2SR[IIF]) is also set. An external interrupt is generated if the interrupt enable (I2CR[IIEN]) is set. The software must first clear the interrupt status (I2SR[IIF]) in the interrupt routine. (See the flowchart in Figure 26-14.) The data transferring bit (I2SR[ICF]) is cleared either by reading from I2DR in receive mode or by writing to this register in transmit mode.

The software can service the I^2C I/O in the main program by monitoring the interrupt status (I2SR[IIF]) if the interrupt enable is de-asserted. In this case, the interrupt status should be polled of the data transferring bit (I2SR[ICF]) because the operation is different when arbitration is lost.

When an interrupt occurs at the end of the address cycle, the master is always in transmit mode; that is, the address is sent. If master receive mode is required, then (I2DR[R/W], I2CR[MTX] should be toggled.

During slave-mode address cycles (I2SR[IAAS] = 1), the slave read/write bit I2SR[SRW] is read to determine the direction of the next transfer. The transmit/receive bit (I2CR[MTX]) should also be programmed accordingly. For slave-mode data cycles (IAAS = 0), SRW is invalid. MTX should be read to determine the current transfer direction.

26.5.4 Generation of STOP

A data transfer ends when the master signals a STOP, which can occur after all data is sent.

For a master receiver to terminate a data transfer, it must inform the slave transmitter by not acknowledging the last data byte. This is done by setting the transmit acknowledge bit (I2CR[TXAK]) before reading the next-to-last byte. Before the last byte is read, a STOP signal must be generated.

26.5.5 Generation of Repeated START

After the data transfer, if the master still wants the bus, it can signal another START followed by another slave address without signalling a STOP.

26.5.6 Slave Mode

In the slave interrupt service routine (see Figure 26-14), the module addressed as slave bit (IAAS) should be tested to check if a calling of its own address has just been received. If IAAS is set, software should set the transmit/receive mode select bit (I2CR[MTX]) according to the I2SR[SRW]. Writing to the I2CR clears the IAAS automatically. The only time IAAS is read as set is from the interrupt at the end of the address cycle where an address match occurred; interrupts resulting from subsequent data transfers will have IAAS cleared. A data transfer can now be initiated by writing information to I2DR for slave transmits, or read from I2DR in slave-receive mode. A dummy read of I2DR in slave/receive mode releases SCL, allowing the master to send data.

In the slave transmitter routine, the receive acknowledge bit (I2SR[RXAK]) must be tested before sending the next byte of data. Setting RXAK means an end-of-data signal from the master receiver, after which the software must switch it from transmitter to receiver mode. Reading the data register (I2DR) then releases SCL so that the master can generate a STOP signal.

26.5.7 Arbitration Lost

If several devices try to engage the bus at the same time, one becomes master. Hardware immediately switches devices that lose arbitration to slave receive mode. Data output to SDA stops, but SCL is still generated until the end of the byte during which arbitration is lost. An interrupt occurs at the falling edge of the ninth clock of this transfer if the arbitration is lost (I2SR[IAL] = 1), and the slave mode is selected (I2CR[MSTA] = 0). See the flowchart in Figure 26-14.

If a device that is not a master tries to transmit or do a START, hardware inhibits the transmission, clears MSTA without signalling a STOP, generates an interrupt to the CPU, and sets IAL to indicate a failed attempt to engage the bus. When considering these cases, the slave service routine should first test IAL, and the software should clear it if it is set.

For Multi-master mode, when an I^2C module is enabled when the bus is busy and asserts START, the IAL bit gets set only for SDA=0, SCL=0/1, SDA=1, and SCL=0, but not for SDA=1 and SCA=1, which is the same as bus idle state.

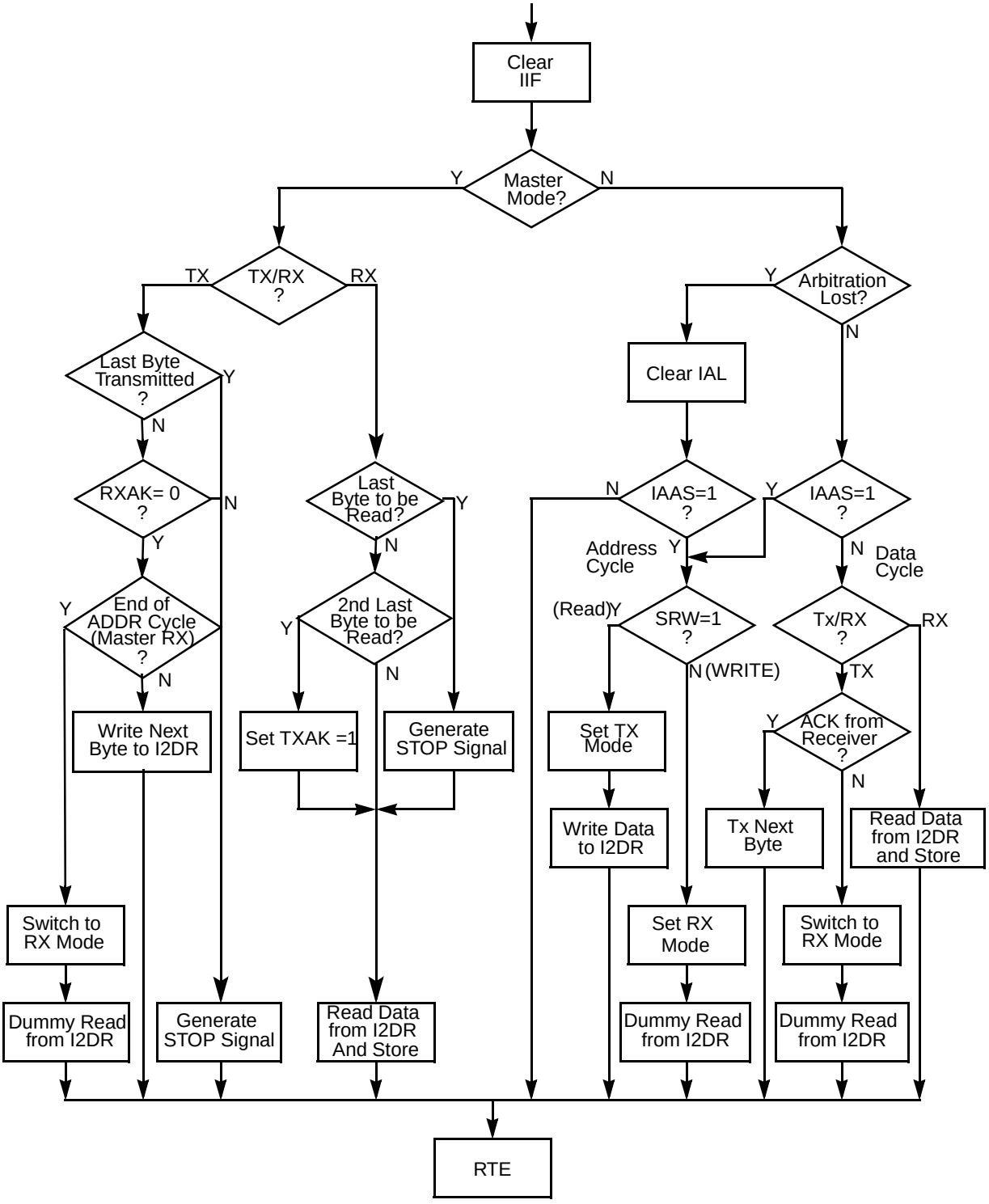


Figure 26-14. Flowchart of Typical I^2C Interrupt Routine

NOTE

For a repeated start-only, the stop generation stage will not occur in master mode. A loop will repeat itself without stopping for the next start.

26.5.8 Timing Section

Figure 26-15 provides an illustration of the timing for the serial data line (SDA) and serial clock line (SCL) devices on the I^2C bus.

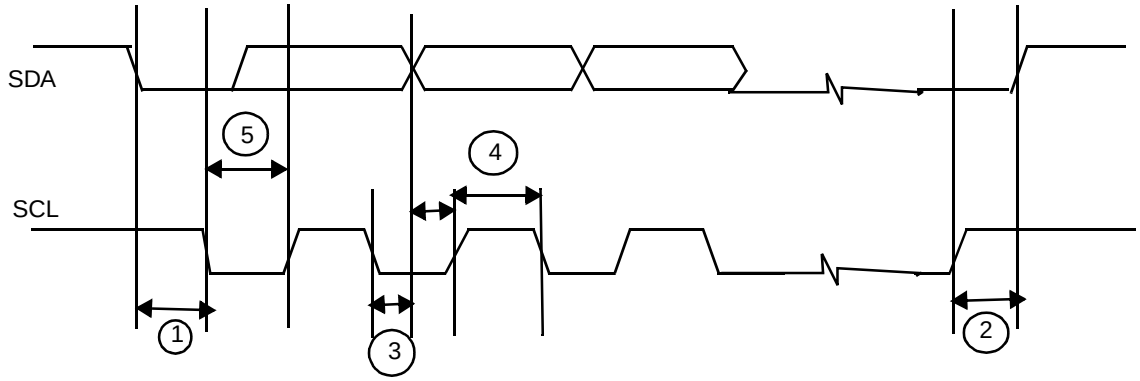


Figure 26-15. Definition of Timing for Devices on I^2C Bus

Table 26-11 provides a list of bus timing parameters.

Table 26-11. I^2C Bus Timing Parameters

Reference Number	Parameter	Maximum (w.r.t ipg_clk_patref)	Minimum (w.r.t ipg_clk_patref)
1	Hold time (repeated) START condition	—	4
2	Setup time for STOP condition	—	4
3	Data hold time	(0.27) * Divider	—
4	HIGH of the SCL Period	—	(0.4) * Divider (Master mode)
5	LOW period of the SCL Clock	—	(0.4) * Divider (Master mode)

NOTE

See Table 26-7 for details for Divider values.

Chapter 27
Keypad Port (KPP)

The Keypad Port (KPP) is a 16-bit peripheral that can be used as a keypad matrix interface or as general purpose input/output (I/O). Figure 27-1 shows the KPP block diagram.

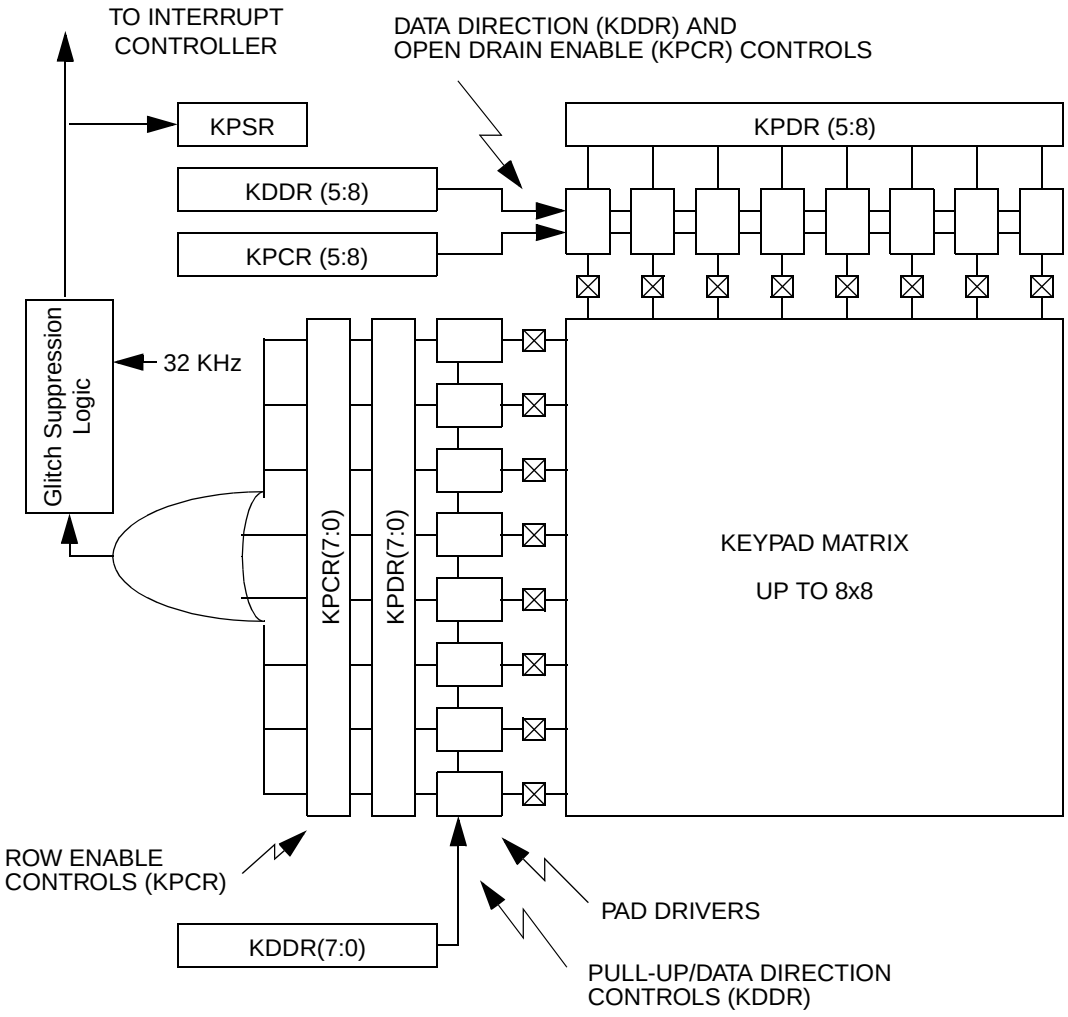


Figure 27-1. KPP Peripheral Block Diagram

27.1 Overview

The KPP is designed to interface with the keypad matrix with 2-point contact or 3-point contact keys. The KPP is designed to simplify the software task of scanning a keypad matrix. With appropriate software

support, the KPP is capable of detecting, debouncing, and decoding one or multiple keys pressed simultaneously on the keypad.

27.1.1 Features

The KPP includes these distinctive features:

- Supports up to an 8 x 8 external key pad matrix
- Port pins can be used as general purpose I/O
- Open drain design
- Glitch suppression circuit design
- Multiple-key detection
- Long key-press detection
- Standby key-press detection
- Synchronizer chain clear
- Supports a 2-point and 3-point contact key matrix

27.1.2 Modes of Operation

This module supports the following modes of operation:

- Run Mode—This is the normal functional mode in which the KPP can detect any key press event.
- Low Power Modes—The keypad can detect any key press even in low power modes (when there is no MCU clock).

27.2 External Signal Description

27.2.1 Overview

There are 16 pins dedicated to the KPP. Keypads of any configuration up to eight rows and eight columns are supported through the software configuration of the peripheral pins. Any pins not used for the keypad are available as general purpose I/O. The registers are configured such that the pins can be treated as an I/O port up to 16 bits wide.

27.2.1.1 Input Pins

Any of the 16 pins associated with the KPP can be configured as inputs by writing a “0” to the appropriate bits in the KDDR. Additionally, the least significant 8 bits (ROW inputs) corresponding to KDDR7:0 have internal pull-ups, which are enabled when the pin is used as an input.

27.2.1.2 Output Pins

Any of the 16 pins associated with the KPP can be configured as outputs by writing the appropriate bits in the KDDR to a “1”. Additionally, the 8 most significant bits (15–8) can be designated as open drain outputs

by writing a “1” to the appropriate bits in the KPCR. The lower 8 bits (7–0) are always in “totem pole” style, driven when configured as outputs. See Table 27-1.

Table 27-1. Keypad Port Column Modes

KDDR (15:8)	KPCR (15:8)	Pin Function
0	x	Input
1	0	Totem-Pole Output
1	1	Open-Drain Output

NOTE

Totem pole capability should be provided for column pins. Totem pole configuration helps for a faster discharge of keypad capacitance when all columns need to be quickly brought to a “1” during the scan routine. With this configuration, a time delay between the scanning of two subsequent columns is reduced.

27.3 Memory Map and Register Definition

The KPP module contains four registers. Section 27.3.3, “Register Descriptions” provides detailed descriptions of the KPP registers.

27.3.1 KPP Memory Map

Table 27-2 shows the KPP memory map.

Table 27-2. KPP Memory Map

Address	Use	Access	Reset Value	Section/Page
0x43FA_8000 (KPCR)	Keypad Control Register	R/W	0x0000	27.3.3.1/27-5
0x43FA_8002 (KPSR)	Keypad Status Register	R/W	0x0000	27.3.3.2/27-5
0x43FA_8004 (KDDR)	Keypad Data Direction Register	R/W	0x0000	27.3.3.3/27-7
0x43FA_8006 (KPDR)	Keypad Data Register	R/W	0x— — — —	27.3.3.4/27-8

27.3.2 Register Summary

Figure 27-2 shows the key to the register fields, and Table 27-3 shows the register figure conventions.

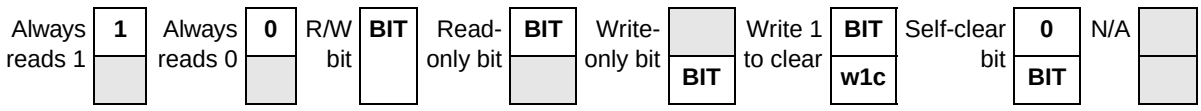


Figure 27-2. Key to Register Fields

Table 27-3. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 27-4 shows the KPP register summary.

Table 27-4. KPP Register Summary

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x43FA_8000 (KPCR)	R	KCO7	KCO6	KCO5	KCO4	KCO3	KCO2	KCO1	KCO0	KRE7	KRE6	KRE5	KRE4	KRE3	KRE2	KRE1	KRE0
	W																
0x43FA_8002 (KPSR)	R	0	0	0	0	0	0	KRIE	KDIE	0	0	0	0	0	0	KPKR	KPKD
	W													KRSS	KDSC	w1c	w1c
0x43FA_8004 (KDDR)	R	KCDD7	KCDD6	KCDD5	KCDD4	KCDD3	KCDD2	KCDD1	KCDD0	KRDD7	KRDD6	KRDD5	KRDD4	KRDD3	KRDD2	KRDD1	KRDD0
	W																
0x43FA_8006 (KPDR)	R	KCD7	KCD6	KCD5	KCD4	KCD3	KCD2	KCD1	KCD0	KRD7	KRD6	KRD5	KRD4	KRD3	KRD2	KRD1	KRD0
	W																

27.3.3 Register Descriptions

This section consists of register descriptions. Each register is listed in the order of its address.

27.3.3.1 Keypad Control Register (KPCR)

The Keypad Control Register determines which of the eight possible column strobes are to be open drain when configured as outputs, and which of the eight row sense lines are considered in generating an interrupt to the core.

It is up to the programmer to ensure that pins being used for functions other than the keypad are properly disabled. The KPCR register is byte- or halfword–addressable.

Figure 27-3 shows the valid bits in the KPCR register, and Table 27-5 provides its field descriptions.

0x43FA_8000 (KPCR) Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	KCO7	KCO6	KCO5	KCO4	KCO3	KCO2	KCO1	KCO0	KRE7	KRE6	KRE5	KRE4	KRE3	KRE2	KRE1	KRE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 27-3. KPCR Register

Table 27-5. Keypad Control Register Field Descriptions

Field	Description
15–8 KCO	Keypad Column Strobe Open-Drain Enable. Setting a column open-drain enable bit (KCO7–KCO0) disables the pull-up driver on that pin. Clearing the bit allows the pin to drive to the high state. This bit has no effect when the pin is configured as an input. 0 Column strobe output is totem pole drive. 1 Column strobe output is open drain. **Note:** Configuration of external port control logic (for example, GPIO) should be done properly so that the KPP module controls an open-drain enable of the pin.
7–0 KCO	Keypad Row Enable. Setting a row enable control bit in this register enables the corresponding row line to participate in interrupt generation. Likewise, clearing a bit disables that row from being used to generate an interrupt. This register is cleared by a reset, disabling all rows. The row-enable logic is independent of the programmed direction of the pin. Writing a “0” to the data register of the pins configured as outputs will cause a keypad interrupt to be generated if the row enable associated with that bit is set. 0 Row is not included in the keypad key press detect. 1 Row is included in the keypad key press detect.

27.3.3.2 Keypad Status Register (KPSR)

The Keypad Status Register reflects the state of the key press detect circuit. The KPSR register is byte- or halfword–addressable.

Figure 27-4 shows the KPSR register, and Table 27-6 provides its field descriptions.

0x43FA_8002 (KPSR) Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	KRIE	KDIE	0	0	0	0	0	0	KPKR	KPKD
W													KRSS	KDSC	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 27-4. KPSR Register

Table 27-6. Keypad Status Register Field Descriptions

Field	Description
15–10	Reserved
9 KRIE	Keypad Release Interrupt Enable. The software should ensure that the interrupt for a Key Release event is masked until it has entered the key pressed state, and vice versa, unless this activity is desired (as might be the case when a repeated interrupt is to be generated). The synchronizer chains are capable of being initialized to detect repeated key presses or releases. If they are not initialized when the corresponding event flag is cleared, false interrupts may be generated for depress (or release) events shorter than the length of the corresponding chain. 0 No interrupt request is generated when KPKR is set. 1 An interrupt request is generated when KPKR is set.
8 KDIE	Keypad Key Depress Interrupt Enable. Software should ensure that the interrupt for a Key Release event is masked until it has entered the key pressed state, and vice-versa, unless this activity is desired (as might be the case when a repeated interrupt is to be generated). The synchronizer chains are capable of being initialized to detect repeated key presses or releases. If they are not initialized when the corresponding event flag is cleared, false interrupts may be generated for depress (or release) events shorter than the length of the corresponding chain. 0 No interrupt request is generated when KPKD is set. 1 An interrupt request is generated when KPKD is set.
7–4	Reserved, should be cleared
3 KRSS	Key Release Synchronizer Set. Self-clear bit. The Key release synchronizer is set by writing a logic one into this bit. Reads return a value of “0”. 0 No effect 1 Set bits which sets keypad release synchronizer chain
2 KDSC	Key Depress Synchronizer Clear. Self-clear bit. The Key depress synchronizer is cleared by writing a logic “1” into this bit. Reads return a value of “0”. 0 No effect 1 Set bits that clear the keypad depress synchronizer chain

Table 27-6. Keypad Status Register Field Descriptions (continued)

Field	Description
1 KPKR	Keypad Key Release. The keypad key release (KPKR) status bit is set when all enabled rows are detected high after synchronization (the KPKR status bit will be set when cleared by a reset). The KPKR bit may be used to generate a maskable key release interrupt. The key release synchronizer may be set high by software after scanning the keypad to ensure a known state. Due to the logic function of the release and depress synchronizer chains, it is possible to see the re-assertion of a status flag (KPKD or KPKR) if it is cleared by software prior to the system exiting the state it represents. 0 No key release is detected. 1 All keys have been released. Reset value of register is “0” as long as reset is asserted. However when reset is de-asserted, the value of the register depends upon the external row pins and can become “1”.
0 KPKD	Keypad Key Depress. The keypad key depress (KPKD) status bit is set when one or more enabled rows are detected low after synchronization. The KPKD status bit remains set until cleared by the software. The KPKD bit may be used to generate a maskable key depress interrupt. If desired, the software may clear the key press synchronizer chain to allow a repeated interrupt to be generated while a key remains pressed. In this case, a new interrupt will be generated after the synchronizer delay (4 cycles of the 32 KHz clock) elapses if a key remains pressed. This functionality can be used to detect a long key press. This allows detection of additional key presses of the same key or other keys. Due to the logic function of the release and depress synchronizer chains, it is possible to see the re-assertion of a status flag (KPKD or KPKR) if it is cleared by the software prior to the system exiting the state it represents. 0 No key presses have been detected. 1 A key has been depressed.

27.3.3.3 Keypad Data Direction Register (KDDR)

The bits in the KDDR control the direction of the keypad port pins. The upper eight bits in the register affect the pins designated as column strobes, while the lower eight bits affect the row sense pins. Setting any bit in this register configures the corresponding pin as an output. Clearing any bit in this register configures the corresponding port pin as an input. For the Keypad Row DDR, an internal pull-up is enabled if the corresponding bit is clear. This register is cleared by a reset, configuring all pins as inputs. The KDDR register is byte- or halfword–addressable.

NOTE

When a pin is used as row pin for keypad purposes, all corresponding pull-ups should be enabled at the upper level (for example, IOMUX) when the bit in row DDR is cleared.

Figure 27-5 shows the valid bits in the KDDR register, and Table 27-7 provides its field descriptions.

0x43FA_8004 (KDDR) Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	KCD7	KCD6	KCD5	KCD4	KCD3	KCD2	KCD1	KCD0	KRD7	KRD6	KRD5	KRD4	KRD3	KRD2	KRD1	KRD 0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 27-5. KDDR Register

Table 27-7. Keypad Data Direction Register Field Descriptions

Field	Description
15–8 KCDD	Keypad Column Data Direction Register. Setting any bit configures the corresponding pin as an output. 0 COLn pin is configured as an input. 1 COLn[1] pin is configured as an output.
7–0 KRDD	Keypad Row Data Direction. Setting any bit configures the corresponding pin as an output. 0 ROWn pin configured as an input. 1 ROWn[2] pin configured as an output.

[1]n=7-0
[2]n=7-0

27.3.3.4 Keypad Data Register (KPDR)

This 16-bit register is used to access the column and row data. Data written to this register is stored in an internal latch, and for each pin configured as an output, the stored data is driven onto the pin. A read of this register returns the value on the pin for those bits configured as inputs. Otherwise, the value read is the value stored in the register.

The KPDR register is byte- or halfword–addressable. This register is not initialized by a reset. Valid data should be written to this register before any bits are configured as outputs.

Figure 27-6 shows the KPDR register, and Table 27-8 provides its field descriptions.

0x43FA_8006 (KPDR) Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	KCD7	KCD6	KCD5	KCD4	KCD3	KCD2	KCD1	KCD0	KRD7	KRD6	KRD5	KRD4	KRD3	KRD2	KRD1	KRD 0
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Figure 27-6. KPDR Register

Table 27-8. Keypad Data Register Field Descriptions

Field	Description
15–8 KCD[	Keypad Column Data. A read of these bits returns the value on the pin for those bits configured as inputs. Otherwise, the value read is the value stored in the register. 0 Read/Write “0” from/to column ports 1 Read/Write “1” from/to column ports
7–0 KRD	Keypad Row Data. A read of these bits returns the value on the pin for those bits configured as inputs. Otherwise, the value read is the value stored in the register. 0 Read/Write “0” from/to row ports 1 Read/Write “1” from/to row ports

27.4 Functional Description

The Keypad Port (KPP) is designed to simplify the software task of scanning a keypad matrix. With appropriate software support and matrix organization, the KPP is capable of detecting, debouncing, and decoding one or more keys pressed simultaneously on the keypad.

Logic in the KPP is capable of detecting a key press even while the processor is in one of the low power standby modes providing that a 32 KHz clock is on. The KPP may generate a CPU interrupt any time a key press or key release is detected. This interrupt is capable of forcing the processor out of a low power mode.

27.4.1 Keypad Matrix Construction

The KPP is designed to interface to a keypad matrix, which shorts the intersecting row and column lines together whenever a key is depressed. The interface is not optimized for any other switch configuration.

27.4.2 Keypad Port Configuration

The software must initialize the KPP for the size of the keypad matrix. Pins connected to the keypad columns should be configured as open-drain outputs. Pins connected to the keypad rows should be configured as inputs. On-chip, pull-up resistors should be implemented for active keypad rows.

In addition to enabled row inputs in the Keypad Control register, corresponding interrupt (depress or/and release) must also be enabled to generate an interrupt.

Discrete switches that are not part of the matrix may be connected to any unused row inputs. The second terminal of the discrete switch is connected to ground. The hardware detects closures of these switches without the need for software polling.

27.4.3 Keypad Matrix Scanning

Keypad scanning is performed by a software loop that walks a zero across each of the keypad columns, reading the value on the rows at each step. The process is repeated several times in succession, with the results of each pass optionally compared to those from the previous pass. When several (3 or 4) consecutive scans yield the same key closures, a valid key press has been detected. Software then can decode exactly which switch was depressed and pass the value up to the next higher software layer.

The basic debouncing period, which must be defined in the software routine, may be controlled with an internal timer. The basic period is the period between the scan of two consecutive columns, so the debouncing time between two consecutive scans of the whole matrix shall be the number of columns multiplied by the basic period.

27.4.4 Keypad Standby

There is no need for the CPU to continually scan the keypad. Between key presses, the keypad can be left in a state that requires no software intervention until the next key press is detected. To place the keypad in a standby state, software should write all column outputs low. Row inputs are left enabled. At this point,

the CPU can attend to other tasks or revert to a low power standby mode. The KPP will interrupt the CPU if any key is pressed.

Upon receiving a keypad interrupt, the CPU should set all the column strobes high, and begin a normal keypad scanning routine to determine which key was pressed. It is important that open-drain drivers be used when scanning to prevent a possible DC path between power and ground through two or more switches.

27.4.5 Glitch Suppression on Keypad Inputs

A glitch suppression circuit qualifies the keypad inputs to prevent noise from inadvertently interrupting the CPU. The circuit is a 4-state synchronizer clocked from a 32 KHz clock source. This clock must continue to run in any low power mode where the keypad is a wake-up source, as the CPU interrupt is generated from the synchronized input. An interrupt is not generated until all four synchronizer stages have latched a valid key assertion. This guarantees the filtering out of any noise less than three clock periods (for 32 KHz clock: 93.75 µs) in duration. Noise filtering of the duration between three to four clock periods (for the 32 KHz clock: between 93.75 µs and 125 µs) cannot be guaranteed. The interrupt output is latched in an S-R latch and remains asserted until cleared by the software. The Set input of the latch is rising-edge clocked. See Figure 27-7.

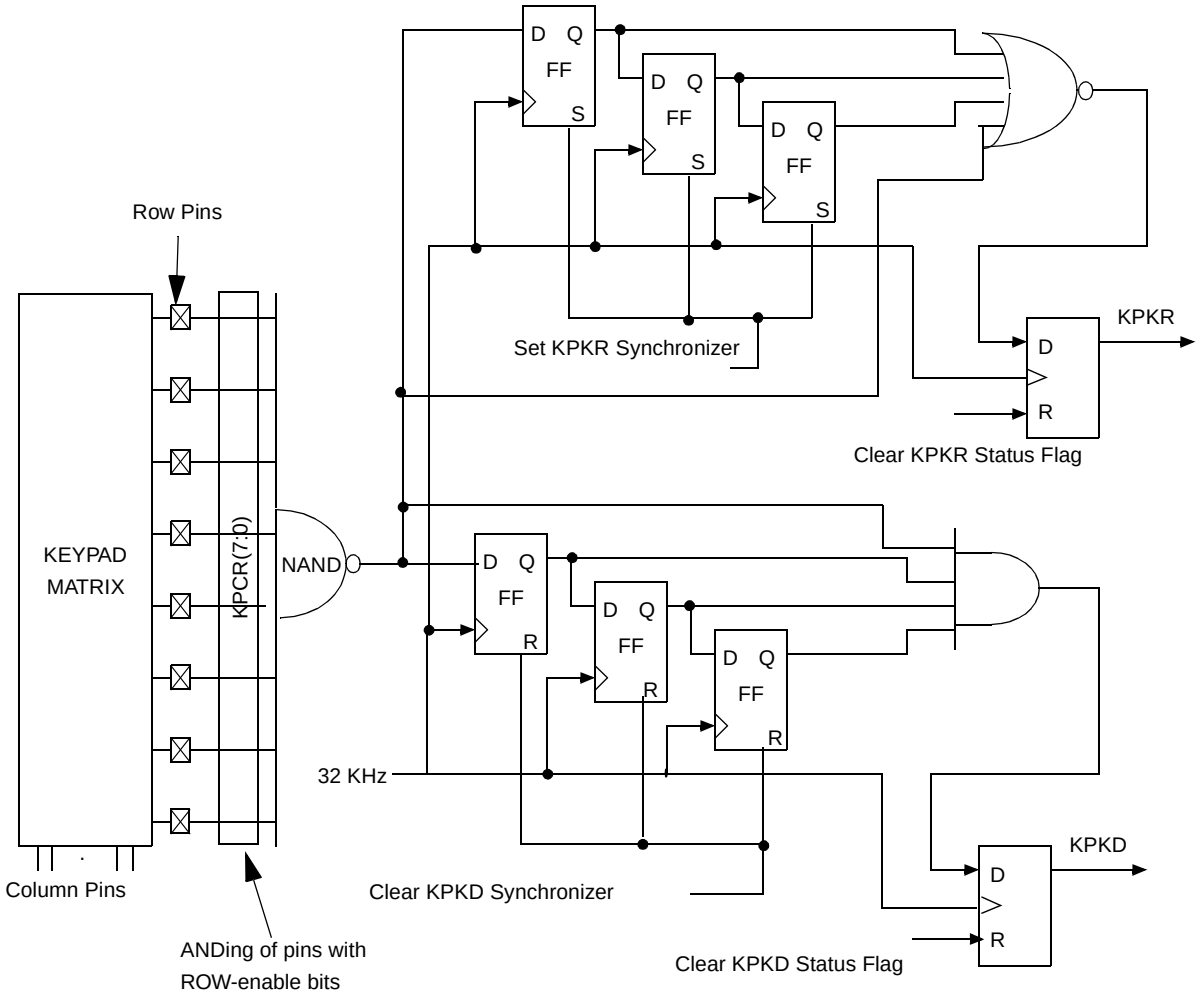


Figure 27-7. Keypad Synchronizer Functional Diagram

27.4.6 Multiple Key Closures

Using the key press and Key release interrupts, the software can detect multiple keys or achieve n key rollover. The key scanning routine can be programmed accordingly. Refer to Section 27.5, "Initialization/Application Information" for more information.

See Figure 27-5 and Figure 27-9 for illustrations of the interfacing of a 2-contact keypad matrix with the KPP controller. With proper enabling of row lines and the performing scan-routine, multiple key presses can be detected. When keys present on the same row are pressed, corresponding row lines (multiple lines) become low when the column is driven low during a scan-routine. By reading the data-register, pressed keys can be detected. Similarly, when keys present on same row line are pressed, the corresponding row line (only one line) becomes low when logic "0" is driven on the column line during a scan-routine.

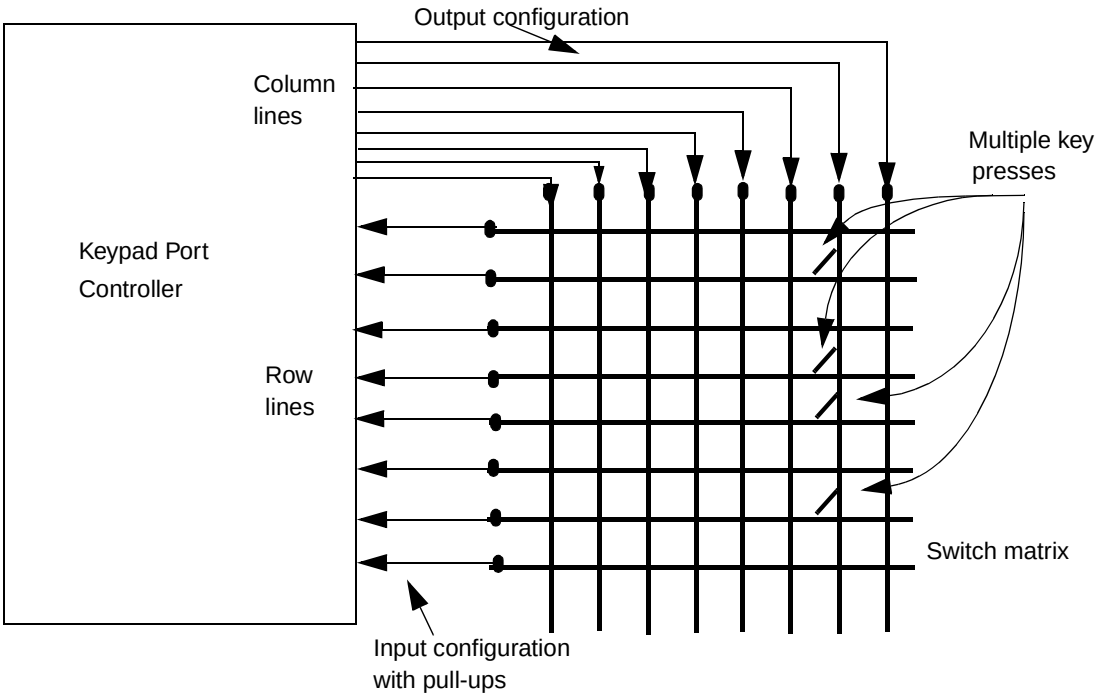


Figure 27-8. Multiple Key Presses on Same Column Line (Simplified View)

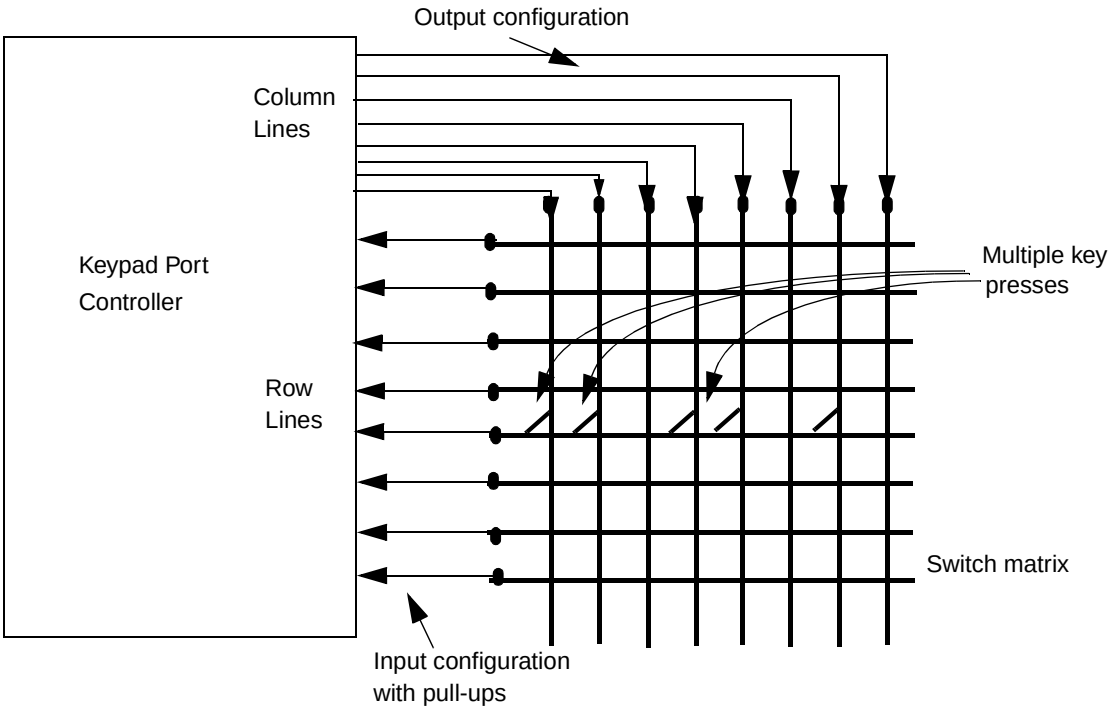


Figure 27-9. Multiple Key Presses on Same Row Line (Simplified View)

NOTE

An n key rollover is a technique the system uses to recognize the order in which keys are pressed.

27.4.6.1 Ghost Key Problem and Correction

The KPP module detects if one or multiple keys are pressed or released. In the case where a simple keypad matrix with two-contact switches is used, there is a chance of “ghost” key detection when three or more keys are pressed. This is a limitation imposed by such a keypad matrix. As can be seen in Figure 27-10, three keys pressed simultaneously can cause a short between the column currently “scanned” by the software and another column. Depending on the location of the third key pressed, a “ghost” key press may be detected.

However, this can be corrected by using a keypad matrix that provides “ghost” key protection. Such a matrix implements a one-way “diode” at all keypad points between rows and columns. This way, the multiple pressing of three keys will not cause a short at a fourth key (see Figure 27-11).

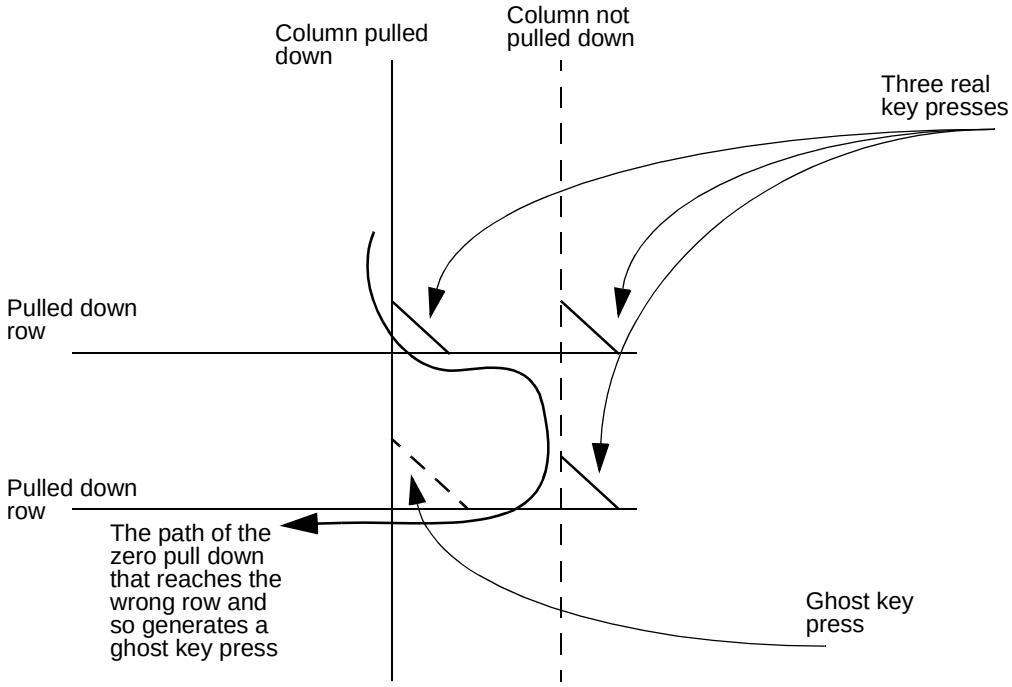


Figure 27-10. Decoding Wrong Three-Key Presses

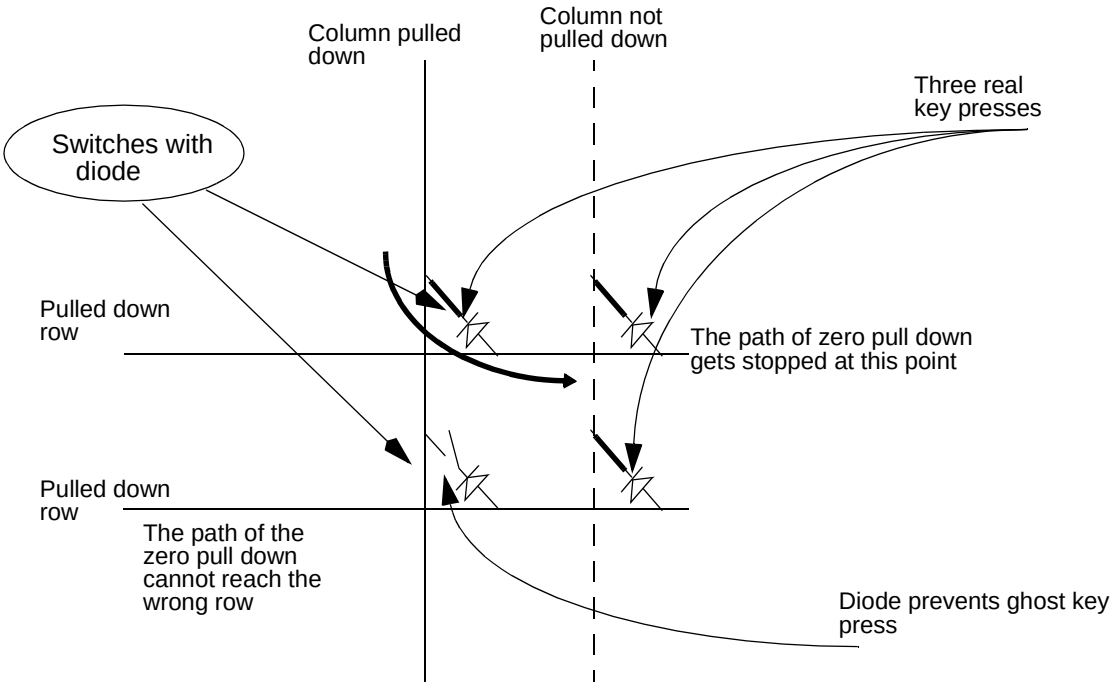


Figure 27-11. Matrix with "Ghost" Key Protections

27.4.7 3-Point Contact Keys Support

The KPP module supports interfacing to a matrix consisting of 3-point contact keys. As shown in Figure 27-12, two points of such a key are connected to keypad lines, while a third point is connected to ground (low logic). The keypad lines should be configured as input and a pull-up should to be present on these lines. When such a key is pressed, corresponding keypad lines go low and an interrupt is generated. There is no need to perform a scanning routine for identification of pressed key as it can be done by reading the keypad data-register. A limitation with such a matrix is that for every key at least one keypad row line should be used.

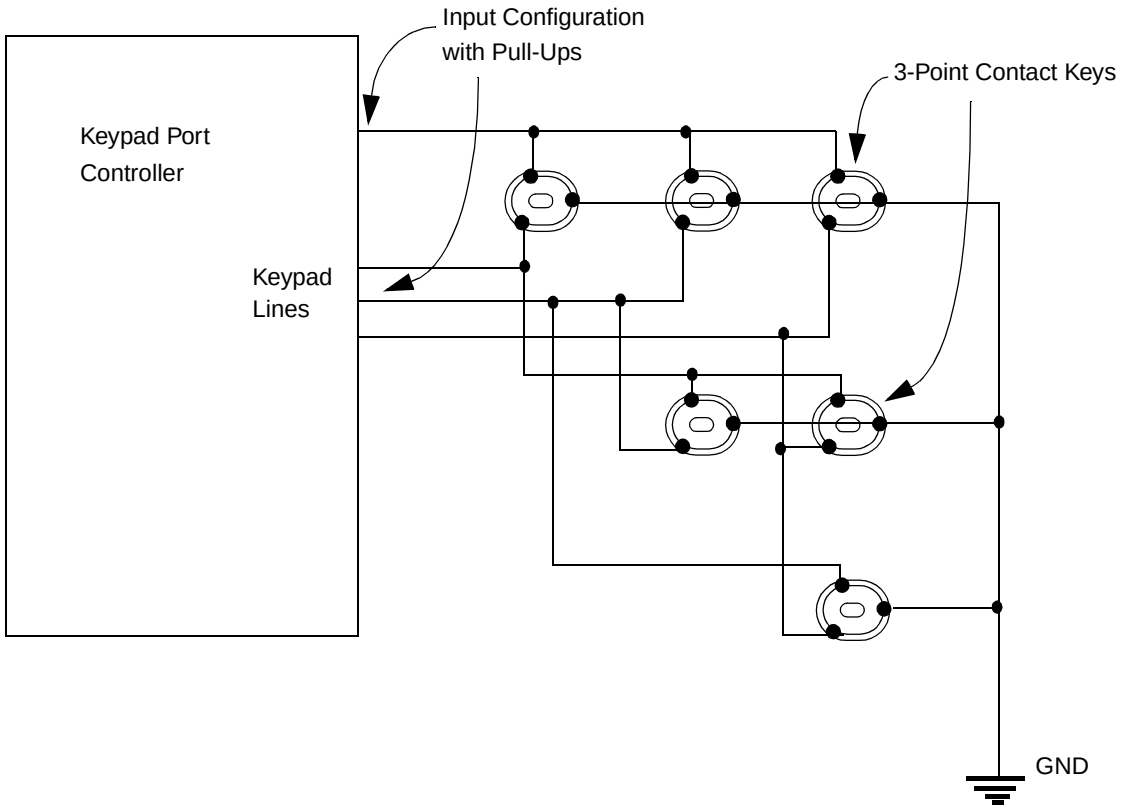


Figure 27-12. KPP Interface with 3-Point Contact Key Matrix (Simplified View)

27.5 Initialization/Application Information

This section provides initialization and application information.

27.5.1 Typical Keypad Configuration and Scanning Sequence

Perform the following steps to configure the keypad:

1. Enable the number of rows in the keypad (KPCR[7:0]).
2. Write 0s to KPDR[15:8].
3. Configure the keypad columns as open-drain (KPCR[15:8]).
4. Configure columns as output and rows as input (KDDR[15:0]).
5. Clear the KPKD Status Flag and Synchronizer chain.
6. Set the KDIE control bit, and clear the KRIE control bit (avoid false release events).

(The system is now in standby mode, and awaiting a key press.)

27.5.2 Key Press Interrupt Scanning Sequence

Perform the following steps to perform a keypad scanning routine:

1. Disable both (depress and release) keypad interrupts.

2. Write 1s to KPDR[15:8], setting column data to 1s.
3. Configure columns as totem pole outputs (for quick discharging of keypad capacitance).
4. Configure columns as open-drain.
5. Write a single column to 0, and other columns to 1.
6. Sample row inputs and save data. Multiple key presses can be detected on a single column.
7. Repeat Steps 2–6 for remaining columns.
8. Return all columns to 0 in preparation for standby mode.
9. Clear KPKD and KPKR status bit(s) by writing to a “1”; set the KPKR synchronizer chain by writing a “1” to the KRSS register; and clear the KPKD synchronizer chain by writing a “1” to the KDSC register.
10. Re-enable the appropriate keypad interrupt(s) so that the KDIE detects a key hold condition, or the KRIE detects a key-release event.

27.5.3 Additional Comments

The order of key press detection can be done in software only. Therefore, the software may need to run the scan routines at very short intervals of time per the application’s demands.

For the keys that require a very precise order (such as game keys), individual GPIO pins may be more useful.

Chapter 28
Memory Stick Host Controller (MSHC)

The Memory Stick Host Controller (MSHC) is placed between the AIPS and the Sony Memory Stick to support data transfer from an applications processor to the memory stick.

28.1 Overview

Figure 28-1 shows the MSHC top-level block diagram with input and output signals.

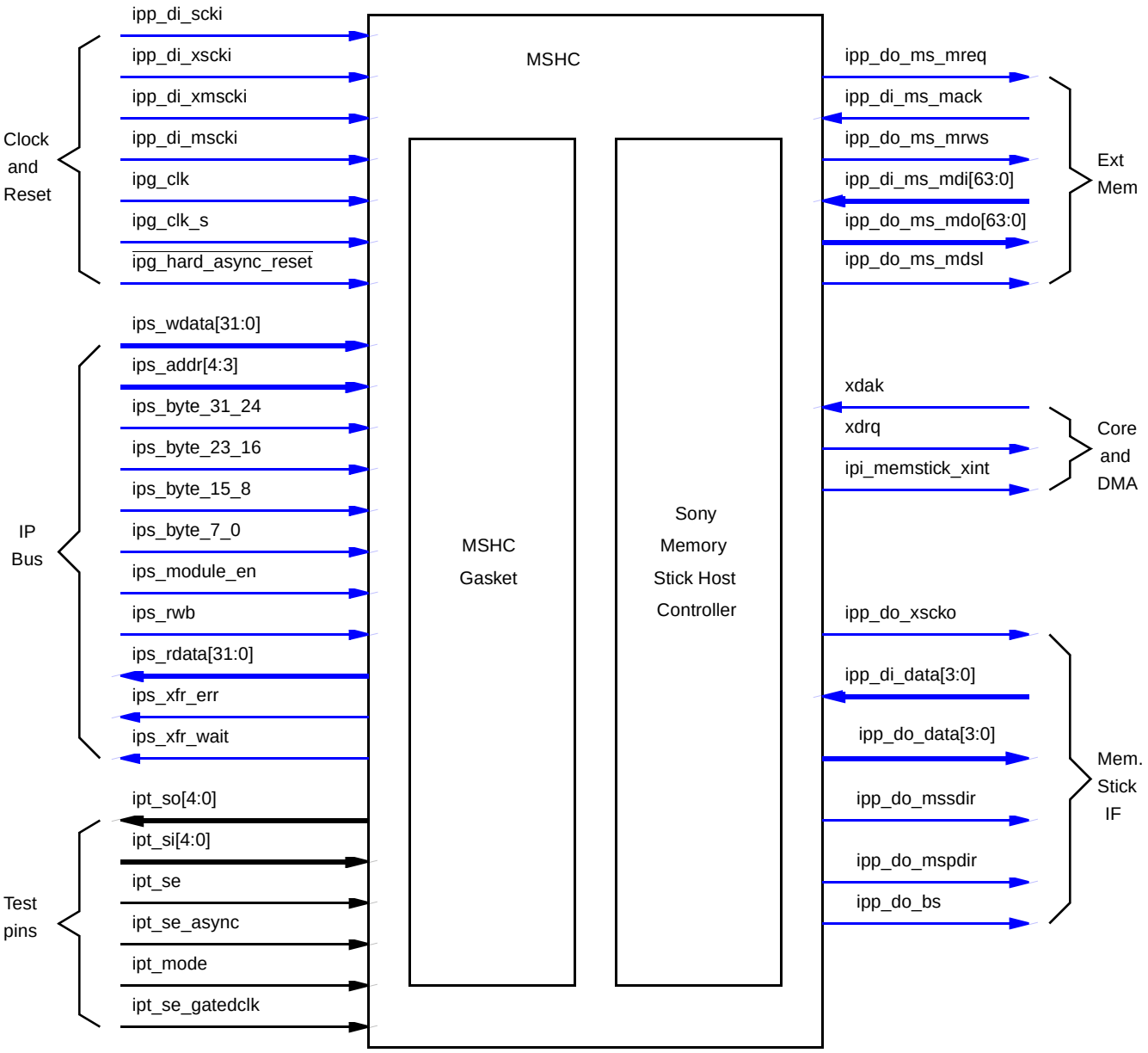


Figure 28-1. Applications Processor Memory Stick Controller Block Diagram

28.1.1 Overview

The memory stick host controller consists of two submodules:

- MSHC gasket
- Sony Memory Stick Host Controller (SMSC)

The SMSC module, which is the actual memory stick host controller, is compatible with Sony Memory Stick Standard Version 1.3 and Memory Stick PRO. The gasket connects the AIPS IP bus to the SMSC interface to allow SkyBlue transfers.

This document describes the MSHC gasket module in detail. All details regarding the SMSC module can be found separately in Sony's "Memory Stick PRO Host Controller IP, Version 1.3."

28.1.2 Features

The MSHC includes the following features:

- A gasket between the IP SkyBlue bus and the SMSC
  - IP SkyBlue interface transfer functionality as slave
  - Three internal registers (timeout, interrupt status/clear, and interrupt enable)
  - Gasket interrupt for transfer errors or wait timeout
  - Timeout function for abnormal transfer wait states
  - Fixed 32-bit data bus
  - Little Endian to IP data bus and Big Endian to SMSC data bus
- SMSC capability to communicate to the Sony Memory Stick
  - Four internal registers structured in 64-bit format
  - FIFO (4 x 64-bit)
  - Interrupt after Memory Stick communication completes
  - DMA in dual address mode

NOTE

SMSC supports single address mode as well, but the applications processor does not use this mode.

- Test mode and DFT implementation

28.1.3 Modes of Operation

The MSHC gasket has a reduced SkyBlue interface and supports the IP bus read/write transfers that include a back-to-back read or write. DMA transfers also take place using the SkyBlue interface.

A transfer can be initiated by the SDMA or the host (through AIPS) in response to an MSHC DMA request or interrupt. The SMSC has two DMA address modes—single and dual.

The MSHC is set to dual address mode for transfers with the SDMA. In dual address mode, when the MSHC requests a transfer with the DMA request (XDRQ), the SDMA initiates a transfer to the MSHC.

The MSHC still has the external memory ports and the DMA acknowledge input (XDAK), even though the single address mode is not used in the applications processors.

28.2 Memory Map and Register Definition

The MSHC has seven internal registers; four registers in the SMSC and three registers in the gasket.

The gasket internal registers do not require any extra addresses, as they share the address space for the SMSC internal registers. IP bus accesses are 32-bit, but all SMSC registers (except the data register) use only the upper 16 bits, so the gasket can utilize the lower eight bits.

The MSHC internal registers are mapped to 64-bit structure, so the required address signal is [4:3]. Due to this address sharing scheme, the host must keep the SMSC register values as necessary while updating the gasket internal registers.

28.2.1 Memory Map

Table 28-1 shows the Memory Map for the MSHC module.

Table 28-1. MSHC Memory Map

Address	Register	Access	Reset Value	Section/Page
0x5002_4000 (TIMEOUT1)	SMSC Timeout Register 1	R/W	0x00	28.2.3.2.1/28-6
0x5002_8000 (TIMEOUT2)	SMSC Timeout Register 2	R/W	0x00	28.2.3.2.1/28-6
0x5002_4014 (INTERRUPT_STATUS1)	Gasket Interrupt Status/ Clear Register 1	R/W	0x— —	28.2.3.2.2/28-7
0x5002_8014 (INTERRUPT_STATUS2)	Gasket Interrupt Status/ Clear Register 2	R/W	0x— —	28.2.3.2.2/28-7
0x5002_401C (INTERRUPT_ENABLE1)	Gasket Interrupt Enable Register 1	R/W	0x— —	28.2.3.2.3/28-8
0x5002_801C (INTERRUPT_ENABLE2)	Gasket Interrupt Enable Register 2	R/W	0x— —	28.2.3.2.3/28-8

28.2.2 Register Summary

Figure 28-2 shows the key to the register fields, and Table 28-2 shows the register figure conventions.

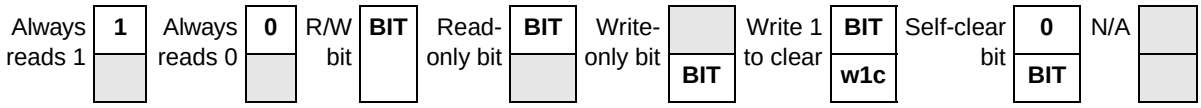


Figure 28-2. Key to Register Fields

Table 28-2. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).

Table 28-2. Register Figure Conventions (continued)

Convention	Description
rwm	A read/write bit modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

See Table 28-3 for summary of all registers in the MSHC module.

Table 28-3. MSHC Register Summary

Name		7	6	5	4	3	2	1	0
0x5002_4000 (TIMEOUT1) 0x5002_8000 (TIMEOUT2)	R	TOVW[7:0]							
	W								
0x5002_4014 (INTERRUPT_STATUS1)0x5002_8014 (INTERRUPT_STATUS2)	R	IDA	IXFR	0	0	WFUL	REMPL	0	0
	W								
0x5002_401C (INTERRUPT_ENABLE1) 0x5002_801C (INTERRUPT_ENABLE2)	R	INTEN_IDA	INTEN_IXFR	0	0	INTEN_WFUL	INTEN_REMPL	0	0
	W								

28.2.3 Register Descriptions

The MSHC module uses Big Endian with 64-bit data while the applications processors use the Little Endian with 32-bit data. Therefore, the data must be assigned per Table 28-4.

Table 28-4. MSHC Data Endianness and Connection to IP Bus

Bus	Endian	Connection							
MSHC data	Big	[63:56]	[55:48]	[47:40]	[39:32]	[31:24]	[23:16]	[15:8]	[7:0]
IP bus data	Little	[7:0]	[15:8]	[23:16]	[31:24]	—	—	—	—

28.2.3.1 SMSC Registers

The SMSC command register contains the MS transfer protocol command (TPC) and transfer data size. It must be set by either the host or SDMA before starting a transfer.

The SMSC data register is used to access the SMSC internal FIFO, which is of size 4 x 64-bit. It takes eight 32-bit transfers to fill or empty the FIFO.

The SMSC status register reflects SMSC status, such as FIFO status, ready flag for transfers, and CRC error flag. The host needs to read this register to start and manage the transfer. This register is readable only through the IP bus.

The SMSC system register has a user command for SMSC modes and options that are required for transfer and communication with the MS. This register must be set before starting a transfer and should not be changed during the communication with the MS.

More details can be found in Sony's "Memory Stick PRO Host Controller IP, Version 1.3."

28.2.3.2 Gasket Registers

28.2.3.2.1 Gasket Timeout Register

This register is readable or writable. The value after synchronous or asynchronous reset is 0. This register gives a mechanism a time-out if long wait states are encountered.

Once the FIFO becomes full while the IP bus has more data to be written, the gasket asserts the wait signal until the FIFO space is available. The gasket also starts to decrement an internal counter from the timeout value for wait, TOVW[7:0]. When the counter reaches zero, the ips_xfr_wait signal is deasserted and an interrupt is generated. If the wait state is cleared before the timeout counter reaches zero, the suspended transfer is resumed.

When the TOVW[7:0] is set to zero, the gasket does not produce wait states if the FIFO is full/empty, but generates an interrupt.

If the gasket interrupt enable bits—INTEN_WFUL and INTEN_REMP—are disabled, the timeout counter has no functionality. (It is effectively set to infinity.) Once the wait state is entered, the MSHC stays in wait until either the AIPS disables the module (using ips_module_en) or the FIFO becomes available for another transfer.

Figure 28-3 shows the Gasket Timeout register, and Table 28-5 shows the register's field descriptions.

0x5002_4000 (TIMEOUT1)
0x5002_8000 (TIMEOUT2)

Access: User read/write

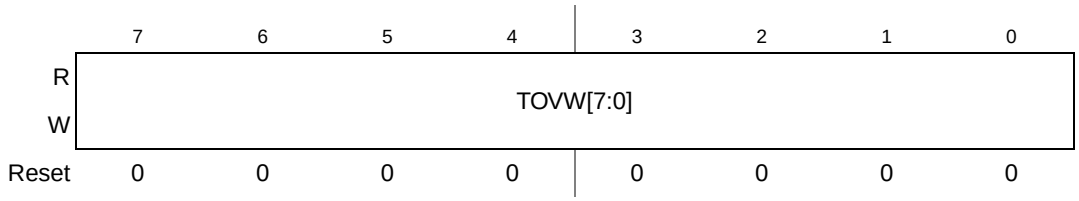


Figure 28-3. Timeout Register

Table 28-5. Timeout Register Field Descriptions

Field	Description
7–0 TOVW[47–40]	Timeout value for wait. When these bits are set to “0”, the gasket does not produce wait states if the FIFO is full/empty, but generates an interrupt.

28.2.3.2.2 Gasket Interrupt Status/Clear Register

This register is readable or writable, and is reset to ‘0’ by asynchronous or synchronous reset. Bits should be written as ‘1’ to clear an interrupt. The unused bits are read as ‘0’. The SMSC status register, which has the same address as this register, is a read-only register.

The interrupt status/clear register reflects the transfer status. It is used for the gasket to decide if it supports a current transfer. It can also be used for the gasket interrupt generation.

The gasket provides four different interrupts, which are all directly related to the IP bus transfer. Once an interrupts occurs, the host needs to check this register to determine which interrupt occurred and also to clear the interrupt (by writing ‘1’ to the register bit).

Figure 28-4 shows the Gasket Interrupt Status/Clear register, and Table 28-6 shows the register’s field descriptions.

0x5002_4014 (INTERRUPT_STATUS1)
0x5002_8014 (INTERRUPT_STATUS2)

Access: User read/write

	7	6	5	4	3	2	1	0
R	IDA	IXFR	0	0	WFUL	REMPL	0	0
W								
Reset	0	0	—	—	0	0	—	—

Figure 28-4. Gasket Interrupt Status/Clear Register

Table 28-6. Gasket Interrupt Status/Clear Register Field Descriptions

Field	Description
7 IDA	Illegal data access. This bit is asserted if a transfer is not 32-bit, that is, if the ips_byte inputs are not all asserted. It is always readable and can be cleared (if the interrupt enable bit INTEN_IDA is enabled) by writing to this register with ips_wdata[23] set to ‘1’. Any illegal data access always asserts IDA, but does not generate a gasket interrupt if the interrupt enable bit INTEN_IDA is disabled.
6 IXFR	Illegal transfer. This bit is asserted if a data transfer direction is not consistent with the SMSC FIFO data direction. It is always readable and can be cleared (if the interrupt enable bit INTEN_IXFR is enabled) by writing to this register with ips_wdata[22] set to ‘1’. Any illegal data transfer always asserts IXFR, but does not generate a gasket interrupt if the interrupt enable bit INTEN_IXFR is disabled.
5–4 Uncommitted	Reserved

Table 28-6. Gasket Interrupt Status/Clear Register Field Descriptions (continued)

Field	Description
3 WFUL	Write to FIFO when full. This bit is asserted if the SMSC FIFO is full and the current write transfer is not finished. It is always readable and can be cleared (if the interrupt enable bit INTEN_WFUL is set) by writing to this register with ips_wdata[19] set to ‘1’. The gasket asserts WFUL when a data write transfer is attempted while the FIFO is full. The update time is different depending on whether INTEN_WFUL is enabled and the value of TOVW[7:0]. If INTEN_WFUL is disabled, WFUL is updated without the timeout delay. The gasket uses this bit to generate a transfer wait state as well as the interrupt.
2 REMP	Read from FIFO when empty. This bit is asserted if the SMSC FIFO is empty and the current read transfer is not finished. It is always readable and can be cleared (if the interrupt enable bit INTEN_REMP is set) by writing to this register with ips_wdata[18] set to ‘1’.The gasket asserts REMP when a normal data read transfer is attempted while the FIFO is empty. The version number read after a reset also asserts REMP as the FIFO is empty at this time. The update time is different, depending on whether INTEN_REMP is enabled and the value of TOVW[7:0]. If INTEN_REMP is disabled, REMP is updated without the timeout delay. The gasket uses this bit to generate a transfer wait state as well as the interrupt.
1–0 Uncommitted	Reserved

28.2.3.2.3 Gasket Interrupt Enable Register

This register is readable or writable, and is reset to ‘0’ by asynchronous or synchronous reset.

This register is used to enable the gasket interrupts. The gasket interrupts are disabled by the default. To enable the gasket interrupt, the relevant bit should be set to ‘1’.

Figure 28-5 shows the Gasket Interrupt Enable register, and Table 28-7 shows the register’s field descriptions.

0x5002_401C (INTERRUPT_ENABLE1)
0x5002_801C (INTERRUPT_ENABLE2)

Access: User read/write

	7	6	5	4	3	2	1	0
R	INTEN_IDA	INTEN_IXFR	0	0	INTEN_WFUL	INTEN_REMPL	0	0
W								
Reset	0	0	—	—	0	0	—	—

Figure 28-5. Gasket Interrupt Enable Register

Table 28-7. Gasket Interrupt Enable Register Field Descriptions

Field	Description
7 INTEN_IDA	Illegal data access interrupt enable bit. If it is set to ‘1’, it enables the gasket interrupt for an illegal data access.
6 INTEN_IXFR	Illegal transfer interrupt enable bit. If it is set to ‘1’, it enables the gasket interrupt for an illegal data transfer.
5–4 Uncommitted	Reserved
3 INTEN_WFUL	Interrupt enable bit for a write transfer to FIFO with full. If it is set to ‘1’, it enables the gasket interrupt and the wait timeout function.
2 INTEN_REMP	Interrupt enable bit for a read transfer from FIFO with empty. If it is set to ‘1’, it enables the gasket interrupt and the wait timeout function.
1–0 Uncommitted	Reserved

28.3 Functional Description

28.3.1 Sony Memory Stick Controller (SMSC)

The details of the SMSC functionality can be found in Sony’s “Memory Stick PRO Host Controller IP, Version 1.3.”

28.3.2 MSHC Gasket

This section describes MSHC clocks, MS interface, the gasket state-machine, IP bus error, wait conditions, the gasket interrupt, and IP bus transfer.

28.3.2.1 Resetting and Clocking

The MSHC uses one asynchronous reset and one synchronous reset to reset gasket internal registers.

The asynchronous reset is the Greenline hardware asynchronous reset that is active low while the synchronous reset is a soft reset from the MSHC system register RST bit that is active high. Once the soft reset is asserted, the SMSC automatically clears the RST bit in the system register after initializing all internal registers. During the soft reset period, the behavior of any IP bus transactions is undefined.

MSHC has several clocks, as shown in Figure 28-6.

The MSHC has three clock inputs—ipg_clk_s directly connected to the gasket, and the others to SMSC through muxing logic. The MSHC also has one clock output that is inverted in the gasket.

The muxing logic allows the MSHC to have only two clock domains during test mode—ipg_clk_s in the gasket and HCKI in the SMSC are connected to one test clock, and all other clocks (SCKI, XSCKI, MSCKI, and XMSCKI) are connected to a different test clock. The test clocks are generated inside the CCM.

In normal operation, XSCKI is an inverted version of SCKI and MSCKI is an inverted version of XMSCKI.

The gasket uses the ipg_clk_s for all logic that is synchronous to the SkyBlue interface signals. This clock is active only when the gasket module enable signal is asserted.

HCKI in the SMSC is the main clock for most of the internal registers and the FIFO. The MS has a slower clock speed and is asynchronous to HCKI, so XSCKO is generated by the SMSC and is output to provide the MS clock.

All the logic shown in Figure 28-6 is implemented in the gasket.

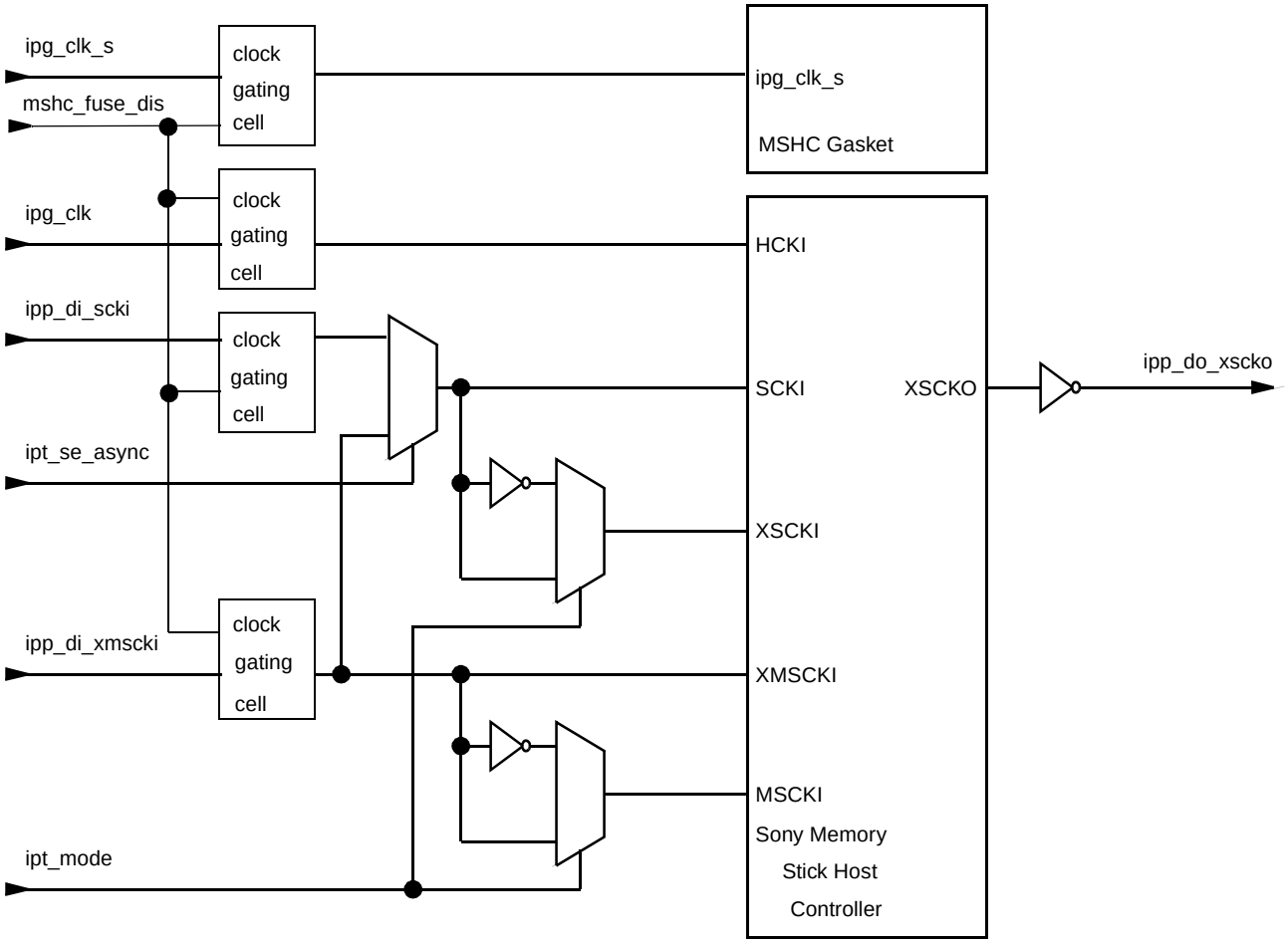


Figure 28-6. MSHC Clock Structure

28.3.2.2 Memory Stick Interface

The gasket includes logic to generate the I/O signals for the MS. This logic is included in the gasket so that it does not need to be provided at the chip level. Figure 28-7 shows multiplex logic and signal connections included in the gasket.

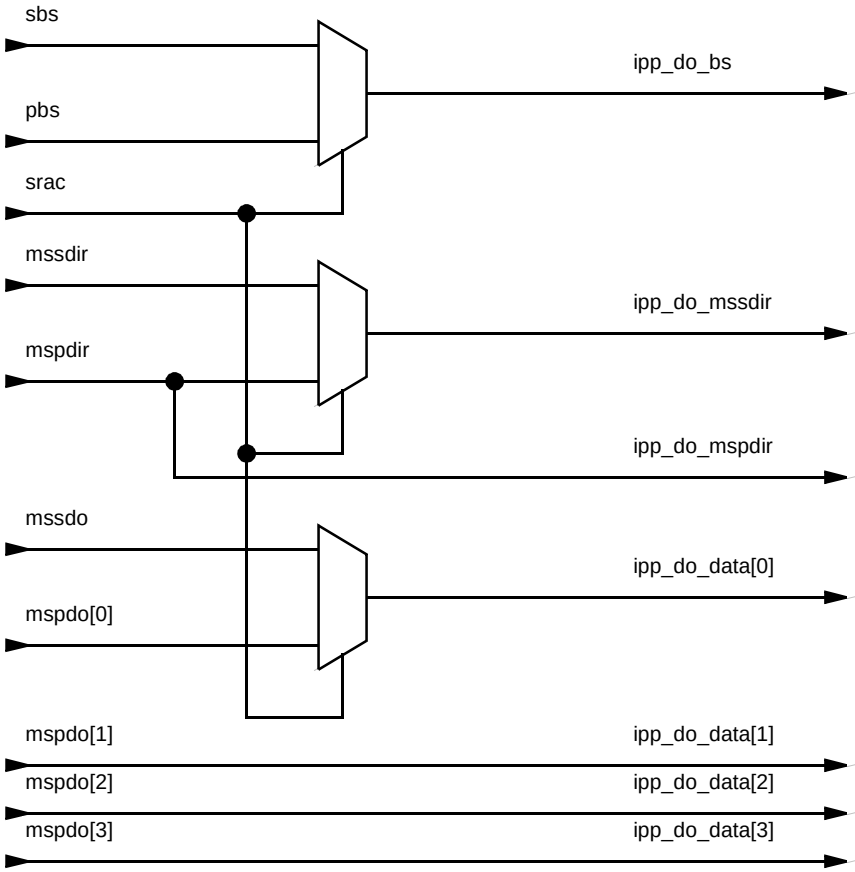


Figure 28-7. Gasket Memory Interface Logic

Chapter 29
Secured Digital Host Controller (SDHC)

The Security Digital Host Controller (SDHC1 and SDHC2) integrates both MultiMediaCard (MMC) support along with Secure Digital (SD) memory and I/O functions, including SD memory and I/O combo card.

The MultiMediaCard (MMC) is a universal low-cost data storage and communication medium that is designed to cover a wide area of applications, such as electronic toys, organizers, PDAs, and smart phones. The MMC communication is based on an advanced 7-pin serial bus designed to operate in a low-voltage range.

The Secure Digital Card (SD), is an evolution of MMC technology, with two additional pins in the form factor. It is specifically designed to meet the security, capacity, performance, and environment requirements inherent in newly emerging audio and video consumer electronic devices. The physical form factor, pin assignment, and data transfer protocol are forward-compatible with the MultiMediaCard with some additions. Under SD protocol, it can be categorized into Memory card, I/O card, and Combo card (has both memory and I/O functionality).

The memory card invokes a copyright-protection mechanism that complies with the security of the SDMI standard, is faster, and provides the capability for a higher memory capacity. The I/O card provides high-speed data I/O with low-power consumption for mobile electronic devices.

Figure 29-1 shows the SDHC block diagram.

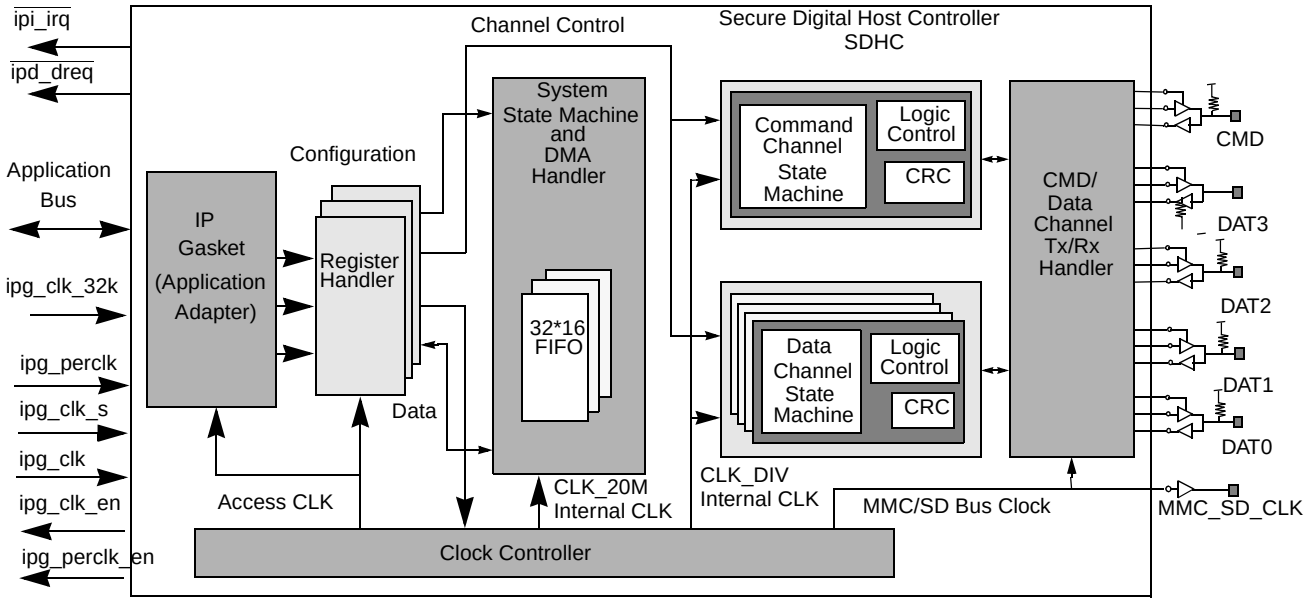


Figure 29-1. Secure Digital Host Controller Block Diagram

Figure 29-2 shows the system interconnection with the SDHC module.

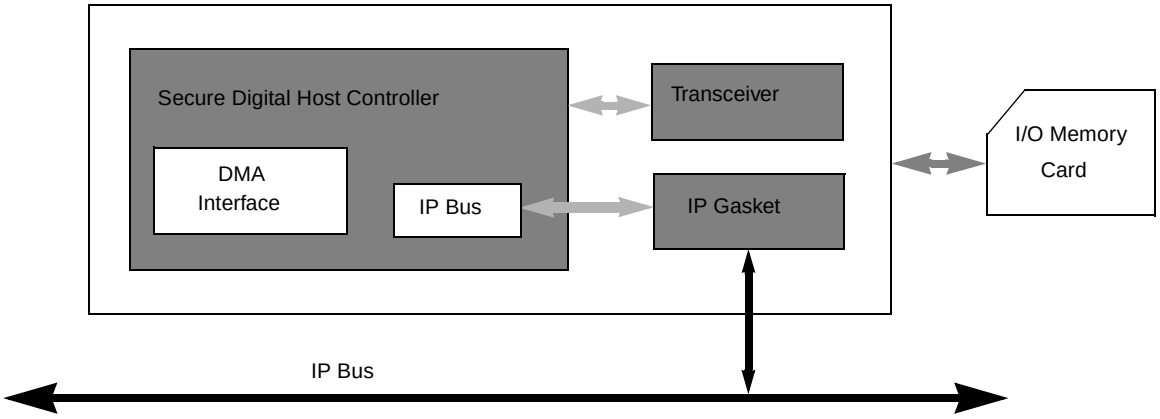


Figure 29-2. System Interconnection with the Secure Digital Host Controller

29.1 Overview

The SDHC controls the MMC, SD memory, and I/O cards by sending commands to cards and performing data accesses to/from the cards. Refer to Section 29.4, "Functional Description" for a detailed description.

29.1.1 Features

The features of the Secure Digital Host Controller module include the following:

- Full compatibility with the MMC system specification version 3.2

- Compatibility with the SD Memory Card specification 1.01, and SD I/O specification 1.1 with 1/4 channel(s)
- 100 Mbps maximum data rate in 4-bit mode, SD bus clock up to 25 MHz
- Built-in programmable frequency counter for SDHC bus
- Maskable hardware interrupt for SDIO Interrupt, Internal status, and FIFO status
- Built-in dual 16 x 32-bit data FIFO buffer
- Plug and play (PnP) support
- Single/multi-block access to the card, including erase operation
- Multi-SD function support, including multiple I/O and combined I/O and memory
- Up to seven I/O functions, plus one memory supported on single SD I/O card (combo card)
- IRQ-supported enable card to interrupt host
- Support of SDIO interrupt detection during 1/4-bit access
- Support of SDIO Read/Wait and suspend/resume operation
- Controller core—Freescale Semiconductor IP bus compatible
- Block-based data transfer between MMC card and SDHC (stream mode not supported)
- Block length of data transfer capability between host and card of approximately 1 to 2048 bytes

29.2 External Signal Description

The SDHC has a six-chip I/O. The MMC_SD_CLK is an internally generated clock used by the MMC/SD card. The command (CMD) I/O is used to send commands and receive responses from the card. Four data lines, DAT3–DAT0, are used to perform data transfers between the host controller and the card.

Table 29-1 shows the signal properties of the I/Os.

Table 29-1. Signal Properties

Name	Port	Function	Reset State	Pull Up
MMC_SD_CLK	O	Clock for MMC/SD/SDIO card	0	
CMD	I/O	CMD line connect to card	1	Pull-Up
DAT3	I/O	Card detect in power up Data line in 4-bit mode Not used in 1-bit mode	0	Pull down DAT3 if need card detection through this bit; otherwise, pull up.
DAT2	I/O	Data line or read wait in 4-bit mode Read wait in 1-bit mode	1	Pull-Up
DAT1	I/O	Data line or interrupt in 4-bit mode Interrupt in 1-bit mode	1	Pull-Up
DAT0	I/O	Data line both in 1-bit and 4-bit mode	1	Pull-Up

29.3 Memory Map and Register Definition

SDHC contains 16 32-bit registers. Section 29.3.3, "Register Descriptions" provides the detailed descriptions for all of the SDHC registers. All the registers can be accessed only in 32-bit sizes. Byte/halfword access is not allowed.

29.3.1 Memory Map

Table 29-2 shows the SDHC memory map. The SDHC memory map space is 4 Kbytes. Only address offsets from 0 x 00 to 0 x 44 are implemented. The address space above the offset of 0 x 44 is reserved. For write access to the reserved address region, the access is ignored by SDHC. For read access to the reserved address region, 0 x 0 is returned on the IP bus. The user should not access the reserved region to ensure compatibility with possible future revisions of this module.

Table 29-2. SDHC Memory Map

Address	Register	Access	Reset Value	Section/Page
0x5000_4000 (STR_STP_CLK) 0x5000_8000 (STR_STP_CLK)	SDHC Clock Control	R/W	0x0000_0000	29.3.3.1/29-8
0x5000_4004 (STATUS) 0x5000_8004 (STATUS)	SDHC Status	R	0x3000_0000	29.3.3.2/29-10
0x5000_4008 (CLK_RATE) 0x5000_8008 (CLK_RATE)	SDHC Card Clock Rate	R/W	0x0000_0008	29.3.3.3/29-14
0x5000_400C (CMD_DAT_CONT) 0x5000_800C (CMD_DAT_CONT)	SDHC Command Data Control	R/W	0x0000_0000	29.3.3.4/29-15
0x5000_4010 (RES_TO) 0x5000_8010 (RES_TO)	SDHC Response Time-out	R/W	0x0000_0040	29.3.3.5/29-17
0x5000_4014 (READ_TO) 0x5000_8014 (READ_TO)	SDHC Read Time-out	R/W	0x0000_FFFF	29.3.3.6/29-18
0x5000_4018 (BLK_LEN) 0x5000_8018 (BLK_LEN)	SDHC Block Length	R/W	0x0000_0000	29.3.3.7/29-19
0x5000_401C (NOB) 0x5000_801C (NOB)	SDHC Number of Block	R/W	0x0000_0000	29.3.3.8/29-20
0x5000_4020 (REV_NO) 0x5000_8020 (REV_NO)	SDHC Revision Number	R	0x0000_0400	29.3.3.9/29-21
0x5000_4024 (INT_CNTR) 0x5000_8024 (INT_CNTR)	SDHC Interrupt Control	R/W	0x0000_0000	29.3.3.10/29-22
0x5000_4028 (CMD) 0x5000_8028 (CMD)	SDHC Command Number	R/W	0x0000_0000	29.3.3.11/29-26
0x5000_402C (ARG) 0x5000_802C (ARG)	SDHC Command Argument	R/W	0x0000_0000	29.3.3.12/29-27

Table 29-2. SDHC Memory Map (continued)

Address	Register	Access	Reset Value	Section/Page
0x5000_4034 (RES_FIFO) 0x5000_8034 (RES_FIFO)	SDHC Command Response FIFO Access	R	0x0000_0000	29.3.3.13/29-28
0x5000_4038 (BUFFER_ACCESS) 0x5000_8038 (BUFFER_ACCESS)	SDHC Data Buffer Access	R/W	0x0000_0000	29.3.3.14/29-29
Note: The following addresses are reserved: 0x5000_4030 0x5000_403C				

29.3.2 Register Summary

Figure 29-3 shows the key to the register fields, and Table 29-3 shows the register figure conventions.

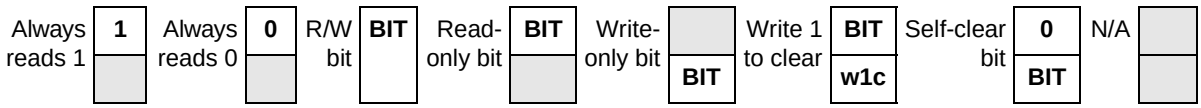


Figure 29-3. Key to Register Fields

Table 29-3. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 29-4 shows the SDHC register summary.

Table 29-4. SDHC Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x5000_4000 (STR_STP_CLK) 0x5000_8000 (STR_STP_CLK)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	slfclr	0	START_CLK	STOP_CLK
	W													SDHC Reset			
0x5000_4004 (STATUS) 0x5000_8004 (STATUS)	R	CARD_INSERTION	CARD_REMOVAL	YBUF_EMPTY	XBUF_EMPTY	YBUF_FULL	XBUF_FULL	BUF_UND_RUN	BUF_OVFL	0	0	0	0	0	0	0	0
	W	w1c	w1c														
	R	0	SDIO_INT_ACTIVE	END_CMD_RESP	WRITE_OP_DONE	READ_TRANS_DONE	WR_CRC_ERR_CODE		CARD_BUS_CLK_RUN	BUF_READ_RDY	BUF_WR_RDY	RESP_CRC_ERR	0	READ_CRC_ERR	WRITE_CRC_ERR	TIME_OUT_RESP	TIME_OUT_READ
	W		w1c	w1c	w1c	w1c	w1c	w1c				w1c		w1c	w1c	w1c	w1c
0x5000_4008 (CLK_RATE) 0x5000_8008 (CLK_RATE)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	CLK_PRESCALER[15:4]												CLK_DIVIDER[3:0]			
	W																
0x5000_400C (CMD_DAT_CONT) 0x5000_800C (CMD_DAT_CONT)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	CMD_RESUME	0	0	CMD_RESP_LONG_OFF	STOP_READWAIT	START_READWAIT	BUS_WIDTH		INIT	0	0	WRITE_READ	DATA_ENABLE	FORMAT_OF_RESPONSE		
	W																

Table 29-4. SDHC Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x5000_4010 (RES_TO) 0x5000_8010 (RES_TO)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	RESPONSE TIME OUT[7:0]							
	W																
0x5000_4014 (READ_TO) 0x5000_8014 (READ_TO)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	DATA_READ_TIME_OUT[15:0]															
	W																
0x5000_4018 (BLK_LEN) 0x5000_8018 (BLK_LEN)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	BLOCK LENGTH[11:0]											
	W																
0x5000_401C (NOB) 0x5000_801C (NOB)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	NOB[15:0]															
	W																
0x5000_4020 (REV_NO) 0x5000_8020 (REV_NO)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	Revision Number[15:0]															
	W																
0x5000_4024 (INT_CNTR) 0x5000_8024 (INT_CNTR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	SDIO_INT_WKP_EN	CARD_INSERT_WKP_EN	CARD_REMOVAL_WKP_EN
	W																
	R	CARD_INSERTION_EN	CARD_REMOVAL_EN	SDIO_IRQ_EN	DAT0_EN	0	0	0	0	0	0	0	BUF_READ_EN	BUF_WRITE_EN	END_CMD_RES	WRITE_OP_DONE	READ_OP_DONE
	W																

Table 29-4. SDHC Register Summary (continued)

Name		31 15	30 14	29 13	28 12	27 11	26 10	25 9	24 8	23 7	22 6	21 5	20 4	19 3	18 2	17 1	16 0
0x5000_4028 (CMD) 0x5000_8028 (CMD)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	COMMAND NUMBER					
	W																
0x5000_402C (ARG) 0x5000_802C (ARG)	R	ARG[31:16]															
	W																
	R	ARG[15:0]															
	W																
0x5000_4034 (RES_FIFO) 0x5000_8034 (RES_FIFO)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	RESPONSE_CONTENT[15:0]															
	W																
0x5000_4038 (BUFFER_ACCESS) 0x5000_8038 (BUFFER_ACCESS)	R	FIFO CONTENT[31:16]															
	W																
	R	FIFO CONTENT[15:0]															
	W																

29.3.3 Register Descriptions

Many of SDHC registers have reserved bits. Reserved bits identified in the registers are read as zero and writes to these bits are ignored. However, the user should write zeros to these bits to ensure compatibility with possible future revisions of this module.

29.3.3.1 SDHC Clock Control Register (STR_STP_CLK)

The SDHC clock control register allows the user to reset the whole module and to enable or disable the MMC_SD_CLK to card.

Figure 29-4 shows the STR_STP_CLK register, and Table 29-5 shows the register's field descriptions.

0x5000_4000 (STR_STP_CLK)
0x5000_8000 (STR_STP_CLK)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	slfclr	0	START_CLK	STOP_CLK
W													SDHC Reset			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 29-4. SDHC Clock Control Register

Table 29-5. SDHC Clock Control Register Field Descriptions

Field	Description
31–4	Reserved
3 SDHC Reset	SDHC reset. Writes to the SDHC reset bit trigger the reset logic inside the SDHC. Reads from this bit always return ‘0’. To reduce the power consumption, the clock to the reset logic in SDHC is off in normal operation. When there is one access to this register, the clock is enabled for one cycle. To complete the entire reset period, it needs at least 8 clock pulses to finish the reset cycle. To reset the SDHC module, it is recommended to write this register with value 0x0008, followed by 0x0009, and then 0x0001 eight times. See Section 29.5.2.2, “Reset” for details on software reset. 0 No effect 1 Reset the SDHC module.
2	Reserved
1 START_CLK	Start clock. Write a ‘1’ to this bit to start the MMC_SD_CLK clock. Setting a value of 11 on Bits [1:0] of this register is not allowed. **Note:** The SDHC bus clock does not start immediately after writing to this bit. Polling needs to be done on the status CARD_BUS_CLK_RUN bit to ensure the SDHC clock is running. Refer to Example 29-1 for procedures to start the SD bus clock. 0 No effect 1 Start MMC/SD clock.
0 STOP_CLK	Stop clock. Write a ‘1’ to this bit to stop the MMC_SD_CLK clock. The MMC_SD_CLK should not be stopped by software during a transmission period. Setting a value of 11 on Bits [1:0] of this register is not allowed. **Note:** A transmission period is defined as the time from when a card data or access related command is submitted to the end of the access operation. The SDHC bus clock does not stop immediately after writing to this bit. Polling needs to be done on the status CARD_BUS_CLK_RUN bit to ensure the SDHC clock is running. Refer to Example 29-1 for procedures to stop the SD bus clock. 0 No effect 1 Stop the MMC/SD clock.

29.3.3.2 SDHC Status Register (STATUS)

The read-only SDHC status register provides the programmer with information about the status of SDHC operations, application FIFO status, error conditions, and interrupt status.

There are eight interrupt status bits—bit-31 card insertion status bit, bit-30 card removal status bit, bit-14 SDIO card interrupt status bit, bit-13 end command and response status bit, bit-12 write operation done status bit, bit-11 read operation done status bit, bit-7 data buffer read ready status bit, and bit-6 data buffer write ready status bit. When the corresponding interrupt enable is enabled in SDHC interrupt control register for any of these interrupts, the SDHC generates an interrupt request to the CPU. The user needs to clear the appropriate status bit to clear the corresponding interrupt. The interrupt status bits are cleared by using a write '1' to clear operation except for the data buffer ready status bits, which can be cleared only by the read or write operation on the data buffer.

Figure 29-5 shows the STATUS register, and Table 29-6 shows the register's field descriptions.

0x5000_4004 (STATUS)
0x5000_8004 (STATUS)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CARD_INSERTION	CARD_REMOVAL	YBUF_EMPTY	XBUF_EMPTY	YBUF_FULL	XBUF_FULL	BUF_UND_RUN	BUF_OVFL	0	0	0	0	0	0	0	0
W	w1c	w1c														
Reset	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	SDIO_INT_ACTIVE	END_CMD_RESP	WRITE_OP_DONE	READ_TRANS_DONE	WR_CRC_ERR_CODE		CARD_BUS_CLK_RUN	BUF_READ_RDY	BUF_WR_RDY	RESP_CRC_ERR	0	READ_CRC_ERR	WRITE_CRC_ERR	TIME_OUT_RESP	TIME_OUT_READ
W		w1c	w1c	w1c	w1c	w1c	w1c				w1c		w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 29-5. SDHC Status Register

Table 29-6. SDHC Status Register Field Descriptions

Field	Description
31 CARD_INSERTION	Card insertion. When this bit is set, the SDHC detects a value transition on the SD_DAT[3:0] from 0111 to 1111. This can be used to detect if a card is inserted to the card socket based on SD_DAT[3] pull-up resistor of the card DAT3. When this bit is set, the SDHC generates an interrupt request if the card detection interrupt is enabled. This bit is read-only and can be cleared by writing a '1' to this bit. 0 No card insertion detected. 1 Card insertion detected based on logic level changed on SD_DAT[3].
30 CARD_REMOVAL	Card removal. When this bit is set, the SDHC detects a logic transition on the SD_DAT[3:0] from 1111 to b0111. This can be used to detect whether a card is removed from the card socket based on the SD_DAT[3] pull-up resistor of the card DAT3. When this bit is set, SDHC generates an interrupt request if the card removal interrupt is enabled. This bit is read-only and can be cleared by writing a '1' to it. The user needs to clear this bit to clear the interrupt request from SDHC when the card detection interrupt is enabled. 0 No card removal detected. 1 Card removal detected based on logic level changed on SD_DAT[3].
29 YBUF_EMPTY	Y data buffer empty. When this bit is set, it indicates that the Y data buffer is empty during a write transfer. This bit is automatically cleared when the first byte of data is moved into the FIFO. Refer to Section 29.4.1, "Data Buffers" for more details about the data buffers. 0 Y buffer is not empty. 1 Y buffer is empty.
28 XBUF_EMPTY	X data buffer empty. When this bit is set, it indicates that the X data buffer is empty during a write transfer. This bit is automatically cleared when the first byte of data is moved into the FIFO. Refer to Section 29.4.1, "Data Buffers" for more details about the data buffers. 0 X buffer is not empty. 1 X buffer is empty.
27 YBUF_FULL	Y data buffer full. When this is set, it indicates that the Y data buffer is full during a read transfer. This bit is automatically cleared when the last byte of data is read out from the FIFO. Refer to Section 29.4.1, "Data Buffers" for more details about the data buffers. 0 Y buffer is not full. 1 Y buffer is full.
26 XBUF_FULL	X data buffer full. When this is set, it indicates that the X data buffer is full during a read transfer. This bit is automatically cleared when the last byte of data is read out from the FIFO. Refer to Section 29.4.1, "Data Buffers" for more details about the data buffers. 0 X buffer is not full. 1 X buffer is full.
25 BUF_UND_RUN	Buffer underrun. When this is set, it indicates that both X and Y data buffers are empty during a write transfer. In this case, the card clock MMC_SD_CLK is stopped automatically by hardware to wait for the DMA or CPU to put data into the buffers. An interrupt is triggered if the corresponding interrupt control bit is enabled. 0 No buffer underrun 1 Buffer underrun during a write operation
24 BUF_OVFL	Buffer overflow. When this is set, it indicates that both data buffers are full during a read operation. In this case, the card clock MMC_SD_CLK is stopped automatically by hardware to wait for the DMA or CPU to remove data out of one of the buffers. An interrupt is triggered if the corresponding interrupt control bit is enabled. Excess data is ignored by the SDHC. 0 No buffer overflow 1 Buffer overflow during a read operation
23–15	Reserved

Table 29-6. SDHC Status Register Field Descriptions (continued)

Field	Description
14 SDIO_INT_ACTIVE	SDIO interrupt active. Indicates if an interrupt from the SDIO card has been detected. When this bit is set, the SDHC generates an interrupt request if the SDIO interrupt is enabled. The user should clear the status to clear the interrupt request. A separate acknowledge command to the card may be required to clear the source of the SDIO interrupt. Write a '1' to this bit to clear it. 0 No interrupt detected 1 Interrupt detected using SDIO card bus
13 END_CMD_RESP	End command response. Indicates that a command was successfully transmitted to the card and the corresponding response stored in the response FIFO. This occurs after each command operation. When this bit is set, the SDHC generates an interrupt request if the End_CMD_RESP interrupt is enabled. The user needs to clear this bit to negate the interrupt request. This bit is cleared by using a write '1' to clear operation. 0 Command not successful, incomplete, or not applicable (no response) 1 Command transmitted successfully (response received) **Note:** When this bit is set, the user also needs to check if the command send and response receive operation completed without error. The user also needs to check the RESP_CRC_ERR (Status[5]) and TIME_OUT_RESP(STATUS[1]) bits to determine if an error occurred.
12 WRITE_OP_DONE	Write operation done. Indicates that a write operation has completed. The FLASH card needs extra idle time for write accesses. This requires the SDHC module to wait until the card writes the buffered data to the inner FLASH memory. The SDHC module automatically detects the status. The WRITE_OP_DONE flag indicates the end of the write operation. When this bit is set, the pre-defined data bytes are written to the card. The user needs to send a STOP command to the card if the write command is a MMC/SD card write multi-block command. When this bit is set, SDHC generates an interrupt request if the WRITE_OP_DONE interrupt enable is enabled. The user needs to clear this bit to clear the interrupt by writing '1' to this bit. 0 Write operation is in progress or incomplete. 1 Write is operation complete. **Note:** When this bit is set, the user also needs to check if the write operation completed without a CRC error. The user needs to check the WR_CRC_ERR_CODE[1:0] (Status[10:9]) and WRITE_CRC_ERR (STATUS[2]) bits to determine if an error has occurred.
11 READ_OP_DONE	Read operation done. The READ_OP_DONE status bit is activated at the end of a read operation, which means all the data is received from the card. When this bit is set, the pre-defined data bytes are read from the card or a READ TIMEOUT occurs. The software needs to send a STOP command to the card if the read command is a MMC or SD card read multi-block command. This bit can be cleared by writing '1' to it. When this bit is set, SDHC generates an interrupt request if the READ_OP_DONE interrupt is enabled. The user needs to clear this bit to clear the interrupt request by writing '1' to this bit. 0 Read operation is in progress or incomplete. 1 Read operation is complete. **Note:** When this bit is set, the user also needs to check if the read operation complete without error. The user needs to check the READ_CRC_ERR (Status[3]) and TIME_OUT_READ(STATUS[0]) bits to determine if an error has occurred.

Table 29-6. SDHC Status Register Field Descriptions (continued)

Field	Description
10–9 WR_CRC_ERROR_CODE	Write CRC error code. Indicates the CRC results from the card at the end of write operations. After receiving a block of data, the card checks the CRC bit and sends the CRC status. These two bits reflect the CRC status of the recent written data. If the card returns a negative CRC status, the data is not written to the card. These two bits can be cleared by writing a value of '11' to them. 00 No transmission error. CRC Status is 010 (positive). 01 Transmission error. CRC Status is 101 (negative). 10 No CRC response 11 Reserved **Note:** These bits have valid value only when the WRITE_CRC_ERR status bit (STATUS[2]) is set.
8 CARD_BUS_CLK_RUN	Card bus clock run. Indicates if the MMC_SD_CLK clock to the card is running. The clock rate setting and system configuration can be modified when the clock is turned off by setting the STOP_CLK bit in STR_STP_CLK register. This bit can be cleared only by writing '1' to STOP_CLK bit in STR_STP_CLK register to stop MMC_SD_CLK. 0 MMC/SD clock is stopped. 1 MMC/SD clock is running. **Note:** Polling needs to be done on this bit to ensure the SDHC clock is running or stopped. Refer to Example 29-1 on page 29-44.
7 BUF_READ_READY	Buffer read ready. This status is set if a buffer (either X buffer or Y buffer) is full during read operations. An interrupt is triggered for non-DMA transfers if BUF_READ_EN is set, or if a DMA request is asserted for DMA transfers. **Note:** The read_op_done status bit is set once all the data is read from the card. At the end of read, the buffer read ready interrupt occurs the same time as the read done. The user can service the buffer read ready interrupt and then handle the read_op_done interrupt. 0 Not ready to read buffer 1 Ready to read buffer
6 BUF_WRITE_READY	Buffer write ready. This status is set if a buffer (either X buffer or Y buffer) is available during write operations. An interrupt is triggered for non-DMA transfers if the BUF_WRITE_EN bit is set, or if a DMA request is asserted for DMA transfers. This bit is set only when SDHC performs a write operation to the card. 0 Not ready to write buffer 1 Ready to write buffer
5 RESP_CRC_ERR	Response CRC error. Indicates that a transmission error occurred on the SD_CMD line during a response transfer. Write '1' to this bit to clear it. 0 No error 1 Response CRC error occurred
4	Reserved
3 READ_CRC_ERR	Read CRC error. Indicates that a transmission error occurred on the SD_DAT line during a card read operation. The user should re-try the transmission. Write '1' to this bit to clear it. 0 No error 1 CRC read error occurred.
2 WRITE_CRC_ERR	Write CRC error. Indicates that a transmission error occurred on the SD_DAT line during a card write operation. The user should check the WR_CRC_ERR_CODE field for more information about the CRC error. Write '1' to this bit to clear it. 0 No error 1 CRC write error occurred

Table 29-6. SDHC Status Register Field Descriptions (continued)

Field	Description
1 TIME_OUT_RESP	Time out response. Indicates that the command response was not received in the time specified in the RES_TO register. This can be caused by the following: • An unsupported command was received at the card(s). • Another MMC/SD_OP_COND command was submitted after all cards had already sent their voltage ranges and the power-up routine was finished. • An identification command was issued when all cards were already in standby state. • No card was on the bus. Write '1' to this bit to clear this condition. 0 No error 1 Time out response error occurred
0 TIME_OUT_READ	Time out read. Indicates that the expected data from the card was not received in the time specified in the READ_TO register. The TIME_OUT_READ is cleared by an internal status change or by removing the source of the error. Write '1' to this bit to clear it. 0 No error 1 Time out read data error occurred

29.3.3.3 SDHC Clock Rate Register (CLK_RATE)

Refer to Section 29.4.8, "System Clock Controller" for the clock scheme.

The high frequency input clock—IPG_PERCLK—is used to derive the low frequency clock to be used by the card and some of its internal logic. The divide circuitry consists of a 4-bit divider followed by a 12-bit prescaler. The IPG_PERCLK is first divided by the 4-bit divider to derive the signal, CLK_DIV. CLK_DIV is then divided by the 12-bit prescaler to derive CLK_20M, which is the source clock to be gated for internal logic and external cards.

CLK_20M is used internally by the SDHC. The MMC_SD_CLK is a buffered and gated version of the CLK_20M clock.

Figure 29-6 shows the CLK_RATE register, and Table 29-7 shows the register's field descriptions.

0x5000_4008 (CLK_RATE)
0x5000_8008 (CLK_RATE)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CLK_PRESCALER[15:4]												CLK_DIVIDER[3:0]			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Figure 29-6. SDHC Clock Rate Register

Table 29-7. SDHC Clock Rate Register Field Descriptions

Field	Description
31–16	Reserved
15–4 CLK_PRESCALER	Clock Prescaler. Specifies the divider value to generate CLK_20M from CLK_DIV. 0x000 CLK_20M is CLK_DIV 0x001 CLK_20M is CLK_DIV/2 0x002 CLK_20M is CLK_DIV/4 0x004 CLK_20M is CLK_DIV/8 0x008 CLK_20M is CLK_DIV/16 0x010 CLK_20M is CLK_DIV/32 0x020 CLK_20M is CLK_DIV/64 0x040 CLK_20M is CLK_DIV/128 0x080 CLK_20M is CLK_DIV/256 0x100 CLK_20M is CLK_DIV/512 0x200 CLK_20M is CLK_DIV/1024 0x400 CLK_20M is CLK_DIV/2048 0x800 CLK_20M is CLK_DIV/4096 Others Reserved
3–0 CLK_DIVIDER	Clock Divider. Specifies the divider value to generate CLK_DIV from input clock IPG_PERCLK. 0x1 CLK_DIV is IPG_PERCLK/2 0x2 CLK_DIV is IPG_PERCLK/3 0x3 CLK_DIV is IPG_PERCLK/4 0x4 CLK_DIV is IPG_PERCLK/5 0x5 CLK_DIV is IPG_PERCLK/6 0x6 CLK_DIV is IPG_PERCLK/7 0x7 CLK_DIV is IPG_PERCLK/8 0x8 CLK_DIV is IPG_PERCLK/9 0x9 CLK_DIV is IPG_PERCLK/10 0xa CLK_DIV is IPG_PERCLK/11 0xb CLK_DIV is IPG_PERCLK/12 0xc CLK_DIV is IPG_PERCLK/13 0xd CLK_DIV is IPG_PERCLK/14 0xe CLK_DIV is IPG_PERCLK/15 0xf CLK_DIV is IPG_PERCLK/16 Others Reserved

NOTE

The maximum frequency of MMC_SD_CLK is IPG_PERCLK/2.

29.3.3.4 SDHC Command and Data Control Register (CMD_DAT_CONT)

The SDHC command and data control register allows the user to specify the format of data and response, and to control the read/wait cycle. After configuring this register, enabling the MMC_SD_CLK causes the command and argument configured in the CMD number register and the CMD argument register to be sent out to the card.

Figure 29-7 shows the CMD_DAT_CONT register, and Table 29-8 shows the register's field descriptions.

0x5000_400C (CMD_DAT_CONT)
0x5000_800C (CMD_DAT_CONT)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMD_RESUME	0	0	CMD_RESP_LONG_OFF	STOP_READWAIT	START_READWAIT	BUS_WIDTH		INIT	0	0	WRITE_READ	DATA_ENABLE	FORMAT_OF_RESPONSE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 29-7. SDHC Command and Data Control Register

Table 29-8. SDHC Command and Data Control Register Field Descriptions

Field	Description
31–16	Reserved
15 CMD_RESUME	Command resume. Used to restore Command and Data Control Register after Read/Wait cycle for SDIO card. 0 Issues command to card. 1 Does not issue command to card.
14–13	Reserved
12 CMD_RESP_LONG_OFF	Command response long off. Allows STATUS[13] END_CMD_RESP bit to be self cleared when the condition to generate this bit disappears. This is utilized in the Read/Wait cycle. For SD/MMC operation, the user should keep this bit set as ‘0’. 0 Bit was not cleared when read. 1 Allows bit to be cleared.
11 STOP_READWAIT	Stop read/wait. Ends the read/wait cycle for SDIO. When this bit is set, the SDHC does not drive DAT2 output low so that the SDIO card would end the Read/Wait cycle. For operation of SD/MMC, the user should keep this bit set as ‘0’. 0 No effect 1 Ends read/wait cycle.
10 START_READWAIT	Start read/wait. Starts the read/wait cycle for SDIO. When this bit is set, the SDHC makes the DAT2 output low and forces the SDIO card to enter READWAIT cycle. For SD/MMC operation, the user should keep this bit set as ‘0’. 0 No effect 1 Starts read/wait cycle.
9–8 BUS_WIDTH	Bus width. Specifies the width of the data bus. These two bits must be set according to current SD/SDIO card bit mode. For MMC card, only 1-bit bus mode is supported. 00 1-bit 01 Reserved 10 4-bit 11 Reserved

Table 29-8. SDHC Command and Data Control Register Field Descriptions (continued)

Field	Description
7 INIT	Initialize. Specifies if the additional 80-clock cycle prefix (to initialize the card) occurs before every command. INIT enables/disables the additional 80-clock initialization time. 0 Disable 80 initialization clocks. 1 Enable 80 initialization clocks.
6–5	Reserved
4 WRITE_READ	Write/read. Specifies whether the data transfer of the current command is a write or read operation. 0 Read 1 Write
3 DATA_ENABLE	Data enable. Specifies if the current command includes a data transfer. 0 No data transfer included. 1 Date transfer included.
2–0 FORMAT_OF_RESPONSE	Format of response. Sets the expected response format for current command. Refer to the SD I/O Specification 1.0 for detail information of the response format. 000 No response for current command 001 48-bit response with CRC7 check. For example: Format R1/R1b/R5/R6. 010 136-bit, CSD/CID read response. For example: Format R2. 011 48-bit response without CRC check. For example: Format R3/R4. Others Reserved

29.3.3.5 SDHC Response Time Out Register (RES_TO)

The MMC/SD response time out register defines an interval within which a response must be returned, or if a time-out error occurs. After the SDHC sends out a command, if the card does not respond within the specified interval, the RESPONSE TIMEOUT status bit (STATUS[1]) and the END_CMD_RESP status bit (STATUS[13]) are set.

Figure 29-8 shows the RES_TO register, and Table 29-9 shows the register's field descriptions.

0x5000_4010 (RES_TO)
0x5000_8010 (RES_TO)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	RESPONSE TIME OUT[7:0]							
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Figure 29-8. MMC/SD Response Time Out Register

Table 29-9. MMC/SD Response Time Out Register Field Descriptions

Field	Description
31–8	Reserved
7–0 RESPONSE TIME OUT	Response time out. This value determines the interval by which response time-out is detected. The clock starts counting when the last bit of the command is sent. The clock counts unit is MMC_SD_CLK to card. 0x01 1 clock count 0x02 2 clock counts ... ... 0xFF 255 clock counts

29.3.3.6 SDHC Read Time Out Register (READ_TO)

The MMC/SD read time out register defines an interval that read data must be returned within or a time out error occurs. After the SDHC sends out the data read command, if no data is returned within the specified interval, the READ TIMEOUT status bit (STATUS[0]) and the READ_OP_DONE status bit (STATUS[11]) is set.

Figure 29-9 shows the READ_TO register, and Table 29-10 shows the register's field descriptions.

0x5000_4014 (READ_TO)
0x5000_8014 (READ_TO)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATA_READ_TIME_OUT[15:0]															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 29-9. SDHC Read Time Out Register

Table 29-10. SDHC Read Time Out Register Field Descriptions

Field	Description
31–16	Reserved
15–0 DATA_READ_TIME_OUT	Data read time out. This value determines the interval by which read data time-outs are detected. The user needs to check the timeout limit of the card and the clock frequency to configure this register. For safety, 0xFFFF is recommended. The timeout clock starts counting when the last bit of the command is sent. The count unit is MMC_SD_CLK/256. The maximum delay the SDHC can tolerate for a data timeout is related to the card clock. If the clock is 25 MHz and this register is 0xFFFF, the maximum delay that the SDHC waits is about 670 ms. If the card does not give data in 670 ms, SDHC issues a data read time-out error and terminates the current data read operation. This is designed to meet the SD physical layer specification, with typical time-out limit to be 100 ms–200 ms. But for some SDIO cards, the time-out limit can be up to 1 s. In this case, the user needs to lower the MMC_SD_CLK frequency to accommodate the delay to 1 s, which requires configuring the MMC_SD_CLK to about 16 MHz and setting this register for 0xFFFF.

29.3.3.7 SDHC Block Length Register (BLK_LEN)

The SDHC block length register defines the number of bytes in a block (block size). Since the stream mode of MMC is not supported, the block length must be set for every transfer. The block length supported by the SDHC ranges from 1 byte to 2048 bytes, but the user needs to check the block size supported by the card before configuring this register. For the SDIO, the block length must be less than the maximum block size defined in the card's CCCR. For the SD/MMC, the block length must be less than the maximum block size defined in the card's CSD register.

NOTE

The software should write to this register only when no SD bus transaction is executing.

Figure 29-10 shows the BLK_LEN register, and Table 29-11 shows the register's field descriptions.

0x5000_4018 (BLK_LEN)
0x5000_8018 (BLK_LEN)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	BLOCK LENGTH[11:0]											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 29-10. SDHC Block Length Register

Table 29-11. SDHC Block Length Register Field Descriptions

Field	Description
31–12	Reserved
11–0 BLOCK LENGTH	Block length. Specifies the number of bytes in a block during data transfer (block size). For the MMC and SD cards, the value set must be the same as the blk_len set in the CARD. For the SDIO, the I/O access is performed through the CMD53 IO_RW_EXTEND command. This command has two modes: • Byte. For byte mode, its operation is similar to a single block transfer command for SD where the block length is the byte count in the command argument. • Block. For block mode, its operation is similar to a multi-block transfer command for the SD where the block length is the block size defined in the command argument. For multi-block data transfers, a block length that is equal to an integer multiple of the data buffer size is preferred. Otherwise, the buffer utilization would be poor. If the data size that needs to be transferred is not an integer multiple of the buffer size, there are two options available to transfer the data: • Option 1: The user needs to split the transaction. The remainder of block size data is transferred by using a single block command for the last transfer. • Option 2: The user needs to add filler data in the last block to fill the block size to be as large as the buffer size. The data buffer size is 64 bytes in 4-bit mode and 16 bytes in 1-bit mode. Refer to Section 29.4.1, “Data Buffers” for more details about data buffers. 0x000 0 byte 0x001 1 byte ... ... 0x7FF 2047 bytes 0x800 2048 bytes 0x801–0XFFF Reserved

29.3.3.8 SDHC Number of Blocks Register (NOB)

The SDHC number of blocks register defines the number of blocks in the multi-block transfer mode. This register and the block length register determines the number of bytes to be transferred during one command. The number is decremented every time a block transfer is completed and stops when the count reaches zero. When all data transfers are completed, the STATUS[11] READ_OP_DONE is set if it is a read (from card) transfer, or the STATUS[12] WRITE_OP_DONE is set if it is a write (to card) transfer.

The software should write to this register only when no MMC/SD transaction is executing.

The NOB and the BLK_LEN defines the maximum data to be transferred in a single data transfer command.

Maximum data size to be transferred in bytes = block length x number of blocks.

Figure 29-11 shows the NOB register, and Table 29-12 shows the register’s field descriptions.

0x5000_401C (NOB)
0x5000_801C (NOB)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	NOB[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 29-11. SDHC Number of Blocks Register

Table 29-12. SDHC Number of Blocks Register Field Descriptions

Field	Description
31–16	Reserved
15–0 NOB	Specifies the number of blocks in a block transfer. One block should be set if the data transfer command is a single block transfer command or IO_RW_EXTEND (CMD53) in byte mode. For multi-block transfer commands to SD/MMC card and IO_RW_EXTEND (CMD53) in block mode to SDIO card, this register should be set for the block count the software expected. The maximum block number to be transferred for command can be as large as 65535. Blocks can range in length from 0 to 65535 bytes. For SD Memory card or memory parts of a SDIO combo card, the user needs to send CMD12 to stop the multi-block transfer. For a SDIO CMD53 in block mode and if user needs to abort the transfer earlier, the user needs to use CMD52 IO-Abort to abort the transfer. 0x0000 0 Block 0x0001 1 Block ... ... 0xFFFF 65535 Blocks **Note:** The maximum transfer blocks is 64 Kbytes. If the user uses infinite transfer command to transfer data, such as multi-block transfer command for memory card or infinite block transfer CMD53 for SDIO card, this register needs to be set to the real number of blocks that the user expected to transfer. The user also needs to abort the transfer through CMD12 or CMD52 IO-Abort to do this.

29.3.3.9 SDHC Revision Number Register (REV_NO)

The read-only SDHC revision number register is a read-only register that displays the revision number of the module.

Figure 29-12 shows the REV_NO register, and Table 29-13 shows the register's field descriptions.

0x5000_4020 (REV_NO)
0x5000_8020 (REV_NO)

Access: User read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Revision Number[15:0]															
W																
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

Figure 29-12. SDHC Revision Number Register

Table 29-13. SDHC Revision Number Register Field Descriptions

Field	Description
31–16	Reserved
15–0 REVISION NUMBER	Revision number. Specifies the revision number of the MMC/SD module. This is fixed at 0x0000_0400.

29.3.3.10 SDHC Interrupt Control Register (INT_CNTR)

When certain events occur in the module, the SDHC has the ability to set an interrupt as well as to set corresponding status register bits. The SDHC interrupt control register allows the user to control if these interrupts should occur.

Figure 29-13 shows the INT_CNTR register, and Table 29-14 shows the register's field descriptions.

0x5000_4024 (INT_CNTR)
0x5000_8024 (INT_CNTR)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	SDIO_INT_WKP_EN	CARD_INSERT_WKP_EN	CARD_REMOVAL_WKP_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CARD_INSERTION_EN	CARD_REMOVAL_EN	SDIO_IRQ_EN	DAT0_EN	0	0	0	0	0	0	0	BUF_READ_EN	BUF_WRITE_EN	END_CMD_RES	WRITE_OP_DONE	READ_OP_DONE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 29-13. SDHC Interrupt Control Register

Table 29-14. SDHC Interrupt Control Register Field Descriptions

Field	Description
31–19	Reserved
18 SDIO_INT_WKP_EN	SDIO interrupt wake-up enable. When INT_CNTR[13] is set to a ‘1’ (SDIO interrupt enabled), this bit controls whether a SDIO card interrupt is detected synchronously or asynchronously. To set this bit, the SDIO card interrupt is detected asynchronously and the SDIO card interrupt is used as a wake-up event. The user should set this bit only in low power mode or when all the clocks to SDHC are off. 0 Disable SDIO card interrupt as wake-up event. 1 Enable SDIO card interrupt as wake-up event.
17 CARD_INSERTION_WKP_EN	Card insertion wake-up enable. When INT_CNTR[15] is set to a ‘1’ (card insertion interrupt enabled), this bit controls whether a card insertion interrupt is detected synchronously or asynchronously. To set this bit, the card insertion interrupt is captured asynchronously and the interrupt is used as a wake-up event. The user should set this bit only in low power mode or when all the clocks to SDHC are off. 0 Disable card insertion as wake-up event. 1 Enable card insertion as wake-up event.
16 CARD_REMOVAL_WKP_EN	Card removal wake-up enable. When INT_CNTR[14] is set to a ‘1’ (card removal interrupt enabled), this bit controls whether the card removal interrupt is detected synchronously or asynchronously. To set this bit, the card removal interrupt is detected asynchronously and the interrupt is used as a wake-up event. The user should set this bit only in low power mode or when all the clocks to SDHC are off. 0 Disable card removal as wake-up event. 1 Enable card removal as wake-up event.

Table 29-14. SDHC Interrupt Control Register Field Descriptions (continued)

Field	Description
15 CARD_INSERTION_EN	Card insertion enable. Setting this bit enables the card insertion interrupt. Since the card detection is through the value of DAT3 data line, if this card is in 4-bit mode, any data transfers in the DAT3 line cause false card insertion interrupts to be generated. The card insertion interrupt should be disabled after the first time the card insertion is detected. To avoid the false status bit generation during data transfer, the card insertion status is masked by this bit. It should be enabled only after the card is removed from the socket. The default of this bit is to disable the card insertion interrupt. When this interrupt is detected, the user needs to write a '1' to the STATUS[31] bit to clear the card insertion status interrupt. 0 Card insertion interrupt is disabled. 1 Card insertion interrupt is enabled. **Note:** INT_CNTR[17] CARD_INSERTION_WKP_EN controls whether this interrupt is detected asynchronously or synchronously.
14 CARD_REMOVAL_EN	Card removal enable. Setting this bit enables the card removal interrupt. Since card detection is through the value of the DAT3 data line, if this card is in 4-bit mode, the data transfer through the DAT3 line causes false card removal interrupt to be generated. The card removal interrupt should be enabled only when there are no active data transfers on the DAT3 line. To avoid the false status bit generation during data transfer, the card insertion status is masked by this bit. The default of this bit is to disable the card removal interrupt. When this interrupt is detected, the user needs to write a '1' to the STATUS[30] bit to clear the card removal status interrupt. 0 Card removal interrupt is disabled. 1 Card removal interrupt is enabled. **Note:** INT_CNTR[16] CARD_REMOVAL_WKP_EN controls whether this interrupt is detected asynchronously or synchronously.
13 SDIO_INT_EN	SDIO interrupt enable. Masks the interrupt from the SD I/O card to the SDHC module interrupt. 0 SD I/O interrupt is disabled. 1 SD I/O interrupt is enabled. **Note:** INT_CNTR[18] SDIO_INT_WKP_EN controls whether this interrupt is detected asynchronously or synchronously.
12 DAT0_EN	Data enable. Identifies how the SD I/O interrupt is detected. An interrupt is determined by SD_DAT [1] low, but this bit is an option setting for the SDIO bit. When the SDHC is preforming data transfer and the SD bus mode is 1-bit mode, the user should set this bit to '0'. 0 SD I/O's Interrupt detection based on SD_DAT[3:0] = 110x. 1 SD I/O's Interrupt detection based on SD_DAT[3:0] = 1101.
11–5	Reserved
4 BUF_READ_EN	Bus read enable. This bit controls the buffer read ready interrupt. If the bit is '1', the interrupt is enabled. When the buffer becomes full during a read operation, an interrupt is generated. The user needs to move the data out of the FIFO and clear the BUF_READ_READY bit to clear the interrupt. 0 Buffer status interrupt is disabled. 1 Buffer status interrupt is enabled.

Table 29-14. SDHC Interrupt Control Register Field Descriptions (continued)

Field	Description
3 BUF_WRITE_EN	Bus write enable. This bit controls the buffer read ready interrupt. If the bit is '1', the interrupt is enabled. When the buffer becomes empty-during a write operation, an interrupt is generated. The user needs to write data to the FIFO and clear the BUF_WRITE_READY bit to clear the interrupt. 0 Buffer status interrupt is disabled. 1 Buffer status interrupt is enabled.
2 END_CMD_RES	End command response. This bit controls the interrupt generation on the status at the end of the command response. When this bit is '1', the SDHC generates an interrupt at the end of the command response status. 0 End command-response interrupt is disabled. 1 End command-response interrupt is enabled.
1 WRITE_OP_DONE	Write operation done. This bit controls the interrupt generation for the status of write operation. When the interrupt is enabled, the SDHC generates an interrupt when the configured bytes of data are transferred to the card. 0 Write_OP_DONE interrupt is disabled. 1 Write_OP_DONE interrupt is enabled.
0 READ_OP_DONE	Read operation done. This bit controls the interrupt generation for the status of read operation completion. When the interrupt is enabled, the SDHC generates an interrupt when the pre-defined bytes of data are transferred from the card. 0 READ_OP_DONE interrupt is disabled. 1 READ_OP_DONE interrupt is enabled.

When an interrupt is generated, there may be some error bits in the STATUS register set as well as the interrupt status. The user needs to check the error status bit to ensure there is no error in the SDHC operation. For example, when the READ_OP_DONE (STATUS[11]) status is set or the READ_OP_DONE interrupt is detected, the user needs to check both the STATUS[3] and STATUS[0] bits as well to ensure the read operation completed without a CRC error or a time out error. Another example is a write operation if both the WRITE_OP_DONE(STATUS[12]) and WRITE_CRC_ERR (STATUS[2]) bits are set. This means the write operation ended with CRC error. See Table 29-15 for a summary of the relationship between the interrupt, interrupt control register, and status registers in the SDHC.

Table 29-15. Interrupt Mechanisms

Source STATUS Bit Name (Status Bit Number)	Does This Status Generate Interrupt Directly?	INT_Control Register Bit Name (INT_CNTR bit Number)	Interrupt/Status Clear Method
TIME_OUT_READ (0)	No, alert by using the READ_OP_DONE bit in the SDHC status register	READ_OP_DONE (0)	Clear status by writing '1'.
TIME_OUT_RESP (1)	No, alert by using the END_CMD_RESP bit in the SDHC status register	END_CMD_RES (2)	Clear status by writing '1'.
WRITE_CRC_ERR (2)	No, alert by using the WRITE_OP_DONE bit in the SDHC status register	WRITE_OP_DONE (1)	Clear status by writing '1'.

Table 29-15. Interrupt Mechanisms (continued)

Source STATUS Bit Name (Status Bit Number)	Does This Status Generate Interrupt Directly?	INT_Control Register Bit Name (INT_CNTR bit Number)	Interrupt/Status Clear Method
READ_CRC_ERR (3)	No, alert by way of the READ_OP_DONE bit in the SDHC status register	READ_OP_DONE (0)	Clear status by writing ‘1’.
RESP_CRC_ERR (5)	No, alert by using the END_CMD_RESP bit in the SDHC status register	END_CMD_RES (2)	Clear status by writing ‘1’.
BUF_WR_RDY (6)	Yes	BUF_WRITE_EN (3)	Clear status by writing data to FIFO buffer.
BUF_READ_RDY (7)	Yes	BUF_READ_EN (4)	Clear status by reading data from FIFO buffer.
READ_OP_DONE(11)	Yes	READ_OP_DONE (0)	Clear status by writing ‘1’.
WRITE_OP_DONE(12)	Yes	WRITE_OP_DONE (1)	Clear status by writing ‘1’.
END_CMD_RESP(13)	Yes	END_CMD_RESP (2)	Clear status by writing ‘1’.
SDIO_INT_ACTIVE(14)	Yes	SDIO_INT_EN (13)	Clear status by writing ‘1’.
CARD_REMOVAL(30)	Yes	CARD_REMOVAL_EN (14)	Clear status by writing ‘1’.
CARD_INSERTION(31)	Yes	CARD_INSERTION_EN (15)	Clear status by writing ‘1’.

29.3.3.11 SDHC Command Number Register (CMD)

The command to the SD card is always 48 bits long. It contains one start bit, one direction bit, six command number bits, 32 argument bits, seven CRC bits, and one end bit. For more details on the format of the command, refer to the SD physical layer specification. Refer to Table 29-20 for MMC/SD/SDIO cards command list and the related card specification for the detailed information about each command.

In SDHC, the command start bit, direction bit, CRC7 bits, and end bit are automatically generated by the hardware. Configure the SDHC command number register and SDHC command argument register to issue a command to the card.

Figure 29-14 shows the CMD register, and Table 29-16 shows the register’s field descriptions.

0x5000_4028 (CMD)
0x5000_8028 (CMD)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	COMMAND NUMBER					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 29-14. SDHC Command Number Register

Table 29-16. SDHC Command Number Register Field Descriptions

Field	Description
31–6	Reserved
5–0 COMMAND Number	Command number. The SDHC module communicates with the MMC/SD/SDIO card(s) by sending commands and arguments. The command to send is set in the MMC/SD command number register (CMD) and the argument is defined in SDHC CMD argument register (ARG). See Table 29-20 for the brief information of the full list of MMC/SD/SDIO commands. 0x00 CMD0 0x01 CMD1 ... ... 0x3F CMD63 **Note:** The user should check the detailed information from the related card specification.

29.3.3.12 SDHC CMD Argument Register (ARG)

This register contains the MMC/SD/SDIO command argument.

Figure 29-15 shows the ARG register, and Table 29-17 shows the register's field descriptions.

0x5000_402C (ARG)
0x5000_802C (ARG)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ARG[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ARG[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 29-15. SDHC Command Argument Register

Table 29-17. SDHC Command Argument Register Field Descriptions

Field	Description
31–0 ARG	Command argument. Specifies the argument for the current command. **Note:** The user should check the detailed command argument information from the related card specification.

29.3.3.13 SDHC Response FIFO Access Register (RES_FIFO)

There is an 8 x 16 bit FIFO to store the response from the card in SDHC. This register is used to access this FIFO. The MSB 16 bits of the response is accessed first and the LSB 16 bits is accessed last.

Figure 29-16 shows the RES_FIFO register, and Table 29-18 shows the register's field descriptions.

0x5000_4034 (RES_FIFO)
0x5000_8034 (RES_FIFO)

Access: User read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RESPONSE_CONTENT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 29-16. SDHC Response FIFO Register

Table 29-18. SDHC Response FIFO Register Field Descriptions

Field	Description
31–16	Reserved
15–0 RESPONSE_CONTENT	Response content FIFO access register. There is a FIFO in the SDHC that is used to store the command response received from card. Every time the host sends a command to a card, the current contents stored in the FIFO are cleared and new response arguments are stored into the response FIFO. According to the SD card specification, the command response size can be 48 bits or 136 bits (R2 response). Refer to the SD Memory Card specification for more detailed information about the command response format. The size of the response FIFO is 8 x 16 bits (128 bits). For a 48-bit response, only 48 bits of the FIFO have valid contents. The user must perform three reads to this response FIFO access register to retrieve the entire 48-bit response content. For a 136-bit R2 response (response for CID[127:0] or CSD[127:0] register), only the contents of the 128-bit CID and CSD register are stored in the response FIFO. This first byte of the R2 response is not stored in the response FIFO. The user can retrieve the CIS/CSD register from the response FIFO through eight accesses to the FIFO access register. All the CRC bits in the response are not stored in the response FIFO. This response FIFO is read-only. **Note:** The CRC7 and end bit for response is hardware checked by SDHC and the corresponding field of the response are not stored in the response FIFOs.

29.3.3.14 SDHC Data Buffer Access Register (BUFFER_ACCESS)

The SDHC uses two 64-byte data buffers in a ping-pong manner—while one buffer is receiving new transmission information, the other buffer is deleting the previous transmission. Data can be transferred by the DMA and the SD card simultaneously to maximize throughput between the two clock domains (that is, the IP peripheral clock, IPG_PERCLK, and the host clock, CLK_20M). These buffers are used as temporary storage for data being transferred between the host system and the card and vice versa. The user can read or write data to the buffers through this buffer access register. Refer to Section 29.4.1, “Data Buffers” for more details about the data buffers.

In the read operation, the SDHC stores the data received from the card into the buffer. The user needs to move the data out of the buffer when the buffer is full.

In the write operation, the SDHC fetches data from the buffer and transfers them to the card. The user can then access the data buffer through the SDHC data buffer access register. The user needs to move data into the buffer when the buffer is empty.

Figure 29-17 shows the BUFFER_ACCESS register, and Table 29-19 shows the register’s field descriptions.

0x5000_4038 (BUFFER_ACCESS)
0x5000_8038 (BUFFER_ACCESS)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	FIFO CONTENT[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	FIFO CONTENT[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 29-17. SDHC Buffer Access Register

Table 29-19. SDHC Buffer Access Register Field Descriptions

Field	Description
31–0 FIFO CONTENT	First In/First Out content. These bits hold 32-bit data upon a read or write transfer. The size of the FIFO is 4 x 32 bits (16 bytes in total) for SD 1-bit mode and 16 x 32 bits (64 bytes in total) for SD 4-bit mode. For reception, the SDHC controller generates a DMA request when the FIFO is full. Upon receiving this request, the DMA starts transferring data from the SDHC FIFO to system memory by reading the data buffer access register for a number of pre-defined bytes. For transmit, the SDHC controller generates a DMA request when the FIFO is empty. Upon receiving this request, the DMA starts moving data from the system memory to the SDHC FIFO by writing to the data buffer access register for a number of pre-defined bytes.

29.4 Functional Description

The following sections provide a brief functional description of the major system blocks, including the DMA interface, memory controller, logic/command controller, and system clock controller.

29.4.1 Data Buffers

The SDHC uses two data buffers in a ping-pong manner so that data can be transferred by the DMA and the SD card simultaneously to maximize throughput between the two clock domains (that is, the IP peripheral clock—IPG_PERCLCK, and the host clock—CLK_20M). Figure 29-18 shows the buffer scheme. These buffers are used as temporary storage for data being transferred between the host system and the card.

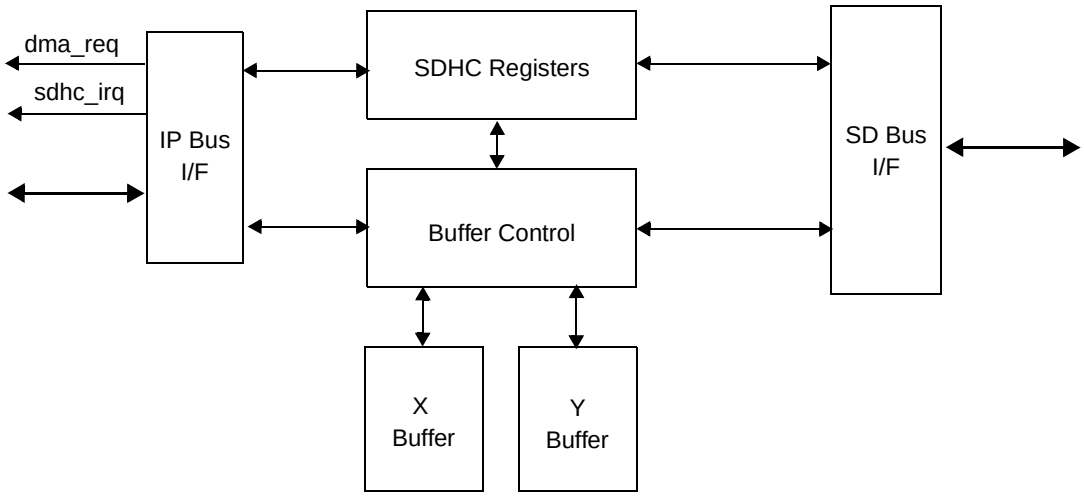


Figure 29-18. SDHC Buffer Scheme

For a host-read operation, the SDHC automatically transfers data into the next available buffer. It is then read out by the DMA and written to the system memory when the DMA request from SDHC becomes highest priority in the DMA arbiter. Conversely, for a host-write operation, the DMA writes data into the next available buffer. The SDHC then reads the data out of the buffer and writes it to the card through the SD Host interface.

29.4.1.1 Data Buffer Access

DMA/CPU accesses the data buffer of SDHC through the 32-bit data buffer access (DBA) register. Internally, the SDHC maintains a pointer into the data buffer. Accesses to the DBA register increases the address value of the pointer. The pointer value of SDHC is not directly accessible by the software. The pointer refers to a 32-bit port-size FIFO, so all the access to the FIFO must be 32-bit size. Sequential and contiguous access is necessary to increase the pointer address value correctly. Random or skipped access is not allowed.

In some cases, when the block length of the data transfer is not a multiple of 32 bits, the last data access to the FIFO can be 24-bit, 16-bit, or 8-bit. Since the SDHC FIFO allows only 32-bit access sizes, the user must put/get the data bytes on the correct byte lanes of the SDHC 32-bit data bus. The byte arrangement order is Little Endian format. For an 8-bit data access to the FIFO, the data is in bit[7–0] of the data bus. For 16-bit data access, the data is in the bit[15–0] of the data bus. For a 24-bit data access, the data is in the bit[23–0] of the data bus.

When data goes to the card, the 32-bit data in the data buffer FIFO is shifted out to the card from the LSB byte to the MSB byte. But for each byte, the bit sequence of the shift is from MSB bit to LSB bit. When read data leaves the card, the data shifted in is stored from LSB byte to MSB byte in the data buffer FIFO. See Figure 29-19 for the bytes lane relationship between the card bus and the SDHC IP bus.

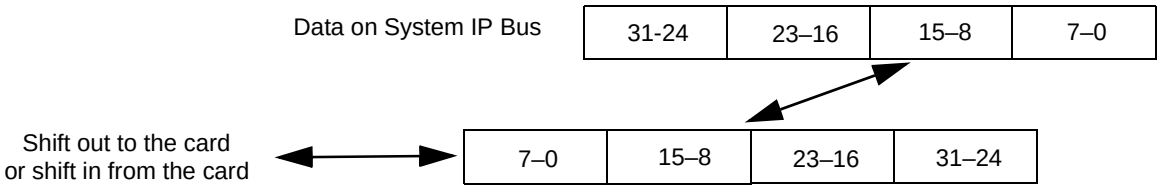


Figure 29-19. Byte Lanes Relationship Between System IP Bus and SD Card Bus

29.4.1.2 Write Operation Sequence

There are two ways to write data to the buffer when the user transfer data to the card. One way is by using DMA through the SDHC DMA request signal. The other way is by using the CPU through the BUF_WR_RDY (STATUS[6]) bit (interrupt or polling).

The SDHC asserts a DMA request when the data buffer is empty and ready to receive new data. At the same time, the SDHC sets the BUF_WR_RDY (STATUS[6]) bit. The buffer write ready interrupt is generated if it is enabled by the software.

The buffer accumulates the data written through the data port until the data count reaches the buffer size. The SDHC does not start data transmission until a full buffer size of data is written to the data buffer. The SDHC starts data transmission when the SD bus is ready for a new transfer. When the other buffer is empty and more data is to be transferred, the SDHC asserts a new DMA request and sets the BUF_WR_RDY bit. In case the DMA does not keep up with moving data into the FIFOs, the SDHC stops the SD_CLK at the block gap to avoid a data buffer underrun situation.

29.4.1.3 Read Operation Sequence

There are two ways to fetch data from the buffer when the user reads data from the card. One way is by using DMA through the SDHC DMA request signal. The other way is by using the CPU through the BUF_READ_RDY (STATUS[7]) bit (interrupt or polling).

The SDHC asserts a DMA request when data buffer is full and ready for DMA/CPU to fetch the data out of the buffer. At the same time, the SDHC sets the BUF_WR_RDY (STATUS[7]) bit. The buffer write ready interrupt is generated if it is enabled by the software.

The SDHC starts receiving data only when either of the dual data FIFO is empty. The buffer accumulates data read from the card until the data count reaches the buffer size. The SDHC asserts a DMA request when either one of the data buffers is full. For multiple block data transfers, while the DMA/CPU is moving data by reading the DBA register, the SDHC receives data into the other FIFO if it is empty and the SD bus is ready. In case the DMA/CPU does not keep up with reading data out of the FIFOs, the SDHC stops the SD_CLK at the block gap to avoid a data overflow situation.

29.4.1.4 Data Buffer Size

The user needs to know the buffer size for buffer operation during data transfer. In SDHC, both of two data buffers are 64 bytes in size. Each data buffer is divided into four 16 bytes data buffers that correspond to

the four data lines of SD bus. Thus, each data line in SDHC contains a dual 16-byte buffer. The data buffer size is 64 bytes in 4-bit SD mode or 16 bytes in 1-bit SD mode.

During multi-block data transfer, the block length, which is an integer multiple of the buffer size, is preferred. The buffer is ready to be read by the CPU/DMA when either of the buffers is full (assumption that STATUS[27] or STATUS[26] is set, and STATUS[7] is set) when full buffer data is written to either of the X or Y buffer. The buffer is ready to be written to by the CPU/DMA (assumption that STATUS[29] or STATUS[28] is set, and STATUS[6] is set) when the full buffer of data is fetched out of the buffer. The buffer ready status bit and DMA request would be set accordingly.

For single-block data transfer, when the block length is smaller than the buffer size or when the block length is not an integer multiple of the buffer size, it is possible that the data size that needs to be written to the buffer or to be fetched out of the buffer is smaller than the buffer size. In this case, the buffer would be full (SDHC set STATUS[27] or STATUS[26]) when this data is written to the buffer. The buffer would be empty (SDHC set STATUS[29] or STATUS[28]) when this buffer of data is fetched out of the buffer. The buffer ready status bit and DMA request would be set accordingly. From the software aspect, the buffer size becomes variable and equal to the real data size that needs to be transferred. This eases the software programming of SDHC. The user does not need to fill dummy data to make the buffer full.

29.4.1.5 Dividing Large Data Transfer

This SDIO command CMD53 definition limits the maximum data size of data transfers according to the following formula:

Maximum data size = block size x block count

The block size can be a multiple of the size of the data buffer. However, it is recommended the block size be set to equal the size of data buffer. This allows the SDHC to stop the SD_CLK during block gaps if an overflow or underrun condition occurs. Stopping the SD_CLK while the DATA lines are active can cause data corruption (when the clock resumes) on some card designs available on the market. If an application or card driver is to transfer larger sizes of data, the host driver divides larger data sizes into multiple blocks.

The length of a multiple block transfer needs to be in block size units. If the total data length cannot be divided evenly to a multiple of the block size, then there are two ways to transfer the data depending on the function and card design. One way is for the card driver to split the transaction. The remainder of block size data is then transferred by using a single block command at the end. Another way is to add dummy data in the last block to fill the block size. In the second method, the card must be able to remove the dummy data.

See Figure 29-20 for an example that shows dividing of large data transfers.

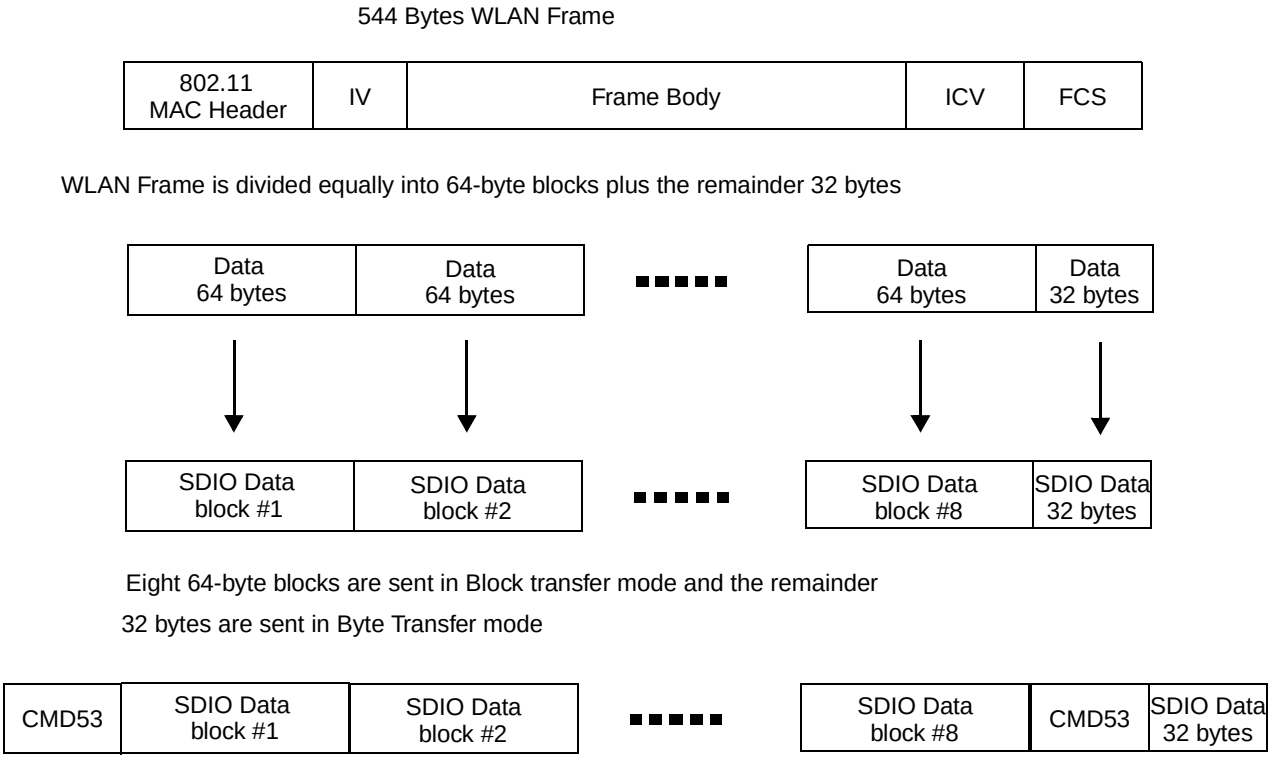


Figure 29-20. Example for Dividing Large Data Transfer

29.4.2 DMA Interface

The DMA interface block controls all data routing between the external data bus (DMA access), internal SDHC module data bus, and internal system FIFO access through a dedicated state machine. This state machine monitors the status of FIFO content (empty or full), FIFO address, and byte/block counters for the SDHC module and the application. Figure 29-21 shows the DMA interface block.

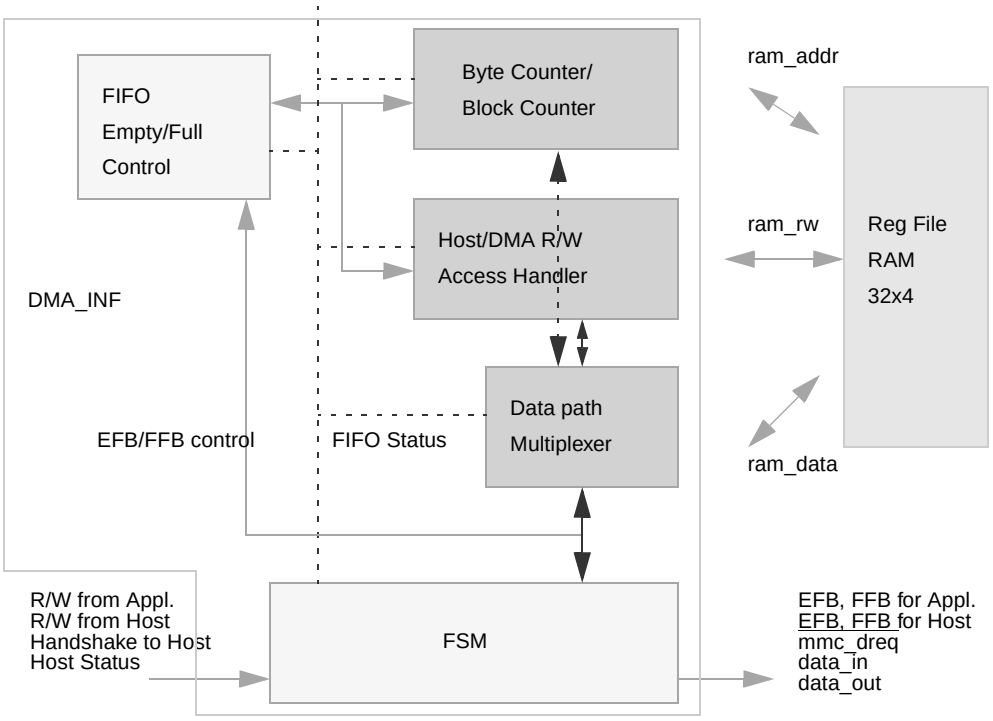


Figure 29-21. DMA Interface Block

In addition, this block also handles the burst request to the external DMA controller, internal register write-error detection, read/wait handling of SDIO, and all IP-related output responses.

29.4.2.1 DMA Request

If the SDHC is in the data transfer state, the SDHC generates DMA requests according to its buffer status. During read operations, the SDHC generates DMA requests if one of its data buffer is full. During write operations, the SDHC generates DMA requests if one of its data buffers is empty.

To avoid buffer underrun conditions during a write operation, the MMC_SD_CLK stops automatically when both buffers are empty. After the DMA or CPU completes writing data into one of the buffers, the MMC_SD_CLK automatically resumes to continue the data transfer.

Similarly, to avoid buffer overflow during read operations, the MMC_SD_CLK stops automatically when both buffers are full. After the DMA or CPU moves the data out of the buffer, the MMC_SD_CLK automatically resumes to continue the data transfer.

29.4.3 Memory Controller

This controller provides the SDIO IRQ and read/wait service handling, card detection, command response handling, and all SDHC interrupt handling. The memory controller also contains the register table. Figure 29-22 shows the block diagram for the memory controller.

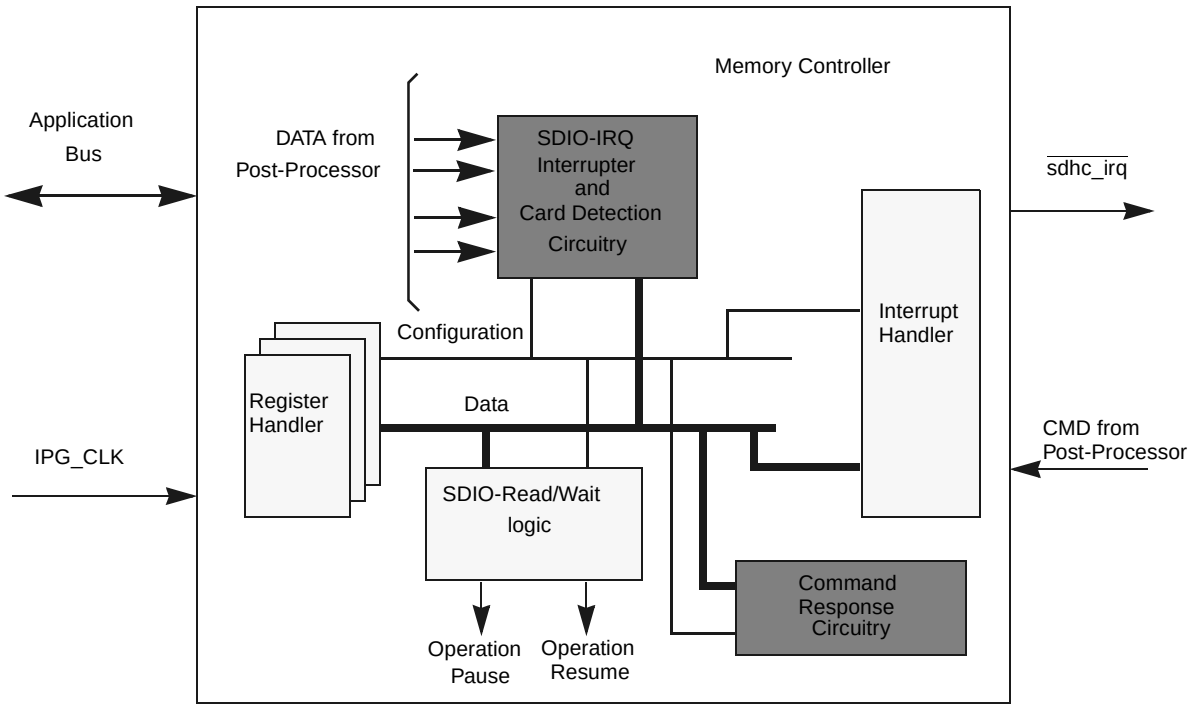


Figure 29-22. Memory Controller Block Diagram

This section summarizes events when an SDIO card generates an interrupt. When an SDIO card generates an interrupt request, it sets its interrupt pending bit in the CSR register and asserts the interrupt line, which is shared with DAT[1] line in 4-bit mode. The SDHC detects and steers the card's interrupt to the selected IRQ line and to the interrupt controller.

29.4.4 SDIO Card Interrupt

29.4.4.1 Interrupts in 1-Bit Mode

In this case, the SD_DAT[1] pin is dedicated to providing the interrupt function. An interrupt is asserted by pulling the SD_DAT[1] low until the host clears the interrupt.

29.4.4.2 Interrupt in 4-Bit Mode

Since the interrupt and data line 1 share pin 8 in 4-bit mode, an interrupt is sent by the card and recognized by the host only during a specific time. This is known as the interrupt period. The SDHC samples the level on pin 8 only during the interrupt period. At all other times, the host interrupt controller ignores the level on pin 8. The definition of the interrupt period is different for operations with single block and multiple block data transfers.

In the case of normal single data block transmissions, the interrupt period becomes active two clock cycles after the completion of a data packet. This interrupt period lasts until after the card receives the end bit of the next command that has a data block transfer associated with it.

For multiple block data transfers in 4-bit mode, there is limited time that the interrupt period can be active because of the limited period of data line availability between the multiple blocks of data. This requires a more strict definition of the interrupt period. For this case, the interrupt period is limited to two clock cycles. This period begins two clock cycles after the end bit of the previous data block. During this two-clock cycle interrupt period, if an interrupt is pending, the SD_DAT[1] line is held low for one clock cycle and the last clock cycle pulls SD_DAT[1] high. On completion of the interrupt period, the card releases the SD_DAT[1] line into the high Z state.

When in 4-bit mode, the SDHC differentiates a data start bit and the interrupt period by checking that all four data lines are low for the start of new data. In the case of an interrupt, only the DAT[1] should have gone low. After the last data block is sent, the interrupt period starts as normal. The interrupt period ends after the next command with data instead of lasting only two cycles.

Refer to SDIO Card Specification v1.0 for further information about SDIO card interrupt.

29.4.4.3 Card Interrupt Handling

When the SDIO bit in the interrupt control register is set to 0, the host controller clears the interrupt request to the system interrupt controller. The SDIO interrupt detection is stopped when this bit is cleared and restarted when this bit is set to ‘1’. The host driver should clear the SDIO interrupt enable bit before servicing the SDIO interrupt. The host driver should set this bit again after all interrupt requests from the card are cleared to prevent inadvertent interrupts.

The SDIO status bit is cleared by resetting the SDIO interrupt. Writing to this bit has no effect in 1-bit mode, as the host controller detects the SDIO interrupt with or without the SD clock (to support wake-up). In 4-bit mode, the interrupt signal is sampled during the interrupt period. There are some sample delays between the interrupt signal from the SDIO card and the interrupt to the host system interrupt controller. When the SDIO status has been set and the host driver needs to start this interrupt service, the SDIO bit in the interrupt control register is set to ‘0’ to clear the SDIO interrupt status latched in the SDHC and to stop driving the interrupt signal to the system interrupt controller. The host driver must issue a CMD52 to clear the interrupts at the card. After completion of the card interrupt service, the SDIO interrupt enable bit is set to ‘1’. The SDHC starts sampling the interrupt signal again.

- Figure 29-23 (a) shows the SDIO card interrupt scheme.
- Figure 29-23 (b) shows the sequences of software and hardware events that occur during the card interrupt handling procedure.

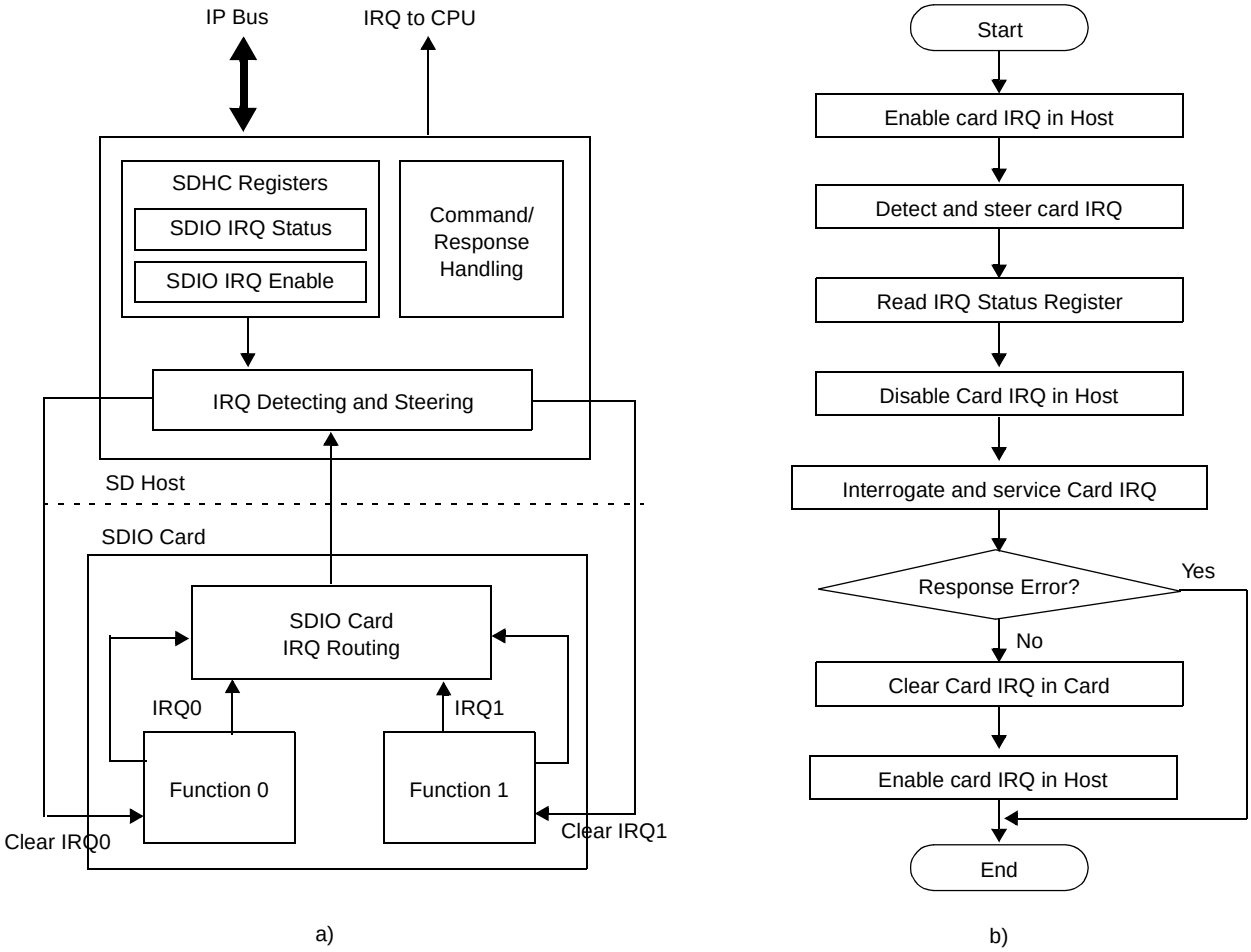


Figure 29-23. a) Card Interrupt Scheme, b) Card Interrupt Detection and Handling Procedure

29.4.5 Card Insertion and Removal Detection

SDHC uses the SD_DAT[3] pin to detect card insertion or removal. To utilize this feature of the SDHC, the chip level integration needs to pull down this pad as a default state. When there is no card on the MMC/SD bus, the SD_DAT[3] defaults to a low voltage level. When any card is inserted or removed from the socket, SDHC detects the logic value changes on the SD_DAT[3] pin and generates an interrupt.

Since the mechanism is based on the value of the SD_DAT[3] line, only single-card systems can benefit from card detection. To avoid the conflict of card insertion/removal detection and the data value changes on SD_DAT[3] because of data transfer, the user should disable the card insertion interrupt when there is a card detected in the socket but enable the interrupt when the card is removed from the socket. The card removal interrupt can be enabled only when there is no bus activity on SD_DAT[3].

To avoid the false status bit generation during data transfer, the card insertion/removal is masked by the corresponding interrupt enable bit in INT_CNTR register.

NOTE

The user can send a command (ACMD42 for SDMem or CMD52 for SDIO) to the card to disable the card internal pull-up resistor after card detection and identification. Since the SD protocol requires that the DAT line must be pulled up for data transfer, the user must disable the host side of the SD_DAT[3] pull-down feature and configure it as pull-up. In the meantime, if the card internal pull-up resistor is disabled, the card removal interrupt cannot be detected through SD_DAT[3].

29.4.6 Power Management and Wake-Up Events

When there is no operation between SDHC and the card through SD bus, the user can disable the ipg_clk and ipg_perclk in chip level clock control module to save power. When the user needs to use SDHC or communicate with the card, he or she can enable the clock and perform the operation.

In some circumstances, when the clocks to SDHC are disabled, or when system is in low power mode, there are some events when the user needs to enable the clock and handle the event. These events are called wake-up interrupts. SDHC can generate these interrupts even if there are no clocks enabled. The three interrupts which can be used as wake-up events are as follows:

- Card removal interrupt
- Card insertion interrupt
- SDIO card interrupt

The SDHC offers a power management feature. By clearing the clock-enabled bits in the clock control register, the clocks are gated in the low position to the SDHC. For maximum power saving, the user can disable all the clocks to SDHC when there is no operation in progress.

While in this state, it is possible that interrupts can occur that require the SDHC to respond. These interrupts are called wake-up events and are defined as follows:

- Wake-up event on SD card removal through card removal interrupt
- Wake-up event on SD card insertion through card insertion interrupt
- Wake-up event on card interrupt through SDIO interrupt

These three wake-up events (or wake-up interrupts) can also be used to wake the system from low-power modes.

NOTE

To use the interrupt as a wake-up event when all the clocks to SDHC are disabled or when whole system is in low power mode, the corresponding wake-up enabled bit needs to be set. Refer to Section 29.3.3.10, “SDHC Interrupt Control Register (INT_CNTR)” for more details.

NOTE

According to the SDIO specification, if the user wants the card interrupt to wake up from low power mode (no clock to SDIO card), the card should be set to 1-bit mode through CMD52.

29.4.6.1 Dynamic Voltage/Frequency Scaling (DVFS) Operation

Any change in ipg_perclk impacts the MMC/SD/SDIO transfer clock rate.

29.4.6.2 Setting Wake-Up Events

For the SDHC to respond to a wake-up event, the software must set the respective wake-up enable bit before the CPU enters sleep mode. Before the software disables the host clock, it should ensure that all of the following conditions have been met:

- No read or write transfer is active.
- Data and command lines are not active.
- No interrupts are pending.
- Internal FIFOs are empty.

The software is responsible to ensure that the clock to the SDHC is fully operational before making accesses to the peripheral.

29.4.7 Command/Data Interpreter

Command and data interpreters are based on similar principles. Both devices consist of three parts:

- Inner state machine
- Submodule controller
- CRC hardware accelerator

The CMD interpreter handles everything related to command line (CMD). The CMD Interpreter includes command data sequence generation, command response extraction, CRC generation and checking, and a response time-out detection. To achieve the above functions, a state machine, a logic control, and a CRC accelerator are used. Figure 29-24 shows the block diagram for the command interpreter.

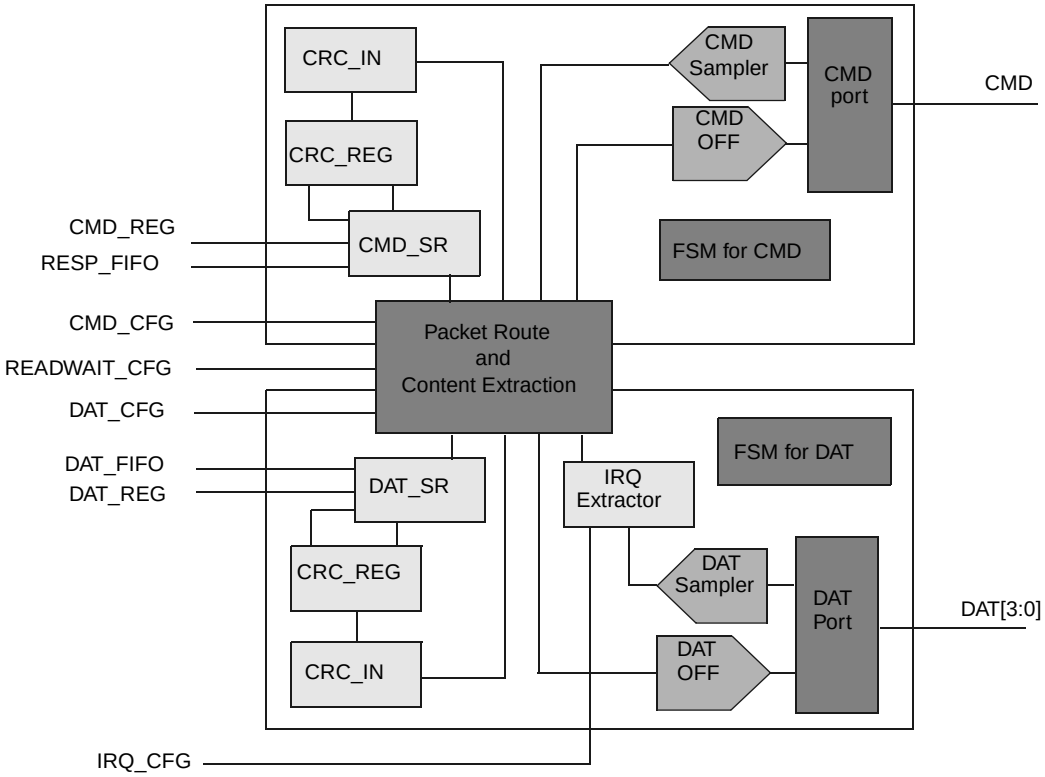


Figure 29-24. Block diagram for Command Interpreter

Figure 29-25 shows the structure for the command CRC shift register.

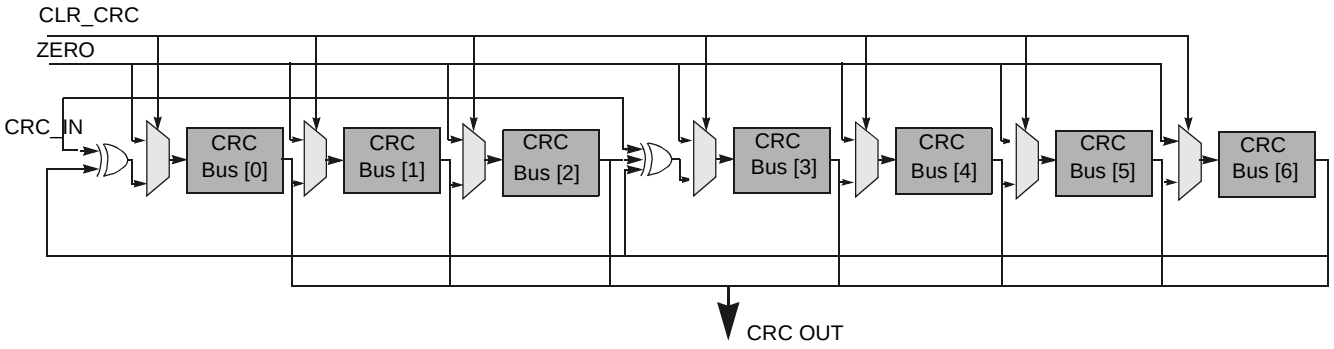


Figure 29-25. Command CRC Shift Register (DATs Have a Similar Structure)

To minimize the gate count, the internal command shift register is re-used for the CRC shift register. The polynomials for the CMD and the DAT are as follows:

For the CMD:

Generator polynomial: $G(x) = x^7 + x^3 + 1$
$M(x) = (\text{first bit}) * x^n + (\text{second bit}) * x^{n-1} + \ldots + (\text{last bit}) * x^0$
$CRC[6:0] = \text{Remainder } [(M(x) * x^7) / G(x)]$

For the DAT:

Generator polynomial: $G(x) = x^{16} + x^{12} + x^5 + 1$
$M(x) = (\text{first bit}) * x^n + (\text{second bit}) * x^{n-1} + \ldots + (\text{last bit}) * x^0$
$CRC[15:0] = \text{Remainder } [(M(x) * x^{16}) / G(x)]$

29.4.8 System Clock Controller

There is one clock divider and one clock prescaler in SDHC to divide the high frequency input clock IPG_PERCLK to a low frequency clock, which can be used by card. See Figure 29-26. The input clock first goes through a 4-bit divider and then a 12-bit prescaler to generate a clock named CLK_20M. This clock is used internally by SDHC and generates the MMC_SD_CLK. The MMC_SD_CLK to the card has the same clock frequency as CLK_20M. CLK_20M is derived from the CLK_DIV by using the 12-bit prescaler. The CLK_DIV is derived from the input clock IPG_PERCLK by using the 4-bit divider. SDHC clock rate register controls the divide rate for both the divider and the prescaler. Refer to Section 29.3.3.3, "SDHC Clock Rate Register (CLK_RATE)" for the clock rate register information.

To maximize power-saving during the operation, the SDHC bus clock pauses and resumes according to the SDHC status. For example, when the FIFO is full during the card read operation, the bus clock is stopped if no further data is written to the FIFO by the card. The bus clock is resumed when the FIFO empty status is cleared by the user (DMA). Also, there are other conditions where the SDHC stops the clock to save power.

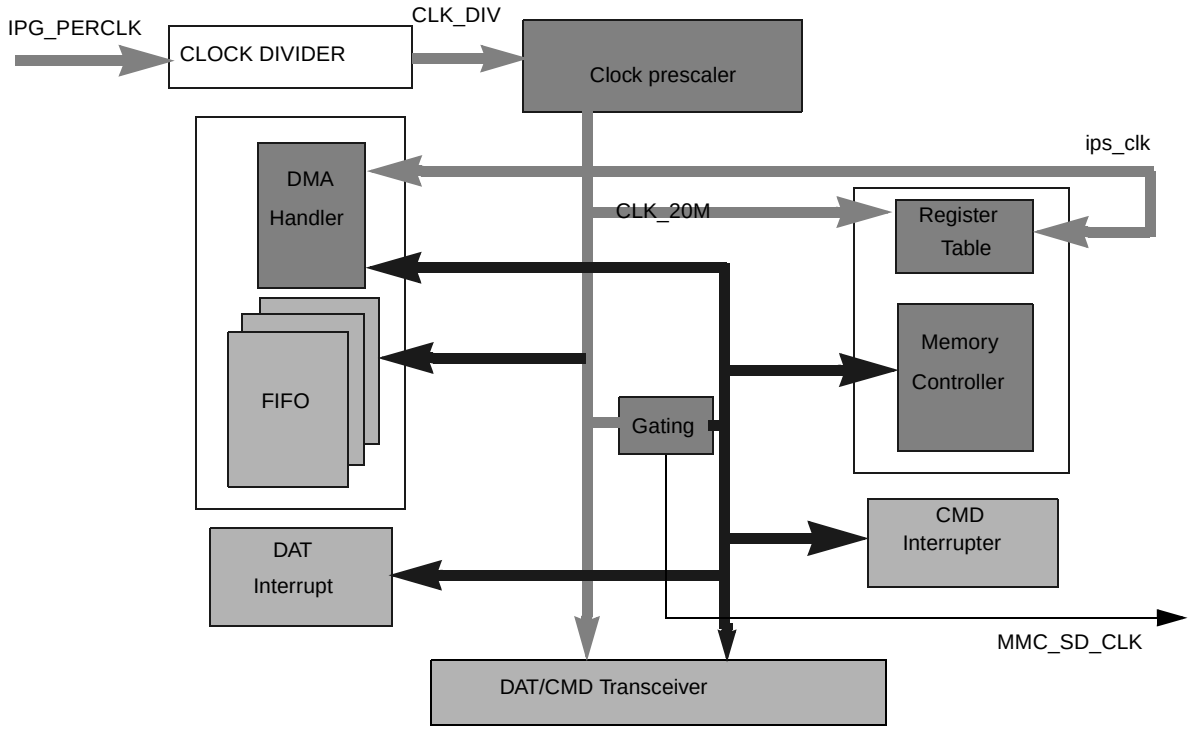


Figure 29-26. Clock used in SDHC

The controller controls the rate of the host main clock and checks whether it is on or off. The clock is turned off by setting the bit[0] of the STR_STP_CLK register and is turned on by setting the bit[1] of the STR_STP_CLK. To change the clock rate, the application must write a new value in the CLK_RATE register.

29.4.9 DAT/CMD Transceiver

The transceiver unit is designed to do the following:

- Control the I/O buffers.
- Synchronize the input data to the system clock domain.

The bidirectional signals CMD and DAT are each connected by OE, OEB, IN, and OUT.

OUT and OEB are used for the high-impedance state output buffer, while OE and IN are used for the input buffer. The use of OE allows the input to be disabled during floating and minimizes the current consumption.

The data buffers are in the system clock domain but the input data is in the MMC_SD_CLK clock domain. The transceiver will synchronize the input data to system clock domain.

29.5 Initialization/Application of SDHC

This section provides initialization and application information for SDHC.

All communication between system and cards are controlled by the host. The host sends commands of two types: broadcast and addressed (point-to-point) commands.

Broadcast commands are intended for all cards, such as: Go_Idle_State, Send_Op_Cond, All_send_CID, and Set_relative_Addr. In broadcast mode, all cards are in the open-drain mode to avoid bus contention. If the socket supports only one card, the broadcast command is similar as the point-to-point command.

After the broadcast command Set_relative_Addr is issued, the cards enter standby mode. Addressed type commands are used from this point. In this mode, the CMD/DAT I/O returns to push-pull mode to have the driving capability for maximum frequency operation.

The MMC and the SD are similar products. Other than the 4x bandwidth, they are programmed similarly. The following example shows how to “initialize” and perform “content access” and “content protection” on the cards.

Example 29-1 shows an example of a program-like function.

Example 29-1. MMC_SD_CLK Control

The MMC_SD_CLK clock to the card is controlled by STR_STP_CLK register. The clock should be supplied to the card for the following tasks:

- Submitting a command to the card and receiving a response
- Transferring data between the SDHC and the card
- Detecting an interrupt from an SD card in 4-bit

The steps below show how to start the MMC_SD_CLK to card:

1. Write 0x2 to STR_STP_CLK register.
2. Poll STATUS[8], then wait until the clock starts.

The steps below show how to stop the MMC_SD_CLK to the card:

1. Write 0x1 to the STR_STP_CLK register.
2. Poll STATUS[8], then wait until the clock is stopped.

NOTE

The user should not change the ipg_clk_gating_disable and ipg_perclk_gating_disable bits when starting and stopping MMC_SD_CLK. And if the SDHC clock gating features is used by the system, the user does not need to stop the MMC_SD_CLK by software.

29.5.1 Command Submit—Response Receive Basic Operation

This section shows the program flow used to submit a command to the card(s). The commands are as follows:

- <command_no>—the targeted command
- <arg_no>—the corresponding argument
- <cmd_dat_cont>—the command configuration required
- <int_Control_value>—the interrupt control used in the user program.

The steps below show how to submit a command to the card:

1. Start MMC_SD_CLK if it is stopped
2. Enable END_CMD_RESP interrupt by write 0 to INT_CNTR[2].
3. Set command number to CMD register.
4. Set the command argument to ARG register.
5. Set the appropriate value to command data control register (CMD_DAT_CONT).
6. Wait for the end command response interrupt and check for the response CRC/time-out status.
7. Read the response from the response FIFO to check the response. Read three or eight times from the response FIFO access register, depending on whether the response is 48-bit or 136-bit.

8. Stop the MMC_SD_CLK if the clock is not needed. (If there is data transfer following the command/response transfer, the clock should not be stopped until the data transfer completes.)

This following is a function defining command submission. This function will be used in the examples in the following subsection:

```
send_cmd_wait_resp(command_no, arg, cmd_dat_cont, int_cntr_value)
{
write_reg(STR_STP_CLK, 0x02);//1. to start mmc_sd_clk
read_reg(STATUS);
while(!STATUS[8]) Read_reg(STATUS); // 2. Wait till the clock has started
write_reg(COMMAND, <command_no>);// 3. configure the CMD
write_reg(ARG, <arg_no>);//4. configure the command argument
write_reg(CMD_DAT_CONT, <cmd_dat_cont>);//5. configure the command data control register,
writing to this register will trigger SDHC send command to the card.
while(irq_status);// 6. Wait interrupt (End Command Response)
Write_reg(INT_CNTR, <int_cntr_value>);//7. negate the irq request from SDHC
read_reg(STATUS); //8. Check whether the interrupt is an End_CMD_RES or a response time out or a
CRC error.
write_reg(STR_STP_CLK, 0x001);// 9. Stop the card clock if the clock is not needed any longer for
this cmd
read_reg(STATUS);
while(STATUS[8]) Read_reg(STATUS); // 10. Wait till the clock is stopped, command - response end.
read_reg(RES_FIFO); // 11. read the response fifo to determine if the command has a response
}
```

29.5.2 Card Identification Mode

When a card is inserted to the socket or the card was reset by the host, the host needs to validate the operation voltage range, identify the cards, and request the cards to publish the relative card address (RCA) or to set the RCA for the MMC cards. All data communications in the card identification mode use the command line (CMD) only.

29.5.2.1 Card Detect

Figure 29-27 shows a flow diagram showing the detection of card using the host controller.

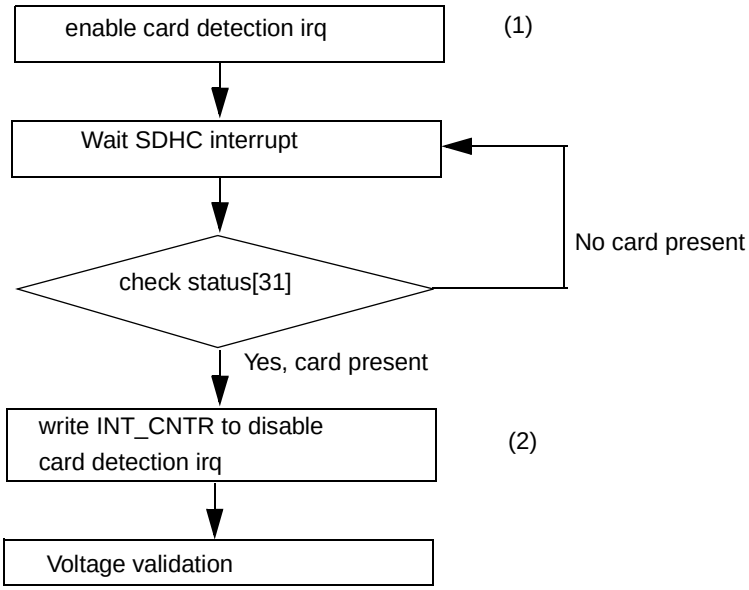


Figure 29-27. Flow Diagram for Card Detection

- Write ‘1’ to INT_CNTR[15] to enable card detection interrupt.
- Write ‘0’ to INT_CNTR[15] to disable card detection interrupt.

29.5.2.2 Reset

The host consists of three types of reset:

- Hardware reset (card and host), which is driven by the POR (power on reset).
- Software reset (host only), which is preceded by the write operation on register “STR_STP_CLK.” Follow the recommended sequence as specified in Section 29.3.3.3, “SDHC Clock Rate Register (CLK_RATE).” The reset resets all of the SDHC registers, but will not reset the card. The card reset is through CMD0. Once the user applies the software reset to SDHC, it should also be using CMD0 to reset the card in case the card is in an unknown state.
- Card reset (card only). The command—“Go_Idle_State” (CMD0)—is the software reset command for both the MMC and the SD memory card. This sets each card into idle state regardless of the current card state. When used as an SD I/O card, CMD52 is used to write I/O reset in CCCR. The cards are initialized with a default relative card address (RCA=0x0000) and with a default driver stage register setting (lowest speed, highest driving current capability).

After the card is reset, the host needs to validate the voltage range of the card. Figure 29-28 shows the software flow to reset both the SDHC and the card.

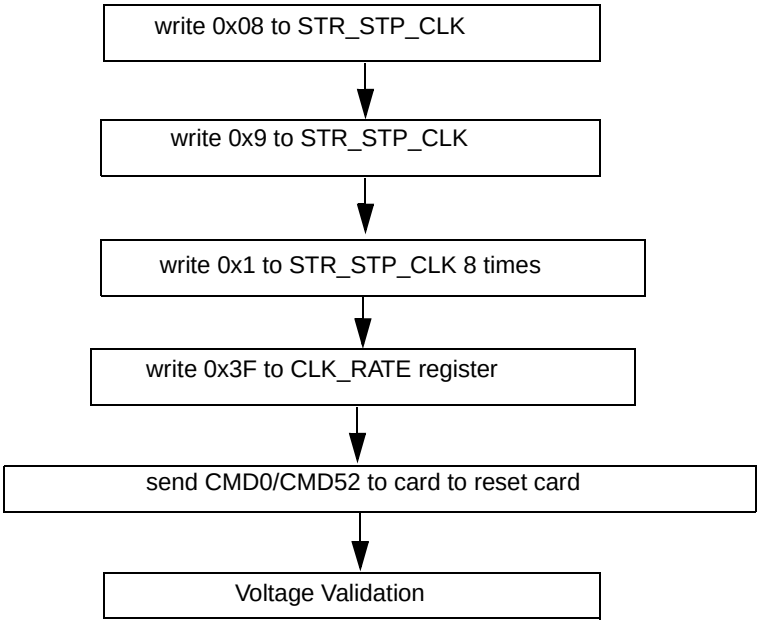


Figure 29-28. Flowchart for Reset of SDHC and SD I/O Card

```
software_reset()
{
write_reg(STR_STP_CLK, 0x8);
write_reg(STR_STP_CLK, 0x9);// 1. reset the SDHC host;
write_reg(STR_STP_CLK, 0x1);
write_reg(STR_STP_CLK, 0x1);
write_reg(STR_STP_CLK, 0x1);
write_reg(STR_STP_CLK, 0x1);
write_reg(STR_STP_CLK, 0x1);
write_reg(STR_STP_CLK, 0x1);
write_reg(STR_STP_CLK, 0x1);
write_reg(STR_STP_CLK, 0x1);// 2. write 0x1 to STR_STP_CLK 8 times;
write_reg(CLK_RATE, 0x3F);// 3. Set the lowest clock for initialization
write_reg(READ_TO, 0x2DB4);// 4. set READ timeout register
send_cmd_wait_resp(CMD_GO_IDLE_STATE, 0x0,0x80, 0x40); //5. reset the card with CMD0
}
```

29.5.2.3 Voltage Validation

All cards must be able to establish communications with the host using any operation voltage in the maximum allowed voltage range specified in this standard. However, the supported minimum and maximum values for Vdd are defined in the operation conditions register (OCR) and might not cover the entire range. Cards that store the CID and CSD data in the preload memory are able to communicate the information only under data transfer Vdd conditions. That means if the host and card have non-compatible Vdd ranges, the card is not able to complete the identification cycle, nor be able to send CSD data.

Commands such as Send_Op_Cont (CMD1 for MMC), SD_Send_Op_Cont (CMD41 for SD Memory), and IO_Send_Op_Cont (CMD5 for SD I/O) are designed to provide a mechanism to identify and reject

cards that do not match the Vdd range desired by the host. This is accomplished by the host sending the required Vdd voltage window as the operand of this command. Cards that cannot perform data transfer in the specified range must detach themselves from further bus operations and go into the inactive state.

By omitting the voltage range in the command, the host can query each card and determine the common voltage range before sending out-of-range cards into the inactive state. This query should be used if the host is able to select a common voltage range or if a notification to the application of non-usable cards in the stack is desired.

The following steps show how to perform voltage validation when a card is inserted:

```
voltage_validation(voltage_range_arguement)
{
send_cmd_wait_resp(IO_SEND_OP_COND, 0x0, 0x04, 0x40); // CMD5, send SDIO operation voltage,
command argument is zero
if(End Command Response true & No. of IO functions> 0)// it is SDIO and have IO function
{IORDY = 0;
while(!(IORDY in I/O ORC response)) {// set voltage range for each IO
send_cmd_wait_resp(IO_SEND_OP_COND, voltage_range_arguement, 0x04, 0x40);}
if(Memory Present flag true)
Card = combo; // that is, SDIO + SD Memory, need to set operation voltage to memory portion as
well
send_cmd_wait_resp(APP_CMD, 0x0, 0x01, 0x40);// CMD55, Application Command follows
send_cmd_wait_resp(SD_APP_OP_COND, voltage_range_arguement, 0x01, 0x40);//ACMD41
else
Card = sdio;

// if No response to CMD5 IO_SEND_OP_COND or No. of IO Function is zero in response
else// the card should be SD or MMC
{send_cmd_wait_resp(APP_CMD, 0x0, 0x01, 0x40);// CMD55, Application Command follows
if(End Command Response true and no response timeout)
{send_cmd_wait_resp(SD_APP_OP_COND, voltage_range_arguement, 0x01, 0x40); // ACMD41, SD card
found
Card = sd;
}
else // the card have no response to APP_CMD, it is not SD card
{send_cmd_wait_resp(SEND_OP_COND, voltage_range_arguement, 0x01, 0x40); //CMD1, MMC card found
if(End Command Response true and no response timeout)
{Card = mmc;}
else{ Card = No card or failed contact;}
}
}
```

29.5.2.4 Card Registry

Card registry between MMC and SD card is different.

For the SD card, the identification process starts at clock rate FoD (below 400 kHz for most of the card), as defined by the card specification (FoD is the initialization clock frequency defined by the card specification). After the bus is activated, the host requests the card to send valid operation conditions. The response to ACMD41 is the operation condition register of the card. The same command is sent to all of the new cards in the system. Incompatible cards are put into the inactive state. The host then issues the command, All_Send_CID (CMD2), to each card to get its unique card identification (CID) number. Cards

that are currently unidentified (that is, in ready state), send their CID number as the response. After the CID is sent by the card, the card goes into the identification state.

The host then issues Send_Relative_Addr (CMD3), requesting that the card publish a new relative card address (RCA), which is shorter that CID. This ID is used to address the card for future data transfer operations. Once the RCA is received, the card state changes to the stand-by state. At this point, if the host wants the card to have an alternative RCA number, it may ask the card to publish a new number by sending another Send_Relative_Addr command to the card. The last published RCA is the actual RCA of the card.

The host repeats the identification process with CMD2 and CMD3 for each card in the system.

For the MMC operation, the host starts the card identification process in open-drain mode with the identification clock rate FoD. (FoD is the initialization clock frequency defined by the card specification.) The open-drain driver stages on the CMD line allow parallel card operations during card identification. After the bus is activated, the host requests the cards to send their valid operation conditions (CMD1). The response to CMD1 is the ‘wired OR’ operation on the condition restrictions of all cards in the system. Incompatible cards are sent into inactive state.

The host then issues the broadcast command All_Send_CID (CMD2), asking all cards for their unique card identification (CID) number. All unidentified cards (that is, those which are in ready state) simultaneously start sending their CID numbers serially, while bit-wise is monitoring their outgoing bitstream. These cards, for which outgoing CID bits do not match the corresponding bits on the command line in any one of the bit periods, stop sending their CIDs immediately. They must wait for the next identification cycle. Since the CID is unique for each card, only one card can successfully send its full CID to the host. This card then goes into identification state.

Thereafter, the host issues Set_Relative_Addr (CMD3) to assign a relative card address (RCA) to this card. Once the RCA is received, the card state changes to the stand-by state, does not react to further identification cycles, and its output switches from open-drain to push-pull. The host repeats the process, that is, CMD2 and CMD3, until the host receives the time-out condition to recognize completion of the identification process.

```
card_registry()
{
while (ResponseTO from STATUS){
if(card==combo or sdio)
{
send_cmd_wait_resp(SET_RELATIVE_ADDR, 0x00, 0x01, 0x40);   //card publish the RCA in response
rca = SDIO_RCA = address from response FIFO;
}
else if(card==sd)
{
send_cmd_wait_resp(ALL_SEND_CID, 0x00, 0x02, 0x40);
send_cmd_wait_resp(SET_RELATIVE_ADDR, 0x00, 0x01, 0x40); //card publish the RCA in response
rca = SD_RCA = address from response FIFO;
}
else if(card==mmc)
{
send_cmd_wait_resp(ALL_SEND_CID, 0x00, 0x00, 0x02, 0x40);
rca = MMC_RCA = 0x1;
send_cmd_wait_resp(SET_RELATIVE_ADDR, MMC_RCA_arguement, 0x01, 0x40);
}
else
```

```
exit due to card not identified;
}
send_cmd_wait_resp(SELECT_CARD, RCA_arguement, 0x41, 0x40);
}
```

29.5.3 Card Access

29.5.3.1 Block Access—Block Write and Block Read

29.5.3.1.1 Block Write

During block write (CMD24–27), one or more blocks of data are transferred from the host to the card with a CRC appended to the end of each block by the host. A card supporting block write will always be able to accept a block of data defined by WRITE_BL_LEN. If the CRC fails, the card shall indicate the failure on the DAT line. The transferred data is discarded and not written, and all further transmitted blocks (in multiple-block write mode) will be ignored.

If the host uses partial blocks for which accumulated length is not block aligned and block misalignment is not allowed (CSD parameter WRITE_BLK_MISALIGN is not set), the card detects the block misalignment error and aborts programming before the beginning of the first misaligned block. The card sets the ADDRESS_ERROR error bit in the status register, and, while ignoring all further data transfer, waits in the Receive-data-State for a stop command. The write operation is aborted if the host tries to write over a write-protected area. In this case, however, the card sets the WP_VIOLATION bit.

Programming of the CID and CSD registers does not require a previous block length setting. The transferred data is also CRC-protected. If a part of the CSD or CID register is stored in ROM, this unchangeable part must match the corresponding part of the receive buffer. If this match fails, the card reports an error and does not change any register contents. Some cards may require long and unpredictable times to write a block of data. After receiving a block of data and completing the CRC check, the card begins writing and holds the DAT line low if its write buffer is full and unable to accept new data from a new WRITE_BLOCK command. The host may poll the status of the card with a SEND_STATUS command (CMD13) at any time, then the card responds with its status. The status bit READY_FOR_DATA indicates whether the card can accept new data or whether the write process is still in progress. The host may deselect the card by issuing CMD7 (to select a different card), which places the card into the disconnect state and releases the DAT line without interrupting the write operation. When re-selecting the card, it reactivates the busy indication by pulling DAT to low if programming is still in progress and the write buffer is unavailable.

The software flow to write to the card with DMA enable is as follows:

1. Start MMC_SD_CLK if it is stopped.
2. Check the card status, wait until card is ready for data.
3. For SD/MMC, set the card block length, using SET_BLOCKLEN (CMD16).
4. Set the SDHC block length register to be same as block length set to the card in Step 2.

— For SDIO, if the CMD53 is in byte mode, the SDHC block length register should be set according to bytes count in CMD53; if the CMD53 is in block mode, the SDHC block length register should be set according to the block size in CCCR registers.

5. Set SDHC number block register (NOB), nob is 1 for single block write or CMD53 in byte mode for SDIO.
6. Disable the buffer ready interrupt, configure the DMA setting, and enable the SDHC DMA channel:
   a) Write '0' to bit[3] of INT_CNTR register in SDHC to disable buffer write ready interrupt.
   b) Set the DMA destination to be SDHC_Buffer access register.
   c) Set the DMA destination port size to be 32-bit.
   d) Set the DMA burst length to be 16 bytes in 1-bit mode or 64 bytes in 4-bit mode.
   e) Set the DMA transfer count to be number of bytes that is a multiple of the Block_length(nob*blk_len = total number of bytes).
7. Check the card status and wait until the card is ready for data.
8. Set the SDHC CMD register to any of the following:
   — CMD24(WRITE_BLOCK)
   — CMD25(WRITE_MULTIPLE_BLOCK)
   — CMD53 in byte mode or block mode
9. Set the SDHC CMD argument register.
10. Set the SDHC command data control register.
11. Wait for end command response and check if there is any CRC error or timeout error.
12. Wait for DMA done.
13. Check for Write_OP_DONE and check the status bit to see if a write CRC error occurred.
14. Send STOP_TRANSMISSION command to the card if the write command is WRITE_MULTIPLE_BLOCK (CMD25).
15. Stop the MMC_SD_CLK. (Finished the write operation.) This step is optional.

If the write operation is without DMA, the system needs to write data to the buffer through buffer write ready interrupt or by polling the buffer write ready status bit (STATUS[6]: BUF_WR_RDY). For high performance, data transfer using DMA is preferred.

29.5.3.1.2 Block Read

For block reads, the basic unit of data transfer is a block for which the maximum size is defined in the CSD (READ_BL_LEN). If READ_BL_PARTIAL is set, smaller blocks for which the starting and ending addresses are entirely contained within one physical block (as defined by READ_BL_LEN) can also be transmitted. A CRC is appended to the end of each block, ensuring data transfer integrity. CMD17 (READ_SINGLE_BLOCK) initiates a block read. After completing the transfer, the card returns to the transfer state. CMD18 (READ_MULTIPLE_BLOCK) starts a transfer of several consecutive blocks. Blocks are continuously transferred until a stop command is issued. If the host uses partial blocks for which accumulated length is not block aligned and block misalignment is not allowed, the card detects a

block misalignment at the beginning of the first mis-aligned block, sets the ADDRESS_ERROR error bit in the status register, aborts transmission, and waits in the data state for a STOP command.

The following is the software flow to write to card with DMA enable:

1. Start MMC_SD_CLK if it is stopped.
2. Check the card status and wait until the card is ready for data.
3. For SD/MMC, set the card block length, using SET_BLOCKLEN (CMD16).
4. Set the SDHC block length register to be same as block length set to the card in Step 3.
   — For SDIO, if the CMD53 is in byte mode, the SDHC block length register should be set according to bytes count in CMD53. If the CMD53 is in block mode, the SDHC block length register should be set according to the block size in CCCR registers.
5. Set the SDHC number block register (NOB) to 1 for single block write or CMD53 in byte mode for SDIO.
6. Disable the buffer ready interrupt, configure the DMA setting, and enable the SDHC DMA channel:
   a) Write ‘0’ to bit[4] of INT_CNTR register in SDHC to disable the buffer read ready interrupt.
   b) Set the DMA source to be SDHC_Buffer access register.
   c) Set the DMA source port size to be 32-bit.
   d) Set the DMA burst length to be 16 bytes in 1-bit mode or 64 bytes in 4-bit mode.
   e) Se the DMA transfer count to be number of bytes that is a number of blocks multiple of the Block_length(nob*blk_len).
7. Check the card status and wait until the card is ready for data.
8. Set the SDHC CMD register to be CMD17(READ_SINGLE_BLOCK) or CMD18 (READ_MULTIPLE_BLOCK) or CMD53 in byte mode or block mode.
9. Set SDHC CMD argument register.
10. Set SDHC command data control register.
11. Wait for the END_CMD_RESP interrupt and check the response FIFO, and check CRC error and timeout error.
12. Wait for DMA done.
13. Check for READ_OP_DONE and check the status bit to see if read CRC error occurred.
14. Send STOP_TRANSMISSION command to the card if the read command is READ_MULTIPLE_BLOCK (CMD18).
15. Stop the MMC_SD_CLK, (Finished the read operation.)

If the read transfer operation does not use DMA, the system needs to fetch data out of the data buffer through utilizing the buffer read ready interrupt or by polling the buffer read ready status bit (STATUS[7]: BUF_READ_RDY). For high performance, data transfer using DMA is preferred.

29.6 Commands for MMC/SD/SDIO

Table 29-20 shows the list of commands for MMC/SD/SDIO.

Refer to corresponding card specifications for details about the command information.

Table 29-20. Commands for MMC/SD/SDIO

CMD INDEX	Type	Argument	Response	Abbreviation	Description
CMD0	bc	[31:0] stuff bits	—	GO_IDLE_STATE	Resets all MMC and SD memory cards to idle state.
CMD1	bcr	[31:0] OCR without busy	R3	SEND_OP_COND	Asks all MMC and SD memory cards in idle state to send their operation conditions register contents in the response on the CMD line.
CMD2	bcr	[31:0] stuff bits	R2	ALL_SEND_CID	Asks all cards to send their CID numbers on the CMD line.
CMD3	ac	[31:6] RCA [15:0] stuff bits	R1 R6(SDIO)	SET_RELATIVE_ADDR	Assigns relative address to the card.
CMD4	bc	[31:0] DSR [15:0] stuff bits	—	SET_DSR	Programs the DSR of all cards.
CMD5	bc	[31:0] OCR without busy	R4	IO_SEND_OP_COND	Asks all SD I/O cards in idle state to send their operation conditions register contents in the response on the CMD line.
CMD6	Reserved				
CMD7	ac	[31:6] RCA [15:0] stuff bits	R1b	SELECT/DESELECT_CARD	Command toggles a card between the stand-by and transfer states or between the programming and disconnect states. In both cases, the card is selected by its own relative address and gets deselected by any other address; address 0 deselects all.
CMD8	Reserved				
CMD9	ac	[31:6] RCA [15:0] stuff bits	R2	SEND_CSD	Addressed card sends its card-specific data (CSD) on the CMD line.
CMD10	ac	[31:6] RCA [15:0] stuff bits	R2	SEND_CID	Addressed card sends its card-identification (CID) on the CMD line.
CMD11	adtc	[31:0] data address	R1	READ_DAT_UNTIL_STOP	MMC reads data stream from the card, starting at the given address, until a STOP_TRANSMISSION follows.
CMD12	ac	[31:0] stuff bits	R1b	STOP_TRANSMISSION	Forces the MMC/SD memory card to stop transmission.
CMD13	ac	[31:6] RCA [15:0] stuff bits	R1	SEND_STATUS	Addressed MMC/SD card sends its status register.

Table 29-20. Commands for MMC/SD/SDIO (continued)

CMD INDEX	Type	Argument	Response	Abbreviation	Description
CMD14	Reserved				
CMD15	ac	[31:6] RCA [15:0] stuff bits	—	GO_INACTIVE_STATE	Sets the card to inactive state to protect the card stack against communication breakdowns.
CMD16	ac	[31:0] block length	R1	SET_BLOCKLEN	Sets the block length (in bytes) for all following block commands (read and write). Default block length is specified in the CSD.
CMD17	adtc	[31:0] data address	R1	READ_SINGLE_BLOCK	Reads a block of the size selected by the SET_BLOCKLEN command.
CMD18	adtc	[31:0] data address	R1	READ_MULTIPLE_BLOCK	Continuously transfers data blocks from card to host until interrupted by a stop command.
CMD19	Reserved				
CMD20	adtc	[31:0] data address	R1	WRITE_DAT_UNTIL_STOP	MMC card writes data stream from the host, starting at the given address, until a STOP_TRANSMISION follows.
CMD21-23	Reserved				
CMD24	adtc	[31:0] data address	R1	WRITE_BLOCK	Writes a block of the size selected by the SET_BLOCKLEN command.
CMD25	adtc	[31:0] data address	R1	WRITE_MULTIPLE_BLOCK	Continuously writes blocks of data until a STOP_TRANSMISSION follows.
CMD26	adtc	[31:0] stuff bits	R1	PROGRAM_CID	Programming of the MMC card identification register. This command to be issued only once per card. The card contains hardware to prevent this operation after the first programming. Normally, this command is reserved for the manufacturer.
CMD27	adtc	[31:0] stuff bits	R1	PROGRAM_CSD	Programming of the programmable bits of the CSD.
CMD28	ac	[31:0] data address	R1b	SET_WRITE_PROT	If the card has write-protection features, this command sets the write-protection bit of the addressed group. The properties of write-protection are coded in the card specific data (WP_GRP_SIZE).

Table 29-20. Commands for MMC/SD/SDIO (continued)

CMD INDEX	Type	Argument	Response	Abbreviation	Description
CMD29	ac	[31:0] data address	R1b	CLR_WRITE_PROT	If the card provides write-protection features, this command clears the write-protection bit of the addressed group.
CMD30	adtc	[31:0] write protect data address	R1	SEND_WRITE_PROT	If the card provides write-protection features, this command asks the card to send the status of the write-protection bits.
CMD31	Reserved				
CMD32	ac	[31:0] data address	R1	ERASE_WR_BLK_START	Sets the address of the first sector of the erase group.
CMD33	ac	[31:0] data address	R1	ERASE_WR_BLK_END	Sets the address of the last sector of the continuous range of the erase group.
CMD34	ac	[31:0] data address	R1	UNTAG_SECTOR	Removes one previously selected sector from the erase selection.
CMD35	ac	[31:0] data address	R1	TAG_ERASE_GROUP_START	Sets the address of the first erase group within a range to be selected for erase.
CMD36	ac	[31:0] data address	R1	TAG_ERASE_GROUP_END	Sets the address of the last erase group within a continuous range to be selected for erase.
CMD37	ac	[31:0] data address	R1	UNTAG_ERASE_GROUP	Removes one previously selected erase group from the erase selection.
CMD38	ac	[31:0] stuff bits	R1b	ERASE	Erases all previously selected sectors.
CMD39	ac	[31:0] RCA [15] register write flag [14:8] register address [7:0] register data	R4	FAST_IO	Used to write and read 8-bit (register) data fields. The command addresses a card and a register and provides the data for writing if the write flag is set. The R4 response contains data read from the address register. This command accesses application dependent registers which are not defined in MMC standard.
CMD40	bcr	[31:0] stuff bits	R5	GO_IRQ_STATE	Sets the system into interrupt mode.
CMD41	Reserved				

Table 29-20. Commands for MMC/SD/SDIO (continued)

CMD INDEX	Type	Argument	Response	Abbreviation	Description
CDM42	adtc	[31:0] stuff bits	R1b	LOCK_UNLOCK	Used to set/reset the password or lock/unlock the card. The size of the data block is set by the SET_BLOCK_LEN command.
CMD43–51	Reserved				
CMD52	—	[31] R/W flag [30:28] Function Number [27] RAW (read after write) flag [26] stuff bit [25:9] Register Address [8] Stuff bit [7:0] Write Data/stuff bits	R5	IO_RW_DIRECT	Used to access a single register within the total 128 Kbytes of register space in any I/O function.
CMD53	—	[31] R/W flag [30:28] Function Number [27] Block mode [26] OP Code [25:9] Register Address [8:0] Byte/Block Count	R5	IO_RW_EXTENDED	Used to access a multiple I/O register with a single command, It allows the reading or writing of a large number of I/O registers.
CMD54	Reserved				
CMD55	ac	[31:16] RCA [15:0] stuff bits	R1	APP_CMD	Indicates to the card that the next command is an application-specific command rather than a standard command.
CMD56	adtc	[31:1] stuff bits [0]: RD/WR	R1b	GEN_CMD	Used either to transfer a data block to the card or to get a data block from the card for general-purpose/application-specific commands. The size of the data block is set by the SET_BLOCK_LEN command.
CMD57–63	Reserved				
ACMDs are preceded with APP_CMD command. (Commands listed below are used for SD only. Other unlisted SD commands are not supported in this module.)					

Table 29-20. Commands for MMC/SD/SDIO (continued)

CMD INDEX	Type	Argument	Response	Abbreviation	Description
ACMD6	ac	[31:2] stuff bits [1:0] bus width	R1	SET_BUS_WIDTH	Defines the SD memory card data bus width (‘00’=1-bit or ‘10’=4-bit bus) to be used for data transfer. The allowed data bus widths are given in the SCR register.
ACMD13	adtc	[31:0] stuff bits	R1	SD_STATUS	Sends the SD memory card status.
ACMD22	adtc	[31:0] stuff bits	R1	SEND_NUM_WR_SECTORS	Sends the number of the written (without errors) sectors. Responds with 32-bit + CRC data block.
ACMD23	ac	[31:23] stuff bits [22:0] Number of blocks	R1	SET_WR_BLK_ERASE_COUNT	Set the number of write blocks to be pre-erased before writing (to be used for faster Multi-block write command).
ACMD41	bcr	[31:0] OCR	R3	SD_APP_OP_COND	Requests the accessed card to send its operating condition register (OCR) content in the response on the CMD line.
ACMD42	ac	[31:1] stuff bits [0] set_cd	R1	SET_CLR_CARD_DETECT	Connects/disconnects the 50 kΩ pull-up resistor on CD/DAT3 of the card.
ACMD51	adtc	[31:0] stuff bits	R1	SEND_SCR	Reads the SD configuration register (SCR).

Chapter 30
Subscriber Identification Module (SIM)

The Subscriber Identification Module (SIM) is designed to facilitate communication to SIM cards or Eurochip pre-paid phone cards. See the SIM block diagram in Figure 30-1. The SIM module has two ports that can be used to interface with the various cards. The interface with the Microcontroller Unit (MCU) is a 32-bit connection as described in the reference document IP Bus Specification.

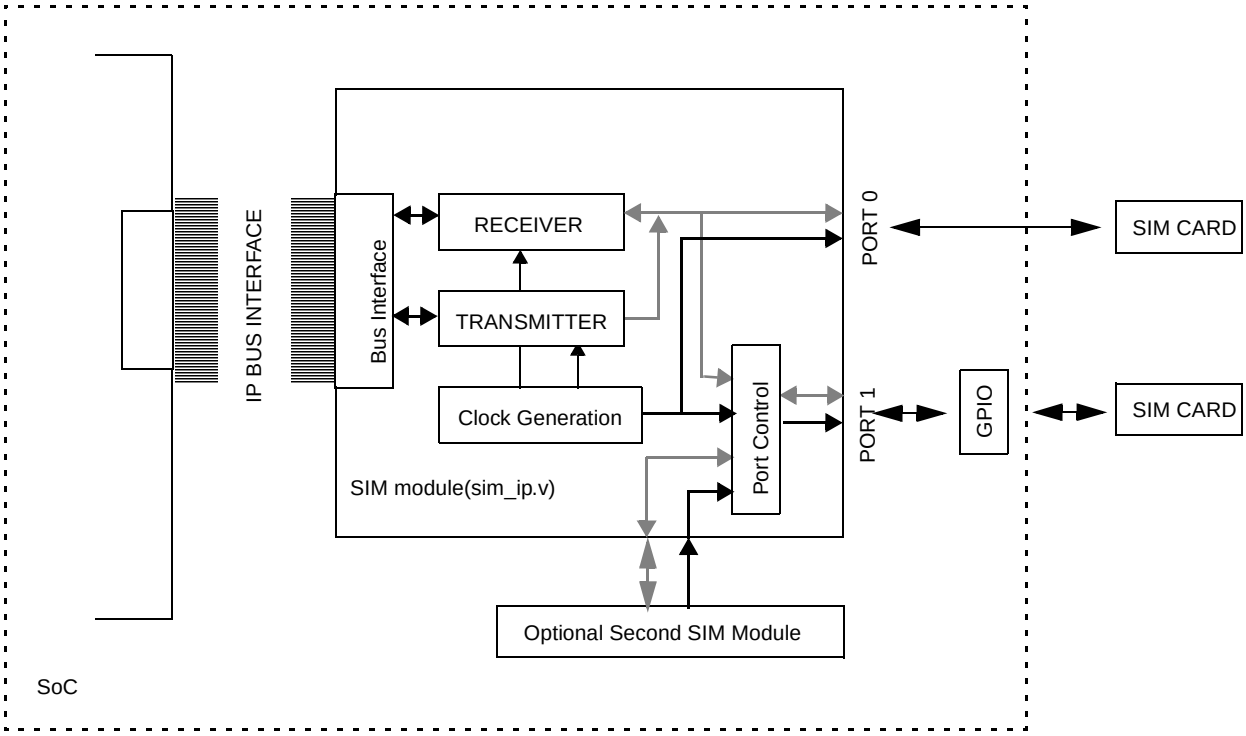


Figure 30-1. SIM Block Diagram

30.1 Introduction

The SIM design is summarized in the following sections.

30.1.1 SIM Features

The SIM design is summarized in the following sections:

- Section 30.1.3, “SIM Bus Interface Overview”
- Section 30.1.4, “SIM Clock Generator Overview”
- Section 30.1.5, “SIM Transmitter Overview”

- Section 30.1.6, “SIM Receiver Overview”
- Section 30.1.7, “SIM Port Control Overview”
- Section 30.1.8, “SIM General Purpose Counter Overview”
- Section 30.1.9, “SIM LRC Block Overview”
- Section 30.1.10, “SIM CRC Block Overview”

30.1.2 SIM Modes of Operation

The SIM module I/O interface can be operated in one of three modes of operation summarized below.

- Two wire interface. In this mode both the IC pin RX and IC pin TX are used to interface to the SmartCard. This is activated by resetting the 3VOLT bit in the port control register to a “0”.
- External one wire interface. In this mode the IC pins RX and TX are tied together external to the IC and routed to the SmartCard. The 3VOLT bit in the port control register is reset to a “0” and the OD bit in the OD_CONFIG register is set to a “1”. For this interface to work properly the IC pin (RX-TX) must be pulled high by a resistor. The value should be selected small enough to give a fast enough rise time.
- Internal one wire interface. In this mode the IC pin TX is routed to the SmartCard. The receive pin RX is connected to the TX pin internal to the IC. The 3VOLT bit in the port control register is reset to a “1” and the OD bit in the OD_CONFIG register is set to a “1”. For this interface to work properly the IC pin TX must be pulled high by a resistor. The value should be selected small enough to give a fast enough rise time.

30.1.3 SIM Bus Interface Overview

The SIM module is designed with a separate block that encompasses the interface to the IP Bus. The bus interface is built in such a way as to be easily modified to other bus definitions. The SIM module is to be considered a 32-bit peripheral. To avoid Endian issues with register access, all read/writes should be done as word access(32 bit). The bus interface has the following features:

- Peripheral Address/chip select decoding
- Illegal Address generation
- Dynamic wait state generation (tied inactive)
- Register organization
- Register bit implementations

See Table 30-1 for summary of the SIM top level interrupts.

Table 30-1. SIM Top Level Interrupt Summary

Flag	Interrupt Sources	Interrupt Masks	Description
SIMIRQ_N	XTE, OEF, SDI0, SDI1, TFO, GPCNT, BWT, BGT, CWT, RTE	XTM, OIM, SDIM0, SDIM1, TFOM, GPCNTM, BWTM, BGTM, CWTM, RTM	SIM General Interrupt
SIMDAT_N	TC, ETC, TFE, RDRF, TDTF	TCIM, ETCIM, TFEIM, RIM, TDTFM	SIM Data Interrupt

30.1.4 SIM Clock Generator Overview

The SIM module contains a block designed specifically for generating the clocks used internal to the SIM module, and the clocks provided to the SIM cards. The clock generator has the following features:

- Divisible by 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 26, 32, 38, 46, 56 for SIM card clock generation (50% duty cycle)
- Receive clock generation (/1, /2, /4, /8, /16, /31) from first stage
- Transmit clock generation (/12, /16) from second stage
- Programmable Divisor for Receive clock generation

There are no interrupt sources generated by the SIM clock generator block.

30.1.5 SIM Transmitter Overview

The SIM Transmitter block contains the transmit state machine, transmit shift register, and transmit FIFO. The transmitter has the following features.

- 16 bytes deep transmit FIFO
- Automatic NACK generation on parity and overrun errors
- Hardware data format support (inverse convention or direct convention)
- Retransmission of data upon SIM card NACK request with programmable maximum threshold of retransmissions
- Programmable guard time between transmitted bytes
- Six interrupt sources (see Table 30-2)

Table 30-2. SIM Transmitter Interrupt Summary

Flag	Flag Register	Mask	Mask Register	Description
TC	XMT_STATUS	TCIM	INT_MASK	Transmit complete
ETC	XMT_STATUS	ETCIM	INT_MASK	Early transmit complete
TFE	XMT_STATUS	TFEIM	INT_MASK	Transmit FIFO empty
XTE	XMT_STATUS	XTM	INT_MASK	Transmit threshold error
TFO	XMT_STATUS	TFOM	INT_MASK	Transmit FIFO Overfill Error
TDTF	XMT_STATUS)	TDTFM	INT_MASK	Transmit Data Threshold

30.1.6 SIM Receiver Overview

The SIM Receiver block contains the receive state machine, receive FIFO, and control logic. The receiver has the following features.

- 32 bytes deep receive FIFO
- Decoding of initial character mode for setting data format
- Hardware data format support (inverse convention or direct convention)
- NACK detection

- 11 ETU character support
- Character Wait Time Counter
- Ten interrupt sources (see Table 30-3)

Table 30-3. SIM Receiver Interrupt Summary

Flag	Flag register	Mask	Mask register	Description
BGT	RCV_STATUS	BGTM	INT_MASK	Block Guard Time flag
BWT	RCV_STATUS	BWTM	INT_MASK	Block Wait Time flag
RTE	RCV_STATUS	RTM	INT_MASK	Receive NACK Threshold
CWT	RCV_STATUS	CWTM	INT_MASK	Character Wait Time flag
OEF	RCV_STATUS	OIM	INT_MASK	Overrun Error Flag
RDRF	RCV_STATUS	RIM	INT_MASK	Receive FIFO full

30.1.7 SIM Port Control Overview

The SIM module supports numerous port control functions necessary for SIM card interaction. Each of the two SIM card ports has the following I/O: CLK, $\overline{\text{RST}}$, DATA_RCV, DATA_XMT, SIMPD (SIM Presence Detect), and SVEN (SIM Vcc enable). The following list gives a number of the important features of the port logic.

- Open-drain or push pull DATA_XMT pin
- Port 1, port 0, or alternate Port selection
- SIM card presence detect with interrupt capability
- Deep sleep wake-up via SIM card presence detect interrupt
- Manual control of all SIM card interface signals
- Automatic power down of port logic on SIM card presence detect
- Two interrupt sources (see Table 30-4)

Table 30-4. SIM Card Detect Interrupts

Flag	Flag register	Mask	Mask Register	Description
SDI1	PORT1_DETECT	SDIM1	PORT1_DETECT	SIM detect interrupt for port 1
SDI0	PORT0_DETECT	SDIM0	PORT0_DETECT	SIM detect interrupt for port 0

30.1.8 SIM General Purpose Counter Overview

The SIM module provides a 16-bit counter that can be configured to count using clock sources from the card clock, receive clock, or transmitter clock (ETU Clock). A programmable 16-bit register is provided to compare with the counter for interrupt generation:

- Selectable clock source
- 16-bit counter and comparator
- Interrupt source (see Table 30-5)

Table 30-5. SIM General Purpose Counter Interrupts

Flag	Flag Register	Mask	Mask Register	Description
GPCNT	XMT_STAT	GPCNTM	INT_MASK	GPCNT Comparator Interrupt

30.1.9 SIM LRC Block Overview

The SIM module contains a block designed to generate Linear Redundancy Checking (LRC) information for both received and transmitted characters. The LRC block has the following features:

- 8-bit LRC value
- Valid LRC Detector

There are no interrupt sources generated by the SIM LRC block.

30.1.10 SIM CRC Block Overview

The SIM module contains a block designed to generate Cyclic Redundancy Checking (CRC) information for both received and transmitted characters. The CRC block has the following features:

- 16-bit CRC value
- 16-bit CRC Polynomial Generator
- Valid CRC Detector
- There are no interrupt sources generated by the SIM CRC block.

30.2 External Signal Description

30.2.1 Overview

See Table 30-6 for summary of the SIM module signals.

Table 30-6. Signal Properties

Name	Port	Function	Reset State	Pull-Up
External Signals (Pads)				
SIM_PIN_SCLK0	out	Clock for the smartcard on port0	0	Active
SIM_PIN_SRST0	out	Reset signal for port0	0	Active
SIM_PIN_SVEN0	out	Vcc enable signal for port0	0	Active
SIM_PIN_DATA0_TX_OUT	in/out	Transmit data signal for port0	1	Active/Passive
SIM_PIN_RCVD0_IN	in	Receive data signal for port0	—	—
SIM_PIN_SIMPD0	in	Card insertion detect signal for port0	—	—
SIM_PIN_SCLK1	out	Clock for the smartcard on port1	0	Active
SIM_PIN_SRST1	out	Reset signal for port1	0	Active

Table 30-6. Signal Properties (continued)

Name	Port	Function	Reset State	Pull-Up
SIM_PIN_SVEN1	out	Vcc enable signal for port1	0	Active
SIM_PIN_DATA1_TX_OUT	in/out	Transmit data signal for port1	1	Active/Passive
SIM_PIN_RCVD1_IN	in	Receive data signal for port1	—	—
SIM_PIN_SIMPD1	in	Card insertion detect signal for port1	—	—
Module Interrupts				
IPI_INT_SIM_IPB_INT	out	Active high SIM Interrupt. Composed of OEF, XTE, SDI1 and SDI0.	0	—
IPI_INT_ SIM_DATA_IRQ	out	Active high SIM Data Interrupt. Composed of TC, ETC, TFE, and RDRF.	0	—

30.2.2 SIM Detailed Signal Descriptions

30.2.2.1 SIM_PIN_SCLK0

The SIM_PIN_SCLK0 is an output from the chip to the SmartCard. This signal is the clock that the SIM module provides for the SmartCard. Typical frequencies are 1 MHz to 5 MHz. This clock is 372 times the data rate that is on pin SIM_PIN_DATA0_TX_OUT. There is no required timing relationship between this clock signal and any of the other data signals. This is because of the asynchronous nature of the protocol.

30.2.2.2 SIM_PIN_SRST0

The SIM_PIN_SRST0 is the reset signal from the SIM to the SmartCard.

30.2.2.3 SIM_PIN_SVEN0

The SIM_PIN_SVEN0 is the SmartCard power supply enable control signal.

30.2.2.4 SIM_PIN_DATA0_TX_OUT

The SIM_PIN_DATA0_TX_OUT is the data transmitted from the SIM module to the SmartCard. In a 1-wire mode of operating, this output port must be bidirectional with a pull-up resister off chip.

30.2.2.5 SIM_PIN_RCVD0_IN

The PIN_SIM_RCVD0_IN is the receive data coming from the SmartCard to the SIM module.

30.2.2.6 SIM_PIN_SIMPD0

The PIN_SIM_SIMPD0 is the SmartCard insertion detect.

30.2.2.7 SIM_PIN__SRST1

The SIM_PIN_SRST1 is the reset signal from the SIM to the SmartCard.

30.2.2.8 SIM_PIN__SVEN1

The SIM_PIN_SVEN1 is the SmartCard power supply enable control signal.

30.2.2.9 SIM_PIN_DATA1_TX_OUT

The SIM_PIN_DATA1_TX_OUT is the data transmitted from the SIM module to the SmartCard. In a one wire mode of operating, this output port must be bidirectional with a pull-up resister off chip.

30.2.2.10 SIM_PIN_RCVD_IN

The PIN_SIM_RCVD1_IN is the receive data coming from the SmartCard to the SIM module.

30.2.2.11 SIM_PIN_SIMPD1

The PIN_SIM_SIMPD1 is the SmartCard insertion detect.

30.3 SIM Memory Map and Register Definition

Section 30.3.3, "SIM Register Descriptions" provides the detailed descriptions for all of the SIM registers.

30.3.1 Memory Map

Table 30-7 shows the SIM memory map.

Table 30-7. SIM Memory Map

Address	Register	Access	Reset Value	Section/Page
0x5001_8000 (PORT1_CNTL)	SIM Port1 Control register	R/W	0x0000_0000	30.3.3.1/30-12
0x5001_8004 (SETUP)	SIM Setup register	R/W	0x0000_0000	30.3.3.2/30-13
0x5001_8008 (PORT1_DETECT)	SIM Port 1 Detect register	R/W	0x0000_000—	30.3.3.3/30-14
0x5001_800C (PORT1_XMT_BUF)	SIM Port1 Transmit Buffer register	R/W	0x0000_0000	30.3.3.4/30-15
0x5001_8010 (PORT1_RCV_BUF)	SIM Port 1 Receive Buffer register	R	0x0000_0000	30.3.3.5/30-16
0x5001_8014 (PORT0_CNTL)	SIM Port0 Control register	R/W	0x0000_0000	30.3.3.6/30-17
0x5001_8018 (CNTL)	SIM Control register	R/W	0x0000_0006	30.3.3.7/30-18
0x5001_801C (CLOCK_SELECT)	SIM Clock Select register	R/W	0x0000_0000	30.3.3.8/30-20
0x5001_8020 (RCV_THRESHOLD)	SIM Receive Threshold register	R/W	0x0000_0001	30.3.3.9/30-21
0x5001_8024 (ENABLE)	SIM Enable register	R/W	0x0000_0000	30.3.3.10/30-22
0x5001_8028 (XMT_STATUS)	SIM Transmit Status register	R/W	0x0000_00B8	30.3.3.11/30-23
0x5001_802C (RCV_STATUS)	SIM Receive Status register	R/W	0x0000_0000	30.3.3.12/30-25

Table 30-7. SIM Memory Map (continued)

Address	Register	Access	Reset Value	Section/Page
0x5001_8030 (INT_MASK)	SIM Interrupt Mask register	R/W	0x0000_1FFF	30.3.3.13/30-27
0x5001_8034 (PORT0_XMT_BUF)	SIM Port0 Transmit Buffer register	R/W	0x0000_0000	30.3.3.14/30-29
0x5001_8038 (PORT0_RCV_BUF)	SIM Port0 Receive Buffer register	R	0x0000_0000	30.3.3.15/30-30
0x5001_803C (PORT0_DETECT)	SIM Port0 Detect register	R/W	0x0000_000—	30.3.3.16/30-31
0x5001_8040 (DATA_FORMAT)	SIM Data Format register	R/W	0x0000_0000	30.3.3.17/30-32
0x5001_8044 (XMT_THRESHOLD)	SIM Transmit Threshold register	R/W	0x0000_0000	30.3.3.18/30-33
0x5001_8048 (GUARD_CNTL)	SIM Transmit Guard Control register	R/W	0x0000_0000	30.3.3.19/30-34
0x5001_804C (OD_CONFIG)	SIM Open Drain Configuration Control register	R/W	0x0000_0000	30.3.3.20/30-35
0x5001_8050 (RESET_CNTL)	SIM Reset Control register	R/W	0x0000_0000	30.3.3.21/30-36
0x5001_8054 (CHAR_WAIT)	SIM Character Wait Time register	R/W	0x0000_FFFF	30.3.3.22/30-37
0x5001_8058 (GPCNT)	SIM General Purpose Counter register	R/W	0x0000_FFFF	30.3.3.23/30-38
0x5001_805C (DIVISOR)	SIM Divisor register	R/W	0x0000_00BF	30.3.3.24/30-39
0x5001_8060 (BWT)	SIM Block Wait Time register	R/W	0x0000_FFFF	30.3.3.25/30-39
0x5001_8064 (BGT)	SIM Block Guard Time register	R/W	0x0000_0000	30.3.3.26/30-40
0x5001_8068 (BWT_H)	SIM Block Wait Time register HIGH	R/W	0x0000_FFFF	30.3.3.27/30-41

30.3.2 SIM Register Summary

Figure 30-2 shows the key to the register fields, and Table 30-8 shows the register figure conventions.

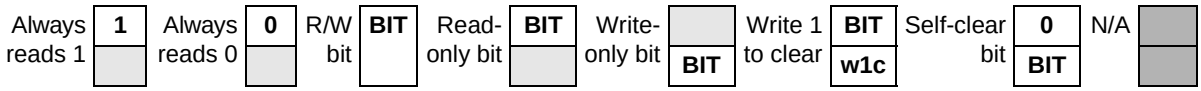


Figure 30-2. Key to Register Fields

Table 30-8. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).

Table 30-8. Register Figure Conventions (continued)

Convention	Description
rwm	A read/write bit modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 30-9 shows SIM register summary.

Table 30-9. SIM Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x5001_8000 (PORT1_CNTL)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0		3VOLT1	SCSP1	SCEN1	SRST1	STEN1	SVEN1	SAPD1
	W									SFP							
0x5001_8004 (SETUP)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	SPS	AMODE
	W																
0x5001_8008 (PORT1_DETECT)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	SPDS1	SPDP1	SDI1	SDIM1
	W															w1c	
0x5001_800C (PORT1_XMT_BUF)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	PORT1_XMT[7:0]							
	W																
0x5001_8010 (PORT1_RCV_BUF)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	CWT	FE	PE	PORT1_RCV[7:0]							
	W																
0x5001_8014 (PORT0_CNTL)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0		3VOLT0	SCSP0	SCEN0	SRST0	STEN0	SVEN0	SAPD0
	W									SFP							

Table 30-9. SIM Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x5001_8018 (CNTL)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	BWTEN	XMTCRC	CRCEN	LRCEN	CWTEN	GPCNTSEL[1:0]		BAUD_SEL[2:0]				SAMPLE1	ONACK	ANACK	ICM	0
	W																
0x5001_801C (CLOCK_SELECT)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	CLOCK_SELECT			
	W																
0x5001_8020 (RCV_THRESHOLD)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	RTH[3:0]				RDT[4:0]				
	W																
0x5001_8024 (ENABLE)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	XMTEN	RCVEN
	W																
0x5001_8028 (XMT_STATUS)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	GPCN	TDTF	TFO	TC	ETC	TFE	0	0	XTE
	W								w1c	w1c	n/a	w1c	w1c	w1c			w1c
0x5001_802C (RCV_STATUS)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	BGT	BWT	RTE	CWT	CRC	LRC	RDR	RFD	0	0	0	OEF
	W					w1c	w1c	w1c	w1c								w1c
0x5001_8030 (INT_MASK)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	BGTM	BWTM	RTM	CWTM	GPCNTM	TDTFM	TFOM	XTM	TFEIM	ETCIM	OIM	TCIM	RIM
	W																
0x5001_8034 (PORT0_XMT_BUF)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	PORT0_XMT[7:0]							
	W																
0x5001_8038 (PORT0_RCV_BUF)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	cwt	fe	pe	PORT0_RCV[7:0]							
	W																
0x5001_803C (PORT0_DETECT)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	SPDS0	SPDP0	SDI0	SDIM0
	W															w1c	

Table 30-9. SIM Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x5001_8040 (DATA_FORMAT)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	IC
	W																
0x5001_8044 (XMT_THRESHOLD)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	XTH[3:0]				TDT3:0]			
	W																
0x5001_8048 (GUARD_CNTL)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	RCVR 11	GETU[7:0]							
	W																
0x5001_804C (OD_CONFIG)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	OD_P 1	OD_P 0
	W																
0x5001_8050 (RESET_CNTL)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	DBUG	STOP	DOZE	KILL CLOC		FLUSH	FLUSH
	W														SOFT		
0x5001_8054 (CHAR_WAIT)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	CHARACTER WAIT TIME[15:0]															
	W																
0x5001_8058 (GPCNT)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	GENERAL PURPOSE COUNTER[15:0]															
	W																
0x5001_805C (DIVISOR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	Divisor[6:0]						
	W																
0x5001_8060 (BWT)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	BLOCK WAIT TIME[15:0]															
	W																
0x5001_8064 (BGT)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	BLOCK GUARD TIME[15:0]															
	W																

Table 30-9. SIM Register Summary (continued)

Name		31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0x5001_8068 (BWT_H)	R	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
	W	
	R	BLOCK WAIT TIME HIGH[15:0]
	W	

30.3.3 SIM Register Descriptions

This section contains the detailed descriptions for the SIM registers.

30.3.3.1 SIM Port1 Control Register (PORT1_CNTL)

See Figure 30-3 for an illustration of valid bits in the SIM Port1 Control Register, and Table 30-10 for descriptions of the bit fields.

0x5001_8000 (PORT1_CNTL) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	3VOLT1	SCSP1	SCEN1	SRST1	STEN1	SVEN1	SAPD1
W									SFPD1							
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 30-3. SIM Port1 Control Register

Table 30-10. SIM Port1 Control Register Field Descriptions

Field	Description
31–8	Reserved
7 SFPD1	Auto Power Down port1. Writing a “1” to this location will start the autopower down sequence for port 1. This bit will autoclear. A read of this bit will give a “0”. 0 No effect 1 Start Auto Powerdown
6 3VOLT1	3 Volt SIM Card port1. Used to configure the Port 1 transmit pin as bidirectional. This allows the Port 1 Receive pin to be re-used as General Purpose I/O. This operation is restricted to 3 Volt SIM cards only. In addition, the SIM I/O must operate at 2.7V in this mode. This bit should not be changed while the receiver or transmitter is enabled! 0 Port1 uses both RCV and XMT pins. 1 Port1 XMT pin bidirectional. Port1 RCV PIN unused.

Table 30-10. SIM Port1 Control Register Field Descriptions (continued)

Field	Description
5 SCSP1	SIM Card Clock Stop Polarity Port1. Used to control the polarity of the idle SIM clock when the clock is disabled by SCEN1. It will be forced low by hardware during the auto power down sequence. This forces the clock be a logic 0 when stopped by auto power down as required by the ISO 7816 specification.
4 SCEN1	SIM card Clock Enable Port 1. Used to enable/disable the clock to the SIM card. It can be forced low by hardware during the auto power down sequence. 0 Clock is logic 0 when stopped by SCEN1. 1 Clock is logic 1 when stopped by SCEN1.
3 SRST1	SIM card Reset. Used to control state of reset line to the SIM card. It can be forced low by hardware during the auto power down sequence. SIM card reset signals are active low. 0 SIM Card reset Port1 inactive (default) 1 SIM Card reset Port1 active
2 STEN1	SIM card Transmit Enable Port 1. Used to enable/disable the XMT data to the SIM card. It can be forced low by hardware during the auto power down sequence. 0 Port1 Transmit Data is forced to zero (default). 1 Port 1 Transmit Data controlled by SIM module.
1 SVEN1	SIM card Vcc Enable Port 1. Used to control the state of the SVEN1 pin on SIM card port 1. The SVEN1 pin controls the SIM card Vcc enable in the power management chip. It can be forced low by hardware during the auto power down sequence. 0 SIM card Voltage Port 1 disabled (default). 1 SIM card Voltage Port 1 enabled.
0 SAPD1	SIM card Auto Power Down Port 1. Used to enable/disable the auto power down function for port 1. It will be forced low at the end of the auto power down sequence. 0 Auto power down Port 1 disabled (default). 1 Auto power down Port 1 enabled.

30.3.3.2 SIM Setup Register (SETUP)

See Figure 30-4 for an illustration of valid bits in the SIM Setup Register, and Table 30-11 for descriptions of the bit fields.

0x5001_8004 (SETUP) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	SPS	AMODE
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 30-4. SIM Setup Register

Table 30-11. SIM Setup Register Field Descriptions

Field	Description
31–2	Reserved
1 SPS	SIM card Port Select. Controls which port the SIM interface uses. **Note:** The AMODE bit must be zero when the SPS bit is set to one. 0 Port 0 Enabled (default) 1 Port 1 Enabled
0 AMODE	Alternate SIM Card Mode enable. Enables an alternate SIM module to control SIM card Port 1. **Note:** The SPS bit must be zero to give the alternate SIM module control. 0 Alternate Port Disabled (default) 1 Alternate Port Enabled

30.3.3.3 SIM Port1 Detect Register (PORT1_DETECT)

See Figure 30-5 for an illustration of valid bits in the SIM Port1 Detect Register, and Table 30-12 for descriptions of the bit fields.

0x5001_8008 (PORT1_DETECT) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	SPDS1	SPDP1	SDI1	SDIM1
W															w1c	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	—	0	1

Figure 30-5. SIM Port 1 Detect Register

Table 30-12. SIM Port 1 Detect Register Field Descriptions

Field	Description
31–4	Reserved
3 SPDS1	SIM Presence Detect Select Port 1. Controls which edge of the SIMPD1 pin is used to detect the presence of the SIM card. 0 Falling edge of SIMPD1 Input (default) 1 Rising edge of SIMPD1 Input
2 SPDP1	SIMPD1 input pin status. This bit reflects the state of the SIMPD1 pin. It is not a latched register bit, but instead a synchronized version of the state of the SIMPD1 pin itself. 0 SIMPD1 pin is logic low. 1 SIMPD1 pin is logic high.

Table 30-12. SIM Port 1 Detect Register Field Descriptions (continued)

Field	Description
1 SDI1	SIM Detect Interrupt Flag Port 1. Status flag to indicate the insertion or removal of a SIM card has been detected on port 1. Can create an interrupt to the MCU if SDIM1 is low. Write a “1” to this bit to clear. 0 No insertion or removal of SIM card detected on Port 1(default) 1 Insertion or removal of SIM card detected on Port 1
0 SDIM1	SIM Detect Interrupt Mask Port 1. Interrupt mask for the SDI1 interrupt flag. 0 SDI1 enabled 1 SDI1 masked (default)
Summary: SPDS1 determines which edge transition of the SIMPD1 pin is used for SIM card presence detection. Presence detection can be used to determine if the card has been inserted or removed. The occurrence of the SIMPD1 edge specified by SPDS1 will cause the following: SDI1 to be set; if the SDIM1 mask is clear, an interrupt on SIMIRQ_N; and if sapd1 in the PORT1_CNTL register is set, an auto power down sequence to begin. If SIM card insertion is expected, SAPD1 can be set low to avoid the auto power down sequence. There is no auto power up sequence. The bit SPDP1 can be used to determine the current state of the SIMPD1 pin.	

30.3.3.4 SIM Port1 Transmit Buffer Register (PORT1_XMT_BUF)

See Figure 30-6 for an illustration of valid bits in the SIM Port1 Transmit Buffer Register, and Table 30-13 for descriptions of the bit fields.

0x5001_800C (PORT1_XMT_BUF) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PORT1_XMT[7:0]							
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 30-6. SIM Port1 Transmit Buffer Register

Table 30-13. SIM Port1 Transmit Buffer Register Field Descriptions

Field	Description
31–8	Reserved
7–0 PORT1_XMT	Port1 Transmit Buffer. Write to the next available location in the transmit buffer. Writes to this register are ignored when the SPS bit is zero.

30.3.3.5 SIM Port1 Receive Buffer Register (PORT1_RCV_BUF)

See Figure 30-7 for an illustration of valid bits in the SIM Port1 Receive Buffer Register, and Table 30-14 for descriptions of the bit fields.

0x5001_8010 (PORT1_RCV_BUF) Access: User read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	CWT	FE	PE	PORT1_RCV[7:0]							
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 30-7. SIM Port 1 Receive Buffer Register

Table 30-14. SIM Port 1 Receive Buffer Register Field Descriptions

Field	Description
31–11	Reserved
10 CWT	Port1 CWT flag. The cwt indicates that this byte was late. It is not necessary to clear the byte since it is overwritten by the next byte received into that location of the FIFO. 0 Byte was on time 1 Byte was late
9 FE	Port1 Frame Error flag. The PORT1 FE flag indicates whether a frame error was detected during the reception of the corresponding byte read in the PORT1 RCV field. The PORT1 FE flag cannot create an interrupt. It need not be cleared since it will be overwritten by the next byte received into that location of the FIFO. 0 Byte contains no framing error (default) 1 Byte contains a framing error
8 PE	Port1 Parity Error flag. The PORT1 FE flag indicates whether a parity error was detected during the reception of the corresponding byte read in the PORT1 RCV field. The PORT1 PE flag cannot create an interrupt. It need not be cleared since it will be overwritten by the next byte received into that location of the FIFO. A parity error can create a NACK pulse. 0 Byte contains no parity error (default) 1 Byte contains a parity error
7–0 PORT1_RCV	Port1 Receive buffer. Read from the next location in the receive buffer. Reads from this register return zero when the SPS bit is zero.

30.3.3.6 SIM Port0 Control Register (PORT0_CNTL)

See Figure 30-8 for an illustration of valid bits in the SIM Port0 Control Register, and Table 30-15 for descriptions of the bit fields.

0x5001_8014 (PORT0_CNTL) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0		3VOLT0	SCSP0	SCEN0	SRST0	STEN0	SVEN0	SAPD0
W									SFPD0							
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 30-8. SIM Port0 Control Register

Table 30-15. SIM Port0 Control Register Field Descriptions

Field	Description
31–8	Reserved
7 SFPD0	Auto Power Down port0. Writing a “1” to this location will start the autopower down sequence for port 0. This bit will autoclear.A read of this bit will give a “0”. 0 No effect 1 Start Auto Powerdown
6 3VOLT0	3 Volt SIM Card port0. Used to configure the Port 0 transmit pin as bidirectional. This allows the Port 0 Receive pin to be re-used as General Purpose I/O. This operation is restricted to 3 Volt SIM cards only. In addition, the SIM I/O must operate at 2.7V in this mode. This bit should not be changed while the receiver or transmitter is enabled! 0 Port0 uses both RCV and XMT pins. 1 Port0 XMT pin bidirectional. Port0 RCV pin unused.
5 SCSP0	SIM Card Clock Stop Polarity port0. Used to control the polarity of the idle SIM clock when the clock is disabled by SCEN0. It will be forced low by hardware during the auto power down sequence. This forces the clock to be a logic 0 when stopped by auto power down as required by the ISO 7816 specification. 0 Clock is logic 0 when stopped by SCEN0 1 Clock is logic 1 when stopped by SCEN0
4 SCEN0	SIM card Clock Enable Port 0. Used to enable/disable the clock to the SIM card. It can be forced low by hardware during the auto power down sequence. 0 SIM Card Clock Disabled Port 0 1 SIM Card Clock Enabled Port 0
3 SRST0	SIM card Reset. Used to control state of reset line to the SIM card. It can be forced low by hardware during the auto power down sequence. SIM card reset signals are active low. 0 SIM Card reset Port0 inactive (default) 1 SIM Card reset Port0 active

Table 30-15. SIM Port0 Control Register Field Descriptions (continued)

Field	Description
2 STEN0	SIM card Transmit Enable Port 0. Used to enable/disable the XMT data to the SIM card. It can be forced low by hardware during the auto power down sequence. 0 Port0 Transmit Data is forced to zero (default). 1 Port 0 Transmit Data controlled by SIM module.
1 SVEN0	SIM card Vcc Enable Port 0. Used to control the state of the SVEN0 pin on SIM card port 0. The SVEN0 pin controls the SIM card Vcc enable in the power management chip. It can be forced low by hardware during the auto power down sequence. 0 SIM card Voltage Port 0 disabled (default) 1 SIM card Voltage Port 0 enabled
0 SAPD0	SIM card Auto Power Down Port 0. Used to enable/disable the auto power down function for port 0. It will be forced low at the end of the auto power down sequence. 0 Auto power down Port 0 disabled (default) 1 Auto power down Port 0 enabled

30.3.3.7 SIM Control Register (CNTL)

See Figure 30-9 for an illustration of valid bits in the SIM Control Register, and Table 30-16 for descriptions of the bit fields.

0x5001_8018 (CNTL) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BWTEN	XMTCRCLRC	CRCEN	LRCEN	CWTEN	GPCNT_CLK_SEL[1:0]			BAUD_SEL[2:0]		0	SAMPLE1	ONACK	ANACK	ICM	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

Figure 30-9. SIM Control Register

Table 30-16. SIM Control Register Field Descriptions

Field	Description
31–16	Reserved
15 BWTEN	Block wait time enable. Writing a “1” to this bit will enable the BWT and BGT functions. The BWT and BGT functions can then be individually selected using the interrupt mask. 0 Disable BWT, BGT 1 Enable BWT, BGT

Table 30-16. SIM Control Register Field Descriptions (continued)

Field	Description
14 xmt_crc_lrc	Transmit CRC or LRC. This bit specifies whether or not to transmit the redundancy checking data at the end of a transmission (that is, when the FIFO becomes empty). 0 No Redundancy check info transmitted (default) 1 Transmit LRC or CRC info when FIFO empties (whichever is enabled)
13 CRCEN	CRC Enable. This bit enables the calculation of the 16-bit CRC value for both receiver and transmitter. The result of the calculation is continuously compared to the expected remainder and reflected in the CRCOK bit in the RCV_STAT register. Clearing this bit resets the current CRC residual value in the SIM hardware. 0 16-bit Cyclic Redundancy Checking disabled (default) 1 16-bit Cyclic Redundancy Checking enabled
12 LRCEN	LRC Enable. This bit enables the calculation of the 8-bit LRC value for both receiver and transmitter. The result of the calculation is continuously compared to zero and reflected in the LRCOK bit in the RCV_STAT register. Clearing this bit resets the current LRC value in the SIM hardware. 0 8-bit Linear Redundancy Checking disabled (default) 1 8-bit Linear Redundancy Checking enabled
11 CWTEN	Character Wait Time Counter Enable. Enables the character wait time counter. Clearing this bit resets the counter to zero. 0 Character Wait time Counter off (default) 1 Character Wait time counter on
10–9 GPCNT_CLK_SEL[1:0]	General Purpose Counter Clock Select. Selects which clock source is used by SIM Module general purpose counter. The only way to reset the counter is to set these bits to zero. The counter will begin counting as soon as the clock input is selected and the clocks are enabled. These input clocks are enabled through other register bits of the SIM module (KILL_CLOCK, RCV_EN, and XMT_EN, respectively). 00 Disabled/Reset 01 Card Clock 10 Receive Clock 11 ETU Clock (transmit clock)
8–6 BAUD_SEL[2:0]	SIM Baud Rate Select. Selects the asynchronous baud rate divisor of the clock When set to “111”, the divisor is set to the value programmed in the DIVISOR register. This allows for more flexible baud rate determination. 000 31 001 16 010 8 011 4 10x 2 110 1 111 DIVISOR Reg
5	Reserved
4 Sample12	Sample12. Set the third stage divider. This sets the corresponding sample rate which is the number of times a bit being received is sampled. 0 divide by 16 (default) 1 divide by 12
3 ONACK	Overrun NACK Enable. Enables overrun NACK generation. 0 NACK generation on overrun is disabled (default) 1 NACK generation on overrun is enabled

Table 30-16. SIM Control Register Field Descriptions (continued)

Field	Description
2 ANACK	Automatic NACK Enable. Enables NACK generation for parity errors or invalid initial characters when in ICM mode. 0 NACK generation on errors disabled (default) 1 NACK generation on errors enabled
1 ICM	Initial Character Mode. Enables initial character mode. Will be automatically cleared by hardware once a valid initial character is received. 0 Initial Character Mode disabled (default) 1 Initial Character Mode enabled
0	Reserved

30.3.3.8 SIM Clock Select Register (CLOCK_SELECT)

See Figure 30-10 for an illustration of valid bits in the SIM Clock Select Register, and Table 30-17 for descriptions of the bit fields.

0x5001_801C (CLOCK_SELECT) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	CLOCK_SELECT[3:0]			
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 30-10. SIM Clock Select Register

Table 30-17. SIM Clock Select Register Field Descriptions

Field	Description
31–4	Reserved
3–0 CLOCK_SELECT	Clock select. This bit field will determine the first stage divider setting. If the ipg_clk is 66 MHz, a typical setting would be 0110. This would set the SIM card clock to 66 MHz/14 = 4.7 MHz. 0000 ref / 2 0001 ref / 4 0010 ref / 6 0011 ref / 8 0100 ref / 10 0101 ref / 12 0110 ref / 14 0111 ref / 16 1000 ref / 18 1001 ref / 20 1010 ref / 22 1011 ref / 26 1100 ref / 32 1101 ref / 38 1110 ref / 46 1111 ref / 56

30.3.3.9 SIM Receive Threshold Register (RCV_THRESHOLD)

See Figure 30-11 for an illustration of valid bits in the SIM Receive Threshold Register, and Table 30-18 for descriptions of the bit fields.

0x5001_8020 (RCV_THRESHOLD) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	RTH[3:0]				RDT[4:0]				
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 30-11. SIM Receive Threshold Register

Table 30-18. SIM Receive Threshold Register Field Descriptions

Field	Description
31–9	Reserved
8–5 RTH[3:0]	Receive NACK Threshold. Used to specify the number of consecutive NACKs transmitted by the SIM module, for a given character, before the receive threshold error (RTE) flag is triggered. A value of 0 indicates that RTE is never set. When a valid character is received by the SIM, the internal counter keeping track of the NACK count resets to zero for the subsequent byte being received. If the ANACK bit is clear in the CNTL register, RTH has no effect.
4–0 RDT[4:0]	Receive Data Threshold. Determines the number of unread bytes that must exist in the FIFO to trigger the receive data register full (RDRF) interrupt flag. If the number of unread bytes in the receive FIFO is greater than or equal to the value in RDT, the RDRF flag in the RCV_STATUS register will be set. A value of zero indicates that there must be 32 unread bytes in the FIFO to trigger RDRF. The RDT value can be altered at any time, and the RDRF flag will be updated accordingly.

30.3.3.10 SIM Enable Register (ENABLE)

See Figure 30-12 for an illustration of valid bits in the SIM Enable register, and Table 30-19 for descriptions of the bit fields.

0x5001_8024 (ENABLE) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	XMTEN	RCVEN
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 30-12. SIM Enable Register

Table 30-19. SIM Enable Register Field Descriptions

Field	Description
31–2	Reserved

Table 30-19. SIM Enable Register Field Descriptions (continued)

Field	Description
1 XMT_EN	SIM Transmit Enable. Used to enable/disable the SIM transmit state machine. When the SIM is being used to receive data, XMT_EN should be deasserted. This bit also enables the xmt_clk and rcv_clk inputs to the General Purpose Counter. **Note:** Setting this bit (transition from 0 to 1) will reset the CRC and LRC values. 0 SIM Transmitter disabled (default) 1 SIM Transmitter enabled
0 RCV_EN	SIM Receiver Enable. Used to enable/disable the SIM receive state machine. The RCV_EN bit should be left high whenever the SIM module is in use. The SIM module has an automatic receive mode operation that disables the reception of characters when the transmitter is operational. Once the transmitter has completed sending the last character, the receiver is automatically enabled. This bit also enables the RCV_CLK input to the General Purpose Counter. 0 SIM Receiver disabled (default) 1 SIM Receiver enabled

30.3.3.11 SIM Transmit Status Register (XMT_STATUS)

See Figure 30-13 for an illustration of valid bits in the SIM Transmit Status register, and Table 30-20 for descriptions of the bit fields.

0x5001_8028 (XMT_STATUS) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	GPCNT	TDTF	TFO	TC	ETC	TFE	0	0	XTE
W								w1c	w1c	n/a	w1c	w1c	w1c			w1c
RESET	0	0	0	0	0	0	0	0	1	0	1	1	1	0	0	0

Figure 30-13. SIM Transmit Status Register

Table 30-20. SIM Transmit Status Register Field Descriptions

Field	Description
31–9	Reserved
8 GPCNT	General purpose Counter Flag. Used to indicate when the General purpose counter has reached the value in the GPCNT register. 0 GPCNT time not reached, or bit has been cleared. (default). 1 General Purpose counter has reached the GPCNT value.

Table 30-20. SIM Transmit Status Register Field Descriptions (continued)

Field	Description
7 TDTF	Transmit Data Threshold Flag. Used to indicate when the number of bytes in the transmit FIFO is less than or equal to the value programmed in the TFT[3:0] bits in the XMT_THRESHOLD register. 0 Number of bytes in FIFO is greater than TFT[3:0], or bit has been cleared (default). 1 Number of bytes in FIFO is less than or equal to TFT[3:0]
6 TFO	Transmit FIFO Overfill Error. Used to indicate when the Transmit FIFO has been written with more than 16 bytes. The TFO bit can only be cleared by setting the FLUSH_XMT or SOFT_RESET bit in the RESET_CNTL register. 0 No transmit FIFO overfill error has occurred (default). 1 A Transmit FIFO overfill error has occurred.
5 TC	Transmit Complete. Used to indicate whether the SIM transmitter is ready for a new transmission. The TC flag becomes set after the guard time has expired for the last byte in the transmit FIFO. The TC flag will create an interrupt if TCIM in the INT_MASK register is low. The TC bit is a write-one-to-clear bit. 0 Transmit pending or in progress 1 Transmit complete (default)
4 ETC	Early Transmit Complete. Used to indicate that the SIM transmitter has finished sending the current byte and the transmit FIFO is empty. This bit differs from the TC bit in that it is set before the guard time of the last byte has elapsed. The ETC flag will create an interrupt if etcim in the INT_MASK register is low. The ETC bit is a write-one-to-clear bit. 0 Transmit pending or in progress 1 Transmit complete (default)
3 TFE	Transmit FIFO Empty. Used to indicate that the SIM transmit FIFO has emptied. This bit will be set when the last byte in the transmit FIFO has been transferred to the SIM transmitter shift register. The TFE flag will create an interrupt if TFEIM in the INT_MASK register is low. The TFE bit is a write-one-to-clear bit. 0 Transmit FIFO is not empty 1 Transmit FIFO is empty
2–1	Reserved
0 XTE	Transmit Threshold Error. Used to indicate the transmit NACK threshold has been reached. When XTE is high, no further transmissions will be done until the XTE flag is cleared. Any data transmissions still pending in the transmit FIFO will be aborted, and the TC, ETC, and TFE flags will be set. The XTE flag will create an interrupt if XTM in the IMT_MASK register is low. The XTE bit is a write-one-to-clear bit. 0 Transmit NACK threshold has not been reached (default) 1 Transmit NACK threshold reached; transmitter frozen

30.3.3.12 SIM Receive Status Register (RCV_STATUS)

See Figure 30-14 for an illustration of valid bits in the SIM Control register, and Table 30-21 for descriptions of the bit fields.

0x5001_802C (RCV_STATUS) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	BGT	BWT	RTE	CWT	CRC OK	LRCO K	RDRF	RFD	0	0	0	OEF
W					w1c	w1c	w1c	w1c								w1c
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 30-14. SIM Receive Status Register

Table 30-21. SIM Receive Status Register Field Descriptions

Field	Description
31–12	Reserved
11 BGT	Block guard time error flag. Used to indicate if the block guard time was to small. The threshold is set by the block guard time register. 0 Block guard time was sufficient. 1 Block guard time was too small.
10 BWT	Block wait time error flag. Used to indicate if the block wait time has been exceeded. The threshold is set by the block wait time registers. 0 Block wait time not exceeded. 1 Block wait time was exceeded.
9 RTE	Receive NACK threshold error flag. Used to indicate whether the number of consecutive NACKs generated by the SIM module in response to receive parity errors, for the byte being received, equals the value programmed in the RTH[3:0] in the RCV_THRESHOLD register. This bit would never be set unless the ANACK bit is set in the CNTL register. The SAPDx bit in the PORTx_CNTL resister must be set to enable the threshold error to trigger the auto power down sequence. RTE is a write once to clear bit. Clearing this bit also resets the internal counter for consecutive NACKs being transmitted for a given byte. 0 Number of NACKs generated by the receiver is less than the value programmed in RTH[3:0] 1 Number of NACKs generated by the receiver is equal to the value programmed in RTH[3:0]
8 CWT	Character Wait Time Counter flag. Used to indicate when the time between received characters is equal to or greater than the value programmed in the CHAR_WAIT register. 0 No CWT violation has occurred (default). 1 Time between two consecutive characters exceeded the value in CHAR_WAIT.

Table 30-21. SIM Receive Status Register Field Descriptions (continued)

Field	Description
7 CRCOK	Cyclic Redundancy Check Okay flag. Used to indicate when the calculated 16-bit CRC value matches the expected value for the current input data stream. The value is calculated across all received characters from the point the CRCEN bit is set in the CNTL register. The current CRC residual can be reset by three mechanisms: • Clear CRCEN bit in CNTL register • Set XMT_EN bit in ENABLE register • Automatically by hardware when ETC flag is set at the end of a transmission. 0 Current CRC value does not match remainder. 1 Current calculated CRC value matches the expected result.
6 LRCOK	Linear Redundancy Check Okay flag. Used to indicate when the calculated 8-bit LRC value is zero value for the current input data stream. The value is calculated across all received characters from the point the LRCEN bit is set in the CNTL register. The current LRC residual can be reset by three mechanisms: • Clear LRCEN bit in CNTL register • Set XMT_EN bit in ENABLE register • Automatically by hardware when ETC flag is set at the end of a transmission. 0 Current LRC value does not match remainder. 1 Current calculated LRC value matches the expected result (that is, zero).
5 RDRF	Receive Data Register Full. Used to indicate whether the SIM receive FIFO has reached the threshold level set by RDT[4:0] in the RCV_THRESHOLD register. The RDRF flag will be set any time the number of unread bytes in the receive FIFO is equal to or greater than the value set by RDT[4:0]. The flag can be cleared by reading enough bytes out of the receive FIFO so as to bring the number of bytes left in the FIFO below the RDT[4:0] level. Another way to clear the flag is to set the RDT[4:0] level higher than the number of unread bytes currently in the FIFO. The RDRF flag will create an interrupt if the RIM bit in the INT_MASK register is cleared. 0 Number of unread bytes in receive buffer < value set by RDT[4:0] (default). 1 Number of unread bytes in receive buffer >= value set by RDT[4:0].
4 RFD	Receive FIFO has unread Data. Used to indicate that there is at least one unread byte in the receive data FIFO. Can only be cleared by reading all bytes out of the receive FIFO. The RFD bit cannot be used to create an interrupt. Normally, the SIM triggers the interrupt with RDRF and software uses RFD to read all of the bytes out of the receive FIFO. 0 There are no unread bytes in the receive FIFO (default). 1 There is at least one unread byte in the receiver FIFO.
3–1	Reserved
0 OEF	Overrun Error Flag. Used to indicate that the SIM was unable to store received data due to already having 16 unread bytes in the FIFO. It does not necessarily indicate that data has been lost. If the ONACK control bit in the CNTL register is set, there will be a NACK pulse generated on bytes that would otherwise cause a loss of data due to a full FIFO. These bytes should be retransmitted by the SIM card which implies that no data has actually been lost. In this case, the OEF flag is just an indicator that this situation has occurred which may be helpful in system debug. For the case where ONACK is not set, a set OEF flag does indicate a loss of data since all bytes received with the OEF flag set will indeed be lost (including the byte that caused the bit to be set). The OEF flag will cause an interrupt if the OIM bit in the INT_MASK register. The OEF flag is a write-one-to-clear bit. 0 No overrun error has occurred (default). 1 A byte was received when the received FIFO was already full.

30.3.3.13 SIM Interrupt Mask Register (INT_MASK)

See Figure 30-15 for an illustration of valid bits in the SIM Interrupt Mask register, and Table 30-22 for descriptions of the bit fields.

0x5001_8030 (INT_MASK) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	BGTM	BWTM	RTM	CWTM	GPCNTM	TDTFM	TFOM	XTM	TFEIM	ETCIM	OIM	TCIM	RIM
W																
RESET	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 30-15. SIM Interrupt Mask Register

Table 30-22. SIM Interrupt Mask Register Field Descriptions

Field	Description
31–13	Reserved
12 BGTM	Block guard time interrupt mask. Used to enable/disable the ability of the BGT flag in the RCV_STATUS register to generate SIM interrupts. 0 BGT interrupt enabled 1 BGT interrupt masked (default)
11 BWTM	Block wait time interrupt mask. Used to enable/disable the ability of the BWT flag in the RCV_STATUS register to generate SIM interrupts. 0 BWT interrupt enabled 1 BWT interrupt masked (default)
10 RTM	Receive NACK Threshold interrupt mask. Used to enable/disable the ability of the RTE flag in the RCV_STATUS register to generate SIM interrupts. 0 RTE interrupt enabled 1 RTE interrupt masked (default)
9 CWTM	Character Wait Time Interrupt Mask. Used to enable/disable the ability of the CWT flag in the RCV_STATUS register to generate SIM interrupts. 0 CWT interrupt enabled 1 CWT interrupt masked (default)
8 GPCNTM	General Purpose Counter Interrupt Mask. Used to enable/disable the ability of the GPCNT flag in the XMT_STATUS register to generate SIM interrupts. 0 GPCNT interrupt enabled 1 GPCNT interrupt masked (default)
7 TDTFM	Transmit Data Threshold Interrupt Mask. Used to enable/disable the ability of the TDTF flag in the XMT_STATUS register to generate SIM interrupts. 0 TDTF interrupt enabled 1 TDTF interrupt masked (default)

Table 30-22. SIM Interrupt Mask Register Field Descriptions (continued)

Field	Description
6 TFOM	Transmit FIFO Overfill Error Interrupt Mask. Used to enable/disable the ability of the TFO flag in the XMT_STATUS register to generate SIM interrupts. 0 TFO interrupt enabled 1 TFO interrupt masked (default)
5 XTM	Transmit Threshold Interrupt Mask. Used to enable/disable the ability of the XTE flag in the XMT_STATUS register to generate SIM interrupts 0 XTE interrupt enabled 1 XTE interrupt masked (default)
4 TFEIM	Transmit FIFO Empty Interrupt Mask. Used to enable/disable the ability of the TFE flag in the XMT_STATUS register to generate SIM interrupts. 0 TFE interrupt enabled 1 TFE interrupt masked (default)
3 ETCIM	Early Transmit Complete Interrupt Mask. Used to enable/disable the ability of the ETC flag in the XMT_STATUS register to generate SIM interrupts. 0 ETC interrupt enabled 1 ETC interrupt masked (default)
2 OIM	Overrun Interrupt Mask. Used to enable/disable the ability of the OEF flag in the RCV_STATUS register to generate SIM interrupts. 0 OEF interrupt enabled 1 OEF interrupt masked (default)
1 TCIM	Transmit Complete Interrupt Mask. Used to enable/disable the ability of the TC flag in the XMT_STATUS register to generate SIM interrupts. 0 TC interrupt enabled 1 TC interrupt masked (default)
0 RIM	Receive Interrupt Mask. Used to enable/disable the ability of the RDRF flag in the RCV_STATUS register to generate SIM interrupts. 0 RDRF interrupt enabled 1 RDRF interrupt masked (default)

30.3.3.14 SIM Port0 Transmit Buffer Register (PORT0_XMT_BUF)

See Figure 30-16 for an illustration of valid bits in the SIM Port0 Transmit Buffer register, and Table 30-23 for descriptions of the bit fields.

0x5001_8034 (PORT0_XMT_BUF) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PORT0_XMT[7:0]							
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 30-16. SIM Port0 Transmit Buffer Register

Table 30-23. SIM Port0 Transmit Buffer Register Field Descriptions

Field	Description
31–8	Reserved
7–0 PORT0_XMT	Port0 Transmit Buffer. Register to write when transmitting using port 0. To use this register to initiate transmissions to the SIM, set SPS = 0. When SPS = 1, a write to this register has no effect. A read of this register will produce the last thing written, or $00 if when SPS = 1. **Note:** Writing more data to the transmit FIFO than it can hold (16 bytes) will cause a transmit FIFO Overfill error.

30.3.3.15 SIM Port0 Receive Buffer Register (PORT0_RCV_BUF)

See Figure 30-17 for an illustration of valid bits in the SIM Port0 Receive Buffer register, and Table 30-24 for descriptions of the bit fields.

0x5001_8038 (PORT0_RCV_BUF) Access: User read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	CWT	FE	PE	PORT0_RCV[7:0]							
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 30-17. SIM Port0 Receive Buffer Register

Table 30-24. SIM Port0 Receive Buffer Register Field Descriptions

Field	Description
31–8	Reserved
10 CWT	Port0 cwt flag. The CWT indicates that this byte was late. It need not be cleared since it will be overwritten by the next byte received into that location of the FIFO. 0 Byte was on time 1 Byte was late
9 FE	Port0 Frame Error flag. The PORT0 FE flag indicates whether a frame error was detected during the reception of the corresponding byte read in the PORT0 RCV field. The PORT1 FE flag cannot create an interrupt. It need not be cleared since it will be overwritten by the next byte received into that location of the FIFO. 0 Byte contains no framing error (default) 1 Byte contains a framing error
8 PE	Port0 Parity Error flag. The PORT0 PE flag indicates whether a parity error was detected during the reception of the corresponding byte read in the PORT0 RCV field. The PORT0 PE flag cannot create an interrupt. It need not be cleared since it will be overwritten by the next byte received into that location of the FIFO. A parity error can create a NACK pulse. 0 Byte contains no parity error (default) 1 Byte contains a parity error
7–0 PORT0_RCV	Port0 Receive buffer. Read from the next location in the receive buffer. Reads from this register return zero when the SPS bit is zero.

30.3.3.16 SIM Port0 Detect Register (PORT0_DETECT)

See Figure 30-18 for an illustration of valid bits in the SIM Port0 Detect register, and Table 30-25 for descriptions of the bit fields.

0x5001_803C (PORT0_DETECT) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	SPDS0	SPDP0	SDI0	SDIM0
W															w1c	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	—	0	1

Figure 30-18. SIM Port0 Detect Register

Table 30-25. SIM Port0 Detect Register Field Descriptions

Field	Description
31-4	Reserved
3 SPDS0	SIM Presence Detect Select Port 0. Controls which edge of the SIMPD0 pin is used to detect the presence of the SIM card. 0 Falling edge of SIMPD0 Input (default) 1 Rising edge of SIMPD0 Input
2 SPDP0	SIMPD0 input pin status. This bit reflects the state of the SIMPD0 pin. It is not a latched register bit, but instead a synchronized version of the state of the SIMPD0 pin itself. 0 SIMPD0 pin is logic low 1 SIMPD0 pin is logic high
1 SDI0	SIM Detect Interrupt flag Port 0. Status flag to indicate the insertion or removal of a SIM card has been detected on port 0. Can create an interrupt to the MCU if SDIM0 is low. Write a “1” to this bit to clear. 0 No insertion or removal of SIM card detected on Port 0 (default) 1 Insertion or removal of SIM card detected on Port 0
0 SDIM0	SIM Detect Interrupt Mask Port 0. Interrupt mask for the SDI0 interrupt flag. 0 SDI0 enabled 1 SDI0 masked (default)

30.3.3.17 SIM Data Format Register (DATA_FORMAT)

See Figure 30-19 for an illustration of valid bits in the SIM Data Format Register, and Table 30-26 for descriptions of the bit fields.

0x5001_8040 (DATA_FORMAT) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	IC
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 30-19. SIM Data Format Register

Table 30-26. SIM Data Format Register Field Descriptions

Field	Description
31–1	Reserved
0 IC	Inverse Convention. Used to configure the SIM to use either inverse convention or direct convention for its data format. The IC bit can be controlled by software, but it is normally set by hardware as a result of the interpretation of the initial character when in ICM mode. 0 Direction convention transfers enabled (default). 1 Inverse convention transfers enabled.

30.3.3.18 SIM Transmit Threshold Register (XMT_THRESHOLD)

See Figure 30-20 for an illustration of valid bits in the SIM Transmit Threshold register, and Table 30-27 for descriptions of the bit fields.

0x5001_8044 (XMT_THRESHOLD) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	XTH[3:0]				TDT[3:0]			
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 30-20. SIM Transmit Threshold Register

Table 30-27. SIM Transmit Threshold Register Field Descriptions

Field	Description
31–8	Reserved
7–4 XTH[3:0]	Transmit NACK Threshold. Used to set the NACK threshold for the transmitter. Once the threshold number set by XTH has been reached for the current byte being transmitted, the error flag XTE in the XMT_STATUS register will be set. Setting of XTE causes the remaining transmissions queued in the transmit FIFO to be aborted and no more transmissions to occur until software clears XTE. To trigger XTE, a given byte being transmitted must reach the XTH threshold itself. Transmit NACKs accumulated on one byte are not carried over to the next. $0 XTE will never be set; retransmission after NACK reception is disabled. $1 XTE will be set after 1 NACK is received; 0 retransmissions occurs. $2 XTE will be set after 2 NACKs are received; at most 1 retransmission occurs. $3 XTE will be set after 3 NACKs are received; at most 2 retransmissions occurs. ... ... $X XTE will be set after X NACKs are received; at most (X–1) retransmissions occurs. ... ... $F XTE will be set after 15 NACKs are received; at most 14 retransmissions occurs.
3–0 TDT	Transmit Data Threshold. Used to set the threshold value for the Transmit FIFO at which the TDTF bit in the XMT_STATUS register will be set. When the number of bytes in the Transmit FIFO is less than or equal to TDT[3:0], TDTF will be set.

30.3.3.19 SIM Transmit Guard Control Register (GUARD_CNTL)

See Figure 30-21 for an illustration of valid bits in the SIM Transmit Guard Control register, and Table 30-28 for descriptions of the bit fields.

0x5001_8048 (GUARD_CNTL) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	RCV R11	GETU[7:0]							
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 30-21. SIM Transmit Guard Control Register

Table 30-28. SIM Transmit Guard Control Register Field Descriptions

Field	Description
31–9	Reserved
8 RCVR11	Receiver use 11 ETUs. Used to configure the SIM module receiver for 11 ETU operation (that is, 1 Stop bit). This bit is provided for support of T=1 cards. 0 Receiver configured for 12 ETU operation (default) 1 Receiver configured for 11 ETU operation
7–0 GETU	Transmit Guard ETUs. Used to control the number of additional elementary time units (ETUs) inserted between bytes transmitted by the SIM transmitter. An ETU is equivalent to one bit time at the given baud rate (for example, the length of a START bit). The guard time has no effect on the SIM receiver. A value of $00 inserts no additional ETUs, while a value of $FE inserts 254 additional ETUs. A value of $FF subtracts one ETU by reducing the number of STOP bits from two to one.

30.3.3.20 SIM Open Drain Configuration Control Register (OD_CONFIG)

See Figure 30-22 for an illustration of valid bits in the SIM Open Drain Configuration Control Register, and Table 30-29 for descriptions of the bit fields.

0x5001_804C (OD_CONFIG) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	OD_PI	OD_P0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 30-22. SIM Open Drain Configuration Control Register

Table 30-29. SIM Open Drain Configuration Control Register Field Descriptions

Field	Description
31–2	Reserved
1 OD_P1	Open Drain control for Port 1. Used to control whether the XMT data line on port 1 is open-drain. If AMODE bit in SETUP register is set, this bit will have no effect. 0 XMT pin on port 1 is push-pull (default). 1 XMT pin on port 1 is open-drain.
0 OD_P0	Open Drain control for Port 0. Used to control whether the XMT data line on port 0 is open-drain. 0 XMT pin on port 0 is push-pull (default). 1 XMT pin on port 0 is open-drain.

30.3.3.21 SIM Reset Control Register (RESET_CNTL)

See Figure 30-23 for an illustration of valid bits in the SIM Reset Control register, and Table 30-30 for descriptions of the bit fields.

0x5001_8050 (RESET_CNTL) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	DBUG	STOP	DOZE	KILL CLOCK	0	FLUSH XMT	FLUSH RCV
W														SOFT RESET		
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 30-23. SIM Reset Control Register

Table 30-30. SIM Reset Control Register Field Descriptions

Field	Description
31-7	Reserved
6 DBUG	DEBUG. Used to configure the operation of the SIM module when a debug event occurs. When set, a debug event (generated by such causes as OnCE module and external control) will make the receive FIFO read pointer to be frozen. 0 Debug event has no affect on SIM module (default). 1 Debug event prohibits read pointer changes for receive FIFO.
5 STOP	STOP. Used to configure the operation of the SIM module when a processor STOP instruction is executed. This bit is added to provide support for SIM cards that do not allow the SIM Card clock to be stopped while power is applied. See Figure 30-35. 0 STOP instruction shuts down all SIM clocks (default). 1 STOP instruction shuts down all clocks except for the BAUD_CLK (clock provided to SIM Card).
4 DOZE	DOZE. Used to configure the operation of the SIM module when a processor DOZE instruction is executed. See Figure 30-35. 0 DOZE instruction has no effect on SIM module (default). 1 DOZE instruction will cause SIM module to gate SIM clocks when the transmit FIFO is empty.
3 KILL_CLK	Kill SIM Clock. Used to enable/disable the SIM clock input to the SIM module. This bit will gate all SIM clocks including the SIM card clock regardless of the state of the STOP bit described above. 0 SIM input clock enabled (default). 1 SIM input clock disabled.

Table 30-30. SIM Reset Control Register Field Descriptions (continued)

Field	Description
2 SOFT_REST	Software Reset. Used to reset the entire SIM module. This acts the same as a hardware reset for the SIM module. This bit is self-clearing. **Note:** Software should allow a minimum of 4 reference clock cycles (CKIH) before attempting to access the SIM module after a software reset. 0 SIM Normal operation (default). 1 SIM held in Reset.
1 FLUSH_XMT	Flush Transmitter. This bit operates as a SIM transmitter reset. The receive portion of the SIM module is not affected. The software must clear this bit before the SIM transmitter can operate. 0 SIM Transmitter normal operation (default). 1 SIM Transmitter held in Reset.
0 FLUSH_RCV	Flush Receiver. This bit operates as a SIM receiver reset. The transmit portion of the SIM module is not affected. The software must clear this bit before the SIM receiver can operate. 0 SIM Receiver normal operation (default). 1 SIM Receiver held in Reset.

30.3.3.22 SIM Character Wait Time Register (CHAR_WAIT)

See Figure 30-24 for an illustration of valid bits in the SIM Character Wait Time register, and Table 30-31 for descriptions of the bit fields.

0x5001_8054 (CHAR_WAIT) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CWT[15:0]															
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 30-24. SIM Character Wait Time Register

Table 30-31. SIM Character Wait Time Register Field Descriptions

Field	Description
31–16	Reserved
15–0 CWT	Character Wait Time. The value written to this register will specify the number of ETU times allowed between characters. Default is 0xFFFF

30.3.3.23 SIM General Purpose Counter Register (GPCNT)

See Figure 30-25 for an illustration of valid bits in the SIM General Purpose Counter register, and Table 30-32 for descriptions of the bit fields.

0x5001_8058 (GPCNT) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GPCNT[15:0]															
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 30-25. SIM General Purpose Counter Register

Table 30-32. SIM General Purpose Counter Register Field Descriptions

Field	Description
31–16	Reserved
15–0 GPCNT	General Purpose Counter. The value written to this register will be used to compare to the general purpose counter in the SIM module. Once the General purpose counter reaches this value, the GPCNT flag in the XMT_STATUS register will be set. This counter is intended to be used for any events that must be monitored for duration based on the card clock, receiver sample rate, or ETU rate (transmit clock). Example: ATR arrival time and ATR duration. Default is 0xFFFF

30.3.3.24 SIM Divisor Register (DIVISOR)

See Figure 30-26 for an illustration of valid bits in the SIM Divisor Register, and Table 30-33 for descriptions of the bit fields.

0x5001_805C (DIVISOR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	DIVISOR[6:0]						
W																
RESET	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1

Figure 30-26. SIM Divisor Register

Table 30-33. SIM Divisor Register Field Descriptions

Field	Description
31–7	Reserved
6–0 DIVISOR	DIVISOR Register. The value written to this register will be used to generate the SIM Receive clock. The BAUD_SEL[2:0] bits in the CNTL register must be set to ‘111’ in order to control the divisor value using the DIVISOR register. Default is 0x7f

30.3.3.25 SIM Block Wait Time Register (BWT)

See Figure 30-27 for an illustration of valid bits in the SIM Block Wait Time register, and Table 30-34 for descriptions of the bit fields.

0x5001_8060 (BWT) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BWT[15:0]															
W																
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 30-27. SIM Block Wait Time Register

Table 30-34. SIM Block Wait Time Register Field Descriptions

Field	Description
31–16	Reserved
15–0 BWT	BWT Register 16 LSB. The value in this register is the block wait time 16 LSB. The time from START bit of last byte sent from the SIM module to the START bit of the first byte sent from the SmartCard must be less than the value formed by the 32-bit register composed by BWT_H and BWT registers. If it is not, then the BWT flag will be set. Default is 0xFFFF

30.3.3.26 SIM Block Guard Time Register (BGT)

See Figure 30-28 for an illustration of valid bits in the SIM Block Guard Time register, and Table 30-35 for descriptions of the bit fields.

0x5001_8064 (BGT) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BGT[15:0]															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 30-28. SIM Block Guard Time Register

Table 30-35. SIM Block Guard Time Register Field Descriptions

Field	Description
31–16	Reserved
15–0 BGT	BGT Register. The value in this register is the block guard time. Time from START bit of last byte sent from the SIM module to the START bit of the first byte sent from the SmartCard must be greater than this value. If it is not, then the BGT flag will be set. Default is 0x0000

30.3.3.27 SIM Block Wait Time Register HIGH (BWT_H)

See Figure 30-29 for an illustration of valid bits in the SIM Block Wait Time Register HIGH, and Table 30-36 for descriptions of the bit fields.

0x5001_8068 (BWT_H) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BWT_H[15:0]															
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 30-29. SIM Block Wait Time Register HIGH

Table 30-36. SIM Block Wait Time Register HIGH Field Descriptions

Field	Description
31–16	Reserved
15–0 BWT_H	BWT Register 16 MSB. The value in this register is the block wait time 16 MSB. The time from START bit of last byte sent from the SIM module to the START bit of the first byte sent from the SmartCard must be less than the value formed by the 32-bit register composed by BWT_H and BWT registers. If it is not, then the BWT flag is set. Default is 0xFFFF

30.4 SIM Functional Description

To best describe the organization of the SIM module from a user's point of view, it is instructive to view the SIM at a number of different levels of hierarchy. See Figure 30-30 for an illustration of the organization of SIM and connection of the signals to the two available serial ports.

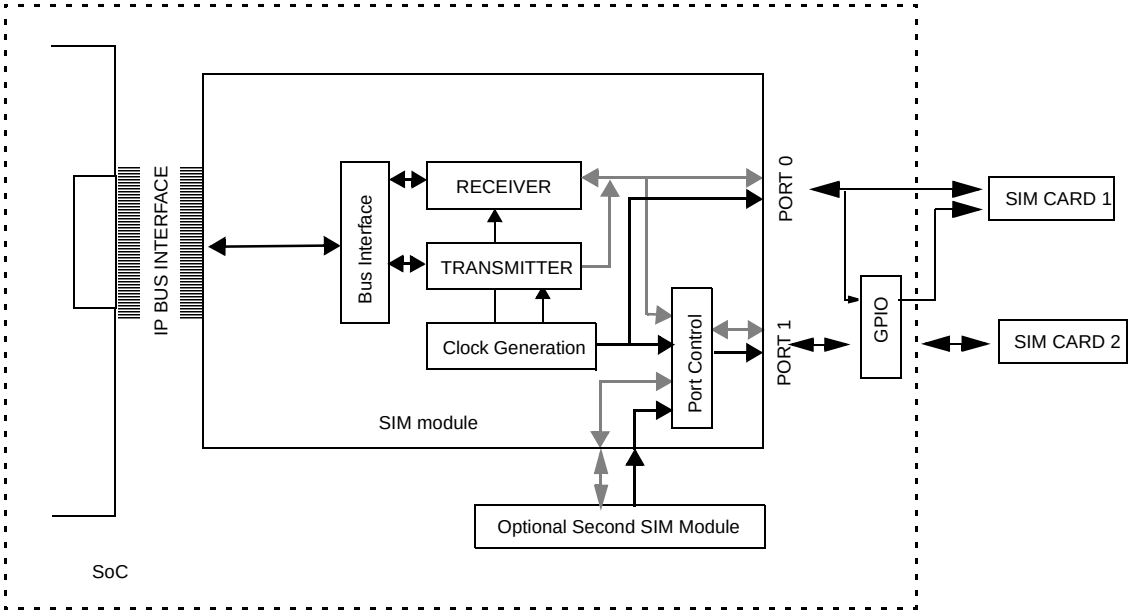


Figure 30-30. Block Diagram for SIM Module

The SIM module is essentially a standard UART with some special provisions made for SIM card communication. The SIM consists of nine main parts:

- IP Bus Interface
- Bus interface
- Clock generator
- Transmitter
- Receiver
- Port controller
- General purpose counter
- LRC blocks
- CRC blocks

30.4.1 Detailed SIM Block Diagram

See Figure 30-31 for an illustration of the block diagram in detail, specifically the nine main parts.

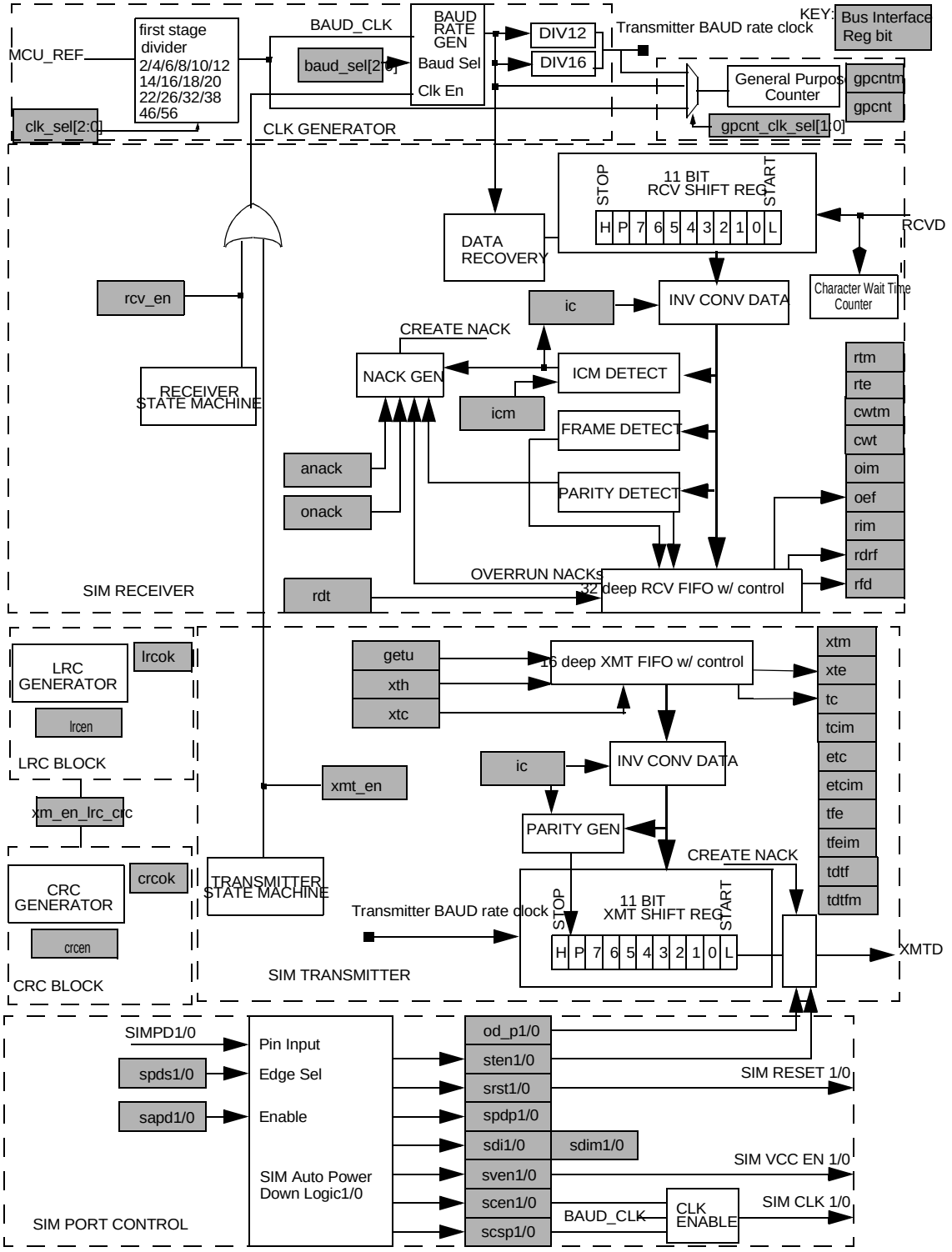


Figure 30-31. SIM Detailed Block Diagram

30.4.2 SIM Bus Interface

The SIM Bus interface block has been designed to enable the SIM module to be easily ported to other MCU cores. The bus interface block contains all of the logic that uses the bus interface signals. See Figure 30-32.

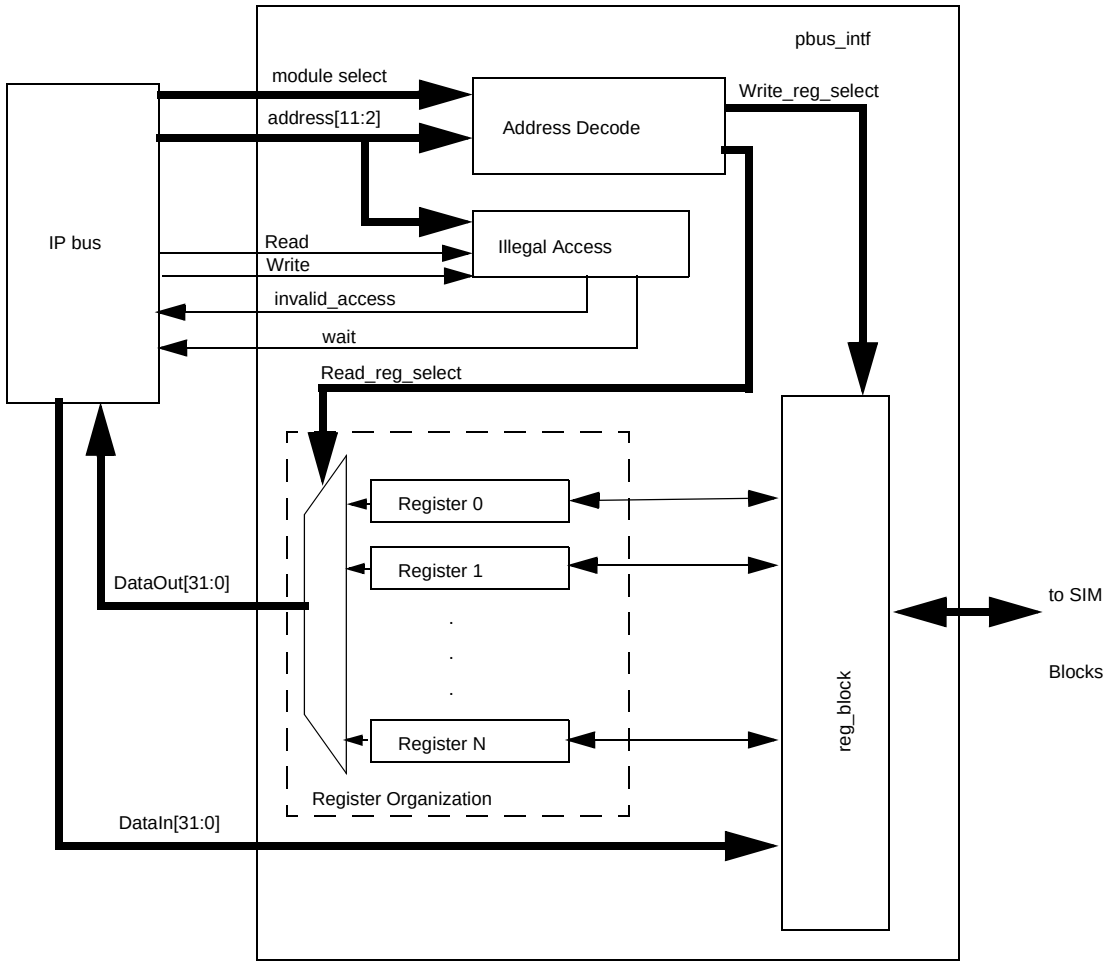


Figure 30-32. SIM Bus Interface

The bus interface block is responsible for peripheral bus address decoding, illegal access detection, dynamic wait-state requests, register organization, and register bit implementations. The address decoding uses the module address bus input along with the module select output of the IP bus to decode if the SIM module is being addressed. This decoding of these signals is done asynchronously. The output of the address decode is used with the read/write signals from the IP bus to determine the action requested by the bus master. If the read signal is active, the data_out bus will be driven with the register selected by the address decoding. If the write signal is active, the data_in bus will be latched into the register selected by the address decoder at the rising edge of the data_strobe signal.

The illegal access detection is implemented similarly to the address decoding. If the module_select signal becomes active while the address inputs are pointing to an unimplemented address, the invalid signal is asserted asynchronously to the bus master. The invalid signal can also be generated under certain register

access conditions such as writing to a full transmit FIFO, or writing to a read-only register. See Figure 30-33.

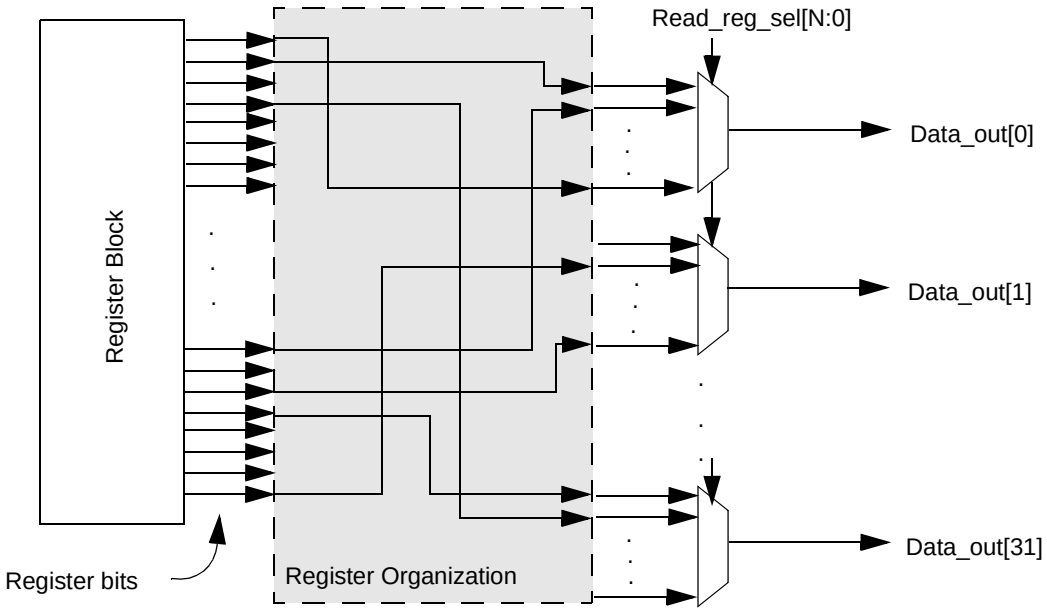


Figure 30-33. SIM Read Registers

The register bit implementations are done inside the reg_block sub-module. This block accepts the decoded read and write signals for the register bits generated in the address decoder. The data_strobe output is used as the clock input to the register bits in order to clock in the new data during a write access. The write_reg_sel signals are used to gate the write action with the address decoding. The clearing mechanisms for status bits are implemented as write-one-to-clear. See Figure 30-34.

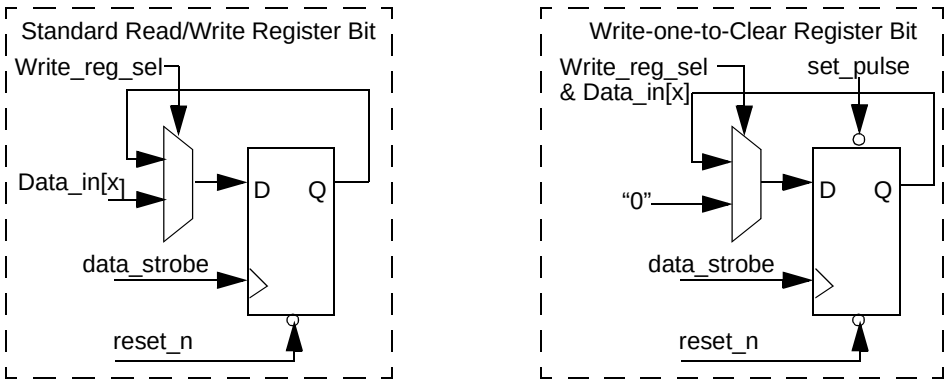


Figure 30-34. Register Bit Diagram

In the "write-one-to-clear" register bit example, "set_pulse" represents a pulsed signal generated in the data_strobe clock domain to set the register output.

30.4.3 SIM Clock Generator

The clock generator is responsible for implementing clock tree synthesis constructs, scan clock muxing, baud rate clock (BAUD_CLK) generation, and providing clocks to the transmitter, receiver, and port controller sections of the SIM module. See Figure 30-35 for the schematic of the SIM clock generator. The dividers outlined in bold, generate a pulsed (gated) clock of the desired frequency. The pulse is equal in duration to one half the MCU_REF period. This clock is used internal to the SIM module to drive the transmit and receive sections of the module.

The dividers outline in normal, generate a clock of 50% duty cycle to drive the clock pin of the SmartCard. This clock is not used internal to the SIM module

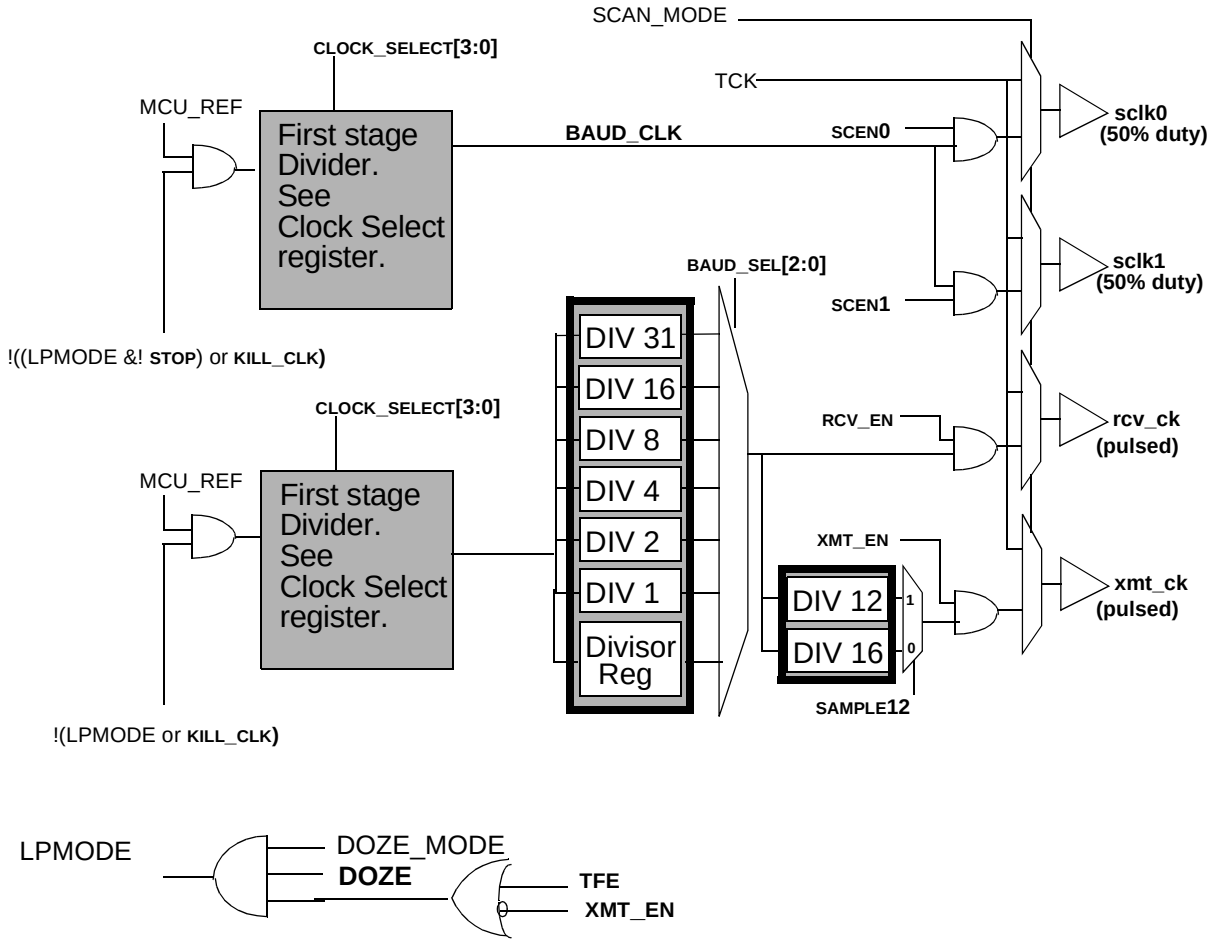


Figure 30-35. SIM Clock Generator

30.4.3.1 Clock Tree Synthesis

The clock tree synthesis constructs used in the current implementation follow the current clock methodology(srs3.1 clock methodology). This can be seen in the RTL of the sim_clocks.v module. There is an instance of the special cell (clk_driver_pos.v) to drive each clock. This cell can then be used to control the clock skew during the layout process.

30.4.3.2 Scan Test

The scan clock muxing is implemented according to the design for testability rules.

30.4.3.3 Baud Clock Generation

The baud rate clock generation performed by the clock generator results in one of four different frequencies. The default frequency is a divide by four of the MCU_REF input. The baud rate can be programmed to be MCU_REF divided by 2,4,6,8,10,12,14,16,18, 20,22,26,32,38,46 or 56 using the CLOCK_SELECT[3:0] bits as shown in the CLOCK_SELECT register. The baud rate clock is generated in two forms in the design. The BAUD_CLK that is used by the SIM cards (SCLK0, SCLK1) is generated so that it is 50% duty at all divide values. This is necessary to meet the requirements of the ISO 7816 specification. The BAUD_CLK that is used internal to the SIM module is generated as a gated version of the MCU_REF clock. The resultant clock will be a pulse of one-half MCU_REF period in width with the expected frequency. The gated baud clock complies with the clock methodology initiative. The 50% duty cycle clock does not comply with the clock methodology initiative but this clock is not used as a clock internal to the SIM module. It is only used a data path.

30.4.3.4 Transmitter Clock Generation

The transmitter clock (xmt_clk) is generated by the clock generator based on the values passed to it for the baud rate select (baud_sel[2:0]) and sample12. The transmit clock is gated by the transmit enable (XMT_EN) register bit. When the transmitter is enabled, the clock generator counts the appropriate number of receive clock (rcv_clk) positive edges to determine when to toggle the transmitter clock output. The transmitter clock is always based upon the receive clock.

30.4.3.5 Receiver Clock Generation

The receiver clock (rcv_clk) is generated by the clock generator based on the value passed to it for the baud rate select (baud_sel[2:0]). The receiver clock is gated by the receiver enable (RCV_EN) register bit. When the receiver is enabled, the clock generator counts the appropriate number of BAUD_CLK positive edges to determine when to toggle the receiver clock output. The number of BAUD_CLK positive edges is determined by the value of the baud rate select input (baud_sel[2:0]) and is programmable to these divisors: 31 (slowest), 16, 8, 4, 2, 1, and divisor[6:0]. The programmable divisor (divisor[6:0]) is programmable via the DIVISOR_REG register.

30.4.3.6 Port Control Clock Generation

The port controller clocks are provided by the clock generator for use on the SIM card ports. These clocks are equivalent in frequency to the BAUD_CLK and are gated by the SIM clock enable (scen0 and scen1) signals. The level at which the card clocks (SCLK0, SCLK1) are stopped when disabled is determined by the SIM clock select polarity (SCSP0, SCSP1) inputs to the clock generator. Synchronizers are implemented to ensure glitch free operation of the card clocks when enabling/disabling, or changing the clock stopped polarity.

30.4.3.7 Low Power Mode Clock Control

The clock generator block is responsible for gating the clocks to the SIM module appropriately whenever a low power mode instruction is decoded. The IP interface provides two signals that encode the two available low-power states of the processor. These states are as follows: STOP and DOZE. The response to the DOZE instruction is controlled through the use of the DOZE bit in the RESET_CNTL register. See the RESET_CNTL register description. The response to the STOP instruction is controlled through the use of the STOP bit in the RESET_CNTL register. See the RESET_CNTL description

30.4.4 SIM Transmitter

The SIM Transmitter block has been designed to localize all transmit related circuitry into one section of hierarchy. The Transmitter block is comprised of the following sections of logic: transmit state machine, transmit shift register, transmit FIFO, guard time generator, transmit NACK control, and transmit data convention.

30.4.4.1 Transmit State Machine

The Transmit state machine is the heart of the transmitter block. The state machine is responsible for sequencing through a transmit operation while reacting to inputs from the receiver, the transmit FIFO, and the guard time circuit. See Figure 30-36 for flow diagram of the transmit state machine.

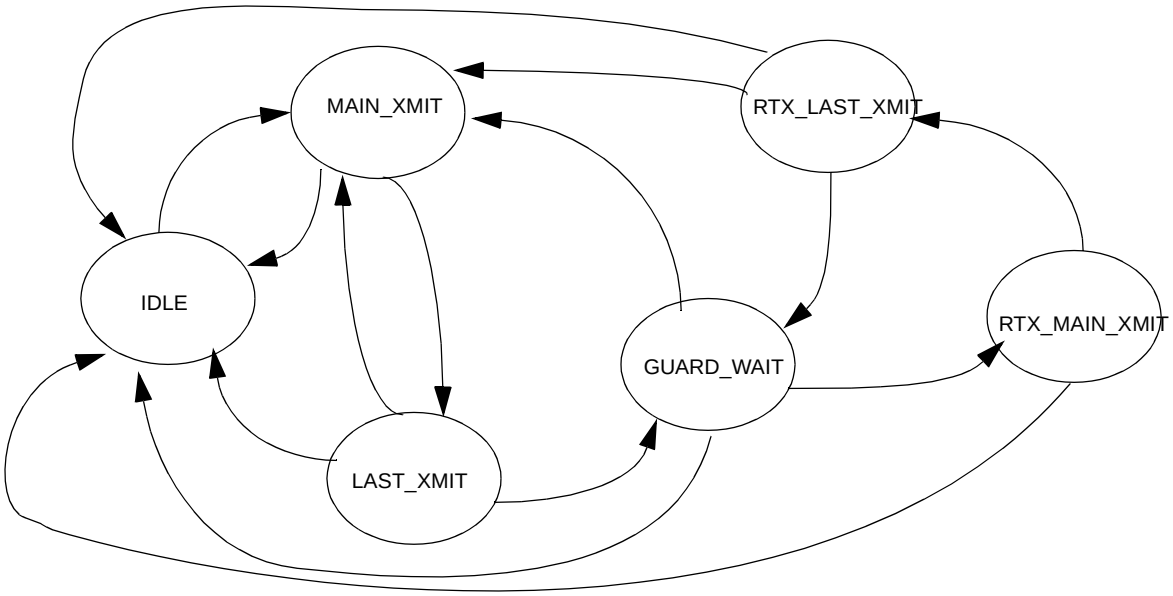


Figure 30-36. Transmit State Machine

The functions performed by each state are as follows:

- IDLE
  - This is the initial state. The state machine waits here until it detects XMT_EN is set and a write to the transmit FIFO has occurred. The data pointed to by the transmit read pointer is loaded into the shift register, and the state machine transitions to the MAIN_XMIT state. Any time XMT_EN is cleared, the state machine will return to this state.

- MAIN_XMIT
  - The transmitter is operating normally in this state. The data in the shift register is shifting once every transmit clock cycle. When the second to last bit of the current transmission is about to be sent, the state machine transitions to the LAST_XMIT state.
- LAST_XMIT
  - This state transmits the last bit of the current transmission and determines the next operation. One of the following will occur.
    - If getu[7:0] is non-zero, jump to the GUARD_WAIT state.
    - If a transmit NACK error occurred, with a zero in getu[7:0], jump to MAIN_XMIT state to retransmit the current byte.
    - If no transmit NACK, and getu[7:0] is zero, load the shift register, jump to MAIN_XMIT to transmit the next byte
    - If no transmit NACK, getu[7:0] is zero, and the FIFO is empty, jump to IDLE state; set the transmit complete (TC) flag.
- GUARD_WAIT
  - The state machine remains in this state until the Guard time counter has expired.
    - If a transmit NACK error occurred on last transmission, jump to RTX_MAIN_XMIT and re-transmit
    - If no transmit NACK and the FIFO is not empty, load the shift register, jump to MAIN_XMIT to transmit the next byte.
    - If no transmit NACK and the FIFO is empty, return to the IDLE state.
    - If transmit NACK threshold is detected, stop transmitter, set XTE flag, and jump to the IDLE state.
- RTX_MAIN_XMIT
  - The transmitter is operating normally in this state. The data in the shift register is shifting once every transmit clock cycle. When the second to last bit of the current transmission is about to be sent, the state machine transitions to the RTX_LAST_XMIT state. This state is identical to the MAIN_XMIT state except that it retransmits the previously NACK'ed byte.
- RTX_LAST_XMIT
  - This state transmits the last bit of the current re-transmission and determines the next operation. One of the following will occur.
    - If getu[7:0] is non-zero, jump to the GUARD_WAIT state.
    - If a transmit NACK error occurred, with a zero in getu[7:0], jump to GUARD_WAIT state to check Transmit NACK threshold.
    - If no transmit NACK, getu[7:0] is zero, and the FIFO is not empty, load the shift register, jump to MAIN_XMIT to transmit the next byte
    - If no transmit NACK, getu[7:0] is zero, and the FIFO is empty, jump to IDLE state; set the transmit complete (TC) flag.

30.4.4.2 Transmit Shift Register

The transmit shift register is 11 bits wide and controlled by the transmit state machine described previously. The shift register shifts out data at the transmit clock frequency (xmt_ck).

30.4.4.3 Transmit FIFO

The transmit FIFO is implemented inside the transmitter block. The FIFO depth is 16 bytes. The FIFO block is shared by both SIM module ports. The transmit FIFO cannot be accessed by the alternate SIM module through the alternate port. Each write to the transmit FIFO increments the transmit FIFO write pointer. Each time the transmit shift register is loaded from the transmit FIFO, the transmit FIFO read pointer is incriminated. When the read and write pointers are equal, the transmit FIFO empty flag (TFE) will be set. Software has no visibility of the transmit FIFO pointers, but a transmit FIFO threshold value can be set to alert the software when the number of bytes in the FIFO has reached a specified level. A read of the transmit FIFO register (PORTx_XMT_BUF) will return the last byte written to the FIFO. Writes to the transmit FIFO can occur at any time.

The transmit FIFO can be flushed by setting the XMT_FLUSH bit in the RESET_CNTL register. A transmit NACK threshold error (XTE) will halt the transmitter, and flush the transmit FIFO. The flush operation resets the transmit read and write pointers to equal values. Everything in the transmitter block is reset by the transmit flush operation. This does not include the control registers associated with the transmitter. The Transmit data threshold flag (TDTF) does not get cleared by the XMT_FLUSH operation.

30.4.4.4 Transmit Guard Time Generator

The guard time generator is simply a counter that is clocked by the transmit clock in order to delay the beginning of the next transmission and the setting of the transmit complete interrupt flag (TC) by a programmable amount of transmit bit widths (ETUs). The duration of the count is controlled by the GUARD_CNTL register (getu[7:0]). See Figure 30-37 for depiction of three transmit operations in order to show the effect of the guard time generator logic

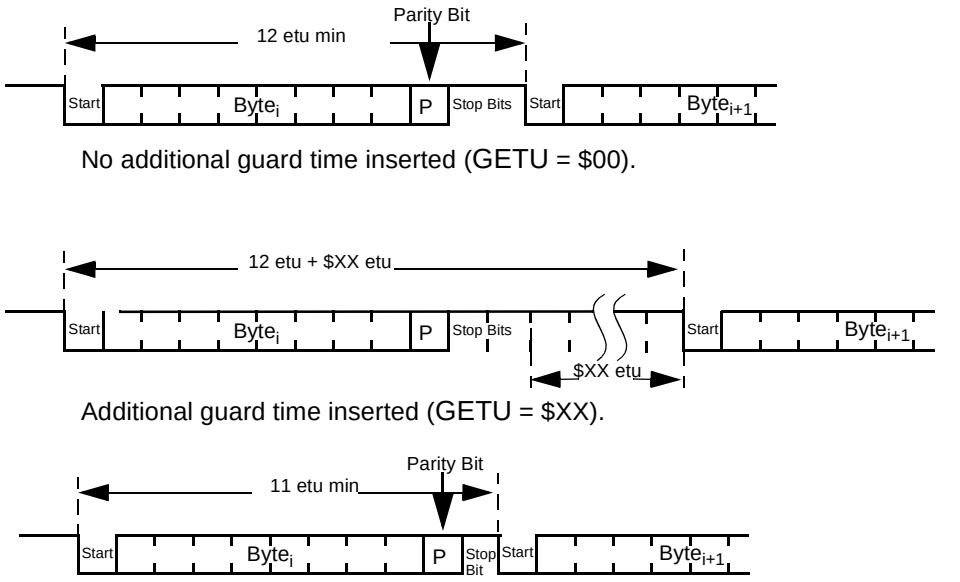


Figure 30-37. Transmit Guard Time

30.4.4.5 Transmit NACK Generator

The transmit NACK generator is responsible for driving the transmitter output low during the STOP bit time to signify an error was detected in the received data from the SIM card. This logic responds to a NACK request generated by the receiver block. See Figure 30-38 shows a typical SIM transaction with the NACK pulse inserted.

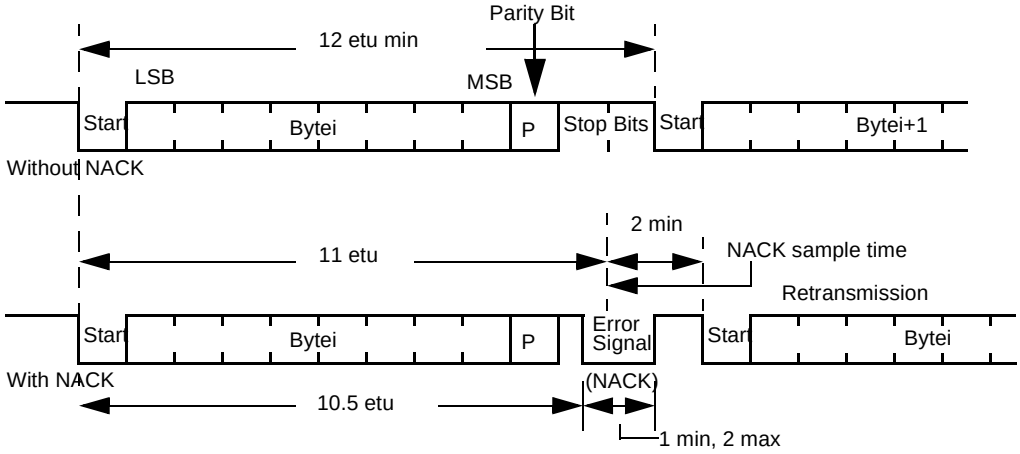
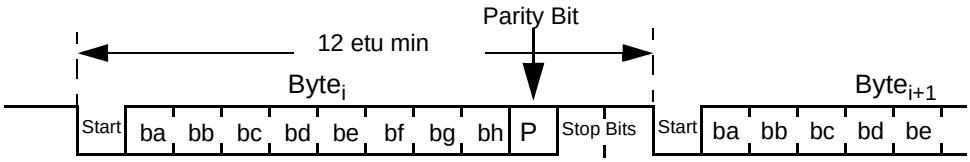


Figure 30-38. Transmit NACK Generator

The transmit NACK generator is also responsible for keeping track of the number of NACKs received during a transmit operation. The SIM receive state machine detects NACKs generated by the SIM card, and reports them to the transmit NACK logic. Once the number of detected NACKs has reached the programmed threshold (xth[3:0]), an interrupt flag is generated, the transmit FIFO is flushed, and the transmitter is disabled.

30.4.4.6 Transmit Data Convention Logic

The transmit data convention logic provides the support for the two different data conventions available in SIM cards. See Figure 30-39 for an illustration of SIM data conventions.



Parity Bit: if configured for even parity, total number of logic ones in the 9 bits (8 bits data, 1 parity bit) will be even
if configured for odd parity, total number of logic ones in the 9 bits (8 bits data, 1 parity bit) will be odd
When configured for inverse convention, the parity bit is inverted by SIM before being transmitted.

Direct Convention: ba is LSB of data byte to be sent. bh is MSB.
Neither the data bits nor parity bit is logically inverted.

Inverse Convention: ba is MSB of data byte to be sent. bh is LSB.
Both the data bits and parity bit are logically inverted by hardware.

Figure 30-39. SIM Data Conventions

The direct data convention is the default. If the inverse convention bit (IC) in the DATA_FORMAT register is set, then the transmit data convention logic will convert the output of the transmit FIFO to the inverse convention before sending it to the transmit shift register.

30.4.5 SIM Receiver

The SIM Receiver block has been designed to localize all receive related circuitry into one section of hierarchy. The receiver block is comprised of the following functions: receive state machine and the receive FIFO.

30.4.5.1 Receive State Machine

The receive state machine is responsible for sampling the receive data pin and capturing the bit value into the receive shift register. Additionally, the receive state machine can detect the START bit, parity errors, framing errors, and initial character when operating in initial character mode.

Once enabled by the RCV_EN bit in the ENABLE register, the receive state machine sequences through the states as shown in Figure 30-40.

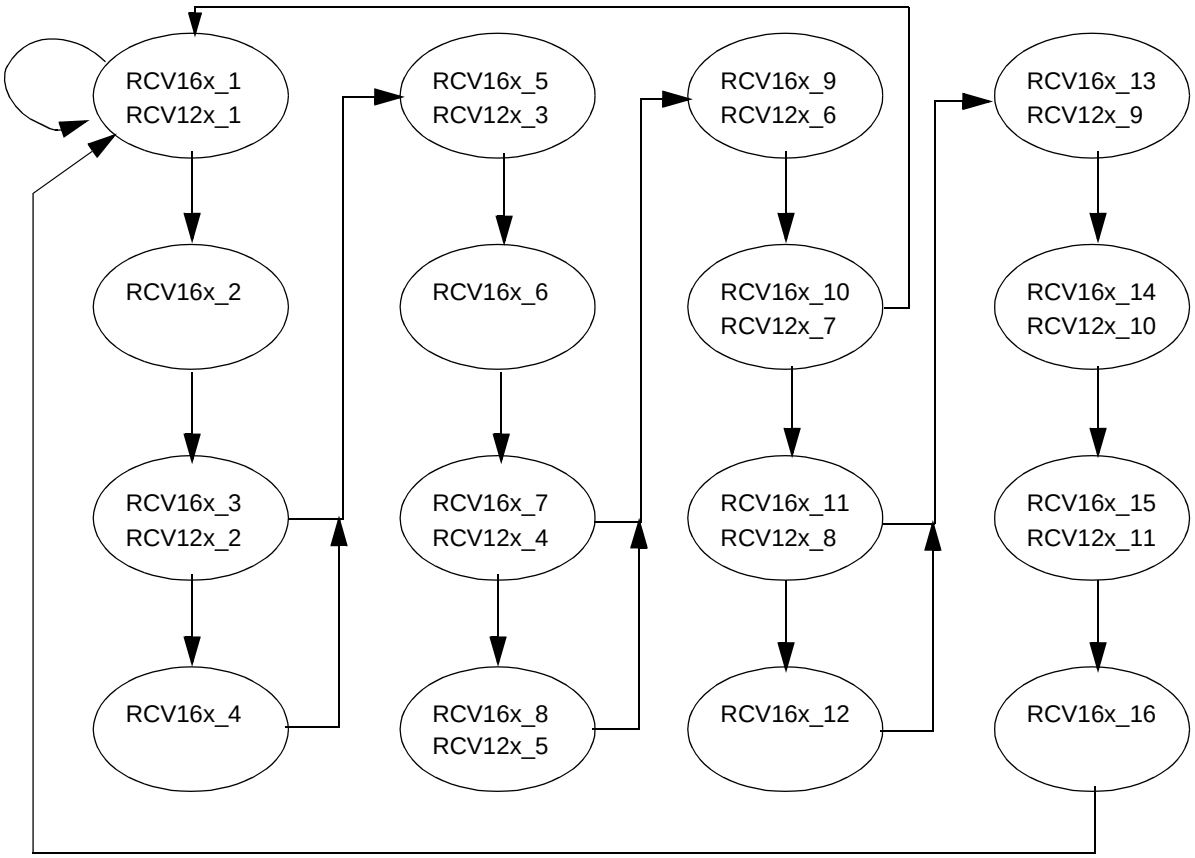


Figure 30-40. Receive State Machine

The states identified with a “16x” are used when operating in a 16X oversampling mode. The states identified with a “12x” are used when operating in a 12X oversampling mode. This mode is only used when sample12 is set to “1”. The number following the oversampling mode identifier represents the state number in the current mode. There are 12 states in “12x” mode, and 16 states in “16x” mode. Some states simply implement a one RCV_CK delay. States that perform additional functions are as follows:

- RCV16x_1, RCV12x_1
  - This is the initial state of the receive state machine. If the first bit has not been received, the state machine remains in this state until a valid START bit is detected. For every subsequent bit, this state is simply a one rck_ck cycle delay.
- RCV16x_3, RCV12x_2
  - This state captures the first sample of the current receive data input.
- RCV16x_7, RCV12x_4
  - This state captures the second sample of the current receive data input.
- RCV16x_8, RCV12x_5
  - This state captures the third sample of the current receive data input.
- RCV16x_9, RCV12x_6

— This state captures the fourth sample of the current receive data input. If the current bit is the 11th bit of the transfer, this state checks the for valid parity and valid initial character when appropriate. If an error is detected, a NACK is generated in this state when enabled.

— When the transmitter is active, the sample captured during this state is the first sample of the NACK window. This is 0.5 ETU into the 11th bit of the transfer.

- RCV16x_10, RCV12x_7
  — If the current bit is the first bit of a transfer, this state will check the validity of the previous four samples and perform a majority vote on whether to accept the data as a valid START bit. If the START bit is valid, the data is shifted into the receive shift register.
  — If this is not the first bit of a transfer, this state does a majority vote on the previous four samples and places the result in the receive shift register.
- RCV16x_11, RCV12x_8
  — If the current bit is the last bit of the transfer (first stop bit), this state will signal the shift register to place its contents in the receive FIFO. Parity and framing errors are also detected in this state.
- RCV16x_13, RCV12x_9
  — If the current bit is the last bit of the transfer (first stop bit), this state takes the second sample of the NACK window.
- RCV16x_14, RCV12x_10
  — If the current bit is the last bit of the transfer (first stop bit), this state takes the third sample of the NACK window.
- RCV16x_15, RCV12x_11
  — If the current bit is the last bit of the transfer (first stop bit), this state takes the fourth sample of the NACK window.
- RCV16x_16, RCV12x_12
  — This state represents the end of the current receive input bit. Several operations occur during this state, including:
    – Increment bit counter
    – Perform a majority vote on the NACK samples and notify the transmitter if a NACK pulse was detected.

30.4.5.2 Data Sampling/Voting

The receive state machine runs at the receive clock rate (RCV_CK). This clock is used to oversample the received data at either a 16X or 12X sample rate. For each input bit, the receive state machine captures 3 samples. A majority voting algorithm is then applied to determine the value of the bit received. The value common to 2 or more of the samples is determined to be the bit value in the receive shift register.

30.4.5.3 Start Bit Detection

The SIM receive input is defined to be high when not active. The data transmission is defined to begin with a low pulse for a bit duration. This is called the START bit. The receive state machine is responsible for detecting and validating the START bit. The receive state machine samples the START bit three times

using a majority voting scheme to determine if the START bit is valid. This will effectively filter out any low receive inputs shorter than one RCV_CK period. See Figure 30-41 for an illustration of a typical SIM data transaction with the START bit identified.

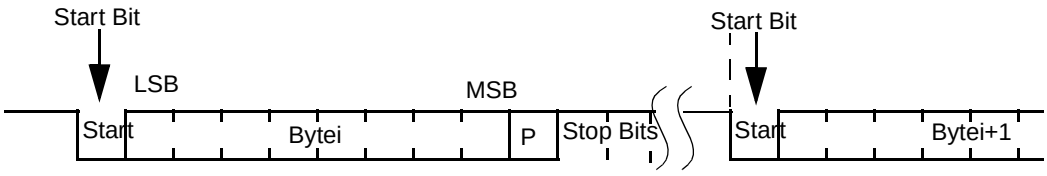


Figure 30-41. Start Bit Diagram

30.4.5.4 Parity Error Detection

The receive state machine is responsible for detecting parity errors in the received data. Data is always transmitted with even parity, except when in inverse convention mode. In inverse convention mode, all data bits and the parity bit are complemented making the data appear to be odd parity. The parity bit is defined as the 10th bit of the received data. The parity of the 2nd through 10th received bits is calculated by the receiver parity logic. This logic determines if the parity of the 9 received bits is correct. See Figure 30-42 for an illustration of a typical SIM data transaction with the parity bit identified.

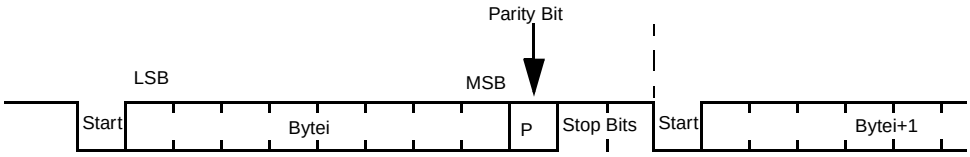


Figure 30-42. Parity Bit Diagram

When a parity error is detected on a given byte, the RCV_PF bit for that byte is set in the receive FIFO. A parity error cannot cause an interrupt, but it can signal the SIM transmitter to create a NACK pulse to the SIM card asking for a retransmission of the corrupted data. NACK generation upon a parity error is enabled by setting the ANACK bit in the CNTL register.

30.4.5.5 Framing Error Detection

The receive state machine is responsible for detecting framing errors in the received data. A SIM data transaction is defined as 11 or 12 bits long consisting of the START bit, 8 data bits, 1 parity bit and 1 or 2 stop bits. A framing error occurs when the stop bit is not detected during the 11th bit time of a data transaction. The stop bit is generally defined as two bit times (ETUs) of a high pulse following the parity bit. When the GUARD_CNTL register is programmed to $FF, the stop bit is defined as one bit time. A framing error can only occur when the parity bit of the current byte is low, and the STOP bit arrives late. See Figure 30-43 for an illustration of a typical SIM data transaction with the stop bits identified. Also, shown is a SIM data transaction with a late arriving STOP bit indicating a framing error.

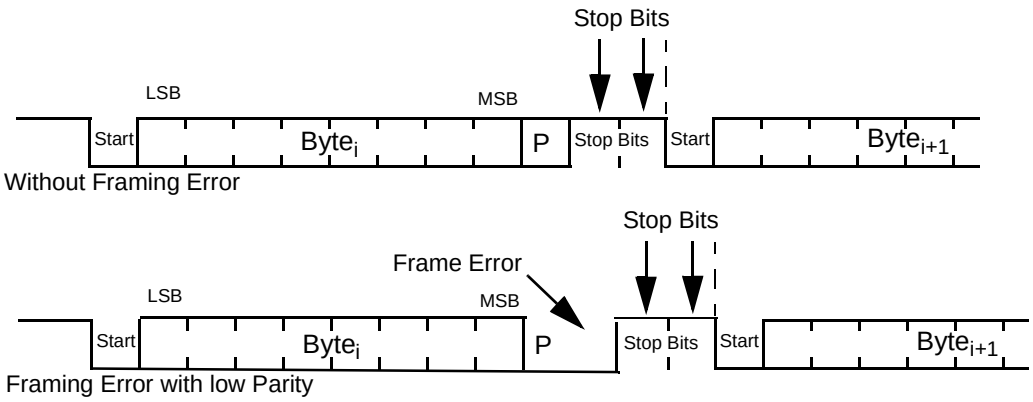


Figure 30-43. Framing Error Diagram

When a framing error is detected on a given byte, the RCV_FE bit for that byte is set in the receive FIFO. A framing error cannot cause an interrupt, nor can it create a NACK pulse to the SIM card asking for a retransmission of the corrupted data.

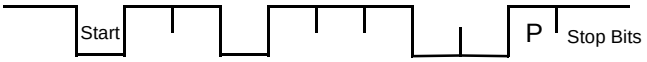
30.4.5.6 NACK Detection

The existence of the NACK pulse is sampled by the receive state machine at 11 elementary time units (ETUs) after the falling edge of the START bit. An ETU is equivalent in time to 1 transmit clock period. Once the receiver detects a NACK, it signals the transmitter that an error occurred. The transmitter will not initiate retransmission for at least another two ETU times as required by the ISO 7816 specification.

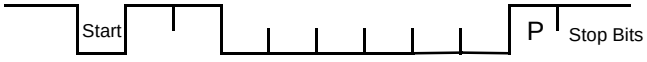
30.4.5.7 Initial Character Detection

The SIM receive state machine supports the detection of special characters that allow it to determine what data format is being used by the connected SIM card. When placed in initial character mode, the SIM expects to receive one of two potential characters that it will use to set the data format control bit, IC, in the DATA_FORMAT register.

The two possible data formats are inverse convention and direct convention. Figure 30-44 and Figure 30-45 illustrate the differences between the two formats. Essentially, inverse convention differs from direct convention in that the order of the data is flipped MSB for LSB and the data bits and parity bit are logically inverted. When receiving inverse convention data, the transformation of the data back to direct convention format is done in hardware, including the inversion of the data and parity bits.

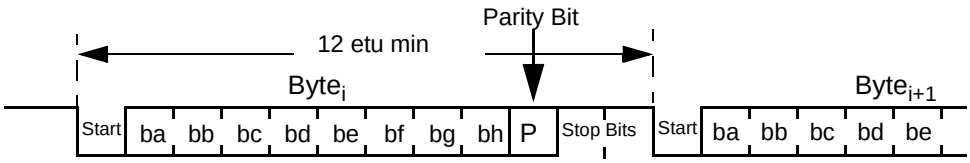


Initial Character from SIM for setting direct convention
(seen as $3B when decoded by direct convention)



Initial Character from SIM for setting inverse convention
(seen as $3F when decoded by inverse convention)

Figure 30-44. Valid Initial Characters



Parity Bit: If configured for even parity, total number of logic ones in the 9 bits (8 bits data, 1 parity bit) will be even
If configured for odd parity, total number of logic ones in the 9 bits (8 bits data, 1 parity bit) will be odd
When configured for inverse convention, the parity bit is inverted by SIM card before being transmitted.

Direct Convention: ba is LSB of data byte to be sent. bh is MSB.
Neither the data bits nor parity bit is logically inverted.

Inverse Convention: ba is MSB of data byte to be sent. bh is LSB.
Both the data bits and parity bit are logically inverted by SIM card.

Figure 30-45. Inverse Convention Versus Direct Convention

30.4.5.8 Receive FIFO

The receive FIFO is implemented inside a sub-block of the receiver block. The FIFO depth is 32 bytes. The FIFO block is shared by both SIM module ports. The receive FIFO *cannot* be accessed by another SIM module through the alternate port. Only the port1 IO pins can be accessed through the alternate port. The secondary SIM would then uses its own internal FIFO while using the primary SIM IO ports. This is only done if it is required to control two SIM cards at the same time. It the two SIM card access only occurs one at a time, then a single SIM module can control both SIM cards (one on port0 and one on port1). The receive FIFO is accessed through the PORTx_RCV_BUF registers.

The receive FIFO is loaded from the receive shift register after the final bit of the current SIM card transmission has been received. The FIFO contains 10 bits per transmission. The lower eight bits contain the received data byte. Bits 8 and 9 contain the parity and framing status for the received byte.

Each read from the receive FIFO increments the receive FIFO read pointer. Each time the receive shift register is transferred to the receive FIFO, the receive FIFO write pointer is incriminated. When the difference between the read and write pointers equals the programmed threshold value (rdt[4:0]), the receive data register full flag (RDRF) will be set. An interrupt can be generated by the RDRF flag if the RIM bit in the INT_MASK register is cleared. Software has no visibility of the receive FIFO pointers. A write to the receive FIFO register (PORTx_XMT_BUF) will generate an invalid access exception to the processor.

The receive FIFO can be flushed by setting the RCV_FLUSH bit in the RESET_CNTL register. The flush operation resets the receive read and write pointers to equal values. All logic associated with the receiver will be reset by the flush operation except for receiver control registers.

30.4.5.9 Overrun Detection

The receive FIFO logic is responsible for detecting an overrun condition. When a received byte is transferred from the receive shift register to a receive FIFO that contains 32 unread bytes, the SIM receiver

will flag an overrun condition. This condition will always set the overrun error flag, OEF in the RCV_STATUS register. The received byte will be discarded leaving the 32 unread bytes in the FIFO unaltered. The SIM module will generate a NACK to the SIM card on an overrun condition if the ONACK bit in the CNTL register is set. The SIM module will continually NACK SIM card transmissions until a read of the receive FIFO occurs.

30.4.5.10 Character Wait Time Counter

The SIM receiver block includes a 16-bit counter that counts the number of bit times (ETU's) between received characters. When enabled, the Character Wait Time Counter (CWT) will not start counting until after the STOP bit(s) of a valid character has been received. The counter is synchronized to the receive character bit positions so that an accurate count of the number of ETU's between characters can be made. The Character Wait Time Counter has a 16-bit programmable comparator. Software can write the expected number of ETU's between characters to the comparator. If the time between characters exceeds this value, an interrupt flag will be set and an interrupt generated if the mask is clear.

30.4.6 SIM Port Control

The SIM Port Control block has been designed to localize all port related circuitry into one section of hierarchy. The port control block is comprised of the following functions: SIM card interface, SIM Card presence detect, and SIM card auto-power down.

30.4.6.1 SIM Card Interface

The SIM module allows for direct control of two separate SIM cards. The SIM module does not support simultaneous communication with two SIM cards. In order to achieve simultaneous communication with two SIM cards, a second SIM module must be used. The SIM module can relinquish control of the inactive port to a secondary SIM module by setting the AMODE bit in the SETUP register.

The SIM card clock is generated in the SIM clock generator. The SIM card reset and SIM card voltage enable are controlled by software through the PORTX_CNTL registers.

See Figure 30-46 for an illustration of a example SIM module hookup to two SIM cards. The power management chip shown is used to provide Vcc for the SIM card, and level translators for the remaining signals when interfacing to a SIM card operating at a different voltage than SIM module. The SIM module can directly access a SIM card if it is operating at the same voltage. In this case, the 3VOLTX bit in the PORTX_CNTL registers can be configured so the SIM Port transmit output as bidirectional. This frees up the SIM port receive pin to be used as general purpose I/O. If the SIM module and the SIM card are operating at different voltages, then the power management level translators must be used. In this case, the 3VOLTX bit must be cleared and then both the RX and TX pins will be used.

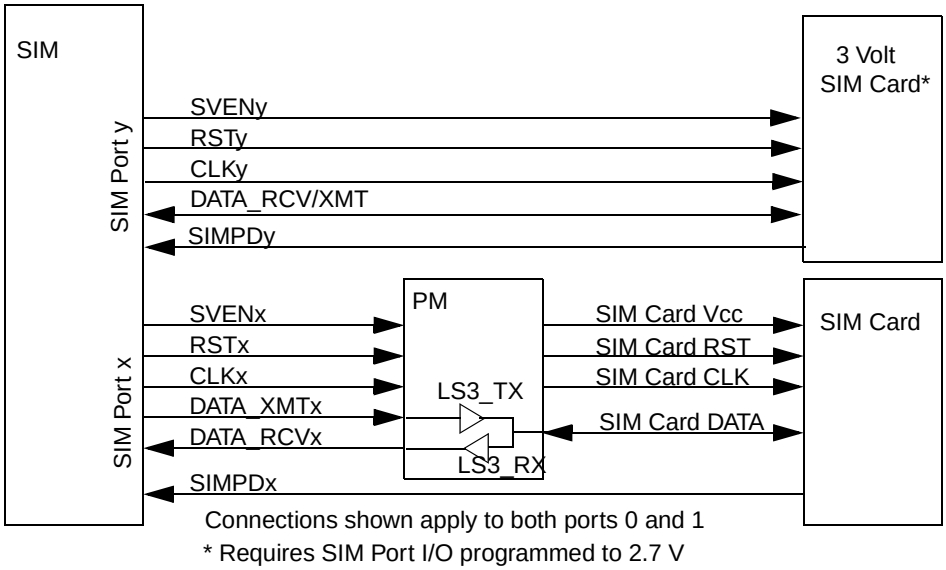


Figure 30-46. SIM Card Hookup

30.4.6.2 SIM Card Presence Detect

The SIMPDx input allows for detection of the insertion or removal of a SIM card. The SPDSX control bit in the PORTX_DETECT register allows the software to configure which edge of the SIMPDx pin will cause a presence detect event. A maskable interrupt can be generated when a SIMPDx event occurs.

An internal pull-down device is present on the SIMPDx pins. This will provide for a high to low transition on the SIMPDx pin when a SIM card is removed.

30.4.6.3 SIM Card Automatic Power Down

When interfacing to the SIM cards, it is necessary to follow a particular sequence when powering them up and down. The SIM port control block contains hardware that provides the correct sequence to power down a SIM card (see Figure 30-47). The power up sequence must be done manually by the software using the pin control bits supplied in the PORTx_CNTL registers.

The power down sequence is specified in ISO 7816 as:

1. RST transitions from high to low
2. CLK is turned off to a low
3. I/O transitions from tri-state to low
4. SIM Vcc is turned off

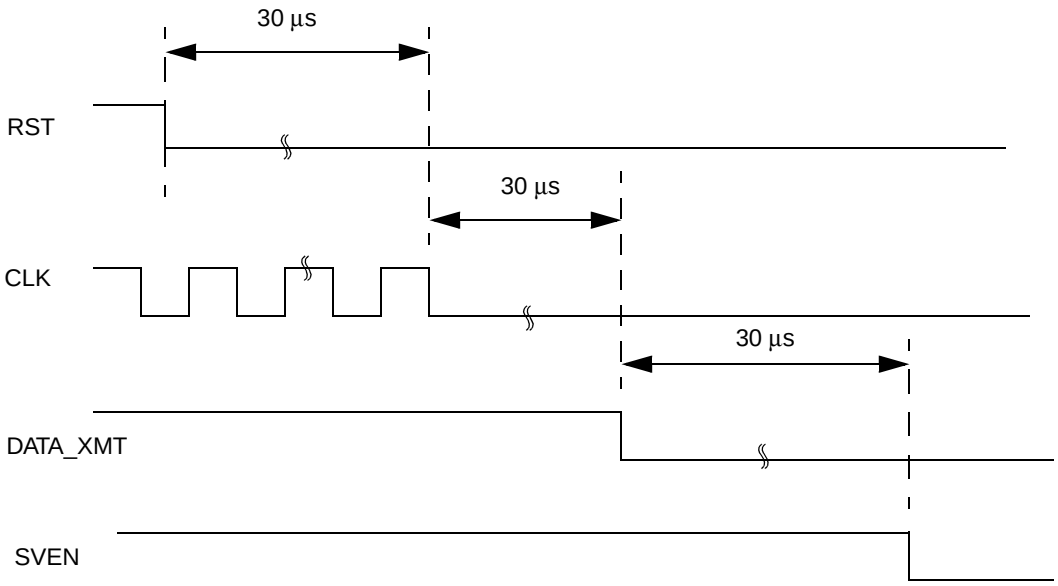


Figure 30-47. Auto Power Down Sequence

30.4.7 SIM General Purpose Counter

The SIM module provides a 16-bit counter for use when timing events during SIM card communication. The clock source for the counter is selectable between three sources: BAUD_CLK, RCV_CLK, or XMT_CK (ETU clock). The GPCNT_CLKSEL[1:0] bits in the CNTL register are used to select the clock input. The counter is enabled as soon as the input clock is selected. The starting of the counter is immediate once the input clock is running. Software can control the three input clock sources by using the KILL_CLK, RCV_EN and XMT_EN bits provided in the RESET_CNTL and ENABLE registers.

To run the counter from the card clock source the following conditions must be met:

1. ‘KILL_CLK = 0’ in the RESET_CNTL register.
2. ‘GPCNT_CLKSET[1:0] = 01’ in the CNTL register

The counter will begin to count at the card clock rate as soon as these conditions are met.

To run the counter from the receive clock source the following conditions must be met:

1. ‘KILL_CLK = 0’ in the RESET_CNTL register.
2. ‘RCV_EN = 1’ or XMT_EN = 1 in the ENABLE register
3. ‘GPCNT_CLKSET[1:0] = 01’ in the CNTL register

The counter will begin to count at the card clock rate as soon as these conditions are met.

To run the counter from the ETU (transmit) clock source the following conditions must be met:

1. ‘KILL_CLK = 0’ in the RESET_CNTL register.
2. Either one of the following:
   “RCV_EN = 1’ in the ENABLE register and ‘CWT_EN = 1’ in the CNTL register
   ‘XMT_EN = 1’ in the ENABLE register

3. ‘GPCNT_CLKSET[1:0] = 11’ in the CNTL register

The counter will begin to count at the card clock rate as soon as these conditions are met.

The counter can be reset by setting GPCNT_CLKSEL[1:0] to 00. A 16-bit comparator value is provided that allows the software to select a count value at which an interrupt flag can be set and an interrupt generated if the mask is clear.

30.4.8 SIM LRC Block

The SIM module provides an 8-bit Linear Redundancy Check (LRC) generator/checker. The block is provided for use with T=1 SIM cards that support LRC. This block can be enabled through the LRCEN bit in the CNTL register. This block performs an 8-bit exclusive-OR on all received or transmitted characters. At the end of the reception of a block of characters, the result is expected to be 00. If so, the LRCOK bit is set in the RCV_STAT register. During transmission, the LRC block Exclusive-OR’s each character that is transmitted with the current value of the LRC. If the XMT_EN_LRC_CRC bit in the CNTL register is set, the LRC value will automatically be sent by the SIM transmitter as the final character when the transmit FIFO empties.

The LRC value can be reset in multiple ways. Clearing the LCREN bit in the CNTL register will reset the LRC value. At the end of a transmission (either after the LRC byte is transmitted, or after the last character in the transmit FIFO is sent when XMT_EN_LRC_CRC is clear), the LRC value is automatically reset by the SIM hardware. Finally, when setting the XMT_EN bit, the SIM hardware resets the LRC value.

30.4.9 SIM CRC Block

The SIM module provides an 16-bit Cyclic Redundancy Check (CRC) generator/checker. The block is provided for use with T=1 SIM cards that support CRC. This block can be enabled through the CRCEN bit in the CNTL register. This block performs a polynomial based check on all received or transmitted characters. The polynomial used is the standard CRC-CCITT where g(x) = x^16 + x^12 + x^5 + 1. The CRC register is initialized to all “1” before the data is shifted in. Before transmission the resulting CRC is inverted. See Figure 30-48 for an illustration of the SIM CRC block diagram.

For example, transmitting a 0xFA results in the following:

- Data = 0x5F (bit reversal of 0xFA)
- crc = 0x4AEA
- invert the crc = 0xB515
- data transmitted = 0x5F, 0xB5, 0x15

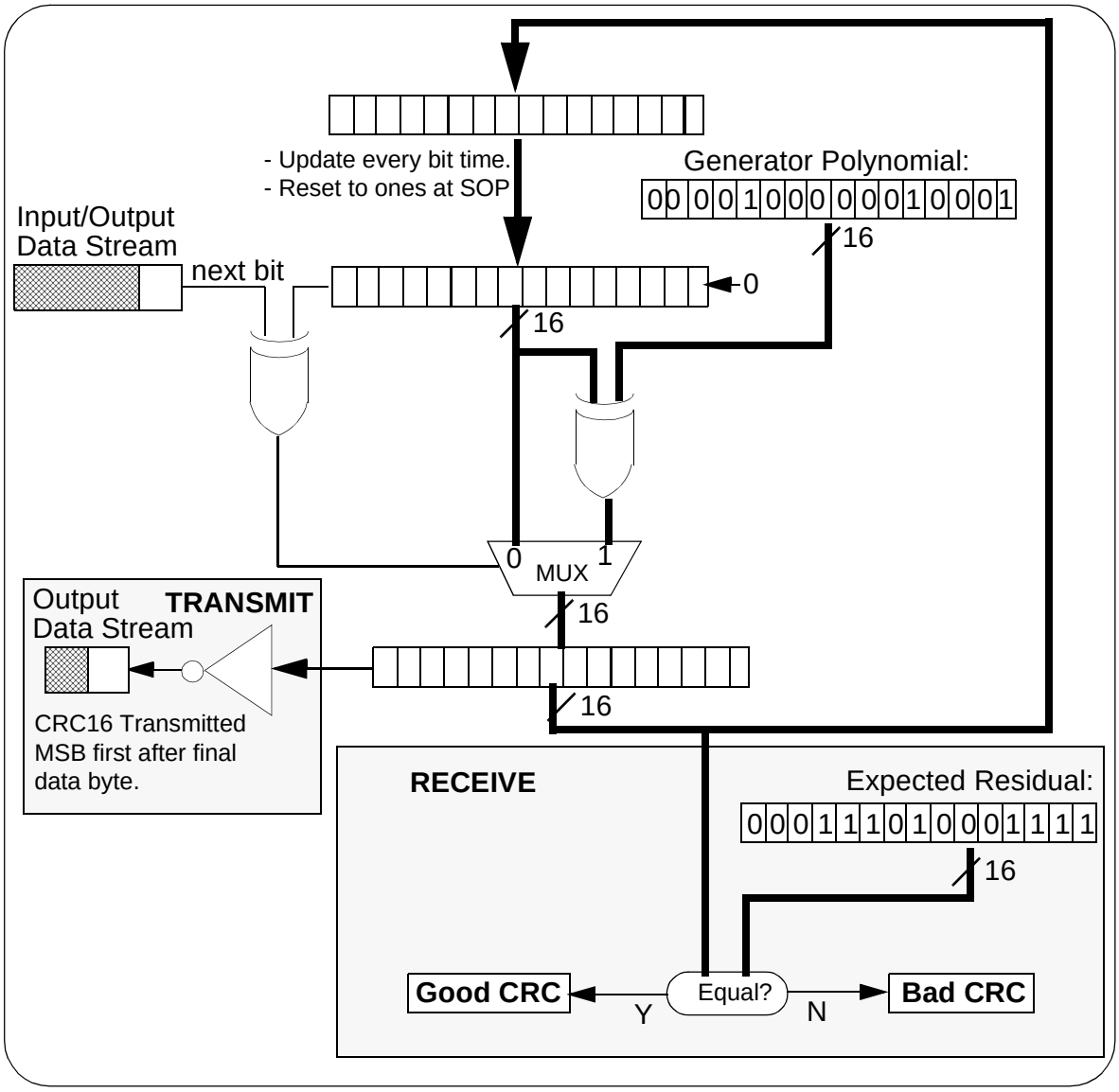


Figure 30-48. CRC Block Diagram

At the end of the reception of a block of characters, the residual from the CRC calculation is compared to $1D0F. If so, the CRCOK bit is set in the RCV_STAT register. During transmission, the CRC block updates the current value of the CRC residual using each character. If the XMT_EN_LRC_CRC bit in the CNTL register is set, the CRC value will automatically be inverted and sent by the SIM transmitter as the final two characters when the transmit FIFO empties.

The CRC value can be reset in multiple ways. Clearing the CRCEN bit in the CNTL register will reset the CRC value. At the end of a transmission (either after the CRC characters are transmitted, or after the last character in the transmit FIFO is sent when XMT_EN_LRC_CRC is clear), the CRC value is automatically reset by the SIM hardware. Finally, when setting the XMT_EN bit, the SIM hardware resets the CRC value.

30.4.10 SIM Interrupts

See Table 30-37 for the list of all possible interrupt sources and their corresponding mask bits. All SIM interrupts are logically OR'ed to create the active low irq_n and data_irq_n signals that go to the ITC module. All mask bits are such that a logic 1 implies that the corresponding interrupt is disabled (masked), whereas a 0 implies that the interrupt enabled (unmasked). All of these mask bits are logic 1 out of reset (all interrupts are masked).

Table 30-37. SIM Module Interrupts

Flag	Flag Register	Mask	Mask Register	Description
TC	XMT_STATUS	TCIM	INT_MASK	Transmit complete
ETC	XMT_STATUS	ETCIM	INT_MASK	Early Transmit complete
TFE	XMT_STATUS)	TFEIM	INT_MASK	Transmit FIFO Empty
XTE	XMT_STATUS	XTM	INT_MASK	Transmit threshold error
TFO	XMT_STATUS	TFOM	INT_MASK	Transmit FIFO Overfill Error
TDTF	XMT_STATUS	TDTFM	INT_MASK	Transmit Data Threshold Flag
GPCNT	XMT_STATUS	GPCNTM	INT_MASK	General Purpose Counter Comparator Flag
RDRF	RCV_STATUS	RIM	INT_MASK	Receive data register full (FIFO threshold level reached)
OEF	RCV_STATUS	OIM	INT_MASK	Overrun error flag
CWT	RCV_STATUS	CWTM	INT_MASK	Character Wait Time Counter Comparator Flag
BWT	RCV_STATUS	BWTM	INT_MASK	Block Wait Time Counter Comparator Flag
BGT	RCV_STATUS	BGTM	INT_MASK	Block Guard Time Counter Comparator Flag
RTE	RCV_STATUS	RTM	INT_MASK	Receive NACK threshold error
SDI1	PORT1_DETECT	SDIM1	PORT1_DETECT	SIM detect interrupt for port 1
SDI0	PORT0_DETECT	SDIM0	PORT0_DETECT)	SIM detect interrupt for port 0

30.5 Initialization and Application Information

This section describes the intended programming model for using the SIM module. The section describes how to begin a typical mode of operation using the registers.

30.5.1 Configuring SIM for Operation

The following is a list of items that need to be performed in order to configure the SIM module for operation:

1. Port selection in the SETUP register.
   a) Select which SIM module port is active by writing the SPS bit.
   b) Select whether the second SIM module port is active at the same time under the control of an alternate SIM module by writing the AMODE bit.

2. Enable the selected port (for port 1, see PORT1_CNTL register; for port 0, see PORT0_CNTL) following the SIM power up procedure specified in ISO 7816.
   a) Enable power to the SIM card using the SVEN1 or SVEN0 bit.
   b) Enable SIM module transmit data output using the STEN1 or STEN0 bit. This is required to allow the SIM receiver to create NACK pulses.
   c) Enable SIM card clock using the scen1 or SCEN0 bit.
   d) Release the SIM card reset using the SRST1 or SRST0 bit (assert SRSTX high).
3. Select XMT port driver type (open-drain or push-pull) in the OD_CONFIG register.
   a) Configure the selected port to be open drain or push-pull by using the OD_P1 or OD_P0 bit.
4. Choose the port Baud rate in the CNTL register.
   a) Select the SIM card clock rate by using the CLK_SEL[1:0] bits.
   b) Select the SIM card baud rate by using the BAUD_SEL[2:0] bits and SAMPLE12.

NOTE

Follow the ISO 7816 specification with regards to card clock frequencies to ensure the maximum frequency specification is not violated.

5. Select Data format type, or place SIM receiver in initial character mode.
   a) Select inverse convention or direct convention by using the IC bit in the DATA_FORMAT register, *or*
   b) Enable initial character mode by using the ICM bit in the CNTL register.

30.5.1.1 Configuring the SIM Receiver

The following is a list of items that need to be performed in order to configure the SIM receiver for operation:

1. Enable NACK capability in CNTL register.
   a) Select NACK generation on parity errors, or invalid initial character by using the ANACK bit.
   b) Select NACK generation on overrun conditions by using the ONACK bit.
2. Select the desired Receive NACK threshold and receive FIFO threshold in RCV_THRESHOLD register.
   a) Program the threshold at which the RDRF flag will be set by using the RDT[4:0] bits.
   b) Program the threshold at which the RTE flag will be set by writing to the RTH[3:0] bits. If an auto power down after RTE flag is set, is desired, then enable the SIM card auto power down by setting the SAPD0,1 bits in the port0,1_cntl register.
3. Configure the Character Wait Time Counter, the Block Wait Timer Counter and the Block Guard Time Counter.
4. Enable interrupts in the INT_MASK register.
   a) Enable the receive data register full interrupt by using the RIM bit
   b) Enable the receive threshold error interrupt by using the RTM bit.
   c) Enable the overrun condition interrupt by using the OIM bit.

d) Enable the Character Wait Time interrupt by using the CWTM bit.

30.5.1.2 Configuring the SIM Transmitter

The following is a list of items that need to be performed in order to configure the SIM transmitter for operation:

1. Select desired re-transmission threshold for NACK'ed characters in XMT_THRESHOLD register.
   a) Program the threshold at which the XTE flag will be set by using the XTH[3:0] bits.
2. Select the guard time between transmissions in the GUARD_CNTL register.
   a. Program the desired guard time between characters transmitted by the SIM module by using the GETU[7:0] bits.
3. Select the desired Transmit FIFO threshold level in the XMT_THRESHOLD register.
   a) Program the desired threshold using the TDT[3:0] bits.
4. Enable interrupts in the INT_MASK register.
   a) Enable the transmit complete interrupt by using the TCIM bit.
   b) Enable the early transmit complete interrupt by using the ETCIM bit.
   c) Enable the transmit FIFO empty interrupt by using the TFEIM bit.
   d) Enable the transmit threshold error interrupt by using the XTM bit.
   e) Enable the transmit FIFO threshold interrupt by using the TDTFM bit.
   f) Enable the transmit FIFO overfill interrupt by using the TFOM bit.

30.5.1.3 Configuring SIM General Purpose Counter

The following is a list of items that need to be performed in order to configure the SIM General Purpose Counter for operation:

1. Select desired clock source for the General Purpose Counter using the CNTL register.
   a) Use the GPCNT_CLKSEL[1:0] bits to select the desired clock source for the counter.
2. Program counter comparator using the GPCNT register.
   a) Use the GPCNT[15:0] bits to select the desired count value at which the GPCNT interrupt flag will be set.
3. Enable the selected clock source for the General Purpose Counter using either the RESET_CNTL or ENABLE registers.
   a) If the GP Counter is configured for the card clock, enable the clock by clearing the KILL_CLOCK bit in the RESET_CNTL register.
   b) If the GP Counter is configured for the receive oversample clock, enable this clock by setting the RCV_EN bit or XMT_EN bit in the ENABLE register.
   c) If the GP Counter is configured for the transmit oversample clock, enable this clock by setting the XMT_EN bit in the ENABLE register.
4. Enable interrupts in the INT_MASK register.
   a) Enable the general purpose counter interrupt by using the GPCNTM BIT

30.5.1.4 Configuring SIM to Measure WWT (Work Wait Time) for type=0 SmartCards

The following is a list of items that need to be performed in order to configure the SIM to measure work wait time. This is intended for type=0 SmartCards. Work wait time is a combination of BWT and CWT. The CWT timer is to measure the time between stop bits in the bytes received from the SmartCard. If this time is larger that the value programmed in the cwt[15:0] register, then a flag will be set and an interrupt generated. This timer can be used for both type=0 and type=1 SmartCards. The BWT timer is used to measure the time between the stop bit of the last byte transmitted from the SIM to the stop bit of the first byte sent from the SmartCard. If we want the SIM to enforce a WWT of 100 bits. Then will need to activate both the CWT and BWT, and program both of them for a value of 100 bits. When measuring WWT, the BGT timer will not be used so it will be masked (inactive) by the BGTM bit. Below is the sequence to program the SIM to send and receive data while checking WWT.

1. Program both the CWT and the BWT (low and high) registers to the WWT (work wait time) value that needs to be enforced. Set BGT register to 0 (this is the default value).
2. Activate both the CWT and BWT functions by setting the CWTEN and BWTEN bit in the CNTL register.
3. Enable the CWT and BWT interrupts by clearing the CWTM and BWTM bits in the INT_MASK register.
4. Program the data to send to the SmartCard by writing data to PORT0,1_XMT_BUFFER
5. Activate the SIM transmit and receive by setting bits XMT_EN and RCV_EN in the ENABLE register.
6. If the software has more data to send, then as the FIFO starts to get near empty, it should write more data to the PORT0,1_XMT_BUFFER. If the software does not have any more data to send then proceed to the next step.
7. When SmartCard has completed its transmission to the SIM module, disable the BWT by clearing the bit BWTEN in the CNTL register. This will prepare SIM to measure the next block wait time.
8. Disable the SIM transmitter by clearing the XMT_EN bit in the ENABLE register.
9. Program the next data to send by writing data to the PORT0,1_XMT_BUFFER.
10. Enable the BWT function by setting the BWTEN bit in the CNTL register.
11. Enable the SIM transmitter by setting the XMT_EN bit in the ENABLE register.
12. Steps 6 to 10 can be repeated for each transmission to the SmartCard.
13. If a cwt or a BWT interrupt occurs, then the software should consider that a WWT violation. The software should clear the cwt or BWT interrupt by writing a “1” to the BWT or cwt bit in the RCV_STATUS register.

30.5.1.5 Configuring SIM to measure CWT, BWT, BGT for type=1 SmartCards

The following is a list of items that need to be performed in order to configure the SIM to measure CWT, BWT, BGT. This is intended for type=1 SmartCards. The CWT timer is to measure the time between stop bits in the bytes received from the SmartCard. If this time is larger that the value programmed in the CWT[15:0] register, then a flag will be set and an interrupt generated. This timer can be used for both type=0 and type=1 SmartCards. The BWT, BGT timer is used to measure the time between the stop bit of

the last byte transmitted from the SIM to the stop bit of the first byte sent from the SmartCard. If this time is larger than the value in the both BWT registers (BWT and BWT_H), then the BWT flag will be set and an interrupt generated. If the time is less than the value in the BGT register, then the BGT flag will be set and an interrupt generated.

1. Program the CWT, BWT, BWT_H and BGT registers to the value that needs to be enforced.
2. Activate both the CWT and BWT functions by setting the CWTEN and BWTEN bit in the CNTL register.
3. Enable the CWT, BWT, BGT interrupts by clearing the CWTM, BWTM, BGTM bits in the INT_MASK register.
4. Program the data to send to the SmartCard by writing data to PORT0_XMT_BUFFER or PORT1_XMT_BUFFER
5. Activate the SIM transmit and receive by setting bits XMT_EN and RCV_EN in the ENABLE register.
6. If the software has more data to send, then as the FIFO starts to get near empty, it should write more data to the PORT0,1_XMT_BUFFER. If the software does not have any more data to send then proceed to the next step.
7. When SmartCard has completed its transmission to the SIM module, disable the BWT by clearing the bit BWTEN in the CNTL register. This will prepare SIM to measure the next block wait time.
8. Disable the SIM transmitter by clearing the XMT_EN bit in the ENABLE register.
9. Program the next data to send by writing data to the PORT0,1_XMT_BUFFER.
10. Enable the BWT function by setting the BWTEN bit in the CNTL register.
11. Enable the SIM transmitter by setting the XMT_EN bit in the ENABLE register.
12. Steps 6 to 10 can be repeated for each transmission to the SmartCard.
13. If a cwt or a BWT interrupt occurs, then the software should read the RCV_STATUS register to determine if the error was caused by a CWT, BWT, or a BGT violation. The software should clear the CWT, BWT, BGT interrupt by writing a “1” to the CWT, BWT, BGT bit in the RCV_STATUS register.

30.5.1.6 Configuring SIM Linear Redundancy Check (LRC) Block

The following is a list of items that need to be performed in order to configure the SIM Linear Redundancy Check Block for operation:

1. Enable the LRC block by using the CNTL register.
   a) Use the LRCEN bit to enable the LRC block.
   b) Use the XMT_EN_LRC_CRC bit to enable the transmission of the LRC Character after the last character in the Transmit FIFO is sent. Refer to the T=1 programming model for more details.

30.5.1.7 Configuring SIM Cyclic Redundancy Check (CRC) Block

The following is a list of items that need to be performed in order to configure the SIM Cyclic Redundancy Check Block for operation:

1. Enable the CRC block by using the CNTL register.
   a) Use the CRCEN bit to enable the CRC block.
   b) Use the XMT_EN_LRC_CRC bit to enable the transmission of the CRC Characters after the last character in the Transmit FIFO is sent. Refer to the T=1 Programming model for more details.

30.5.2 Using the SIM Receiver

Once the SIM has been properly configured (such as correct baud rate and correct data format), SIM receptions can be enabled by setting the receive enable bit, RCV_EN, in the ENABLE register. As bytes are received, they will be placed in the 32 byte deep receive data FIFO. Unread bytes can be accessed from this FIFO at any time. There is no need to disable the receiver to access the FIFO. The FIFO should only be read when the receive FIFO data flag, RFD, in the RCV_STATUS register is set. The RFD flag, which cannot create an interrupt, is high any time there is at least one unread byte in the receive FIFO. If the receive FIFO is read when RFD is low, it will simply produce the last byte read.

The correct address to read the data contained in the receive FIFO depends on which SIM port is selected. The SPS control bit in the SETUP register controls port selection. If SPS = 0, read the data from the PORT0_RCV_BUF register; if SPS = 1, read the data from the PORT1_RCV_BUF register.

The receive data register full flag, RDRF, in the RCV_STATUS register can be used to determine when the receive FIFO has reached a given threshold value. This flag will create an interrupt if the RIM bit in the INT_MASK register is clear. To control at which point RDRF is set, program the receive data threshold, RDT[4:0], in the RCV_THRESHOLD register. If the number of unread bytes in the receive FIFO is equal to or greater than the value set by RDT[4:0], RDRF will be set.

NOTE

A value of $0 in RDT[4:0] implies that there must be 32 unread bytes in the receive FIFO to trigger RDRF.

The value in RDT[4:0] can be changed at any time to alter this threshold level. The comparison between the number of unread bytes in the FIFO and the value set by RDT[4:0] is continuously updated so that any change in either will be immediately reflected in the state of RDRF. For instance, if RDT[4:0] is set to 5, and there are 3 unread bytes in the FIFO, changing RDT[4:0] to 2 will immediately cause RDRF to be set. Likewise, setting RDT[4:0] back to 5 will cause RDRF to clear. Similarly, if there are 5 unread bytes in the receive FIFO and RDT[4:0] is set to 3, RDRF will remain high until 3 reads are complete, assuming RDT[4:0] is left as a constant and no new data is received.

The standard flow for receiving bytes from the SIM card is to set RDT[4:0] to the appropriate value, wait for RDRF to cause an interrupt (RIM clear), and then read bytes out of the receive FIFO as long as RFD is high. In addition to checking RFD between every byte, it is also recommended that software check for the existence of a set OEF flag as well.

30.5.2.1 Receive Parity Errors and Parity NACK Generation

The SIM receiver checks every byte received for proper parity. The IC control bit in the DATA_FORMAT register controls whether it checks for odd parity or even parity. When checking for odd parity, the number of logic ones contained in the 9 received bits (8 data bits and 1 parity bit) should be odd. Likewise, when checking for even parity, the number of logic ones contained in the 9 received bits (8 data bits and 1 parity bit) should be even.

When a parity error is detected on a given byte, the PORTX_PE bit for that byte is set in the FIFO. The PORTX_PE flag for each byte is read out of the FIFO when the data itself is read. There is no need to attempt to clear the parity error flag in the FIFO. It is simply overwritten the next time a byte is received into that position of the FIFO. A parity error cannot cause an interrupt.

It is possible for the SIM to automatically request that the SIM card re-send a byte found to have a parity error by generating a NACK pulse on the SIM XMT pin. The SIM will generate a NACK pulse on a byte received with a parity error if the control bit ANACK in the CNTL register is set. Bytes with parity errors that cause a NACK pulse are still placed into the FIFO just as bytes that do not cause a NACK pulse are. It is up to the software to discard data bytes with parity errors.

To control NACK generation by the SIM receiver use the RTH[3:0], Receive NACK threshold, in the RCV_THRESHOLD register. This set of bits specify the number of consecutive NACKs generated by the SIM module on a received byte, before generating an RTE (receive threshold interrupt flag) in the RCV_STATUS register. The RTE flag would also force the SIM port to power down the card if the SAPD0,1 bit is set in the PORT0,1_CNTL register.

NOTE

The ANACK bit must be set in the CNTL register to enable this feature. The control bit ANACK is also used in initial character mode to enable the retransmission of initial characters in the event that an invalid initial character is received.

When a valid character has been received by the SIM, the internal counter keeping track of the number of NACKs transmitted on the current byte resets to zero. Clearing the receive threshold error (RTE) bit would also clear that counter.

When generating a NACK pulse, the SIM will generate the low pulse starting at 10.5 ETUs and lasting for 1 ETU.

30.5.2.2 Receive Frame Errors

The SIM receiver checks every byte received for a proper stop bit. A stop bit should exist during at least the first half of the 11th ETU time after the start of the character. If this is not true, a frame error is flagged. When a frame error is detected on a given byte, the PORTX_FE bit for that byte is set in the FIFO. The PORTX_FE flag for each byte is read out of the FIFO when the data itself is read. There is no need to clear the frame error flag in the FIFO. It is simply overwritten the next time a byte is received into that position of the FIFO. A frame error cannot cause an interrupt, nor can it create a NACK pulse to the receiver asking for a retransmission of the corrupted data.

30.5.2.3 Receive Overrun Errors and Overrun NACK Generation

When there already exists 32 unread bytes in the FIFO, a received character will cause the SIM receiver to flag an overrun condition. This condition will always set the overrun error flag, OEF in the RCV_STATUS register. The received byte will be discarded leaving the 32 unread bytes in the FIFO unaltered.

It is possible for the SIM to automatically request that the SIM card re-send the byte that caused the overrun condition by generating a NACK pulse on the SIM XMT pin. The SIM will generate a NACK pulse for the byte that caused the overrun condition if the control bit ONACK in the CNTL register is set. In this case, the existence of an OEF flag does not indicate the loss of data, but rather a NACK (that is, retransmission request) due to a full receive FIFO. As opposed to transmit NACK generation, there is no limit to the number of times an overrun condition will cause a NACK other than to disable ONACK itself. When generating a NACK pulse, the SIM will generate the low pulse starting at 10.5 ETUs and lasting for 1 ETU.

If the control bit ONACK is clear, the existence of a high OEF flag indicates the loss of data. The OEF flag can create an interrupt if the OIM bit in the INT_MASK register is clear.

To clear the OEF flag, software must simply write a “one” to the OEF bit position in the RCV_STATUS register. A high OEF flag has no effect on the operation of the SIM receiver other than to create an interrupt if OIM is clear.

30.5.2.4 Using Initial Character Mode and Resulting Receive Data Formats

The SIM receiver supports the detection of special characters that allow it to determine what data format is being used by the connected SIM card. When placed in initial character mode, the SIM expects to receive one of two potential values that it will use to set the data format control bit, IC, in the DATA_FORMAT register.

The two possible data formats are inverse convention and direct convention. Essentially, inverse convention differs from direct convention in that the order of the data is flipped MSB for LSB, and the data bits and parity bit are logically inverted. When receiving inverse convention data, the transformation of the data back to direct convention format is done in hardware, including the inversion of the data and parity bits.

To place the SIM into initial character mode, set the ICM bit in the CNTL register. Once a valid initial character is received, the IC bit in the DATA_FORMAT register will be set accordingly by the hardware, and the ICM bit will be cleared. Software can read the state of the IC bit to determine which mode the SIM is currently using.

The $3B (as decoded by direct convention) with parity bit high will cause direct convention to be used (IC set to logic 0), whereas a $3F (as decoded by inverse convention) with parity bit high will cause inverse convention to be used (IC set to logic 1).

When the receiver is in initial character mode, all received bytes will continue to be placed into the receive FIFO whether they be valid initial characters or not. If a valid initial character is received that causes the data format being used to change, all subsequent bytes will be decoded with that format before being placed into the FIFO, including the initial character byte itself. That is, if the IC bit is low, and the correct

initial character for setting inverse convention is received, that character and all subsequently received characters will be stored in the FIFO after having been decoded using inverse convention (for example, the initial character will be stored as $3F).

If the receiver is in initial character mode (ICM is high), and an invalid initial character is received, the SIM can be configured to automatically request that the initial character be retransmitted by setting the ANACK control bit in the CNTL register. When generating a NACK pulse, the SIM will generate the low pulse starting at 10.5 ETUs and lasting for 1 ETUs. The invalid initial character will be placed into the receive FIFO and marked with a parity error to signify that this is an invalid initial character.

30.5.2.5 Initial Character Mode Programming Notes

The usage of the initial character mode requires close attention to the programming model. There is a condition where a parity error in the initial byte for direct convention ($3B) could be decoded as what appears to be a valid initial character for inverse convention (that is, $3F). The SIM module will not recognize this as a valid initial character for inverse convention and will mark the character by setting the parity error flag. The software must look for the existence of a parity error before recognizing a character as a valid initial character.

30.5.2.6 Automatic Receiver Mode

The SIM module has an automatic receive mode that inhibits the data being transmitted by the SIM module from entering the SIM receive buffer through the feedback path of the SIM data pin. The SIM module receiver should normally be enabled while the transmitter is operational. Automatic receive mode saves the software from having to actively manage the transition from transmitter to receiver. The auto receive mode is always active whenever the receiver is enabled.

30.5.2.7 Using the SIM Receiver with “T=1” SIM Cards

The SIM module provides hardware support for “T=1” type SIM cards. These type of cards present several requirements above and beyond the standard “T=0” cards. The features provided to meet the requirements that pertain to the SIM receiver are as follows:

- 11 ETU Characters
  - — “T=1” cards can transmit with character lengths of 11 ETUs (that is, one STOP bit). The SIM module provides the RCVR11 bit in the GUARD_CNTL register in order to configure the receiver state machine to accept 11 ETU characters.
- Character Wait Time Counter
  - — The character waiting time (CWT) is defined as the time between the start bits of two consecutive characters received from the SmartCard. The value of CWT can range from 12 ETU to 32779 ETU. The SIM module provides a 16-bit counter with programmable comparator clocked at the ETU bit rate to identify when the CWT has been exceeded by the SIM card.
- Block Waiting Time
  - — The block waiting time (BWT) is defined as the time between the start bits of the last character of a received block and the first character of the next received block. The block wait time must

not exceed a value that is programmable. The SIM module provides a 32-bit Block Wait Timer that can be used to identify when the BWT has been violated (divided in two registers).

- Block Guard Time
  - The block waiting time (BGT) is defined as the time between the start bits of the last character of a received block and the first character of the next received block. The block guard must be greater than a value that is programmable. The SIM module provides a 16-bit Block Guard Timer that can be used to identify when the BGT has been violated.
- Error Detection Code
  - "T=1" cards can specify LRC or CRC error detection codes to be used. The SIM module provides hardware support for both the LRC and CRC operations.

30.5.3 Using the SIM Transmitter

Once the SIM has been properly configured (such as data XMT enabled, correct baud rate, and correct data format), the transmitter can be enabled by setting the XMT_EN bit in the ENABLE register. If data had been previously written to the transmit FIFO, the transmitter will begin to send the first character. If no data is written to the transmit FIFO before enabling the transmitter, then the transmitter will wait until the first character is written before beginning transmission. Clearing the XMT_EN bit while the transmitter is in operation, will halt any transmission in progress, flush the transmit FIFO, and reset the transmit state machine.

Data can be written to the transmit FIFO at any time.

The transmit data threshold flag, TDTF, in the XMT_STATUS register can be used to determine when the transmit FIFO has reached a given threshold value. This flag will create an interrupt if the TDTFM bit in the INT_MASK register is clear. To control at which point TDTF is set, program the transmit data threshold, TDT[3:0], in the XMT_THRESHOLD register. If the number of bytes remaining in the transmit FIFO is equal to or less than the value set by TDT[4:0], TDTF will be set.

NOTE

A value of $0 in TDT[3:0] implies that the transmit FIFO must be empty to trigger TDTF.

The value in TDT[3:0] can be changed at any time to alter this threshold level. The comparison between the number of remaining bytes in the transmit FIFO and the value set by TDT[3:0] is continuously updated so that any change in either will be immediately reflected in the state of TDTF. Unlike the RDRF flag for the receive FIFO, TDTF is latched an remains set until the software writes a one to the TDTF bit location in the XMT_STATUS register. For instance, if TDT[3:0] is set to 5, and there are 6 bytes remaining in the FIFO, changing TDT[3:0] to 6 will immediately cause TDTF to be set. However, setting TDT[3:0] back to 5 will not cause TDTF to clear.

The standard flow for transmitting bytes from the SIM card is to set TDT[3:0] to the appropriate value, write up to 16 bytes to the transmit FIFO, enable the transmitter, wait for TDTF to cause an interrupt (TDTFM clear), and then write additional bytes to the transmit FIFO.

NOTE

For the SIM module to transmit successfully, the transmit pin must be connected to the receive pin of the same port. This connection is necessary for the SIM to decode transmit NACKs sent to it by the SIM card. Without this connection, all transmissions will appear to have a NACK thereby causing the first byte to be transmitted continuously. When operating in 3VOLT mode (3VOLT0 or 3VOLT1), this connection is made internal to the SoC.

30.5.3.1 Transmit Data Formats

There are two possible data formats that the SIM module will use when transferring data to the SIM card-inverse convention or direct convention. The format used depends on the state of inverse convention bit (IC in the DATA_FORMAT register). Software can set the IC bit, or it can be set automatically by hardware when using the initial character mode.

30.5.3.2 Transmit NACK

The SIM transmitter can respond to NACKs created by the SIM card. A NACK will be decoded if the SIM card creates a logic low level on the SIM receive pin during the STOP bit time at the end of a transmitted byte. To prevent a situation where the SIM interface is stalled by an infinite number of NACK pulses on a given byte, the SIM module can be configured to limit the number of times it will respond to NACKs. The XTH[3:0] control field in the XMT_THRESHOLD register allows software to set a threshold on the number of times a given byte will be retransmitted. If the threshold is reached, a transmit threshold error, XTE in the XMT_STATUS register, is asserted.

When XTE is set, the SIM transmitter is halted, and all pending transfers are aborted, and the TC, ETC, AND TFE flags are set. All bytes remaining in the transmit FIFO will be lost. There is no way to restart the transmission on the next byte in the FIFO. The transmitter will remain frozen until XTE is cleared by software. The only way to clear XTE is to write a 1 to the XTE bit in the XMT_THRESHOLD register. The XTE flag can create an interrupt if the XTM mask in the XMT_THRESHOLD register is clear.

It is possible to disable the detection of NACKs from the SIM card by setting XTH[3:0] to $0. By setting XTH[3:0] to $1, it is possible to disable all retransmissions while still setting XTE on the first NACK received. In general, XTE is set on the NACK that causes the threshold set by XTH[3:0] to be reached. This final NACK will not cause a retransmission, whereas all previous NACKs will cause a retransmission.

30.5.3.3 Transmit Guard Time

The time between data bytes sent from the SIM transmitter can be altered using the transmit guard time control. By default, the minimum time between start bits of successive transmitted bytes is 12 ETUs (Elementary Time Units). An ETU is equivalent in time to 1 bit time. Therefore, the amount of time an ETU consumes is determined by the baud rate chosen. The 12 ETUs that form the default character transmission time consist of a start bit, 8 data bits, a parity bit and two stop bits. The number of stop bits (idle bits) can be extended by an integer number of ETUs. The number of additional ETUs can be

programmed directly into the GETU[7:0] bits of the GUARD_CNTL register. Programming the getu[7:0] bits to $FF will configure the SIM transmitter to use only one stop bit for each character transmission.

30.5.3.4 Using SIM Transmit with “T=1” SIM Cards

The SIM module provides hardware support for “T=1” type SIM cards. These type of cards present several requirements above and beyond the standard “T=0” cards. The features provided to meet the requirements that pertain to the SIM transmitter are as follows:

- 11 ETU Characters
  - The SIM module transmitter has a programmable guard time register that allows the programmer to specify the number of ETUs between character transmissions. Programming a value of 255 ($FF) in the GETU[7:0] bits in the GUARD_CNTL register will set the number of ETUs per character transmitted to 11.
- Character Waiting Time
  - The character waiting time (CWT) is defined as the time between the start bits of two consecutive characters. The value of CWT can range from 12 ETU to 32779 ETU. The time between transmitted characters is controlled by the programmable guard time in the GUARD_CNTL register. However, the time between the last byte in the transmit FIFO, and the next transmitted byte can be largely affected by software response time to the transmit interrupts. The SIM transmitter provides a Transmit FIFO threshold (TDTF) interrupt to signal the system when the expected number of characters have been transmitted from the transmit FIFO. The minimum CWT is achieved only if the software can respond to the TDTF interrupt and write new data to the transmit FIFO before the last character in the Transmit FIFO has been sent.
- Block Waiting Time
  - The block waiting time (BWT) is defined as the time between the start bits of the last character of a received block and the first character of the next received block. The value of BWT is always greater than 1800 ETU. The SIM transmitter provides a General Purpose Counter that can be used to track the BWT. The BWT is purely determined by software response time to the transmit interrupts.
- Block Guard Time
  - The block guard time (BGT) is defined as the minimum delay between the leading edges of two consecutive characters sent in opposite directions. The value of BGT is 22 ETU. The SIM module supports the BGT by providing the ability to generate an interrupt when the last byte is received, and transmitting within 2 ETU after the XMT_EN bit is set. The BGT will be determined by the speed at which the software can react to an interrupt and enable the transmitter.
- Error Detection Code
  - “T=1” cards can specify LRC or CRC error detection codes to be used. The SIM module provides hardware support for both the LRC and CRC operation.

30.5.4 Suggested T=1 Compliant Programming Model

This section will describe the suggested programming model for supporting “T=1”, “T=0”, and known “special” cards using the SIM module on the SoC. This should be used as a rough guide for how to configure the SIM module for use with SIM cards specified by ISO 7816-3 and EMV. Some details are not addressed. Other uses for some of the SIM features are not included (for example, GP Counter uses for some ISO timing requirements).

30.5.4.1 Answer To Reset (ATR) Detection

The first step to communicating with a SIM card is to provide power and a clock signal to the card. Once the card is detected as present (using the presence detect features, or some other method), the SIM card should be powered up according to the power up sequence specified in the ISO 7816-3 specification.

1. Apply voltage to the SIM card by setting the SVEN0 or SVEN1 bit in the PORTx_CNTL register.
2. Select the appropriate clock frequency for the SIM card by programming the CLK_SEL[1:0] bits in the CNTL register.
3. Enable the clock to the SIM card by setting the SCEN0 or SCEN1 bit in the PORTx_CNTL register.
4. Remove the card from reset by setting the SRST0 or SRST1 bit in the PORTx_CNTL register.

The first communication between the SIM card and the SIM module will be a block of data sent from the SIM card to the SIM module after the card is powered and the card reset is removed. This block is call the Answer To Reset (ATR). In order to receive the ATR, the SIM module should be configured for 12 ETU character reception. According to the ISO 7816-3 specification, both “T=0” and “T=1” cards will communicate initially using 12 ETU character durations.

NOTE

We are aware of some card manufacturers that communicate at 11.5 ETU character durations (Geldkarte). This will complicate the initial card detection sequence shown below.

5. Clear RCVR11 bit in the GUARD_CNTL register

The next item to configure for ATR reception is the NACK capability of the SIM module. The ISO 7816-3 specification allows the SIM module to NACK any communication errors that occur during the initial communication at 12 ETU.

NOTE

The Europay Mastercard and VISA (EMV) cards are similar to “T=1”, but do not allow the SIM module to NACK during the initial communication. This will again complicate the initial card detection sequence shown below.

6. Set ANACK bit in the CNTL register to enable NACK generation.

In order for the SIM module to notify when characters are received, the receive interrupt(s) can be enabled. A threshold can be set for the number of characters to receive before generating an interrupt.

7. Enable RDRF and OEF interrupts by setting RIM and OIM bits in INT_MASK register.

8. Set desired threshold for received characters by writing RDT[4:0] in RCV_THRESHOLD register.

The SIM should be setup to perform in initial character mode. This will cause the hardware to identify the first valid character sent during the ATR as an initial character. This character will automatically configure the hardware for the data convention used by the SIM card.

9. Set Initial Character Mode by setting the ICM bit in the CNTL register

The ISO 7816-3 specification requires that SIM cards meet certain timing restrictions. One of these is the time from the deassertion of the card reset to the beginning of the ATR sequence. The SIM module General Purpose Counter can be used to verify that the SIM card begins its ATR within the 400 to 40,000 clock cycle range.

a) Set General Purpose Counter Comparator to $9C40 using GPCNT register.
b) Enable General Purpose Counter Interrupt by clearing GPCNTM in INT_MASK register.
c) Enable General Purpose Counter by programming the GPCNT_CLKSEL[1:0] bits to 01 so the card clock is used for counting.

The ISO 7816-3 specification states that the maximum allowed time between two characters during the ATR is 9600 ETUs (Initial Waiting Time). The Character Wait Time Counter should be setup to detect any errors for this condition.

a) Set Character Wait Time Counter Comparator to 9600 using the CHAR_WAIT register.
b) Enable the Character Wait Time Counter Interrupt by clearing CWTM in INT_MASK register.
c) Enable the Character Wait Time Counter by setting the CWTEN bit in the CNTL register.

The last step in preparing for ATR reception is to enable the receiver.

10. Set RCV_EN bit in ENABLE register.

The SIM module will generate interrupts once a threshold number of characters is received. The software should react to these interrupts and read the characters from the receive FIFO (PORTx_RCV_BUF) until the complete ATR has been received. If a General Purpose Counter interrupt occurs before the final ATR character is received, then the card should be deactivated according to the ISO 7816-3 specification. Otherwise, once a valid ATR is received, the software will know from the ATR information the specific characteristics for this card (refer to the ISO 7816-3 specification for details).

30.5.4.2 Programming Considerations for Geldkarte Cards

There is at least one manufacturer of SIM cards (Geldkarte) that we know of that does not send the ATR in 12 ETU mode or support NACKs. This creates an issue with detecting a valid ATR from a SIM card. Since any kind of card can be attached to the SIM module, the software and hardware do not know how to begin communication. Basically, if the card fails to send a valid ATR on the first try, disable NACKs, configure the SIM module for 11 ETU and try again. The software should toggle between 12 and 11 ETU modes with and without NACKs enabled until a valid ATR is received, or the number of attempts to communicate passes a predetermined error threshold. Figure 30-49 shows the flowchart for the suggested Geldkarte Compliant SIM Initialization.

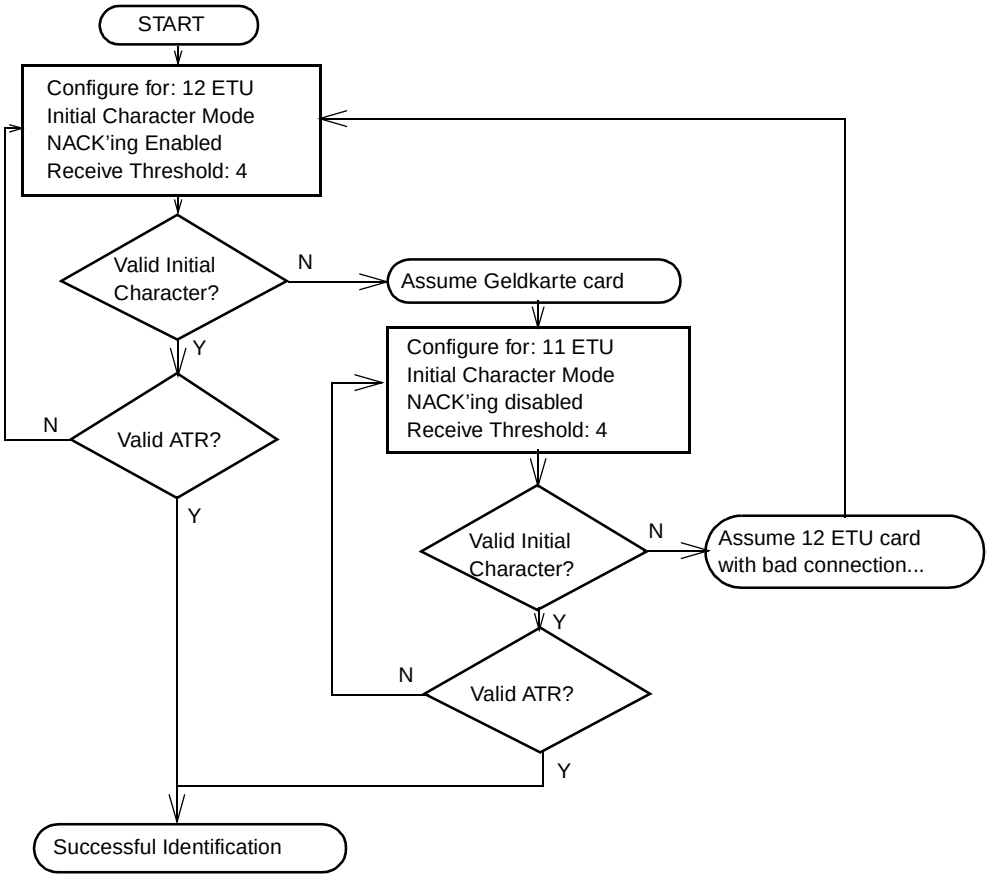


Figure 30-49. Suggested T=1, EMV, Geldkarte Compliant SIM Initialization

30.5.4.3 Programming Considerations for T=0 SIM Cards

If the card is of type T=0, the software should adjust the following parameters according to the information in the ATR:

1. Adjust the baud rate by changing the values of BAUD_SEL[2:0] and SAMPLE12 in the CNTL register.
2. Adjust the guard time between characters by changing the value of GETU[7:0] in the GUARD_CNTL register.
3. Adjust NACK capability by modifying the values of the ONACK and ANACK bits in the CNTL register.
4. Adjust the stop clock polarity by modifying the values of the SCSP0 or SCSP1 bits in the PORTx_CNTL register.
5. Adjust the level of transmit NACK re-transmissions allowed by modifying the value of the XTH[3:0] bits in the XMT_THRESHOLD register.
6. Adjust the level for the Receive NACK threshold by modifying the RTH[3:0] bits in the RCV_THRESHOLD register.

If a negotiation with the SIM card is desired, the software will send a PPS response to the SIM card. In order to send the response, the following steps should be performed:

1. Set the desired transmit FIFO threshold level by writing the TDT[3:0] bits in the XMT_THRESHOLD register.
2. Write the characters to be sent as response (max 16) to the transmit FIFO using the PORTx_XMT_BUF register.
3. Clear all transmit interrupt flags in the XMT_STAT register by writing a one to them.
4. Enable the transmit interrupts desired by clearing the mask bits in the INT_MASK register. If more than 16 character are to be sent, it is suggested that the TDTF interrupt be used to signify when to write more characters to the transmit FIFO. This results in the most efficient transfer times to the SIM card.
5. Enable the transmitter by setting the XMT_EN bit in the ENABLE register.

At this point, the SIM module will transmit the characters in the transmit FIFO. If more than 16 characters are to be sent, the transmit threshold interrupt will be set when the threshold number of characters are remaining in the FIFO. The software can then write an additional number of characters to be sent without interrupting transmission to the SIM card.

Once the transmission is complete, the SIM module should be completely configured for standard operation with the T=0 SIM card. The software can continue to service RDRF interrupts for received characters, and TDTF interrupts for transmitted characters.

30.5.4.4 Programming Considerations for T=1 SIM Cards

If the card is of type T=1, the software should adjust the following parameters according to the information in the ATR:

Adjust the baud rate by changing the values of BAUD_SEL[2:0] and SAMPLE12 in the CNTL register.

1. Adjust the guard time between characters by changing the value of GETU[7:0] in the GUARD_CNTL register. Setting GETU[7:0] to $FF configures the SIM transmitter for 11 ETU transmissions.
2. Disable NACK capability by clearing the ONACK and ANACK bits in the CNTL register. T=1 cards do not allow NACKs.
3. Adjust the stop clock polarity by modifying the values of the SCSP0 or SCSP1 bits in the PORTx_CNTL register.
4. Set Character Wait Time Counter Comparator to value specified in the ATR by using the CHAR_WAIT register.
5. Enable the Character Wait Time Counter Interrupt by clearing CWTM in INT_MASK register.
6. Enable the Character Wait Time Counter by setting the CWTEN bit in the CNTL register.
7. Enable CRC or LRC error checking according to the ATR information by setting either the CRCEN or LRCEN bit in the CNTL register. These bits will never be set at the same time!

For T=1 cards, the ATR is sent using a T=0 type of structure (12 ETU, no LRC or CRC). If a negotiation with the SIM card is desired, the software will send a PPS response to the SIM card. Otherwise, the

protocol is initiated with a block transfer from the SIM module. In order to send the response or the first block, the following steps should be performed:

1. Set the desired transmit FIFO threshold level by writing the TDT[3:0] bits in the XMT_THRESHOLD register.
2. Write the characters to be sent as response (max 16) to the transmit FIFO using the PORTx_XMT_BUF register.
3. Clear all transmit interrupt flags in the XMT_STAT register by writing a one to them.
4. Enable the transmit interrupts desired by clearing the mask bits in the INT_MASK register. If more than 16 character are to be sent, it is suggested that the TDTF interrupt be used to signify when to write more characters to the transmit FIFO. This results in the most efficient transfer times to the SIM card.
5. Enable the transmission of the error checking characters (LRC or CRC) by setting the XMT_EN_LRC_CRC bit in the CNTL register.

NOTE

If the card supports PPS, the software may not be allowed to send the LRC/CRC information until the PPS exchange is completed. If so, do not set the XMT_EN_LRC_CRC bit during the PPS exchange.

6. Enable the transmitter by setting the XMT_EN bit in the ENABLE register

At this point, the SIM module will transmit the characters in the transmit FIFO. If more than 16 characters are to be sent, the transmit threshold interrupt will be set when the threshold number of characters are remaining in the FIFO. The software can then write an additional number of characters to be sent without interrupting transmission to the SIM card.

Once the transmission is complete, the SIM module should be completely configured for standard operation with the T=1 SIM card. The software can continue to service RDRF interrupts for received characters, and TDTF interrupts for transmitted characters.

Chapter 31
Universal Asynchronous Receiver/Transmitter (UART)

The i.MX31 and i.MX31L contain five UART modules. The Universal Asynchronous Receiver/Transmitter (UART) module is capable of standard RS-232 non-return-to-zero (NRZ) encoding format and IrDA-compatible infrared modes. The UART provides serial communication capability with external devices through an RS-232 cable or through use of an external circuitry that converts infrared signals to electrical signals (for reception) or transforms electrical signals to signals that drive an infrared LED (for transmission) to provide low-speed IrDA compatibility.

The UART transmits and receives characters containing either 7 or 8 bits (program selectable). To transmit, data is written by way of the IP data bus (SkyBlue line interface) to a 32-byte transmitter FIFO (TxFIFO). This data is passed to the shift register and shifted serially out on the transmitter pin (TXD). To receive, data is received serially from the receiver pin (RXD) and stored in a 32-halfwords-deep receiver FIFO (RxFIFO). The received data is retrieved from the RxFIFO by way of the IP data bus. The RxFIFO and TxFIFO generate maskable interrupts as well as DMA requests when the data level in one of the FIFO reaches a programmed threshold level.

The UART generates baud rates based on a dedicated input clock and its programmable divisor. The UART also contains programmable auto baud detection circuitry to receive 1 or 2 stop bits as well as odd, even, or no parity. The receiver detects framing errors, idle conditions, BREAK characters, parity errors, and overrun errors.

The module has a serial infrared capability allowing it to encode and decode standard serial IrDA data.

Figure 31-1 shows the UART block diagram.

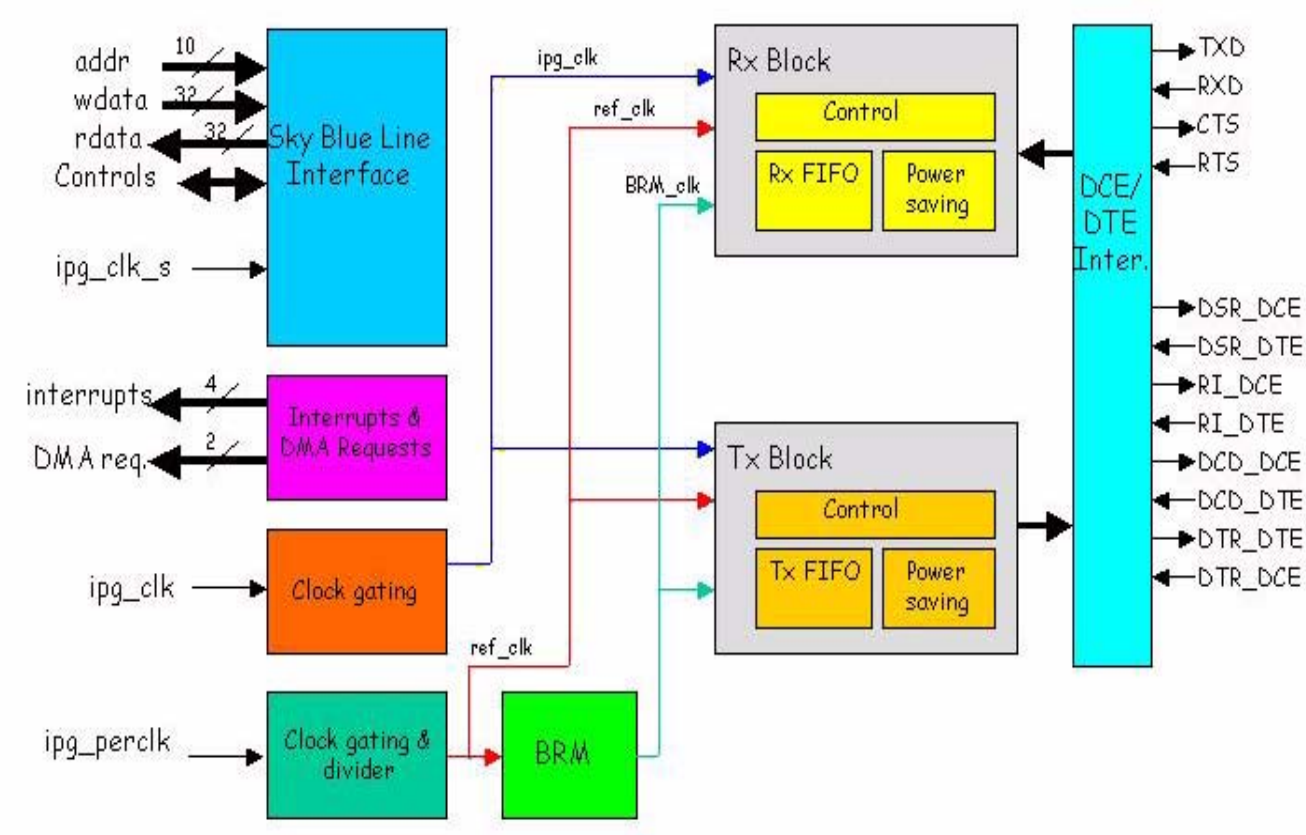


Figure 31-1. UART Block Diagram

31.1 Features

The UART includes the following features:

- High-speed TIA/EIA-232-F compatible, up to 1.875 Mbit/s
- 7 or 8 data bits
- 1 or 2 stop bits
- Programmable parity (even, odd, and no parity)
- Hardware flow control support for request to send ($\overline{\text{RTS}}$) and clear to send ($\overline{\text{CTS}}$) signals
- Edge-selectable $\overline{\text{RTS}}$ and edge-detect interrupts
- Status flags for various flow control and FIFO states
- Serial IR interface low-speed, IrDA-compatible (up to 115.2 Kbit/s).
- Voting logic for improved noise immunity (16x oversampling)
- Transmitter FIFO empty interrupt suppression
- UART internal clocks enable/disable
- Auto baud rate detection (up to 115.2 Kbit/s)
- Receiver and transmitter enable/disable for power saving
- DCE/DTE capability

- $\overline{\text{RTS}}$, IrDA asynchronous wake (AIRINT), receive asynchronous wake (AWAKE), $\overline{\text{RI}}$ (DTE only), $\overline{\text{DCD}}$ (DTE only), $\overline{\text{DTR}}$ (DCE only) and $\overline{\text{DSR}}$ (DTE only) interrupts wake the processor from STOP mode
- Maskable interrupts
- Two DMA requests (TxFIFO DMA request and RxFIFO DMA request)
- Escape character sequence detection
- Software reset ($\overline{\text{SRST}}$)
- Two independent, 32-entry FIFOs for transmit and receive
- Dedicated BRM clock (*ipg_perclk*) to allow frequency scaling on main clock (*ipg_clk*) without reprogramming BRM registers

31.1.1 Modes of Operation

The following are two modes of operation for the UART:

- Serial RS-232 NRZ format
- IrDA

31.2 External Signal Description

31.2.1 Overview

See Table 31-1 for the list of interface signals.

Table 31-1. Interface Signals

Interface Signals				
Interrupts				
IPI_UART_RX	O	Low	Receiver interrupt	High
IPI_UART_TX	O	Low	Transmitter interrupt	High
Serial/IrDA Signals				
IPP_UART_RXD	I		Serial data receive	
IPP_UART_TXD	O		Serial data transmit	High
Modem Control Signals				
IPP_UART_CTS	O	Low	Clear to send	High
IPP_UART_RTS	I	Low	Request to send	
IPP_UART_DSR_DTE_I	I	Low	Data set ready (DTE)	
IPP_UART_DSR_DCE_O	O	Low	Data set ready (DCE)	High
IPP_UART_DCD_DTE_I	I	Low	Data carrier detected (DTE)	
IPP_UART_DCD_DCE_O	O	Low	Data carrier detected (DCE)	High

Table 31-1. Interface Signals (continued)

Interface Signals				
Interrupts				
IPP_UART_DTR_DCE_I	I	Low	Data terminal ready (DCE)	
IPP_UART_DTR_DTE_O	O	Low	Data terminal ready (DTE)	High
IPP_UART_RI_DTE_I	I	Low	Ring indicator (DTE)	
IPP_UART_RI_DCE_O	O	Low	Ring indicator (DCE)	High

31.2.2 Detailed Signal Descriptions

31.2.2.1 Interrupt Signals

31.2.2.1.1 $\overline{\text{IPI\_UART\_RX}}$—Receiver Interrupt

Output interrupt signal for receive interface.

31.2.2.1.2 $\overline{\text{IPI\_UART\_TX}}$—Transmitter Interrupt

Output interrupt signal for transmit interface.

31.2.2.2 Serial/IrDA Signals

31.2.2.2.1 IPP_UART_RXD—Serial Data Receive

Input asynchronous data receive in serial mode. This signal is left unconnected if *ipp_uart_txd_mux* is used.

31.2.2.2.2 IPP_UART_TXD—Serial Data Transmit

Output asynchronous data transmit in serial mode. This signal is left unconnected if *ipp_uart_txd_mux* is used.

31.2.2.3 Modem Control Signals

31.2.2.3.1 $\overline{\text{IPP\_UART\_CTS}}$—Clear To Send

Output in DCE and DTE mode. This signal informs the remote modem that UART is ready to receive data.

31.2.2.3.2 $\overline{\text{IPP\_UART\_RTS}}$—Request To Send

Input in DCE and DTE mode. This signal informs UART that the remote modem is ready to receive data.

31.2.2.3.3 $\overline{\text{IPP\_UART\_DSR\_DTE\_I}}$—Data Set Ready (DTE Mode)

Input in DTE mode. Indicates to UART that the remote modem is operational.

31.2.2.3.4 IPP_UART_DSR_DCE_O—Data Set Ready (DCE Mode)

Output in DCE mode. Indicates to the remote modem that UART is operational.

31.2.2.3.5 IPP_UART_DCD_DTE_I—Data Carrier Detected (DTE Mode)

Input in DTE mode. Indicates to UART that a good carrier is being received from the remote modem (in DCE mode).

31.2.2.3.6 IPP_UART_DCD_DCE_O—Data Carrier Detected (DCE Mode)

Output in DCE mode. Indicates to remote device that a good carrier is being received from the UART (in DCE mode).

31.2.2.3.7 IPP_UART_DTR_DCE_I—Data Terminal Ready (DCE Mode)

Input in DCE mode. Indicates to UART (in DCE mode) that the remote device (in DTE mode) is operational.

31.2.2.3.8 IPP_UART_DTR_DTE_O—Data Terminal Ready (DTE Mode)

Output in DTE mode. Indicates to the remote modem (in DCE mode) that UART (in DTE mode) is operational.

31.2.2.3.9 IPP_UART_RI_DTE_I—Ring Indicator (DTE Mode)

Input in DTE mode. Indicates to UART that the remote modem is detecting a ringing tone.

31.2.2.3.10 IPP_UART_RI_DCE_O—Ring Indicator (DCE Mode)

Output in DCE mode. Indicates to the remote device that UART is detecting a ringing tone.

31.3 Memory Map and Register Definition

Section 31.3.3, “Register Descriptions” provides the detailed descriptions for all of the UART registers.

31.3.1 UART Memory Map

Table 31-2 shows the UART memory map.

Table 31-2. UART Memory Map

Address	Register	Access	Reset Value	Section/Page
0x43F9_0000 (URXD1) 0x43F9_4000 (URXD2) 0x5000_C000 (URXD3)) 0x43FB_0000 (URXD4) 0x43FB_4000 (URXD5)	UART Receiver Register	R	0x0000_40– –	31.3.3.1/31-12
0x43F9_0040 (UTXD1) 0x43F9_4040 (UTXD2) 0x5000_C040 (UTXD3)) 0x43FB_0040 (UTXD4) 0x43FB_4040 (UTXD5)	UART Transmitter Register	W	0x0000_00– –	31.3.3.2/31-14
0x43F9_0080 (UCR1_1) 0x43F9_4080 (UCR1_2) 0x5000_C080 (UCR1_3) 0x43FB_0080 (UCR1_4) 0x43FB_4080 (UCR1_5)	UART Control Register 1	R/W	0x0000_0000	31.3.3.3/31-15
0x43F9_0084 (UCR2_1) 0x43F9_4084 (UCR2_2) 0x5000_C084 (UCR2_3) 0x43FB_0084 (UCR2_4) 0x43FB_4084 (UCR2_5)	UART Control Register 2	R/W	0x0000_0001	31.3.3.4/31-17
0x43F9_0088 (UCR3_1) 0x43F9_4088 (UCR3_2) 0x5000_C088 (UCR3_3) 0x43FB_0088 (UCR3_4) 0x43FB_4088 (UCR3_5)	UART Control Register 3	R/W	0x0000_0700	31.3.3.5/31-19
0x43F9_008C (UCR4_1) 0x43F9_408C (UCR4_2) 0x5000_C08C (UCR4_3) 0x43FB_008C (UCR4_4) 0x43FB_408C (UCR4_5)	UART Control Register 4	R/W	0x0000_8000	31.3.3.6/31-22
0x43F9_0090 (UFCR1) 0x43F9_4090 (UFCR2) 0x5000_C090 (UFCR3) 0x43FB_0090 (UFCR4) 0x43FB_4090 (UFCR5)	UART FIFO Control Register	R/W	0x0000_0801	31.3.3.7/31-23
0x43F9_0094 (USR1_1) 0x43F9_4094 (USR1_2) 0x5000_C094 (USR1_3) 0x43FB_0094 (USR1_4) 0x43FB_4094 (USR1_5)	UART Status Register 1	R/W	0x0000_2040	31.3.3.8/31-25
0x43F9_0098 (USR2_1) 0x43F9_4098 (USR2_2) 0x5000_C098 (USR2_3) 0x43FB_0098 (USR2_4) 0x43FB_4098 (USR2_5)	UART Status Register 2	R/W	0x0000_4028	31.3.3.9/31-27

Table 31-2. UART Memory Map (continued)

Address	Register	Access	Reset Value	Section/Page
0x43F9_009C (UESC1) 0x43F9_409C (UESC2) 0x5000_C09C (UESC3) 0x43FB_009C (UESC4) 0x43FB_409C (UESC5)	UART Escape Character Register	R/W	0x0000_002B	31.3.3.10/31-29
0x43F9_00A0 (UTIM1) 0x43F9_40A0 (UTIM2) 0x5000_C0A0 (UTIM3) 0x43FB_00A0 (UTIM4) 0x43FB_40A0 (UTIM5)	UART Escape Timer Register	R/W	0x0000_0000	31.3.3.11/31-30
0x43F9_00A4 (UBIR1) 0x43F9_40A4 (UBIR2) 0x5000_C0A4 (UBIR3) 0x43FB_00A4 (UBIR4) 0x43FB_40A4 (UBIR5)	UART BRM Incremental Register	R/W	0x0000_0000	31.3.3.12/31-31
0x43F9_00A8 (UBMR1) 0x43F9_40A8 (UBMR2) 0x5000_C0A8 (UBMR3) 0x43FB_00A8 (UBMR4) 0x43FB_40A8 (UBMR5)	UART BRM Modulator Register	R/W	0x0000_0000	31.3.3.13/31-32
0x43F9_00AC (UBRC1) 0x43F9_4 0AC (UBRC2) 0x5000_C 0AC (UBRC3) 0x43FB_0 0AC (UBRC4) 0x43FB_4 0AC (UBRC5)	UART Baud Rate Count Register	R	0x0000_0004	31.3.3.14/31-33
0x43F9_00B0 (ONEMS1) 0x43F9_4 0B0 (ONEMS2) 0x5000_C 0B0 (ONEMS3) 0x43FB_0 0B0 (ONEMS4) 0x43FB_4 0B0 (ONEMS5)	UART One Millisecond Register	R/W	0x0000_0000	31.3.3.15/31-34
0x43F9_00B4 (UTS1) 0x43F9_40B4 (UTS2) 0x5000_C0B4 (UTS3) 0x43FB_00B4 (UTS4) 0x43FB_40B4 (UTS5)	UART Test Register	R/W	0x0000_0060	31.3.3.16/31-35

31.3.2 Register Summary

The UART supports 8-bit and 16-bit accesses to 32-bit memory-mapped addresses only. Any access to unmapped memory location results in a transfer error (assuming that resp_sel primary input is tied to 1’b0).

All memory mapped registers are 32 bits wide; the 16 MSB are not used:

- For 32-bits write access, the 16 MSB are not taken into account.
- For 32-bits read access, the 16 MSB are read as 0.

All registers described in this section are for 16 LSB.

Figure 31-2 shows the key to the register fields, and Table 31-3 shows the register figure conventions.

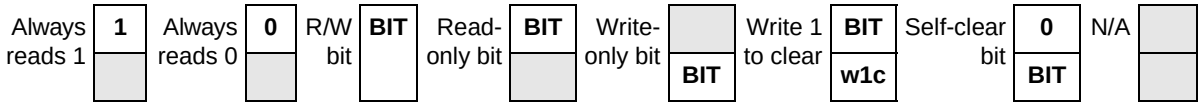


Figure 31-2. Key to Register Fields

Table 31-3. Register Figure Convention

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 31-4 shows the UART register summary.

Table 31-4. UART Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x43F9_0000 (URXD1) 0x43F9_4000 (URXD2) 0x5000_C000 (URXD3)) 0x43FB_0000 (URXD4) 0x43FB_4000 (URXD5)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	CHARRDY	ERR	OVRRUN	FRMERR	BRK	PRERR	0	0	RX_DATA							
	W																

Table 31-4. UART Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x43F9_0040 (UTXD1) 0x43F9_4040 (UTXD2) 0x5000_C040 (UTXD3) 0x43FB_0040 (UTXD4) 0x43FB_4040 (UTXD5)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0								
	W	TX_DATA															
0x43F9_0080 (UCR1_1) 0x43F9_4080 (UCR1_2) 0x5000_C080 (UCR1_3) 0x43FB_0080 (UCR1_4) 0x43FB_4080 (UCR1_5)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	ADEN	ADBR	TRDYEN	IDEN	ICD		RRDYEN	RXDMAEN	IREN	TXMPTYEN	RTSDEN	SNDBRK	TXDMAEN	ATDMAEN	DOZE	UARTEN
	W																
0x43F9_0084 (UCR2_1) 0x43F9_4084 (UCR2_2) 0x5000_C084 (UCR2_3) 0x43FB_0084 (UCR2_4) 0x43FB_4084 (UCR2_5)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	ESCI	IRTS	CTSC	CTS	ESCEN	RTEC		PREN	PROE	STPB	WS	RTSEN	ATEN	TXEN	RXEN	SRST
	W																
0x43F9_0088 (UCR3_1) 0x43F9_4088 (UCR3_2) 0x5000_C088 (UCR3_3) 0x43FB_0088 (UCR3_4) 0x43FB_4088 (UCR3_5)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	DPEC		DTREN	PARERREN	FRAERREN	DSR	DCD	RI	ADNIMP	RXDSEN	AIRINTEN	AWAKEN	DTRDEN	RXDMUXSEL	INVT	ACIEN
	W																

Table 31-4. UART Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x43F9_008C (UCR4_1) 0x43F9_408C (UCR4_2) 0x5000_C08C (UCR4_3) 0x43FB_008C (UCR4_4) 0x43FB_408C (UCR4_5)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	CTSTL						INVR	ENIRI	WKEN	IDDMAEN	IRSC	LPBYP	BKEN	BKEN	OREN	DREN
	W																
0x43F9_0090 (UFCR1) 0x43F9_4090 (UFCR2) 0x5000_C090 (UFCR3) 0x43FB_0090 (UFCR4) 0x43FB_4090 (UFCR5)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	TXTL						RFDIV			DCEDTE	RXTL					
	W																
0x43F9_0094 (USR1_1) 0x43F9_4094 (USR1_2) 0x5000_C094 (USR1_3) 0x43FB_0094 (USR1_4) 0x43FB_4094 (USR1_5)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	PARITYERR	RTSS	TRDY	RTSD	ESCF	FRAMERR	RRDY	AGTIM	DTRD	RXDS	AIRINT	AWAKE	0	0	0	0
	W	w1c			w1c	w1c	w1c		w1c	w1c		w1c	w1c				
0x43F9_0098 (USR2_1) 0x43F9_4098 (USR2_2) 0x5000_C098 (USR2_3) 0x43FB_0098 (USR2_4) 0x43FB_4098 (USR2_5)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	ADET	TXFE	DTRF	IDLE	ACST	RIDELT	RIIN	IRINT	WAKE	DCDDELT	DCDIN	RTSF	TXDC	BRCD	ORE	RDR
	W	w1c			w1c		w1c			w1c			w1c		w1c	w1c	

Table 31-4. UART Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x43F9_009C (UESC1) 0x43F9_409C (UESC2) 0x5000_C09C (UESC3) 0x43FB_009C (UESC4) 0x43FB_409C (UESC5)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	ESC_CHAR							
	W																
0x43F9_00A0 (UTIM1) 0x43F9_40A0 (UTIM2) 0x5000_C0A0 (UTIM3) 0x43FB_00A0 (UTIM4) 0x43FB_40A0 (UTIM5)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	TIM											
	W																
0x43F9_00A4 (UBIR1) 0x5000_C0A4 (UBIR3) 0x43FB_00A4 (UBIR4) 0x43FB_00A4 (UBIR4) 0x43FB_40A4 (UBIR5)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	INC															
	W																
0x43F9_00A8 (UBMR1) 0x43F9_40A8 (UBMR2) 0x5000_C0A8 (UBMR3) 0x43FB_00A8 (UBMR4) 0x43FB_40A8 (UBMR5)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	MOD															
	W																

Table 31-4. UART Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x43F9_00AC (UBRC1) 0x43F9_40AC (UBRC2) 0x5000_C0AC (UBRC3) 0x43FB_00AC (UBRC4) 0x43FB_40AC (UBRC5)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	BCNT															
	W																
0x43F9_00B0 (ONEMS1) 0x43F9_40B0 (ONEMS2) 0x5000_C0B0 (ONEMS3) 0x43FB_00B0 (ONEMS4) 0x43FB_40B0 (ONEMS5)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	ONEMS															
	W																
0x43F9_00B4 (UTS1) 0x43F9_40B4 (UTS2) 0x5000_C0B4 (UTS3) 0x43FB_00B4 (UTS4) 0x43FB_40B4 (UTS5)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	FRCPERR	LOOP	DBGEN	LOOPIR	RXDBG	0	0	TXEMPTY	RXEMPTY	TXFULL	RXFULL	0	0	SOFTRST
	W																

31.3.3 Register Descriptions

This section contains the detailed register descriptions for the UART registers.

31.3.3.1 UART Receiver Register (URXD)

Figure 31-3 shows the URXD register, and Table 31-5 shows the register's field descriptions.

0x43F9_0000 (URXD1)
0x43F9_4000 (URXD2)
0x5000_C000 (URXD3)
0x43FB_0000 (URXD4)
0x43FB_4000 (URXD5)

Access: User read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CHARRDY	ERR	OVRRUN	FRMERR	BRK	PRERR	0	0	RX_DATA							
W																
Reset	0	1	0	0	0	0	0	0	—	—	—	—	—	—	—	—

Figure 31-3. UART Receiver Register

NOTE

Memory space between URXD and UTXD registers (for example, BASE + 0x04 up to BASE + 0x3c) cannot be accessed. Any read or write access to this space is considered an invalid address and results in a transfer error.

Table 31-5. UART Receiver Register Field Descriptions

Field	Description
31–16	Reserved
15 CHARRDY	Character ready. This read-only bit indicates an invalid read when the FIFO becomes empty and the software tries to read the same old data. This bit should not be used for polling for data written to the RxFIFO. 0 Character in RX_DATA field and associated flags are invalid. 1 Character in RX_DATA field and associated flags valid and ready for reading.
14 ERR	Error detect. Indicates whether the character present in the RX_DATA field has an error (OVRRUN, FRMERR, BRK or PRERR) status. The ERR bit is updated and valid for each received character. 0 No error status was detected. 1 An error status was detected.
13 OVRRUN	Receiver overrun. This read-only bit, when HIGH, indicates that the corresponding character was stored in the last position (32nd) of the RxFIFO. Even if a 33rd character has not been detected, this bit is set to ‘1’ for the 32nd character. 0 No RxFIFO overrun was detected. 1 A RxFIFO overrun was detected.
12 FRMERR	Frame error. Indicates whether the current character had a framing error (a missing stop bit) and is possibly corrupted. FRMERR is updated for each character read from the RxFIFO. 0 The current character has no framing error. 1 The current character has a framing error.

Table 31-5. UART Receiver Register Field Descriptions (continued)

Field	Description
11 BRK	BREAK detect. Indicates whether the current character was detected as a BREAK character. The data bits and the stop bit are all 0. The FRMERR bit is set when BRK is set. When odd parity is selected, PRERR is also set when BRK is set. BRK is valid for each character read from the RxFIFO. 0 The current character is not a BREAK character. 1 The current character is a BREAK character.
10 PRERR	Parity error. Indicates if the current character was detected with a parity error and is possibly corrupted. PRERR is updated for each character read from the RxFIFO. When parity is disabled, PRERR always reads as 0. 0 = No parity error was detected for data in the RX_DATA field. 1 = A parity error was detected for data in the RX_DATA field.
9–8	Reserved
7–0 RX_DATA	Received data. Holds the received character. In 7-bit mode, the most significant bit (MSB) is forced to 0. In 8-bit mode, all bits are active.

31.3.3.2 UART Transmitter Register (UTXD)

Figure 31-4 shows the UTXD register, and Table 31-6 shows the register's field descriptions.

0x43F9_0040 (UTXD1)
0x43F9_4040 (UTXD2)
0x5000_C040 (UTXD3)
0x43FB_0040 (UTXD4)
0x43FB_4040 (UTXD5)

Access: User write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0								
W									TX_DATA							
Reset	0	0	0	0	0	0	0	0	—	—	—	—	—	—	—	—

Figure 31-4. UART Transmitter Register

NOTE

Memory space between URXD and UTXD registers (for example, BASE + 0x04 up to BASE + 0x3c) cannot be accessed. Any read or write access to this space is considered an invalid address and results in a transfer error.

Table 31-6. UART Transmitter Register Field Descriptions

Field	Description
31–16	Reserved
15–8	Reserved
7–0 TX_DATA	Transmit data. Holds the parallel transmit data inputs. In 7-bit mode, D7 is ignored. In 8-bit mode, all bits are used. Data is transmitted least significant bit (LSB) first. A new character is transmitted when the TX_DATA field is written. The TX_DATA field must be written only when the TRDY bit is high to ensure that corrupted data is not sent.

31.3.3.3 UART Control Register 1 (UCR1)

Figure 31-5 shows the UCR1 register, and Table 31-7 shows the register's field descriptions.

0x43F9_0080 (UCR1_1)
0x43F9_4080 (UCR1_2)
0x5000_C080 (UCR1_3)
0x43FB_0080 (UCR1_4)
0x43FB_4080 (UCR1_5)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADEN	ADBR	TRDYEN	IDEN	ICD		RRDYEN	RXDMAEN	IREN	TXMPTYEN	RTSDEN	SNDBRK	TXDMAEN	ATDMAEN	DOZE	UARTEN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 31-5. UART Control Register 1

Table 31-7. UART Control Register 1 Field Descriptions

Field	Description
31–16	Reserved
15 ADEN	Automatic baud rate detection interrupt enable. Enables/disables the automatic baud rate detect complete (ADET) bit to generate an interrupt ($\overline{ipi\_uart\_mint}$ = 0). 0 Disable the automatic baud rate detection interrupt. 1 Enable the automatic baud rate detection interrupt.
14 ADBR	Automatic detection of baud rate. Enables/disables automatic baud rate detection. When the ADBR bit is set and the ADET bit is cleared, the receiver detects the incoming baud rate automatically. The ADET flag is set when the receiver verifies that the incoming baud rate is detected properly by detecting an ASCII character "A" or "a" (0x61 or 0x41). 0 Disable the automatic baud rate detection. 1 Enable the automatic baud rate detection.

Table 31-7. UART Control Register 1 Field Descriptions (continued)

Field	Description
13 TRDYEN	Transmitter ready interrupt enable. Enables/disables the transmitter ready interrupt (TRDY) when the transmitter has one or more slots available in the TxFIFO. The fill level in the TxFIFO at which an interrupt is generated is controlled by TxTL bits. When TRDYEN is negated, the transmitter ready interrupt is disabled. 0 Disable the transmitter ready interrupt. 1 Enable the transmitter ready interrupt.
12 IDEN	Idle condition detected interrupt enable. Enables/disables the IDLE bit to generate an interrupt (ipi_uart_rx = 0). 0 Disable the IDLE bit. 1 Enable the IDLE bit.
11–10 ICD	Idle condition detect. Controls the number of frames RXD is allowed to be idle before an idle condition is reported. 00 Report idle of more than 4 frames. 01 Report idle of more than 8 frames. 10 Report idle of more than 16 frames. 11 Report idle of more than 32 frames.
9 RRDYEN	Receiver ready interrupt enable. Enables/disables the RRDY interrupt when the RxFIFO contains data. The fill level in the RxFIFO at which an interrupt is generated is controlled by the RXTL bits. When RRDYEN is negated, the receiver ready interrupt is disabled. 0 Disable the RRDY interrupt. 1 Enable the RRDY interrupt.
8 RXDMAEN	Receive ready DMA enable. Enables/disables the receive DMA request ipd_uart_rx_dmareq when the receiver has data in the RxFIFO. The fill level in the RxFIFO at which a DMA request is generated is controlled by the RXFL bits. When negated, the receive DMA request is disabled. 0 Disable the DMA request. 1 Enable the DMA request.
7 IREN	Infrared interface enable. Enables/disables the IR interface. Refer to the IR interface description in Section 31.4.9, “Infrared Interface” for more information. 0 Disable the IR interface. 1 Enable the IR interface.
6 TXMPTYEN	Transmitter empty interrupt enable. Enables/disables the transmitter FIFO empty (TXFE) interrupt ipi_uart_tx. When negated, the TXFE interrupt is disabled. 0 Disable the transmitter FIFO empty interrupt. 1 Enable the transmitter FIFO empty interrupt.
5 RTSDEN	RTS delta interrupt enable. Enables/disables the RTSD interrupt. The current status of the *ipp_uart_rts* pin is read in the RTSS bit. 0 Disable the RTSD interrupt. 1 Enable the RTSD interrupt.
4 SNDBRK	Send BREAK. Forces the transmitter to send a BREAK character. The transmitter finishes sending the character in progress (if any) and sends BREAK characters until SNDBRK is reset. Because the transmitter samples SNDBRK after every bit is transmitted, it is important that SNDBRK is asserted high for a sufficient period of time to generate a valid BREAK. After the BREAK transmission completes, the UART transmits 2 mark bits. The user can continue to fill the TxFIFO. Any characters remaining are transmitted when the BREAK is terminated. 0 Do not send a BREAK character. 1 Send a BREAK character (continuous 0s).

Table 31-7. UART Control Register 1 Field Descriptions (continued)

Field	Description
3 TXDMAEN	Transmitter ready DMA enable. Enables/disables the transmit DMA request ipd_uart_tx_dmareq when the transmitter has one or more slots available in the TxFIFO. The fill level in the TxFIFO that generates the ipd_uart_tx_dmareq is controlled by the TXTL bits. 0 Disable the transmit DMA request. 1 Enable the transmit DMA request.
2 ATDMAEN	Aging DMA timer enable. Enables/disables the receive DMA request ipd_uart_rx_dmareq for the aging timer interrupt (triggered with AGTIM flag in USR1[8]). 0 Disable the DMA AGTIM interrupt. 1 Enable the DMA AGTIM interrupt.
1 DOZE	DOZE. Determines the UART enable condition in the doze state. When ipg_doze input pin is at '1', meaning the CPU executes a doze instruction and the system is placed in the doze state, the DOZE bit affects operation of the UART. While in the doze state, if this bit is asserted, the UART is disabled. Refer to the description in Section 31.4.10, "UART Operation in Low-Power System States." 0 Enable the UART when it is in doze state. 1 Disable the UART when it is in doze state.
0 UARTEN	UART Enable. Enables/disables the UART. If UARTEN is negated in the middle of a transmission, the transmitter stops and pulls the TXD line to a logic 1. UARTEN must be set to 1 before any access to UTXD and URXD registers; otherwise, an ipg_xfr_error is returned. Output ipg_uart_clk_en is internally connected to UARTEN and can be used for software controlled clock gating purpose. 0 Disable the UART. 1 Enable the UART.

31.3.3.4 UART Control Register 2 (UCR2)

Figure 31-6 shows the UCR2 register, and Table 31-8 shows the register's field descriptions.

0x43F9_0084 (UCR2_1)
0x43F9_4084 (UCR2_2)
0x5000_C084 (UCR2_3)
0x43FB_0084 (UCR2_4)
0x43FB_4084 (UCR2_5)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ESCI	IRTS	CTSC	CTS	ESCEN	RTEC		PREN	PROE	STPB	WS	RTSEN	ATEN	TXEN	RXEN	SRST
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 31-6. UART Control Register 2

Table 31-8. UART Control Register 2 Field Descriptions

Name	Description
31–16	Reserved
15 ESCI	Escape sequence interrupt enable. Enables/disables the ESCF bit to generate an interrupt. 0 Disable the escape sequence interrupt. 1 Enable the escape sequence interrupt.
14 IRTS	Ignore RTS pin. Forces the RTS input signal presented to the transmitter to always be asserted (set to low), effectively ignoring the external pin. When in this mode, the RTS pin serves as a general purpose input. 0 Transmit only when the RTS pin is asserted. 1 Ignore the RTS pin.
13 CTSC	CTS pin control. Controls the operation of the ipp_uart_cts output pin. When CTSC is asserted, the ipp_uart_cts output pin is controlled by the receiver. When the RxFIFO is filled to the level of the programmed trigger level and the start bit of the overflowing character (TRIGGER LEVEL + 1) is validated, the ipp_uart_cts output pin is negated to indicate to the far-end transmitter to stop transmitting. When the trigger level is programmed for less than 32, the receiver continues to receive data until the RxFIFO is full. When the CTSC bit is negated, the ipp_uart_cts output pin is controlled by the CTS bit. On reset, because CTSC is cleared to 0, the ipp_uart_cts pin is controlled by the CTS bit, which again is cleared to 0 on reset. This means that on reset, the ipp_uart_cts signal is negated. 0 The ipp_uart_cts pin is controlled by the CTS bit. 1 The ipp_uart_cts pin is controlled by the receiver.
12 CTS	Clear to send. Controls the ipp_uart_cts pin when the CTSC bit is negated. CTS has no function when CTSC is asserted. 0 The ipp_uart_cts pin is high (inactive). 1 The ipp_uart_cts pin is low (active).
11 ESCEN	Escape enable. Enables/disables the escape sequence detection logic. 0 Disable escape sequence detection. 1 Enable escape sequence detection.
10-9 RTEC	Request to send edge control. Selects the edge that triggers the RTS interrupt. This has no effect on the RTS delta interrupt. RTEC has an effect only when RTSEN = 1 (See Table 31-26.) 00 Trigger interrupt on a rising edge 01 Trigger interrupt on a falling edge 1X Trigger interrupt on any edge
8 PREN	Parity enable. Enables/disables the parity generator in the transmitter and parity checker in the receiver. When PREN is asserted, the parity generator and checker are enabled, and disabled when PREN is negated. 0 Disable the parity generator and checker. 1 Enable the parity generator and checker.
7 PROE	Parity odd/even. Controls the sense of the parity generator and checker. When PROE is high, odd parity is generated and expected. When PROE is low, even parity is generated and expected. PROE has no function if PREN is low. 0 Even parity 1 Odd parity
6 STPB	Stop. Controls the number of stop bits transmitted after a character. When STPB is high, 2 stop bits are sent. When STPB is low, 1 stop bit is sent. STPB has no effect on the receiver, which expects 1 or more stop bits. 0 Transmit 1 stop bit. 1 Transmit 2 stop bits.

Table 31-8. UART Control Register 2 Field Descriptions (continued)

Name	Description
5 WS	Word size. Controls the character length. When WS is high, the transmitter and receiver are in 8-bit mode. When WS is low, they are in 7-bit mode. The transmitter ignores bit 7 and the receiver sets bit 7 to 0. WS can be changed in-between transmission (reception) of characters, but not when a transmission (reception) is in progress. During transmission (reception), the length of the current character being transmitted (received) is unpredictable. 0 7-bit transmit and receive character length (not including START, STOP or PARITY bits) 1 8-bit transmit and receive character length (not including START, STOP or PARITY bits)
4 RTSEN	Request to send interrupt enable. Controls the RTS edge sensitive interrupt. When RTSEN is asserted and the programmed edge is detected on the ipp_uart_rts pin, the RTSF bit is asserted. (See Table 31-26.) 0 Disable the request to send interrupt. 1 Enable the request to send interrupt.
3 ATEN	Aging timer enable. This bit is used to enable the aging timer interrupt (triggered with AGTIM). 0 Disable the AGTIM interrupt. 1 Enable the AGTIM interrupt.
2 TXEN	Transmitter enable. Enables/disables the transmitter. When TXEN is negated, the transmitter is disabled and idle. When the UARTEN and TXEN bits are set, the transmitter is enabled. If TXEN is negated in the middle of a transmission, the UART disables the transmitter immediately and starts marking 1s. The transmitter FIFO cannot be written to when this bit is cleared. 0 Disable the transmitter. 1 Enable the transmitter.
1 RXEN	Receiver enable. Enables/disables the receiver. When the receiver is enabled, if the RXD input is already low, the receiver does not recognize BREAK characters. The receiver requires a valid 1-to-0 transition before it can accept any character. 0 Disable the receiver. 1 Enable the receiver.
0 $\overline{\text{SRST}}$	Software reset. Resets the transmitter and receiver state machines, all FIFOs, status registers USR1 and USR2, and BRM registers UBIR and UBMR. Once the software writes 0 to $\overline{\text{SRST}}$, the software reset remains active for 4 clock cycles of IPG_CLK before the hardware deasserts $\overline{\text{SRST}}$. The software can write only 0 to $\overline{\text{SRST}}$. Writing 1 to $\overline{\text{SRST}}$ is ignored. 0 Reset the transmit and receive state machines, all FIFOs, and all status registers. 1 No reset

31.3.3.5 UART Control Register 3 (UCR3)

Figure 31-7 shows the UCR3 register, and Table 31-9 shows the register's field descriptions.

0x43F9_0088 (UCR3_1)
0x43F9_4088 (UCR3_2)
0x5000_C088 (UCR3_3)
0x43FB_0088 (UCR3_4)
0x43FB_4088 (UCR3_5)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DPEC		DTREN	PARERREN	FRAERREN	DSR	DCD	RI	ADNIMP	RXDSEN	AIRINTEN	AWAKEN	DTRDEN	RXDMUXSEL	INVT	ACIEN
Reset	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0

Figure 31-7. UART Control Register 3

Table 31-9. UART Control Register 3 Field Descriptions

Name	Description
31–16	Reserved
15–14 DPEC	DTR/DSR interrupt edge control. These bits control the edge of ipp_uart_dtr_dce_i (DCE) or ipp_uart_dsr_dte_i (DTE), where an interrupt would be generated. An interrupt is generated only if the DTREN bit is set. 00 Generate interrupt on rising edge. 01 Generate interrupt on falling edge. 1X Generate interrupt on either edge.
13 DTREN	Data terminal ready interrupt enable. When this bit is set, it enables the status bit DTRF (USR2 [13]) (DTR/DSR edge sensitive interrupt) to cause an interrupt. 0 Disable the data terminal ready interrupt. 1 Enable the data terminal ready interrupt.
12 PARERREN	Parity error interrupt enable. Enables/disables the interrupt. When asserted, PARERREN causes the PARITYERR bit to generate an interrupt. 0 Disable the parity error interrupt. 1 Enable the parity error interrupt.
11 FRAERREN	Frame error interrupt enable. Enables/disables the interrupt. When asserted, FRAERREN causes the FRAMERR bit to generate an interrupt. 0 Disable the frame error interrupt. 1 Enable the frame error interrupt.
10 DSR	Data set ready. This bit is used by software to control the DSR/DTR output pin for the modem interface. In DCE mode, it applies to ipp_uart_dsr_dce_o. In DTE mode, it applies to ipp_uart_dtr_dte_o. 0 DSR/ DTR pin is logic zero. 1 DSR/ DTR pin is logic one.

Table 31-9. UART Control Register 3 Field Descriptions (continued)

Name	Description
9 DCD	Data carrier detect. In DCE mode, this bit is used by software to control the ipp_uart_dcd_dce_o output pin for the modem interface. In DTE mode, when this bit is set, it enables the status bit DCDDELT (USR2 (6)) to cause an interrupt. 0 ipp_uart_dcd_dce_o pin is logic zero (DCE mode). 1 ipp_uart_dcd_dce_o pin is logic one (DCE mode). 0 Disable the DCDDELT interrupt (DTE mode). 1 Enable the DCDDELT interrupt (DTE mode).
8 RI	Ring indicator. In DCE mode, this bit is used by software to control the ipp_uart_ri_dce_o output pin for the modem interface. In DTE mode, when this bit is set, it enables the status bit RIDELT (USR2 (10)) to cause an interrupt. 0 ipp_uart_ri_dce_o pin is logic zero (DCE mode). 1 ipp_uart_ri_dce_o pin is logic one (DCE mode). 0 Disable the RIDELT interrupt (DTE mode). 1 Enable the RIDELT interrupt (DTE mode).
7 ADNIMP	Autobaud detection not improved. Disables new features of autobaud detection. Refer to Section 31.4.7.2, “Baud Rate Automatic Detection Protocol Improved” for more details. 0 Select autobaud detection new features. 1 Keep old autobaud detection mechanism.
6 RXDSEN	Receive status interrupt enable. Controls the receive status interrupt (ipi_uart_rx). When this bit is enabled and RXDS status bit is set, the interrupt ipi_uart_rx is generated. 0 Disable the RXDS interrupt. 1 Enable the RXDS interrupt.
5 AIRINTEN	Asynchronous IR WAKE interrupt enable. Controls the asynchronous IR WAKE interrupt. When AIRINTEN is asserted and a pulse is detected on the UART_RX pin, an interrupt is generated. 0 Disable the AIRINT interrupt. 1 Enable the AIRINT interrupt.
4 AWAKEN	Asynchronous WAKE interrupt enable. Controls the asynchronous WAKE interrupt. When AWAKEN is asserted and a falling edge is detected on the RXD pin, an interrupt is generated. 0 Disable the AWAKE interrupt. 1 Enable the AWAKE interrupt.
3 DTRDEN	Data terminal ready delta enable. Enables/disables the asynchronous DTRD interrupt. When DTRDEN is asserted and an edge (rising or falling) is detected on ipp_uart_dtr_dce_i (in DCE mode) or on ipp_uart_dsr_dte_i (in DTE mode), an interrupt is generated. 0 Disable the DTRD interrupt. 1 Enable the DTRD interrupt.
2 RXDMUXSEL	RXD muxed input selected. Selects the ipp_uart_rxd_mux input pin for serial and Infrared input signal. 0 Serial input pin is ipp_uart_rxd and IrDA input pin is ipp_uart_rxd_ir. 1 Input pin is ipp_uart_rxd_mux for serial and IR interfaces.

Table 31-9. UART Control Register 3 Field Descriptions (continued)

Name	Description
1 INVT	Inverted infrared transmission. Sets the active level for the transmission. When INVT is cleared, the infrared logic block transmits a positive IR 3/16 pulse for all 0s and 0s are transmitted for 1s. When INVT is set (INVT = 1), the infrared logic block transmits an active low or negative infrared 3/16 pulse for all 0s and 1s are transmitted for 1s. 0 Active low transmission 1 Active high transmission.
0 ACIEN	Autobaud counter interrupt enable. This bit is used to enable the autobaud counter stopped interrupt (triggered with ACST (USR2[11]). 0 Disable ACST interrupt. 1 Enable ACST interrupt

31.3.3.6 UART Control Register 4 (UCR4)

Figure 31-8 shows the UCR4 register, and Table 31-10 shows the register's field descriptions.

0x43F9_008C (UCR4_1)
0x43F9_408C (UCR4_2)
0x5000_C08C (UCR4_3)
0x43FB_008C (UCR4_4)
0x43FB_408C (UCR4_5)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CTSTL						INVR	ENIRI	WKEN	IDDMAEN	IRSC	LPBYP	TCEN	BKEN	OREN	DREN
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 31-8. UART Control Register 4

Table 31-10. UART Control Register 4 Field Descriptions

Name	Description
31–16	Reserved
15–10 CTSTL	CTS trigger level. Controls the threshold at which the $\overline{\text{ipp\_uart\_cts}}$ pin is deasserted by the RxFIFO. After the trigger level is reached and the $\overline{\text{ipp\_uart\_cts}}$ pin is deasserted, the RxFIFO continues to receive data until it is full. The CTSTL bits are encoded as shown in the settings column. 000000 0 characters received 000001 RxFIFO has 1 character. ... ... 100000 RxFIFO has 32 characters (maximum). All other settings reserved

Table 31-10. UART Control Register 4 Field Descriptions (continued)

Name	Description
9 INVR	Inverted infrared reception. Determines the logic level for the detection. When cleared, the infrared logic block expects an active low or negative IR 3/16 pulse for 0s and 1s are expected for 1s. When INVR is set (INVR1), the infrared logic block expects an active high or positive IR 3/16 pulse for 0s and 0s are expected for 1s. 0 Detect active low. 1 Detect active high.
8 ENIRI	Serial infrared interrupt enable. Enables/disables the serial infrared interrupt. 0 Disable the serial infrared interrupt 1 Enable the serial infrared interrupt.
7 WKEN	WAKE interrupt enable. Enables/disables the WAKE bit to generate an interrupt. The WAKE bit is set at the detection of a start bit by the receiver. 0 Disable the WAKE interrupt. 1 Enable the WAKE interrupt.
6 IDDMAEN	DMA IDLE condition detected interrupt enable. Enables/disables the receive DMA request ipd_uart_rx_dmareq for the IDLE interrupt (triggered with IDLE flag in USR2[12]). 0 Disable the DMA IDLE interrupt. 1 Enable the DMA IDLE interrupt.
5 IRSC	IR special case. Selects the clock for the vote logic. When set, IRSC switches the vote logic clock from the sampling clock to the UART reference clock. The IR pulses are counted a predetermined amount of time depending on the reference frequency. Refer to Section 31.4.9.3, "Infrared Special Case (IRSC) Bit." 0 The vote logic uses the sampling clock (16x baud rate) for normal operation. 1 The vote logic uses the UART reference clock.
4 LPBYP	Low power bypass. Allows bypassing the low power new features in the UART. Use during debug phase. 0 Enable low power features. 1 Disable low power features.
3 TCEN	Transmit complete interrupt enable. Enables/disables the TXDC bit to generate an interrupt. (ipi_uart_tx = 0) 0 Disable the TXDC interrupt. 1 Enable the TXDC interrupt.
2 BKEN	BREAK condition detected interrupt enable. Enables/disables the BRCD bit to generate an interrupt. 0 Disable the BRCD interrupt. 1 Enable the BRCD interrupt.
1 OREN	Receiver overrun interrupt enable. Enables/disables the ORE bit to generate an interrupt. 0 Disable the ORE interrupt. 1 Enable the ORE interrupt.
0 DREN	Receive data ready interrupt enable. Enables/disables the RDR bit to generate an interrupt. 0 Disable the RDR interrupt. 1 Enable the RDR interrupt.

31.3.3.7 UART FIFO Control Register (UFCR)

Figure 31-9 shows the UFCR register, and Table 31-11 shows the register's field descriptions.

0x43F9_0090 (UFCR1)
0x43F9_4090 (UFCR2)
0x5000_C090 (UFCR3)
0x43FB_0090 (UFCR4)
0x43FB_4090 (UFCR5)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TXTL						RFDIV			DCE DTE	RXTL					
Reset	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1

Figure 31-9. UART FIFO Control Register

Table 31-11. UART FIFO Control Register Field Descriptions

Name	Description
31–16	Reserved
15–10 TXTL	Transmitter trigger level. Controls the threshold at which a maskable interrupt is generated by the TxFIFO. A maskable interrupt is generated whenever the data level in the TxFIFO falls below the selected threshold. The bits are encoded as shown in the settings column. 000000 Reserved 000001 Reserved 000010 TxFIFO has 2 or fewer characters. ... ... 011111 TxFIFO has 31 or fewer characters. 100000 TxFIFO has 32 characters (maximum). All other settings reserved
9–7 RFDIV	Reference frequency divider. Controls the divide ratio for the reference clock. The input clock is ipg_perclk. The output from the divider (ref_clk) is used by BRM to create the 16x baud rate oversampling clock. 000 Divide input clock by 6. 001 Divide input clock by 5. 010 Divide input clock by 4. 011 Divide input clock by 3. 100 Divide input clock by 2. 101 Divide input clock by 1. 110 Divide input clock by 7. 111 Reserved

Table 31-11. UART FIFO Control Register Field Descriptions (continued)

Name	Description
6 DCEDTE	DCE/DTE mode select. Selects data communication equipment (DCE) or DTE data terminal equipment (DTE) mode. 0 Select DCE mode. 1 Select DTE mode.
5–0 RXTL	Receiver trigger level. Controls the threshold at which a maskable interrupt is generated by the RxFIFO. A maskable interrupt is generated whenever the data level in the RxFIFO reaches the selected threshold. The RXTL bits are encoded as shown in the settings column. 000000 0 characters received. 000001 RxFIFO has 1 character. ... ... 011111 RxFIFO has 31 characters. 100000 RxFIFO has 32 characters (maximum). All other settings reserved

31.3.3.8 UART Status Register 1 (USR1)

Figure 31-10 shows the USR1 register, and Table 31-12 shows the register's field descriptions.

0x43F9_0094 (USR1_1)
0x43F9_4094 (USR1_2)
0x5000_C094 (USR1_3)
0x43FB_0094 (USR1_4)
0x43FB_4094 (USR1_5)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PARITYERR	RTSS	TRDY	RTSD	ESCF	FRAMERR	RRDY	AGTIM	DTRD	RXDS	AIRINT	AWAKE	0	0	0	0
W	w1c			w1c	w1c	w1c		w1c	w1c		w1c	w1c				
Reset	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0

Figure 31-10. UART Status Register 1

Table 31-12. UART Status Register 1 Field Descriptions

Name	Description
31–16	Reserved
15 PARITYERR	Parity error interrupt flag. Indicates a parity error is detected. Clear PARITYERR by writing 1 to it. Writing 0 to PARITYERR has no effect. When parity is disabled, PARITYERR always reads 0. At reset, PARITYERR is set to 0. 0 No parity error detected. 1 Parity error detected.
14 RTSS	$\overline{\text{RTS}}$ pin status. Indicates the current status of the $\overline{\text{ipp\_uart\_rts}}$ pin. A “snapshot” of the pin is taken immediately before RTSS is presented to the data bus. RTSS cannot be cleared because all writes to RTSS are ignored. At reset, RTSS is set to 0. 0 The $\overline{\text{ipp\_uart\_rts}}$ pin is high (inactive). 1 The $\overline{\text{ipp\_uart\_rts}}$ pin is low (active).
13 TRDY	Transmitter ready interrupt/DMA flag. Indicates that the TxFIFO emptied below its target threshold and requires data. TRDY is automatically cleared when the data level in the TxFIFO goes above the set threshold level by TXFL bits. At reset, TRDY is set to 1. 0 The transmitter does not require data. 1 The transmitter requires data (interrupt posted).
12 RTSD	RTS delta. Indicates whether the $\overline{\text{ipp\_uart\_rts}}$ pin changed state. It (RTSD) generates a maskable interrupt. When in STOP mode, RTS assertion sets RTSD and can be used to wake the processor. The current state of the $\overline{\text{ipp\_uart\_rts}}$ pin is available on the RTSS bit. Clear RTSD by writing 1 to it. Writing 0 to RTSD has no effect. At reset, RTSD is set to 0. 0 $\overline{\text{ipp\_uart\_rts}}$ pin did not change state since last cleared. 1 $\overline{\text{ipp\_uart\_rts}}$ pin changed state. (Write 1 to clear.)
11 ESCF	Escape sequence interrupt flag. Indicates if an escape sequence was detected. ESCF is asserted when the ESCEN bit is set and an escape sequence is detected in the RxFIFO. Clear ESCF by writing 1 to it. Writing 0 to ESCF has no effect. 0 No escape sequence detected. 1 Escape sequence detected.
10 FRAMERR	Frame error interrupt flag. Indicates that a frame error was detected. The $\overline{\text{ipi\_uart\_mint}}$ interrupt is generated by this. Clear FRAMERR by writing 1 to it. Writing 0 to FRAMERR has no effect. 0 No frame error detected. 1 Frame error detected.
9 RRDY	Receiver ready interrupt/DMA flag. Indicates that the RxFIFO data level is above the threshold set by the RXFL bits. (See the RXFL bits description in Table 31-11 for setting the interrupt threshold.) When asserted, RRDY generates a maskable interrupt or DMA request. RRDY is automatically cleared when data level in the RxFIFO goes below the set threshold level. At reset, RRDY is set to 0. 0 No character ready. 1 Character(s) ready (interrupt posted).
8 AGTIM	Aging timer interrupt flag. Indicates that data in the RxFIFO has been idle for a time of 8 character lengths (where a character length consists of 7 or 8 bits, depending on the setting of the WS bit in UCR2, with the bit time corresponding to the baud rate setting) and FIFO data level is less than RxFIFO threshold level (RxTL in the UFCR). Clear AGTIM by writing 1 to it. 0 AGTIM is not active 1 AGTIM is active.

Table 31-12. UART Status Register 1 Field Descriptions (continued)

Name	Description
7 DTRD	DTR delta. Indicates whether ipp_uart_dtr_dce_i (in DCE mode) or ipp_uart_dsr_dte_i (in DTE mode) pins changed state. DTRD generates a maskable interrupt if DTRDEN (UCR3[3]) is set. Clear DTRD by writing 1 to it. Writing 0 to DTRD has no effect. 0 ipp_uart_dtr_dce_i (DCE) or ipp_uart_dsr_dte_i (DTE) pin did not change state since last cleared. 1 ipp_uart_dtr_dce_i (DCE) or ipp_uart_dsr_dte_i (DTE) pin changed state.
6 RXDS	Receiver IDLE interrupt flag. Indicates that the receiver state machine is in an IDLE state, the next state is IDLE, and the receive pin is high. RXDS is automatically cleared when a character is received. RXDS is active only when the receiver is enabled. 0 Receive is in progress. 1 Receiver is IDLE.
5 AIRINT	Asynchronous IR WAKE interrupt flag. Indicates that the IR WAKE pulse was detected on the ipp_uart_rxd_ir pin, or on ipp_uart_rxd_mux if RXDMUXSEL is set to 1. Clear AIRINT by writing 1 to it. Writing 0 to AIRINT has no effect. 0 No pulse was detected on the RXD IrDA pin. 1 A pulse was detected on the RXD IrDA pin.
4 AWAKE	Asynchronous WAKE interrupt flag. Indicates that a falling edge was detected on the ipp_uart_rxd pin, or on ipp_uart_rxd_mux if RXDMUXSEL is set to 1. Clear AWAKE by writing 1 to it. Writing 0 to AWAKE has no effect. Caution: AWAKE Interrupt flag cannot be used in loopback mode (UTS[12] = 1'b1) as RXD pin will be ignored. (See Table 31-20.) 0 No falling edge was detected on the RXD serial pin. 1 A falling edge was detected on the RXD serial pin.
3–0	Reserved

31.3.3.9 UART Status Register 2 (USR2)

Figure 31-11 shows the USR2 register, and Table 31-13 shows the register's field descriptions.

0x43F9_0098 (USR2_1)
0x43F9_4098 (USR2_2)
0x5000_C098 (USR2_3)
0x43FB_0098 (USR2_4)
0x43FB_4098 (USR2_5)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADET	TXFE	DTRF	IDLE	ACST	RIDELT	RIIN	IRINT	WAKE	DCD DELT	DCDIN	RTSF	TXDC	BRCD	ORE	RDR
W	w1c			w1c		w1c			w1c			w1c		w1c	w1c	
Reset	0	1	0	0	0	0	0	0	0	0	1	0	1	0	0	0

Figure 31-11. UART Status Register 2

Table 31-13. UART Status Register 2 Field Descriptions

Name	Description
31–16	Reserved
15 ADET	Automatic baud rate detect complete. Indicates that an “A” or “a” was received and that the receiver detected and verified the incoming baud rate. Clear ADET by writing 1 to it. Writing 0 to ADET has no effect. 0 ASCII “A” or “a” was not received. 1 ASCII “A” or “a” was received.
14 TXFE	Transmit buffer FIFO empty. Indicates that the transmit buffer (TxFIFO) is empty. TXFE is cleared automatically when data is written to the TxFIFO. Even though TXFE is high, the transmission might still be in progress. 0 The transmit buffer (TxFIFO) is not empty. 1 The transmit buffer (TxFIFO) is empty.
13 DTRF	DTR edge triggered interrupt flag. This bit is asserted, when the programmed edge is detected on the ipp_uart_dtr_dce_i pin (DCE mode) or on ipp_uart_dsr_dte_i (DTE mode). This flag can cause an interrupt if DTREN (UCR3[13]) is enabled. Clear DTR/DSR by writing 1 to it. 0 Programmed edge not detected on DTR/DSR. 1 Programmed edge detected on DTR/DSR.
12 IDLE	Idle condition. Indicates that an idle condition has existed for more than a programmed amount frame. (See Section 31.4.5.2.1, “Idle Line Detect.”) An interrupt can be generated by this IDLE bit if IDEN (UCR1[12]) is enabled.Clear IDLE by writing 1 to it. Writing 0 to IDLE has no effect. 0 No idle condition detected. 1 Idle condition detected.
11 ACST	Autobaud counter stopped. in autobaud detection (ADBR=1). Indicates the counter that determines the baud rate was running and is now stopped. This means either START bit is finished (if ADNIMP=1), or Bit 0 is finished (if ADNIMP=0). See Section 31.4.7.2.2, “New Autobaud Counter Stopped Bit and Interrupt” for more details. An interrupt can be flagged on ipi_uart_mint if ACIEN=1. Clear ACST by writing 1 to it. 0 Measurement of bit length not finished (in autobaud). 1 Measurement of bit length finished (in autobaud).
10 RIDELT	Ring Indicator Delta. This bit is used in DTE mode to indicate that the Ring Indicator input (ipp_uart_ri_dte_i) has changed state. This flag can generate an interrupt if RI (UCR3[8]) is enabled. Clear RIDELT by writing 1 to it. Writing 0 to RIDELT has no effect. 0 Ring Indicator input has not changed state. 1 Ring Indicator input has changed state.
9 RIIN	Ring indicator input. This bit is used in DTE mode to reflect the status if the ring indicator input (ipp_uart_ri_dte_i). The ring indicator input is used to indicate that a ring has occurred. In DCE mode, this bit is always zero. 0 Ring detected 1 No ring detected.
8 IRINT	Serial infrared interrupt flag. When an edge is detected on the RX pin during SIR mode, this flag will be asserted. This flag can cause an interrupt on ipi_uart_mint, which can be masked using the control bit ENIRI: UCR4 [8]. Clear IRINT by writing 1 to it. 0 No edge detected. 1 Valid edge detected.
7 WAKE	Wake. Indicates the start bit is detected. WAKE can generate an interrupt on ipi_uart_mint that can be masked using the WKEN bit. Clear WAKE by writing 1 to it. Writing 0 to WAKE has no effect. 0 Start bit not detected. 1 Start bit detected.

Table 31-13. UART Status Register 2 Field Descriptions (continued)

Name	Description
6 DCDDELT	Data carrier detect delta. This bit is used in DTE mode to indicate that the data carrier detect input has changed state (ipp_uart_dcd_dte_i). This flag can cause an interrupt if DCD (UCR3[9]) is enabled. When in STOP mode, this bit can be used to wake the processor. In DCE mode, this bit is always zero. Clear DCDDELT by writing 1 to it. 0 Data carrier detect input has not changed state. 1 Data carrier detect input has changed state.
5 DCDIN	Data carrier detect input. This bit is used in DTE mode to reflect the status of the data carrier detect input (ipp_uart_dcd_dte_i). The data carrier detect input is used to indicate that a carrier signal has been detected. In DCE mode, this bit is always zero. 0 Carrier signal detected 1 No carrier signal detected
4 RTSF	RTS edge triggered interrupt flag. Indicates if a programmed edge was detected on the ipp_uart_rts pin. The RTEC bits select the edge that generates an interrupt. (See Table 31-26.) The RTSF can generate an interrupt on ipi_uart_mint that can be masked using the RTSEN bit. Clear RTSF by writing 1 to it. Writing 0 to RTSF has no effect. 0 Programmed edge not detected on ipp_uart_rts. 1 Programmed edge detected on ipp_uart_rts.
3 TXDC	Transmission complete. Indicates that the transmit buffer (TxFIFO) and shift register are empty; therefore, the transmission is complete. TXDC is cleared automatically when data is written to the TxFIFO. 0 Transmission is incomplete. 1 Transmission is complete.
2 BRCD	BREAK condition detected. Indicates that a BREAK condition was detected by the receiver. Clear BRCD by writing 1 to it. Writing 0 to BRCD has no effect. 0 No BREAK condition was detected. 1 A BREAK condition was detected.
1 ORE	Overrun error. When set to 1, ORE indicates that the receive buffer (RxFIFO) was full (32 chars inside), and a 33rd character has been fully received. This 33rd character has been discarded. Clear ORE by writing 1 to it. Writing 0 to ORE has no effect. 0 No overrun error. 1 Overrun error.
0 RDR	Receive data ready. Indicates that at least 1 character is received and written to the RxFIFO. If the URXD register is read and there is only 1 character in the RxFIFO, RDR is automatically cleared. 0 No receive data ready. 1 Receive data ready.

31.3.3.10 UART Escape Character Register (UESC)

Figure 31-12 shows the UESC register, and Table 31-14 shows the register's field descriptions.

0x43F9_009C (UESC1)
0x43F9_409C (UESC2)
0x5000_C09C (UESC3)
0x43FB_009C (UESC4)
0x43FB_409C (UESC5)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	ESC_CHAR							
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	1

Figure 31-12. UART Escape Character Register

Table 31-14. UART Escape Character Register Field Descriptions

Name	Description
31–8	Reserved
7–0 ESC_CHAR	UART escape character. Holds the selected escape character that all received characters are compared against to detect an escape sequence.

31.3.3.11 UART Escape Timer Register (UTIM)

Figure 31-13 shows the UTIM register, and Table 31-15 shows the register's field descriptions.

0x43F9_00A0 (UTIM1)
0x43F9_40A0 (UTIM2)
0x5000_C0A0 (UTIM3)
0x43FB_00A0 (UTIM4)
0x43FB_40A0 (UTIM5)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	TIM											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 31-13. UART Escape Timer Register

Table 31-15. UART Escape Timer Register Field Descriptions

Name	Description
31–12	Reserved
11–0 TIM	UART escape timer. Holds the maximum time interval (in ms) allowed between escape characters. The escape timer register is programmable in intervals of 2 ms. See Section 31.4.8, “Escape Sequence Detection” and Table 31-31 for more information on the UART escape sequence detection. Reset value 0x000 = 2 ms up to 0xFFF = 8.192 s.

31.3.3.12 UART BRM Incremental Register (UBIR)

Figure 31-14 shows the UBIR register, and Table 31-16 shows the register’s field descriptions.

0x43F9_00A4 (UBIR1)
0x43F9_40A4 (UBIR2)
0x5000_C0A4 (UBIR3)
0x43FB_00A4 (UBIR4)
0x43FB_40A4 (UBIR5)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	INC															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 31-14. UART BRM Incremental Register

Table 31-16. UART BRM Incremental Register Field Descriptions

Name	Description
31–16	Reserved
15–0 INC	Incremental numerator. Holds the numerator value minus one of the BRM ratio. See Section 31.4.6, “Binary Rate Multiplier (BRM).” The UBIR register MUST be updated before the UBMR register is updated for the baud rate to be updated correctly. If only one register is written to by the software, the BRM ignores this data until the other register is written to by the software. Updating this field using byte accesses is not recommended and is undefined.

31.3.3.13 UART BRM Modulator Register (UBMR)

Figure 31-15 shows the UBMR register, and Table 31-17 shows the register’s field descriptions.

0x43F9_00A8 (UBMR1)
0x43F9_40A8 (UBMR2)
0x5000_C0A8 (UBMR3)
0x43FB_00A8 (UBMR4)
0x43FB_40A8 (UBMR5)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MOD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 31-15. UART BRM Modulator Register

Table 31-17. UART BRM Modulator Register Field Descriptions

Name	Description
31–16	Reserved
15–0 MOD	Modulator denominator. Holds the value of the denominator minus one of the BRM ratio. See Section 31.4.6, “Binary Rate Multiplier (BRM).” The UBIR register MUST be updated before the UBMR register is updated for the baud rate to be updated correctly. If only one register is written to by the software, the BRM ignores this data until the other register is written to by the software. Updating this register using byte accesses is not recommended and undefined.

31.3.3.14 UART Baud Rate Count Register (UBRC)

Figure 31-16 shows the UBRC register, and Table 31-18 shows the register’s field descriptions.

0x43F9_00AC (UBRC1)
0x43F9_40AC (UBRC2)
0x5000_C0AC (UBRC3)
0x43FB_00AC (UBRC4)
0x43FB_40AC (UBRC5)

Access: User read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BCNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Figure 31-16. UART Baud Rate Count Register

Table 31-18. UART Baud Rate Count Register Field Descriptions

Name	Description
31–16	Reserved
15–0 BCNT	Baud rate count register. This read-only register is used to count the start bit of the incoming baud rate (if ADNIMP=1), or start bit + bit0 (if ADNIMP=0). When the measurement is done, the baud rate count register contains the number of the UART internal clock cycles (clock after divider) present in an incoming bit. BCNT retains its value until the next automatic baud rate detection sequence has been initiated. The 16-bit baud rate count register is reset to 4 and stays at hex FFFF in case of an overflow.

31.3.3.15 UART One Millisecond Register (ONEMS)

Figure 31-17 shows the ONEMS register, and Table 31-19 shows the register's field descriptions.

0x43F9_00B0 (ONEMS1)
0x43F9_40B0 (ONEMS2)
0x5000_C0B0 (ONEMS3)
0x43FB_00B0 (ONEMS4)
0x43FB_40B0 (ONEMS5)

Access: User read/write

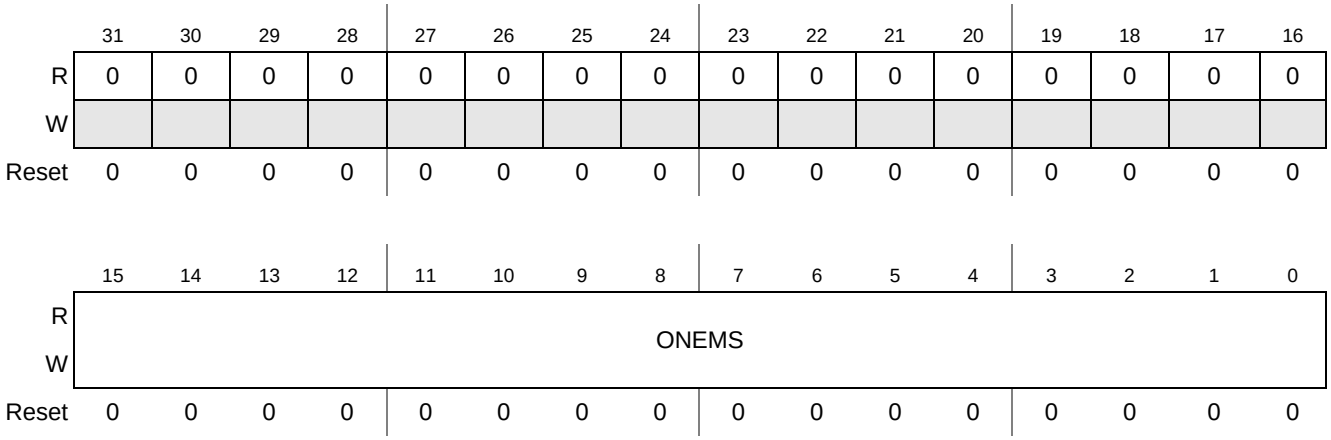


Figure 31-17. UART One Millisecond Register

Table 31-19. UART One Millisecond Register Field Descriptions

Name	Description
31–16	Reserved
15–0 ONEMS	One millisecond register. This 16-bit register must contain the value of the UART internal frequency (ref_clk in Figure 31-1) divided by 1000. The internal frequency is obtained after the UART BRM internal divider (F (ref_clk) = F(ipg_perclk) / RFDIV). This register contains the value corresponding to the number of the UART BRM internal clock cycles present in one millisecond. The ONEMS (and UTIM) registers value are used in the escape character detection feature (Section 31.4.8, “Escape Sequence Detection”) to count the number of clock cycles left between two escape characters. The ONEMS register is also used in infrared special case mode (IRSC = UCR4[5] = 1’b1). See Section 31.4.9.3, “Infrared Special Case (IRSC) Bit.”

31.3.3.16 UART Test Register (UTS)

Figure 31-18 shows the UTS register, and Table 31-20 shows the register’s field descriptions.

0x43F9_00B4 (UTS1)
0x43F9_40B4 (UTS2)
0x5000_C0B4 (UTS3)
0x43FB_00B4 (UTS4)
0x43FB_40B4 (UTS5)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	FRCPERR	LOOP	DBGEN	LOOPIR	RXDBG	0	0	TXEMPTY	RXEMPTY	TXFULL	RXFULL	0	0	SOFTRST
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

Figure 31-18. UART Test Register

Table 31-20. UART Test Register Field Descriptions

Name	Description
31–14	Reserved
13 FRCPERR	Force parity error. Forces the transmitter to generate a parity error if parity is enabled. FRCPERR is provided for system debugging. 0 Generate normal parity. 1 Generate inverted parity (error).
12 LOOP	Loop TX and RX for test. Controls loopback for test purposes. When LOOP is high, the receiver input is internally connected to the transmitter and ignores the RXD pin. The transmitter is unaffected by LOOP. If RXDMUXSEL (UCR3[2]) is set to 1, the loopback is applied on serial and IrDA signals. If RXDMUXSEL is set to 0, the loopback is applied only on serial signals. 0 Normal receiver operation. 1 Internally connect the transmitter output to the receiver input.
11 DBGEN	$\overline{\text{debug\_enable}}$. This bit controls whether to respond to the $\overline{\text{debug}}$ input signal. 0 UART will go into debug mode when $\overline{\text{debug}}$ is LOW. 1 UART will not go into debug mode even if $\overline{\text{debug}}$ is LOW.
10 LOOPIR	Loop TX and RX for IR test (LOOPIR). This bit controls loopback from transmitter to receiver in the Infrared interface. 0 No IR loop. 1 Connect IR transmitter to IR receiver.
9 RXDBG	RX_fifo_debug_mode. This bit controls the operation of the RxFIFO read counter when in debug mode. 0 Read pointer for rx_fifo does not increment. 1 Read pointer for rx_fifo increments as normal.
8–7	Reserved
6 TXEMPTY	TxFIFO empty. Indicates that the TxFIFO is empty. 0 The TxFIFO is not empty. 1 The TxFIFO is empty.

Table 31-20. UART Test Register Field Descriptions (continued)

Name	Description
5 RXEMPTY	RxFIFO empty. Indicates the RxFIFO is empty. 0 The RxFIFO is not empty. 1 The RxFIFO is empty.
4 TXFULL	TxFIFO full. Indicates the TxFIFO is full. 0 The TxFIFO is not full. 1 The TxFIFO is full.
3 RXFULL	RxFIFO full. Indicates the RxFIFO is full. 0 The RxFIFO is not full. 1 The RxFIFO is full.
2–1	Reserved
0 SOFTRST	Software reset. Indicates the status of the software reset ($\overline{\text{SRST}}$ bit of UCR2). 0 Software reset is inactive. 1 Software reset is active.

31.4 Functional Description

31.4.1 Integration Guide

31.4.1.1 Connections of Signals Used During Serial and Infrared Transfer

The signals used in serial and infrared transfers must be connected to the pads at the IC top level. Those interface signals are multiplexed based on UART modes. For instance, DCE/DTE mode is selected by enabling bit DCEDTE in the UFCR, and the signaling reflected on output pin *ipg_uart_dce_oe_o*. As shown in the description of signals in Table 31-21, certain signals are used in DCE mode while they are not used in DTE mode, and vice versa.

Table 31-21. Usage of Serial/Infrared Signal in DCE/DTE

Pin Name	I/O	Used in DCE	Used in DTE
Serial signals			
ipp_uart_rxd	I	X	X
ipp_uart_rxd_ir	I	X	X
ipp_uart_rxd_mux	I	See note.	See note.
ipp_uart_txd	O	X	X
ipp_uart_txd_ir	O	X	X
ipp_uart_txd_mux	O	See note.	See note.
Modem Control Signals			
ipp_uart_cts	O	X	X
ipp_uart_rts	I	X	X
ipp_uart_dsr_dte_i	I		X

Table 31-21. Usage of Serial/Infrared Signal in DCE/DTE (continued)

Pin Name	I/O	Used in DCE	Used in DTE
ipp_uart_dsr_dce_o	O	X	
ipp_uart_dcd_dte_i	I		X
ipp_uart_dcd_dce_o	O	X	
ipp_uart_dtr_dce_i	I	X	
ipp_uart_dtr_dte_o	O		X
ipp_uart_ri_dte_i	I		X
ipp_uart_ri_dce_o	O	X	
Special pins			
ipg_uart_dce_oe_o	O	X	X
Note: If ipp_uart_rxd_mux pin is used, then ipp_uart_rxd and ipp_uart_rxd_ir are not used. In this case ipp_uart_rxd_mux pin receives a MUX of rxd serial and rxd infrared signals. Pin ipp_uart_txd_mux is a MUX of ipp_uart_txd and ipp_uart_txd_ir.			

31.4.1.2 Special Multiplexing at IC Top Level

Table 31-22 shows the special muxing that must be implemented outside the UART to connect its signals to the IC pads. It is assumed the pads are bidirectional. Pad names and their directions are defined in the table. When a pin is not used in the selected mode, it must be tied at its inactive level. This special muxing is required only if the UART is used in both modes (DCE and DTE). For a usage in only one mode, muxing is not necessary; connections between the UART pins and chip pads can be direct.

Table 31-22. Multiplexing at IC Top Level

	DCE		DTE	
UART Pin Name	**IC Pad Connection**	**I/O**	**IC Pad connection**	**I/O**
Serial Signals				
ipp_uart_rxd	RXD	I	TXD	I
ipp_uart_rxd_ir	RXD_IR	I	RXD_IR	I
ipp_uart_rxd_mux	RXD/RXD_IR	I	TXD/RXD_IR	I
ipp_uart_txd	TXD	O	RXD	O
ipp_uart_txd_ir	TXD_IR	O	TXD_IR	O
ipp_uart_txd_mux	TXD/TXD_IR	O	RXD/TXD_IR	O
Modem Control Signals				
ipp_uart_cts	CTS	O	RTS	O
ipp_uart_rts	RTS	I	CTS	I
ipp_uart_dsr_dte_i	Tie to ‘1’	I	DSR	I

Table 31-22. Multiplexing at IC Top Level (continued)

	DCE		DTE	
ipp_uart_dsr_dce_o	DSR	O	Do not connect.	O
ipp_uart_dtr_dce_i	DTR	I	Tie to ‘1’	I
ipp_uart_dtr_dte_o	Do not connect	O	DTR	O
ipp_uart_dcd_dte_i	Tie to ‘1’	I	DCD	I
ipp_uart_dcd_dce_o	DCD	O	Do not connect.	O
ipp_uart_ri_dte_i	Tie to ‘1’	I	RI	I
ipp_uart_ri_dce_o	RI	O	Do not connect.	O

Figure 31-19 and Figure 31-20 show the UART interface connection in different modes.

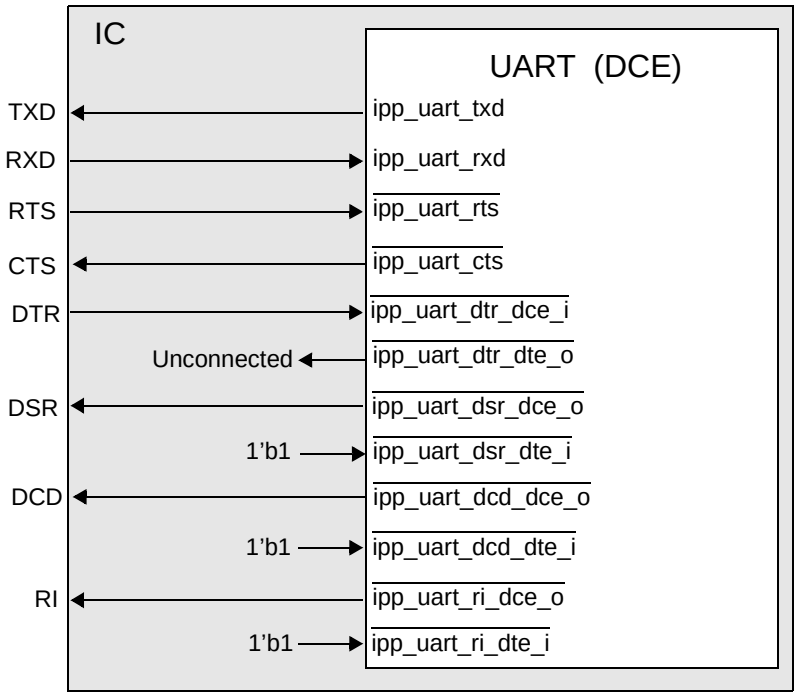


Figure 31-19. UART in DCE Mode

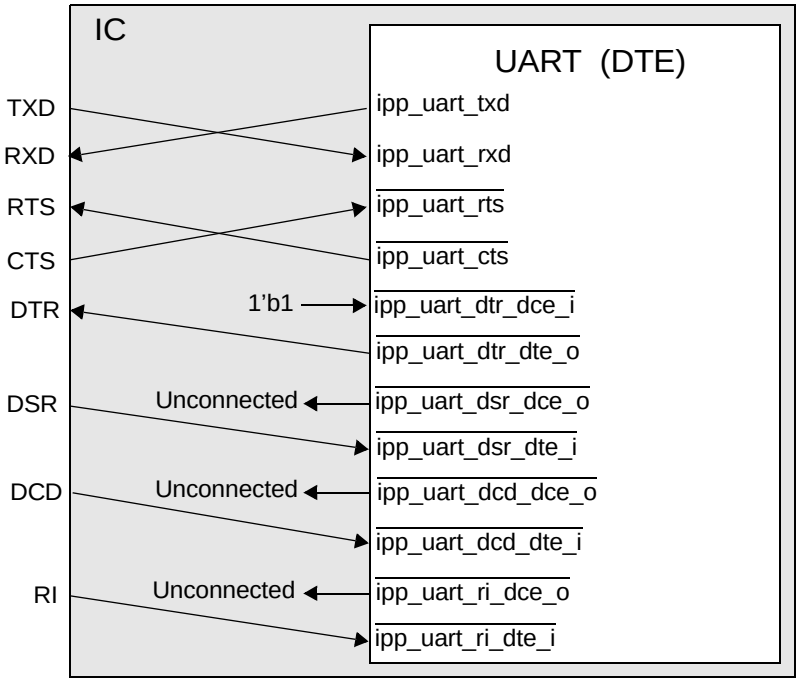


Figure 31-20. UART in DTE Mode

NOTE

The RS-232 Standard (TIA/EIA-232-F) specifies the following:

– For a DCE device, RXD is the output of the serial signal and TXD is the input of the serial signal.
– For a DTE device, RXD is the input of the serial signal and TXD is the output of the serial signal.

31.4.1.3 Special Case of 4-Wire Mode

The UART can send or receive data using only 4 wires. In this case, TXD, RXD, CTS, and RTS are used and the other signals are unused. The unused inputs must be connected to their inactive level, while the unused outputs must be left unconnected.

Typically, 4-wire interface is used for UART operation in DCE mode. Table 31-23 shows top level signal connections and status of inactive signals.

Table 31-23. 4-Wire Connections in DCE Mode and with No IrDA

UART pin name	Connection	I/O
Serial signals		
ipp_uart_rxd	RXD	I
ipp_uart_rxd_ir	Tie to '1'	I
ipp_uart_rxd_mux	Tie to '1'	I

Table 31-23. 4-Wire Connections in DCE Mode and with No IrDA (continued)

UART pin name	Connection	I/O
ipp_uart_txd	TXD	O
ipp_uart_txd_ir	Do not connect.	O
ipp_uart_txd_mux	Do not connect.	O
Modem Control Signals		
ipp_uart_cts	CTS	O
ipp_uart_rts	RTS	I
ipp_uart_dsr_dte_i	Tie to ‘1’.	I
ipp_uart_dsr_dce_o	Do not connect.	O
ipp_uart_dtr_dce_i	Tie to ‘1’.	I
ipp_uart_dtr_dte_o	Do not connect.	O
ipp_uart_dcd_dte_i	Tie to ‘1’.	I
ipp_uart_dcd_dce_o	Do not connect.	O
ipp_uart_ri_dte_i	Tie to ‘1’.	I
ipp_uart_ri_dce_o	Do not connect.	O

31.4.1.4 Special Case of IrDA Mode (2-Wire Connection)

The UART can be used in IrDA mode only. In this case, only 2 pins must be connected to top level. The other pins must be either tied to their inactive level (inputs) or left unconnected (outputs). See Table 31-24 for the summary of the connections.

Table 31-24. 2-Wire Connections in IrDA Mode Only

UART Pin Name	Connection	I/O
Serial signals		
ipp_uart_rxd	Tie to ‘1’.	I
ipp_uart_rxd_ir	RXD_IR	I
ipp_uart_rxd_mux	Tie to ‘1’.	I
ipp_uart_txd	Do not connect.	O
ipp_uart_txd_ir	TXD_IR	O
ipp_uart_txd_mux	Do not connect.	O
Modem Control Signals		
ipp_uart_cts	Do not connect.	O
ipp_uart_rts	Tie to ‘1’.	I
ipp_uart_dsr_dte_i	Tie to ‘1’.	I
ipp_uart_dsr_dce_o	Do not connect.	O

Table 31-24. 2-Wire Connections in IrDA Mode Only (continued)

UART Pin Name	Connection	I/O
ipp_uart_dtr_dce_i	Tie to ‘1’.	I
ipp_uart_dtr_dte_o	Do not connect.	O
ipp_uart_dcd_dte_i	Tie to ‘1’.	I
ipp_uart_dcd_dce_o	Do not connect.	O
ipp_uart_ri_dte_i	Tie to ‘1’.	I
ipp_uart_ri_dce_o	Do not connect.	O

31.4.2 Interrupts and DMA Requests

See Table 31-25 for the lists of all interrupt signals and associated interrupt sources of the UART. See the register description section for explanation of interrupt enable and status.

Table 31-25. Interrupts and DMA

Interrupt Output	Interrupt Enable	Enable Register Location	Interrupt Flag	Flag Register Location
ipi_uart_rx	RRDYEN IDEN DREN RXDSEN ATEN	UCR1 (bit 9) UCR1 (bit 12) UCR4 (bit 0) UCR3 (bit 6) UCR2 (bit 3)	RRDY IDLE RDR RXDS AGTIM	USR1 (bit 9) USR2 (bit 12) USR2 (bit 0) USR1 (bit 6) USR1 (bit 8)
ipi_uart_tx	TXMPTYEN TRDYEN TCEN	UCR1 (bit 6) UCR1 (bit 13) UCR4 (bit 3)	TXFE TRDY TXDC	USR2 (bit 14) USR1 (bit 13) USR2 (bit 3)
ipi_uart_mint	OREN BKEN WKEN ADEN ACIEN ESCI ENIRI AIRINTEN AWAKEN FRAERREN PARERREN RTSDEN RTSEN DTREN (DCE) RI (DTE) DCD (DTE) DTRDEN	UCR4 (bit 1) UCR4 (bit 2) UCR4 (bit 7) UCR1 (bit 15) UCR3 (bit 0) UCR2 (bit 15) UCR4 (bit 8) UCR3 (bit 5) UCR3 (bit 4) UCR3 (bit 11) UCR3 (bit 12) UCR1 (bit 5) UCR2 (bit 4) UCR3 (bit 13) UCR3 (bit 8) UCR3 (bit 9) UCR3 (bit 3)	ORE BRCD WAKE ADET ACST ESCF IRINT AIRINT AWAKE FRAERR PARITYERR RTSD RTSF DTRF RIDELT DCDDELT DTRD	USR2 (bit 1) USR2 (bit 2) USR2 (bit 7) USR2 (bit 15) USR2 (bit 11) USR1 (bit 11) USR2 (bit 8) USR1 (bit 5) USR1 (bit 4) USR1 (bit 10) USR1 (bit 15) USR1 (bit 12) USR2 (bit 4) USR2 (bit 13) USR2 (bit 10) USR2 (bit 6) USR1 (bit 7)
ipd_uart_rx_dmareq	RXDMAEN ATDMAEN IDDMAEN	UCR1 (bit 8) UCR1 (bit 2) UCR4 (bit 6)	RRDY AGTIM IDLE	USR1 (bit 9) USR1 (bit 8) USR2 (bit 12)
ipd_uart_tx_dmareq	TXDMAEN	UCR1 (bit 3)	TRDY	USR1 (bit 13)

31.4.3 Clocking Considerations

31.4.3.1 Min/Max Clock Frequency

The UART module receives 3 clocks, *ipg_clk*, *ipg_clk_s*, and *ipg_perclk*. The Standard Clock Methodology document (v1.4) specifies the roles and constraints on *ipg_clk* and *ipg_clk_s*.

- *ipg_clk* and *ipg_clk_s*—synchronous and have the same frequency.
- *ipg_clk*—the main clock, which must always be running when the UART is enabled. There is an exception in stop mode. See Section 31.4.3.2, "Clocking in Low-Power Modes."
- *ipg_clk_s*—the bus clock, which is active only when there is a bus access (read/write) to the UART registers.

The UART receives also another clock, *ipg_perclk*. This clock is the binary multiplier clock. It must always be running when the UART is sending or receiving characters. This clock has been added to allow frequency scaling on *ipg_clk* (and *ipg_clk_s*) without changing the configuration of BRM (*ipg_perclk* staying at a fixed frequency).

Constraints:

- The clocks *ipg_clk*, *ipg_clk_s* and *ipg_perclk* must be synchronous and their clock trees must be balanced. But *ipg_perclk* frequency is not necessary equal to *ipg_clk* frequency (and *ipg_clk_s*).This specific relationship between *ipg_perclk* and *ipg_clk* is obtained by extracting those clocks from the same source clock but for which different dividers have been applied. The dividers' ratios must always be integer values. With this constraint, the UART receives either *ipg_perclk* and *ipg_clk* rising edges at the same time or, at minimum, separated by a fixed guard-band.

  For example:

  The main source clock frequency is 270 MHz.

  The *ipg_perclk* frequency is fixed at 16.8 MHz (270 MHz /16).

  The *ipg_clk* (and *ipg_clk_s*) frequency can vary from 16.8 MHz to a maximum value. This maximum frequency must always come from an integer division of main source clock (270 MHz/n). It must also be used to constrain the design during synthesis phase.
- At any moment, *ipg_clk* (and *ipg_clk_s*) frequency must be higher or equal to the *ipg_perclk* frequency.

  See Figure 31-21 for examples of working configurations.
- Due to the 16x oversampling of the incoming characters, *ipg_perclk* frequency must always be greater or equal to 16x the maximum baud rate. For example, if max baud rate is 1.875 Mbit/s, *ipg_perclk* must be greater or equal to 1.875 M x 16 = 30 MHz

NOTE

If the architecture of the IC does not require a clock dedicated to BRM, the *ipg_perclk* input pin must receive the same clock as *ipg_clk*. Clock trees between both pins must be balanced.

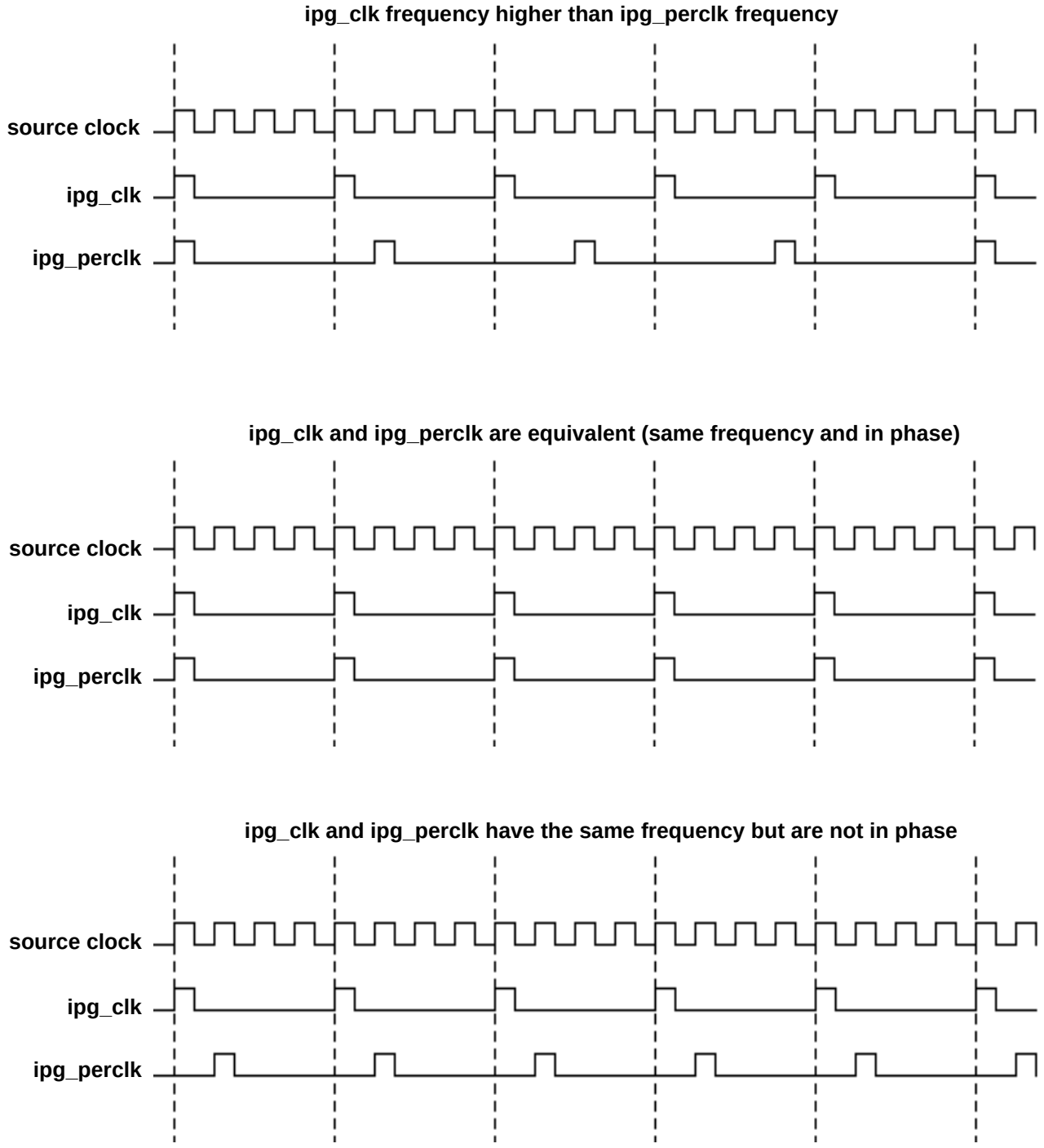


Figure 31-21. Examples of Working Relations Between ipg_clk and ipg_perclk

31.4.3.2 Clocking in Low-Power Modes

The UART supports 2 low-power modes: DOZE and STOP.

In STOP mode (input pin ipg_stop at ‘1’), the UART does not need a clock. In this mode, the UART can wake up the MCU with the asynchronous interrupts. See Section 31.4.10, “UART Operation in Low-Power System States.” An application of this feature is the system needing to be awakened by the arrival of a frame of characters.

- If, before entering stop mode, the software has enabled the RTSDEN interrupt, then RTS changes state (putting at ‘0’ by external device started to send), the asynchronous interrupt wakes up the system, and *ipg_clk* (and *ipg_perclk*) is provided to the UART before the first start bit so that no data is lost.
- If RTS does not change state (already at ‘0’ before entering in stop mode), then wake-up interrupt (AWAKE) is sent at the arrival of the first start bit (on falling edge). In this case, the UART must receive the *ipg_clk* and *ipg_perclk* during the first half of start bit to correctly receive this character. For example, at 115.2 Kbit/s, the UART must receive *ipg_clk* and *ipg_perclk* at maximum 4.3 microseconds after the falling edge of the start bit). If the UART receives *ipg_clk* and *ipg_perclk* too late, the first character is lost, and so should be dropped. Also, if autobaud detection is enabled, the first character will not be correctly received and another autobaud detection will need to be initiated.

In doze mode, the UART behavior is programmable through the DOZE bit (UCR1[1]). If the DOZE bit is set to ‘1’, then the UART is disabled in doze mode, and in consequence, the UART clocks can be switched off (after ensuring the UART is not transmitting nor receiving). However, if the DOZE bit is set to ‘0’, the UART is enabled and must receive *ipg_clk* and *ipg_perclk* (and *ipg_clk_s* during accesses to registers).

31.4.4 General UART Definitions

Definitions of terms that occur in the following discussions are as follows:

- Bit time—the period of time required to serially transmit or receive 1 bit of data (1 cycle of the baud rate frequency).
- Start bit—the bit time of a logic 0 that indicates the beginning of a data frame. A start bit begins with a 1-to-0 transition, and is preceded by at least 1 bit time of logic 1.
- Stop bit—1 bit time of logic 1 that indicates the end of a data frame.
- BREAK—a frame in which all of the data bits, including the stop bit, are logic 0. This type of frame is usually sent to signal the end of a message or the beginning of a new message.
- Frame—a start bit followed by a specified number of data or information bits and terminated by a stop bit. The number of data or information bits depends on the format specified. The format must be the same for the transmitting device and the receiving device. The most common frame format is 1 start bit followed by 8 data bits (least significant bit first) and terminated by 1 stop bit. An additional stop bit and a parity bit also can be included.
- Framing error—an error condition that occurs when the stop bit of a received frame is missing, usually when the frame boundaries in the received bit stream are not synchronized with the receiver bit counter. Framing errors can go undetected if a data bit in the expected stop bit time happens to be a logic 1. A framing error is always present on the receiver side when the transmitter is sending BREAKs. However, when the UART is programmed to expect 2 stop bits and only the first stop bit is received, this is not a framing error by definition.

- Parity error—an error condition that occurs when the calculated parity of the received data bits in a frame does not match the parity bit received on the RXD input. Parity error is calculated only after an entire frame is received.
- Idle—one in NRZ encoding format and selectable polarity in IrDA mode.
- Overrun error—an error condition that occurs when the latest character received is ignored to prevent overwriting a character already present in the UART receive buffer (RxFIFO). An overrun error indicates that the software reading the buffer (RxFIFO) is not keeping up with the actual reception of characters on the RXD input.

31.4.4.1 $\overline{\text{RTS}}$—UART Request To Send

The UART request to send input controls the transmitter. The modem or other terminal equipment signals the UART when it is ready to receive by setting $\overline{\text{RTS}}$ to '0' on the *ipp_uart_rts* pin. Normally, the transmitter waits until this signal is active (low) before transmitting a character, However, when the ignore RTS (IRTS) bit is set, the transmitter sends a character as soon as it is ready to transmit. An interrupt (RTSD) can be posted on any transition of this pin and can wake the MCU from STOP mode on its assertion. When $\overline{\text{RTS}}$ is set to '1' during a transmission, the UART transmitter finishes transmitting the current character and shuts off. The contents of the TxFIFO (characters to be transmitted) remain undisturbed. The operation of this input is the same regardless of whether the UART is in DTE or DCE mode.

31.4.4.2 $\overline{\text{RTS}}$ Edge-Triggered Interrupt

The input to the *ipp_uart_rts* pin can be programmed to generate an interrupt on a selectable edge. See Table 31-26 for a summary of the operation of the $\overline{\text{RTS}}$ edge triggered interrupt (RTSF).

To enable the *ipp_uart_rts* pin to generate an interrupt, set the request to send interrupt enable (RTSEN) bit (UCR2[4]) to 1. Writing 1 to the $\overline{\text{RTS}}$ edge-triggered interrupt flag (RTSF) bit (USR2[4]) clears the interrupt flag. The interrupt can occur on the rising edge, falling edge, or either edge of the $\overline{\text{RTS}}$ input. The request to send edge control (RTEC) field (UCR2[10:9]) programs the edge that generates the interrupt. When RTEC is set to 0x00 and RTSEN = 1, the interrupt occurs on the rising edge (default). When RTEC is set to 0x01 and RTSEN = 1, the interrupt occurs on the falling edge. When RTEC is set to 0x1X and RTSEN = 1, the interrupt occurs on either edge. This is a synchronous interrupt. The RTSF bit is cleared by writing 1 to it. Writing 0 to RTSF has no effect.

Table 31-26. $\overline{\text{RTS}}$ Edge Triggered Interrupt Truth Table

RTS	RTSEN	RTEC [1]	RTEC [0]	RTSF	Interrupt Occurs On…	ipi_uart_mint
X	0	X	X	0	Interrupt disabled	1
1–>0	1	0	0	0	Rising edge	1
0–>1	1	0	0	1	Rising edge	0
1–>0	1	0	1	1	Falling edge	0
0–>1	1	0	1	0	Falling edge	1

Table 31-26. $\overline{\text{RTS}}$ Edge Triggered Interrupt Truth Table (continued)

RTS	RTSEN	RTEC [1]	RTEC [0]	RTSF	Interrupt Occurs On…	ipi_uart_mint
1–>0	1	1	X	1	Either edge	0
0–>1	1	1	X	1	Either edge	0

There is another $\overline{\text{RTS}}$ interrupt that is not programmable. The status bit RTSD asserts the *ipi_uart_mint* interrupt when the $\overline{\text{RTS}}$ delta interrupt enable = 1. This is an asynchronous interrupt. The RTSD bit is cleared by writing 1 to it. Writing 0 to the RTSD bit has no effect.

31.4.4.3 $\overline{\text{DTR}}$—Data Terminal Ready

This signal indicates the general readiness of the data terminal equipment (DTE). This signal is an input in DCE mode and an output in DTE mode. If the connection between the DCE and the DTE is established once, the $\overline{\text{DTR}}$ signal must remain active throughout the whole connection time. The $\overline{\text{DTR}}$ and $\overline{\text{DSR}}$ signals are responsible for establishing the connection. The $\overline{\text{RTS}}$ and $\overline{\text{CTS}}$ signals are responsible for the data transfer and the transfer direction in the case of a half-duplex configuration. The $\overline{\text{DTR}}$ signal is like a “main switch.” If the $\overline{\text{DTR}}$ signal is inactive, the $\overline{\text{RTS}}$ and $\overline{\text{CTS}}$ signals have no effect. In DCE mode, an interrupt (DTRD) can be posted on any transition of this pin and can wake the MCU from STOP mode on its assertion.

31.4.4.4 $\overline{\text{DSR}}$—Data Set Ready

This signal indicates the general readiness of the DCE. This signal is an output in DCE mode and an input in DTE mode. The DCE uses this signal to inform the DTE that it is switched on, has completed all preparations, and can communicate with the DTE. In DTE mode, an interrupt (DTRD) can be posted on any transition of this pin and can wake the MCU from STOP mode on its assertion.

31.4.4.5 $\overline{\text{DTR/DSR}}$ Edge Triggered Interrupt

The $\overline{\text{DTR}}$ input pin (DCE mode) or $\overline{\text{DSR}}$ input pin (DTE mode) can be configured to cause an interrupt on a selectable edge. See Table 31-27 for summary of the operation of the $\overline{\text{DTR/DSR}}$ edge triggered interrupt. To enable the interrupt, set the DTREN bit (UCR3[13]) to ‘1’. Write a “1” to the DTRF bit (USR2[13]) to clear the interrupt flag.

The interrupt can be configured to occur on either the rising, falling, or either edge of the $\overline{\text{DTR/DSR}}$ input. Write to the DPEC[1:0] bits (UCR3[15:14]) to program which edge causes an interrupt. If the bits are set to 00b and DTREN = 1, the interrupt occurs on the rising edge (default). If the bits are set to 01b and DTREN = 1, the interrupt occurs on the falling edge. If the bits are set to 1Xb and DTREN = 1, the interrupt occurs on either edge.

Table 31-27. $\overline{\text{DTR/DSR}}$ Edge Triggered Interrupt Truth Table

$\overline{\text{DTR/DSR}}$	DTREN	DPEC[1]	DPEC[0]	DTRF	Interrupt Occurs On:	ipi_uart_mint
X	0	X	X	0	turned off	1
1->0	1	0	0	0	rising edge	1

Table 31-27. $\overline{\text{DTR/DSR}}$ Edge Triggered Interrupt Truth Table (continued)

$\overline{\text{DTR/DSR}}$	DTREN	DPEC[1]	DPEC[0]	DTRF	Interrupt Occurs On:	ipi_uart_mint
0->1	1	0	0	1	rising edge	0
1->0	1	0	1	1	falling edge	0
0->1	1	0	1	0	falling edge	1
1->0	1	1	X	1	either edge	0
0->1	1	1	X	1	either edge	0

31.4.4.6 $\overline{\text{DCD}}$—Data Carrier Detect

This signal is an output in DCE mode and an input in DTE mode. If used, the DCE device uses this signal to inform the DTE it has detected the carrier signal and the connection will be set up. This signal remains active while the connection remains established. In DTE mode, this input can trigger an interrupt (on *ipi_uart_mint*) on changing state. This is achieved by setting to ‘1’ the interrupt enable bit (DCD, UCR3[9]). The change state is reflected in DCDDELT (USR2[6]). Also, the state of the data carrier detect input is mirrored in the status register DCDIN (USR2[5]).

31.4.4.7 $\overline{\text{RI}}$—Ring Indicator

This signal is an output in DCE mode and an input in DTE mode. If used, the DCE device uses this signal to inform the DTE that a ring just occurred. In DTE mode, this input can trigger an interrupt (on *ipi_uart_mint*) on changing state. This is achieved by setting to ‘1’ the interrupt enable bit (RI, UCR3[8]). The change state is reflected in RIDELT (USR2[10]). Also, the state of the ring indicator input is mirrored in the status register RIIN (USR2[9]).

31.4.4.8 $\overline{\text{CTS}}$—Clear To Send

This output pin serves two purposes. Normally, the receiver indicates that it is ready to receive data by asserting this pin (low). When the $\overline{\text{CTS}}$ trigger level is programmed to trigger at 32 characters received and the receiver detects the valid start bit of the 33 character, it de-asserts this pin. The operation of this output is the same regardless of whether the UART is in DTE or DCE mode.

31.4.4.9 Programmable $\overline{\text{CTS}}$ Deassertion

The $\overline{\text{CTS}}$ output can also be programmed to deassert when the RxFIFO reaches a certain level. Setting the CTS trigger level (UFCR[15:10]) at any value less than 32 deasserts the $\overline{\text{CTS}}$ pin on detection of the valid start bit of the N + 1 character (where N is the trigger level setting). However, the receiver continues to receive characters until the RxFIFO is full.

31.4.4.10 TXD—UART Transmit

This is the transmitter serial output. When operating in normal mode, NRZ-encoded data is output. When operating in infrared mode, a 3/16 bit-period pulse is output for each ‘0’ bit transmitted, and no pulse is output for each ‘1’ bit transmitted. For RS-232 applications, this pin must be connected to an RS-232

transmitter. The operation of this output is the same regardless of whether the UART is in DTE or DCE mode. See Figure 31-22.

31.4.4.10.1 RXD—UART Receive

This is the receiver serial input. When operating in normal mode, NRZ-encoded data is expected. When operating in infrared mode, a narrow pulse is expected for each ‘0’ bit received and no pulse is expected for each ‘1’ bit received. External circuitry must convert the IR signal to an electrical signal. RS-232 applications require an external RS-232 receiver to convert voltage levels. The operation of this input is the same regardless of whether the UART is in DTE or DCE mode. See Figure 31-22.

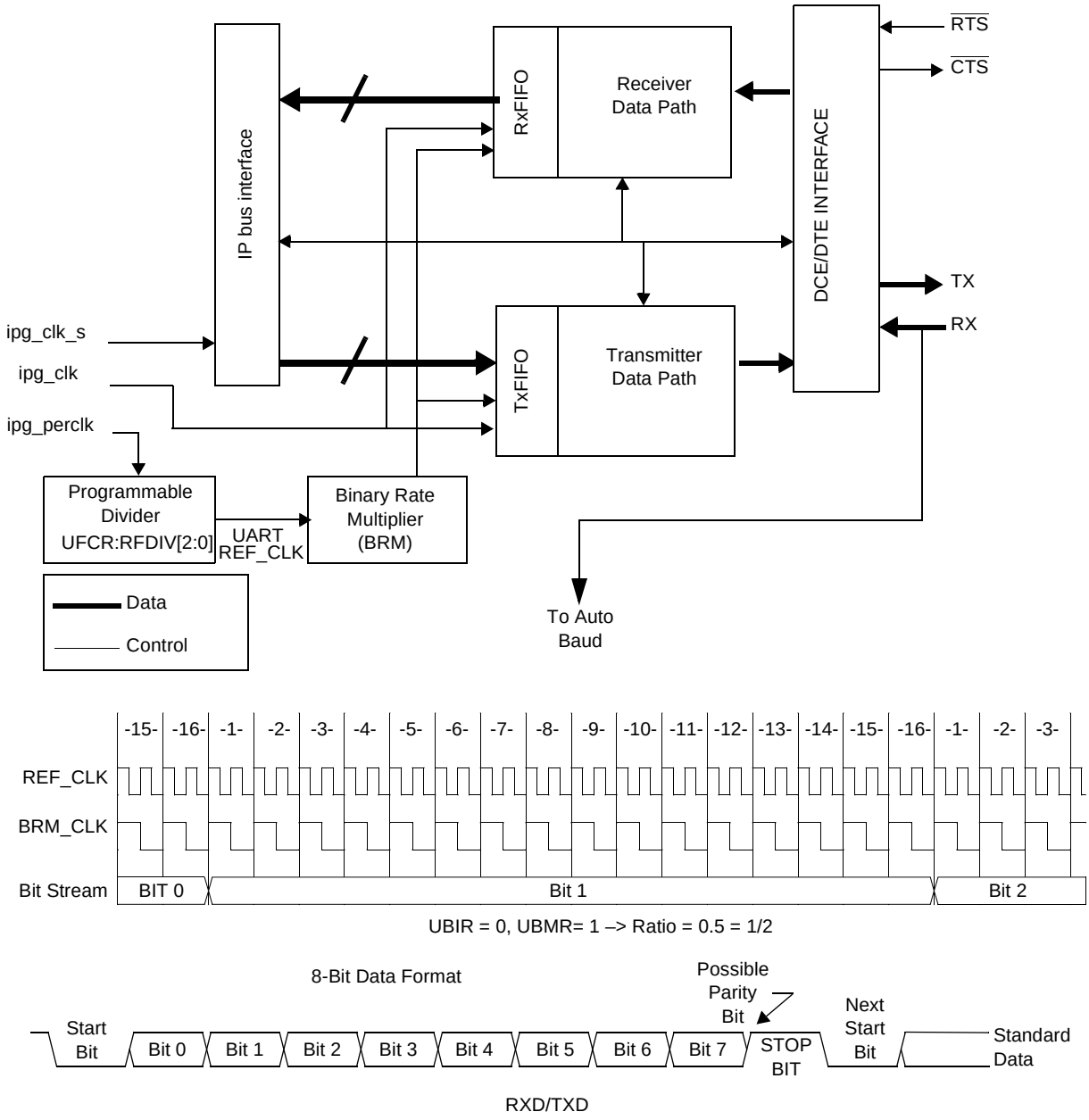


Figure 31-22. UART Simplified Block and Clock Generation Diagrams

31.4.5 Sub-Block Description

31.4.5.1 Transmitter

The transmitter accepts a parallel character from the MCU and transmits it serially. The start, stop, and parity (when enabled) bits are added to the character. When the ignore RTS bit (IRTS) is set, the transmitter sends a character as soon as it is ready to transmit. $\overline{\text{RTS}}$ can be used to provide flow-control of the serial

data. When $\overline{\text{RTS}}$ is set to ‘1’, the transmitter finishes sending the character in progress (if any), stops, and waits for $\overline{\text{RTS}}$ to be set to ‘0’ again. Generation of BREAK characters and parity errors (for debugging purposes) is supported. The transmitter operates from the clock provided by the BRM. Normal NRZ encoded data is transmitted when the IR interface is disabled.

The transmitter FIFO (TxFIFO) contains 32 bytes. The data is written to TxFIFO by writing to the UTXD register with the byte data to the [7:0] bits. The data is written consecutively if the TxFIFO is not full. It is read (internally) consecutively if the TxFIFO is not empty. The TXFULL bit (UTS[4]) can be used to control whether TxFIFO is full or not.

31.4.5.1.1 Transmitter FIFO Empty Interrupt Suppression

The transmitter FIFO empty interrupt suppression logic suppresses the TXFE interrupt between writes to the TxFIFO. When TxFIFO is empty, the software can either send one or several characters. If the software sends one character, it would write the character into the UTXD register, then that character is immediately transferred to the transmitter shift register, assuming the transmitter is already enabled. Without interrupt suppression logic, the TXFE interrupt would be set immediately. But with this logic, for example, the interrupt is set when the last bit of the character has been transmitted before the transmission of the parity bit (if exists) and the stop bit(s).

The suppression logic does not immediately send the TXFE interrupt. It allows the software to write another character to the TxFIFO before the interrupt is asserted.

When the transmitter shift register empties before another character is written to the TxFIFO, the interrupt is asserted. Writing data to the TxFIFO would release the interrupt. The interrupt is asserted with the following conditions:

- System reset
- UART software reset
- When a single character has been written to transmitter FIFO and then the transmitter FIFO and the transmitter shift register become empty until another character is written to the transmitter FIFO
- That the last character in the TxFIFO is transferred to the shift register, when TxFIFO contains two or more characters. See Figure 31-23.

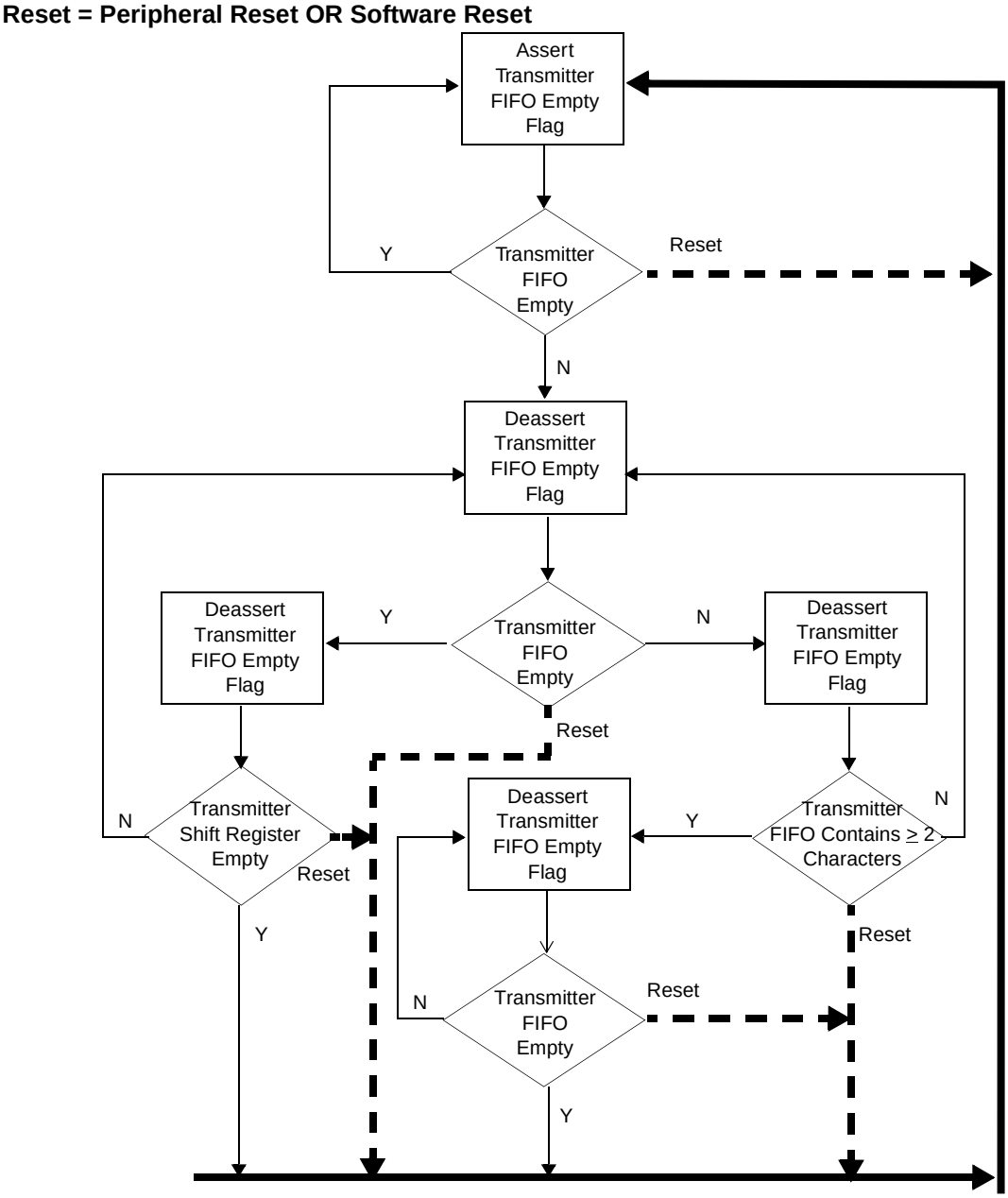


Figure 31-23. Transmitter FIFO Empty Interrupt Suppression Flowchart

31.4.5.1.2 Transmitting a Break Condition

Asserting SNDBRK bit of the UCR1 register forces the transmitter to send a break character (continuous zeros). The transmitter finishes sending the character in progress (if any) before sending BREAK until this bit is reset. The user is responsible for ensuring that this bit is high long enough for it to generate a valid BREAK. The transmitter samples SNDBRK after every bit is transmitted. Following completion of the BREAK transmission, the UART transmits two mark bits. The user can continue to fill the FIFO. Any character remaining is transmitted when the break is terminated.

31.4.5.2 Receiver

See Figure 31-24 for the receiver flowchart. The receiver accepts a serial data stream and converts it into parallel characters. When enabled, it searches for a start bit, qualifies it, and samples the following data bits at the bit-center. Jitter tolerance and noise immunity are provided by sampling at a 16x rate and using voting techniques to clean up the samples. Once the start bit is found, the data bits, parity bit (if enabled), and stop bits (either 1 or 2 depending on user selection) are shifted in. Parity is checked and its status reported in the URXD register when parity is enabled. Frame errors and BREAKs are also checked and reported.

When a new character is ready to be read by the MCU from the RxFIFO, the receive data ready (RDR = USR2[0]) bit is asserted and an interrupt is posted (if DREN = UCR4[0] = 1). If the receiver trigger level is set to 2 (RXTL[5:0] = UFCR[5:0] = 2), and 2 characters have been received into RxFIFO, the receiver ready interrupt flag (RRDY = USR1[9]) is asserted and an interrupt is posted if the receiver ready interrupt enable bit is set (RRDYEN = UCR1[9] = 1). If the UART Receiver Register (URXD) is read once, there is only 1 character in the RxFIFO, the interrupt generated by the RRDY bit is automatically cleared. The RRDY bit is cleared when the data in the RxFIFO falls below the programmed trigger level.

Normal NRZ-encoded data is expected when the IR interface is disabled.The RxFIFO contains 32 halfword entries. Characters received are written consecutively into this FIFO. If the FIFO is full and a 33rd character is received, this character is ignored and the ORE bit (USR2[1]) register is set.

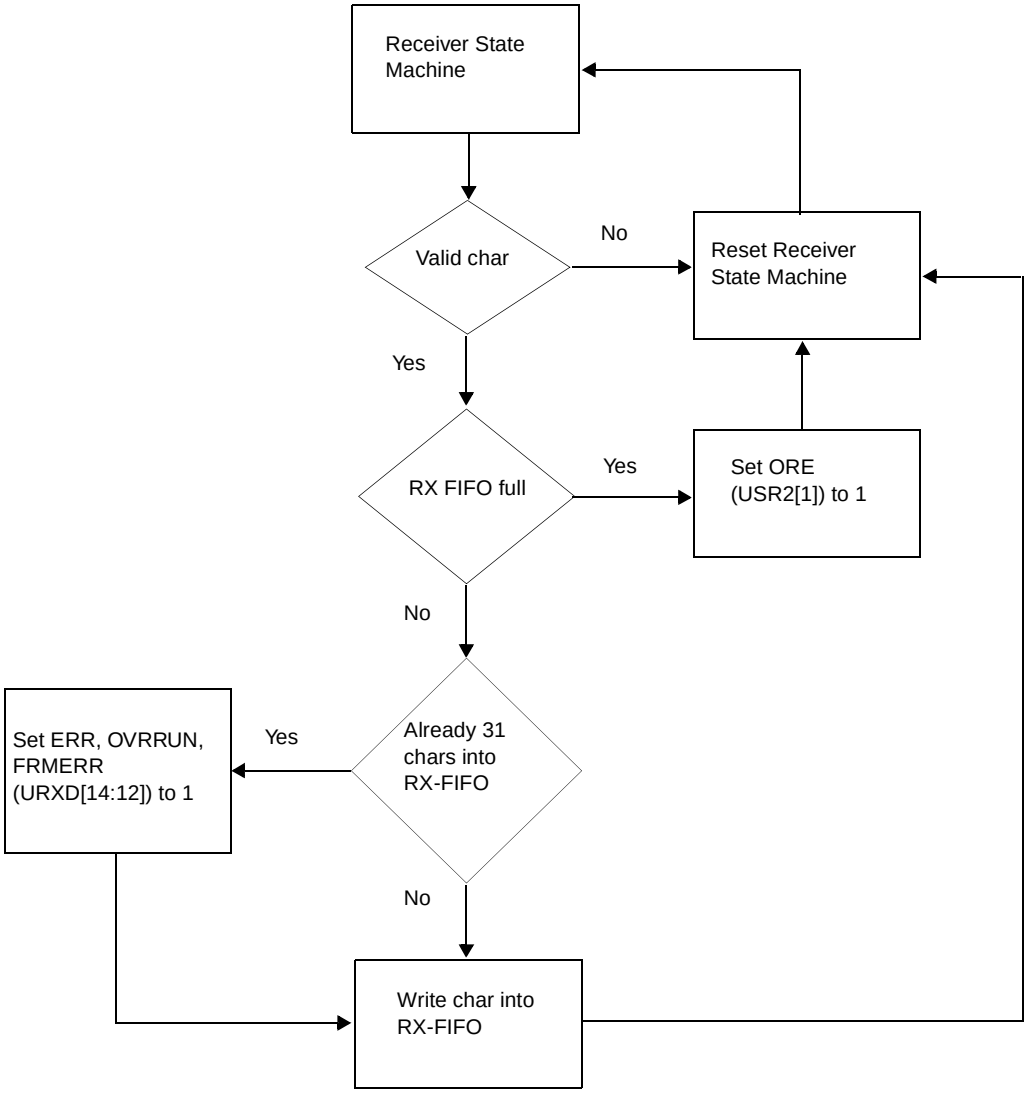


Figure 31-24. Receiver Flowchart

31.4.5.2.1 Idle Line Detect

The receiver logic block includes the ability to detect an idle line. Idle lines indicate the end or the beginning of a message.

For an idle condition to occur, both of the following conditions apply:

- RxFIFO must be empty.
- RXD pin must be idle for more than a configured number of frames (ICD[1:0] = UCR1[11:10]).

When the idle condition detected interrupt enable (IDEN = UCR1[12]) is set and the line is idle for 4 (default), 8, 16, or 32 (maximum) frames, the detection of an idle condition flags an interrupt. When an idle condition is detected, the IDLE (USR2[12]) bit is set. Clear the IDLE bit by writing 1 to it. Writing 0 to the IDLE bit has no effect.

31.4.5.2.2 Idle Condition Detect Configuration

The idle condition detect ICD [1:0] field is located in the UCR1[11:10]. If the bits are set to 00b, RXD must be idle for more than 4 frames before the IDLE bit is asserted. If the bits are set to 01b, RXD must be idle for more than 8 frames before the IDLE bit is asserted. If the bits are set to 10b, RXD must be idle for more than 16 frames before the IDLE bit is asserted. If the bits are set to 11b, RXD must be idle for more than 32 frames before the IDLE bit is asserted (see Table 31-28).

Table 31-28. Detection Truth Table

IDEN	ICD [1]	ICD [0]	IDLE	ipi_uart_rx
0	X	X	0	1
1	0	0	asserted after 4 idle frames	asserted after 4 idle frames
1	0	1	asserted after 8 idle frames	asserted after 8 idle frames
1	1	0	asserted after 16 idle frames	asserted after 16 idle frames
1	1	1	asserted after 32 idle frames	asserted after 32 idle frames
Note: This table assumes that no other interrupt is set at the same time this interrupt is set for the ipi_uart_rx signal. This table shows how this interrupt affects the ipi_uart_rx signal.				

During a normal message, there is no idle time between frames. When all of the information bits in a frame are logic 1s, the start bit ensures that at least one logic 0 bit time occurs for each frame so that the IDLE bit is not asserted.

31.4.5.2.3 Aging Character Detect

The receiver block also includes the possibility to detect when at least one character has been sitting in the RxFIFO for a time corresponding to 8 characters. This aging character capability allows the UART to inform the MCU that there are fewer characters in the RxFIFO than the Rx trigger and no new character has been detected on the RXD line. The aging capability is a timer that starts to count as soon as there is one character in RxFIFO. This counter is reset when either a RxFIFO read is performed or another character has been received in RxFIFO. If neither of those two events occurs, the bit AGTIM (USR1[8]) is set when the counter has measured a time corresponding to 8 characters. The AGTIM bit is cleared by writing a 1 to it. The AGTIM bit can flag an interrupt to the MCU on *ipi_uart_rx* if ATEN (UCR2[3]) has been set.

To summarize, AGTIM is set under the following conditions:

- There is at least one character in RxFIFO.
- No read has occurred on RxFIFO and the RXD line has stayed high for a time corresponding to 8 characters.
- The RxFIFO trigger is not reached (RRDY=0).

31.4.5.2.4 Receiver Wake

The WAKE bit (USR2[7]) is set when the receiver detects a qualified start bit. For this, two conditions must be fulfilled. First, a falling edge on the RXD line must be detected. Second, the RXD line must stay

at low level for more than a half-bit duration. When the wake interrupt enable WKEN (UCR4[7]) bit is enabled, the receiver flags an interrupt ($\overline{ipi\_uart\_mint}$) if the WAKE status bit is set. The WAKE bit is cleared by writing 1 to it. Writing 0 to the WAKE bit has no effect.

When the asynchronous wake interrupt (AWAKE) is enabled (AWAKEN = UCR3[4] = 1), the MCU is in STOP mode, and the UART clocks have been shut-off, then a falling edge detected on the receive pin (RXD) asserts the AWAKE bit (USR1[4]) and the $\overline{ipi\_uart\_mint}$ interrupt to wake the MCU from STOP mode. Re-enable the UART clocks and clear the AWAKE bit by writing 1 to it. Writing 0 to the AWAKE bit has no effect.

In IR mode, if the asynchronous IR WAKE interrupt is enabled (AIRINTEN = UCR3[5] = 1), and if MCU is in STOP mode (meaning the UART clocks are off when MCU in STOP mode), then the detection of a falling edge on the receive pin (RXD_IR) asserts the AIRINT bit (USR1[5]) and the $\overline{ipi\_uart\_mint}$ interrupt. This interrupt wakes the MCU from STOP mode. The software re-enables the UART clocks and clear the AIRINT bit by writing 1 to it. Writing 0 to the AIRINT bit has no effect.

The recommended procedure for programming the asynchronous interrupts is performing the following:

1. Clear the interrupts by writing 1 to the appropriate bit in the UART Status Register 1 (USR1).
2. Poll or enable the interrupt for the receiver IDLE interrupt flag (RXDS) in the USR1. When asserted, the RXDS bit indicates to the software that the receiver state machine is in the idle state, the next state is idle, and the RXD pin is idle (high)

After following the procedure, enable the asynchronous interrupt and enter STOP mode.

31.4.5.2.5 Receiving a BREAK Condition

A BREAK condition is received when the receiver detects all 0s (including a 0 during the bit time of the stop bit) in a frame. The BREAK condition asserts the BRCD bit (USR2[2]) and writes only the first BREAK character to the RxFIFO. Clear the BRCD bit by writing 1 to it. Writing 0 to the BRCD bit has no effect.

Asserting BRCD would generate an interrupt on $\overline{ipi\_uart\_mint}$. The interrupt generation can be masked using the control bit BKEN (UCR4[2]). Receiving a break condition also effects the following bits in the receiver register URXD:

URXD(11) = BRK. While high, this bit indicates that the current char was detected as a break.

URXD(12) = FRMERR. The frame error bit is always be set when BRK is set.

URXD(10) = PRERR. If odd parity was selected, the parity error bit is also be set when BRK is set.

URXD(14) = ERR. The error detect bit indicates that the character present in the receive data field has an error status. This can be asserted by a break.

31.4.5.2.6 Vote Logic

The vote logic block provides jitter tolerance and noise immunity by sampling with respect to a 16x clock (BRM_CLK) and using voting techniques to clean up the samples. The voting is implemented by sampling the incoming signal constantly on the rising edge of the BRM_CLK. See Figure 31-25. The receiver is

provided with the majority vote value, which is 2 out of the 3 samples. Table 31-29 shows examples of majority vote results of the vote logic.

Table 31-29. Majority Vote Results

Samples	Vote
000	0
101	1
001	0
111	1

The vote logic captures a sample on every rising edge of BRM_CLK; however, the receiver uses 16x oversampling to take its value in the middle of the sample character.

The receiver starts to count when the start bit is set; however, it does not capture the contents of the RxFIFO at the time the start bit is set. The start bit is validated when 0s are received for 7 consecutive 1/16 of bit times following the 1-to-0 transition. Once the counter reaches 0xF, it starts counting on the next bit and captures it in the middle of the sampling frame as noted in Table 31-29. All data bits are captured in the same manner. Once the stop bit is detected, the receiver shift register (SIPO_OUT) data is parallel shifted to the RxFIFO.

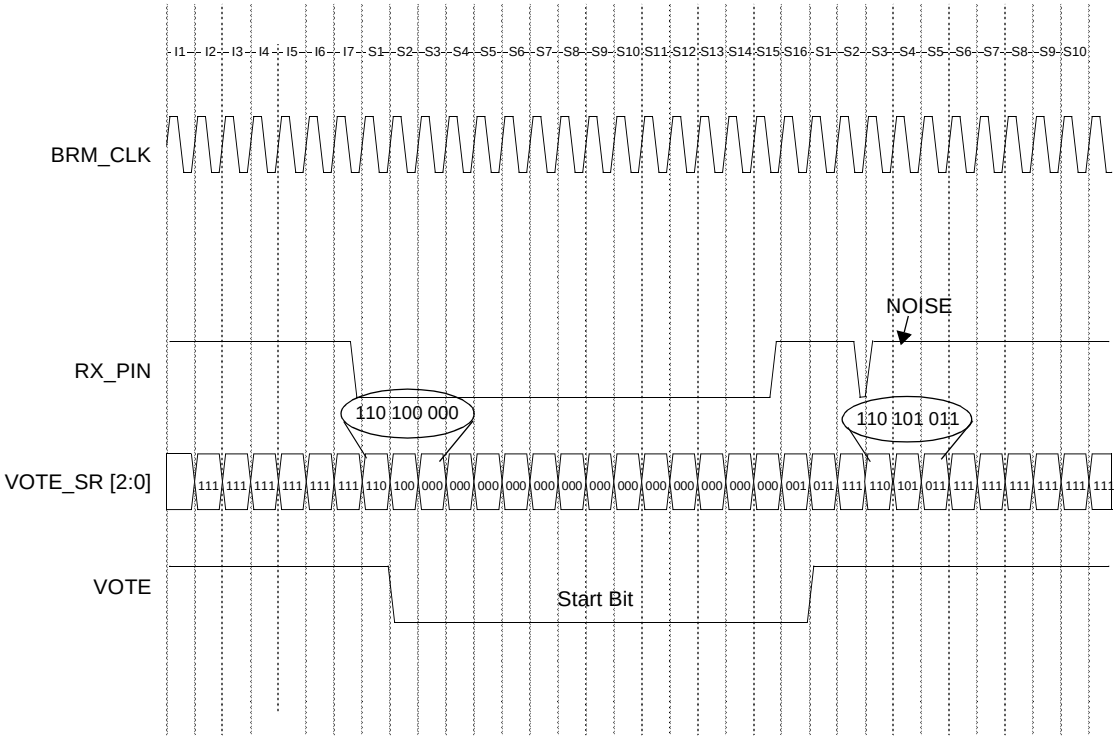


Figure 31-25. Majority Vote Results

A new feature has been recently implemented, which allows resynchronization of the counter on each edge of the RXD line. This is automatic and allows improvement of the immunity of the UART against signal distortion.

There is a special case when the BRM_CLK frequency is too low and is unable to capture a 0 pulse in IrDA. In this case, the software must set the IRSC (UCR4[5]) bit so that the reference clock (after internal divider) is used for the voting logic. The pulse is validated by counting the length of the pulse.

See Section 31.4.9, "Infrared Interface" for more details.

31.4.6 Binary Rate Multiplier (BRM)

The BRM submodule receives *ref_clk* (*ipg_perclk* clock after divider). Form this clock, and with integer and non-integer division, the BRM generates all baud rates. The input and output frequency ratio is programmed in the UART BRM incremental register (UBIR) and UART BRM MOD register (UBMR). The output frequency is divided by the input frequency to produce this ratio. For integer division, set the UBIR = 0x000F and write the divisor to the UBMR register. All values written to these registers must be one less than the actual value to eliminate division by 0 (undefined), and to increase the maximum range of the registers.

Updating the BRM registers requires writing to both registers. The UBIR register must be written to before writing to the UBMR register. If only one register is written to by the software, the BRM continues to use the previous values.

The following examples show how to determine the values that are to be programmed into the UBIR and UBMR for a given reference frequency and desired baud rate. Equation 31-1 can be used to help determine these values:

Frequency and Baud Rate for UBIR and UBMR ***Eqn. 31-1***

$$BaudRate = \frac{RefFreq}{\left(16 \times \frac{UBMR + 1}{UBIR + 1}\right)}$$

With

reference frequency (Hz): UART reference frequency (*ipg_perclk* after RFDIV divider)

baud rate (bit/s): Desired baud rate.

Example 31-1. Integer Division ÷ 21

```
Reference Frequency = 19.44 MHz
UBIR = 0x000F
UBMR = 0x0014
Baud Rate = 925.7 kbit/s
```

NOTE

Observe that each value written to the registers is one less than the actual value.

Example 31-2. Non-Integer Division

```
Reference Frequency = 16 MHz
Desired Baud Rate = 920 Kbits/s
```

$$\frac{UBMR+1}{UBIR+1} = \frac{RefFreq}{16 \times BaudRate} = \frac{16 \times 10^6}{16 \times 920 \times 10^3} = 1.087$$

```
Ratio = 1.087 = 1087 / 1000
UBIR = 999 (decimal)= 0x3E7
UBMR = 1086 (decimal)= 0x43E
Non-Integer Division
Reference Frequency = 25 MHz
Desired Baud Rate = 920 kbit/s
Ratio = 1.69837 = 625 / 368
UBIR = 367 (decimal)= 0x16F
UBMR = 624 (decimal)= 0x270
```

Example 31-3. Non-Integer Division

```
Reference Frequency: 30 MHz
Desired Baud Rate = 115.2 kbit/s
Ratio = 16.276043 = 65153 / 4003
UBIR = 4002 (decimal) = 0x0FA2
UBMR = 65152 (decimal) = 0xFE80
```

31.4.7 Baud Rate Automatic Detection Logic

When the baud rate automatic detection logic is enabled, the UART locks onto the incoming baud rate. To enable this feature, set the automatic detection of baud rate bit (ADBR = UCR1[14] = 1) and write 1 to the ADET bit (USR2[15]) to clear it. When ADET=0 and ADBR = 1, the detection starts. Then, once the beginning of start bit (transition from 1-to-0 of RXD) has been detected, the UART starts a counter (UBRC) working at reference frequency. Once the end of start bit is detected (transition from 0-to-1 of RXD), the value of UBRC–1 is directly copied into UBMR register. The UBIR register is filled with 0x000F.

So, at the end of start bit, registers gets following values:

```
UBRC = number of reference clock periods (after divider) during Start bit.
UBIR = 0x000F
UBMR = UBRC - 1
```

The updated values of the 3 registers can be read.

Table 31-30 shows parameters for baud rate detection and Figure 31-26 shows the baud rate detection protocol diagram.

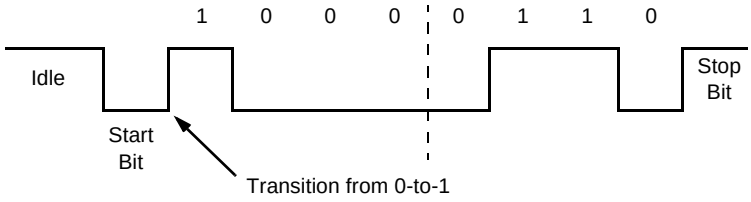
If any of the UART BRM registers are simultaneously written by the baud rate automatic detection logic and by the peripheral data bus, the peripheral data bus would have higher priority.

Table 31-30. Baud Rate Automatic Detection

ADBR	ADET	Baud Rate Detection	ipi_uart_mint
0	X	Manual Configuration	1
1	0	Auto Detection Started	1

Table 31-30. Baud Rate Automatic Detection (continued)

ADBR	ADET	Baud Rate Detection	ipi_uart_mint
1	1	Auto Detection Complete	0
Note: This table assumes that no other interrupt is set at the same time this interrupt is set for the ipi_uart_mint signal.			



Note: LSB transmitted first.

Figure 31-26. Baud Rate Detection Protocol Diagram

31.4.7.1 Baud Rate Automatic Detection Protocol

The receiver must receive an ASCII character “A” or “a” to verify proper detection of the incoming baud rate. When an ASCII character “A” (0x41) or “a” (0x61) is received and no error occurs, the automatic detect baud rate bit is set (ADET=1). And if the interrupt is enabled (ADEN=UCR1[15]=1), an interrupt *ipi_uart_mint* is generated.

When an ASCII character “A” or “a” is not received (because of a bit error or the reception of another character), the auto detection sequence restarts and waits for another 1-to-0 transition.

As long as ADET = 0 and ADBR = 1, the UART continues to try to lock onto the incoming baud rate. Once the ASCII character “A” or “a” is detected and the ADET bit is set, the receiver ignores the ADBR bit and continues normal operation with the calculated baud rate.

The UART interrupt is active ($\overline{ipi\_uart\_mint}$ = 0) as long as ADET = 1 and ADBR = 1. This can be disabled by clearing the automatic baud rate detection interrupt enable bit (ADEN = 0). Before starting an automatic baud rate detection sequence, set ADET = 0 and ADBR = 1.

The RxFIFO must contain the ASCII character “A” or “a” following the automatic baud rate detection interrupt.

The 16-bit UART baud rate count register (UBRC) is reset to 4 and stays at 0xFFFF when an overflow occurs. The UBRC register counts (measures) the duration of start bit. When the start bit is detected and counted, the UBRC retains its value until the next automatic baud rate detection sequence is initiated.

The UBRC counts only when auto detection is enabled.

31.4.7.2 Baud Rate Automatic Detection Protocol Improved[1]

31.4.7.2.1 New Baud Rate Determination

To reduce distortion and noise on the RXD line, the duration of the baud rate measurement has been extended. Previously, as described above, this determination was based on the measurement of the START bit duration. Now, this measurement is based on the duration of START bit + bit0. Bit0 is the first bit following the START bit. The counter that is started at the falling edge of START bit is no longer stopped at the next rising edge (end of START bit), it is stopped at the next falling edge (end of bit 0). As the character sent is always an “A” (41h) or an “a” (61h), this second falling edge is always present and it indicates the end of bit 0. Once this counter is stopped, the result is divided by 2 and used by the BRM to determine the incoming baud rate.

NOTE

The UBRC register contains the result of this division by two and reflects the measurement of the duration of one bit.

31.4.7.2.2 New Autobaud Counter Stopped Bit and Interrupt

A new bit has been added in USR2 register: ACST (USR2[11]). This bit is set immediately after the determination of the baud rate.

- If ADNIMP is not set (default), ACST is set to 1 after the end of bit 0,
- If ADNIMP is set to 1, ACST is set to 1 at the end of START bit.

If ACIEN (UCR3[0]) is set to 1, ACST flags an interrupt on $\overline{ipi\_uart\_mint}$ signal. This interrupt informs the MCU that the BRM has just been set with the result of the bit length measurement. If needed, the MCU can perform a read of UBMR (or UBRC) register and determine the baud rate measured. Then the MCU has the possibility to correct the BRM registers with the nearest standardized baud rate.

NOTE

- ACST is set only if ADBR is set to 1, for example, the UART is autobauding.
- Clear the ACST bit by writing 1 to it. Writing 0 to the ACST bit has no effect.

31.4.8 Escape Sequence Detection

An escape sequence typically consists of 3 characters entered in rapid succession (such as +++). Because these are valid characters by themselves, the time between characters determines if it is a valid escape

1.Several issues have been reported for ICs using the existing autobaud protocol, especially for 57.6 Kbit/s and 115.2 Kbit/s. As a consequence, this protocol has been improved. The old one is still available in the current UART IP, but several modifications can also be used to make this autobaud detection more reliable. If the user wants to keep with the old method, he has to set the bit ADNIMP (UCR3[7]) to 1. If this bit is not set (default), the autobaud improvements protocol is used. Those improvements are grouped in two categories: the new baud rate measurement and the new ACST bit (and associated interrupt).

sequence. Too much time between two of the “+” characters is interpreted as two “+” characters, and not part of an escape sequence.

The software chooses the escape character and writes its value to the UART escape character register (UESC). The software must also enable escape detection feature by setting ESCEN (UCR2[11]) to 1. The hardware compares this value to incoming characters in the RxFIFO. When an escape character is detected, the internal escape timer starts to count. The software specifies a time-out value for the maximum allowable time between 2 successive escape characters. Table 31-31 lists these times. The escape timer is programmable in intervals of 2 ms to a maximum interval of 8.192 s.

Table 31-31. Escape Timer Scaling

UTIM Register	Maximum Time Between Specified Escape Characters
0x000	2 ms
0x001	4 ms
0x002	6 ms
0x003	8 ms
0x004	10 ms
…	…
0F8	498 ms
0F9	500 ms
…	…
9C3	5 s
…	…
FFD	8.188 s
FFE	8.190 s
FFF	8.192 s
Note: To calculate the time interval: (UTIM_Value + 1) × 0.002 = Time_Interval Example: (09C3 + 1) × 0.002 = 5 s.	

The escape sequence detection feature is available for all the reference frequencies. Before using the escape sequence detection, the user must fill the ONEMS register. This 16-bit register must contain the value of the UART internal frequency divided by 1000. The internal frequency is obtained after the UART internal divider, which is applied on *ipg_perclk* clock.

The following is an example of a calculation of frequency:

- The UART BRM (*ipg_perclk*) input clock frequency is 66.5 MHz.
- The UART BRM input clock is divided by 2 with the internal divider: UFCR[9:7] = 3’b100.

Calculation of Frequency for ONEMS Register ***Eqn. 31-2***

$$ONEMS = \frac{66.5 \times 10^6}{2 \times 1000} = 33250 = 81E2h$$

The escape sequence detection interrupt is asserted when the escape sequence interrupt enable (ESCI) bit is set and an escape sequence is detected (ESCF set). Clear the ESCF bit by writing 1 to it. Writing 0 to the ESCF bit has no effect.

31.4.9 Infrared Interface

See Section 31.5, "Programming the IrDA Interface."

31.4.9.1 Generalities

The infrared interface is selected when IREN (UCR1[7]) is set to 1.

The infrared interface is compatible with the IrDA Serial Infrared Physical Layer specification. In this specification, a "zero" is represented by a positive pulse, and a "one" is represented by no pulse (line remains low).

In the UART:

- In TX: For each "zero" to be transmitted, a narrow positive pulse that is 3/16 of a bit time is generated. For each "one" to be transmitted, no pulse is generated (output is low). External circuitry must be provided to drive an infrared LED.
- In RX: When receiving, a narrow negative pulse is expected for each "zero" transmitted while no pulse is expected for each "one" transmitted. (Input is high.)

NOTE

The receive part of the IR block expects to receive an inverted signal compared to the IrDA specification. Circuitry external to the IC transforms the infrared signal to an electrical signal.

The IR interface has an edge-triggered interrupt (IRINT). This interrupt validates a zero bit being received. This interrupt is enabled by writing a "one" to the ENIRI bit.

The behavior of the infrared interface is determined by 3 bits—INVT (UCR3[1]), INVR (UCR4[9]) and IRSC (UCR4[5])

31.4.9.2 Inverted Transmission and Reception Bits (INVT and INVR)

The values of INVT and INVR depend on the IrDA transceiver connected to the TXD_IR and RXD_IR pins of the UART. If this transceiver is not inverting on both paths Tx and Rx, a '0' is represented by a positive pulse and a '1' is represented by no pulse. (The line remains low.) In this case, the bit INVT must be set to '0' and the bit INVR must be set to '1' (because Rx IR block expects an inverted signal).

The user must set INVT=1 and INVR= 0 if both paths of the transceiver are inverting, that is, a ‘0’ is represented as a negative pulse and a ‘1’ is represented by no pulse (line remains high). The transceiver can also be inverting on only one path (Tx or Rx). In this case, INVT and INVR must be together equal to 1 or 0, depending on which path is inverted.

31.4.9.3 Infrared Special Case (IRSC) Bit

The value to apply to IRSC bit is based on 2 parameters: the baud rate and the minimum pulse duration (MPD) of the transceiver. According to IrDA Standard specification, for SIR (Serial IR) baud rates from 2.4 Kbit/s to 115.2 Kbit/s, this nominal pulse duration is equal to 3/16 of a bit duration (at the selected baud rate). For all the baud rates, a minimum pulse duration is also specified. According to the IrDA Standard specification, a ‘0’ is represented by a light pulse, so the IrDA transceiver cannot emit a light pulse shorter than the MPD. For SIR, the MPD is constant and equal to 1.41 µs.

The user must take into account the electrical MPD associated to the transceiver on the receiver path. Typically, this value is 2.0 µs, but for some manufacturers, MPD can go down as low as 1.0 µs.

When UART is in IrDA mode, a zero is not only detected by the state of the RXD_IR line, but also with the duration of the pulse. This pulse duration can be measured with 2 different clocks. In this case, the clock is selected with the IRSC bit.

- If IRSC = 0, the clock used is the BRM clock.
- If IRSC = 1, the clock used is the UART internal clock (UART clock after the divider (RFDIV)).

In normal operation, IRSC = 0. This means at any time, the user must ensure the frequency of BRM_clock is high enough to measure the pulse. In the UART and for IRSC= 0, the pulse must last at least 2 BRM clock cycles.

If this condition is not fulfilled, IRSC must be set to 1.

The following are two examples with the minimum pulse duration equal to the MPD of the IrDA specification (in SIR).

Example 31-4. Calculation of BRM Clock Period (Clock Period < 1.41 µs)

In this instance, the user wants to receive IrDA data at 115.2 Kbit/s. The UBIR and UBMR registers are set to create the BRM_clock with a frequency of 16*baud rate = 16 * 115.2 = 1.843 MHz. At the same time, to correctly detect the pulse, the user must ensure that 2* BRM_clock period is lower than 1.41 us. Confirm the calculation as follows:

BRM_clock period = 1/1843000 = 542 ns

Thus, 2*BRM_clock period = 1.09 µs < 1.41 µs, which is a good result.

Example 31-5. Calculation of BRM Clock Period (Clock Period > 1.41 µs)

In this instance, the user wants to receive IrDA data at 19.2 Kbit/s. The BRM_clock is set to 16*19200 = 307.2 kHz. Confirm if 2* BRM_clock period < 1.41 µs:

BRM_clock period =1/307200 = 3.25 µs

Thus, 2*BRM_clock period = 6.50 µs > 1.41 µs, which is not a good result.

So, in the last case, the BRM clock cannot be used to measure the pulse duration and the user must select the UART internal clock by setting IRSC =1.

NOTE

As for escape character detection, when the IR special case is enabled (IRSC=1), the UART must measure a duration. To do that, the user must fill the ONEMS register. See Section 31.4.8, "Escape Sequence Detection."

31.4.9.4 IrDA Interrupt

Serial infrared mode (SIR) uses an edge triggered interrupt flag IRINT (USR2[8]). When INVR =0, detection of a falling edge on the UART_RXD pin asserts the IRINT bit. When INVR=1, detection of a rising edge on the UART_RXD pin asserts the IRINT bit. When IRINT and ENIRI bits are both asserted, the *ipi_uart_mint* interrupt is asserted. Clear the IRINT bit by writing 1 to it. Writing 0 to the IRINT bit has no effect.

31.4.9.5 Conclusion about IrDA

Before using the UART in IrDA, the baud rate limit must be calculated. This baud rate limit informs the user if IRSC bit must be set or not.

As already described, if IRSC = 0, the following condition must always be fulfilled:

Calculation of Baud Rate: ***Eqn. 31-3***

$$2 \times \mathrm{BRMClockPeriod} < \mathrm{MinPulseDuration}$$

So,

$$\mathrm{BRMClockFrequency} > \frac{2}{\mathrm{MPD}}$$

So, knowing BRM_clock frequency = 16 * baud rate, the following:

$$\mathrm{BaudRate} > \frac{1}{8 \times \mathrm{MinPulseDuration}}$$

The user needs to set IRSC = 0 under the following conditions:

- If minimum pulse duration = 2.5 µs and baud rate > 50 Kbit/s.
- If minimum pulse duration = 2.0 µs and baud rate > 62.5 Kbit/s.
- If minimum pulse duration = 1.41 µs and baud rate > 88.6 Kbit/s.

NOTE

For baud rates lower than the limit, IRSC must be set to1.

31.4.10 UART Operation in Low-Power System States

The serial interface of the UART operates as long as *ipg_clk* and *ipg_perclk* are provided. The RXEN (UCR2[1]), TXEN (UCR2[2]) and UARTEN (UCR1[0]) bits are set by the user and provide software control of low-power modes.

Table 31-32 shows the UART functionality while in hardware controlled low-power modes. These modes are controlled by the signals *ipg_doze* and *ipg_stop*.

Table 31-32. UART Low Power State Operation

	Normal State (ipg_doze = 1'b0 and ipg_stop = 1'b0)	Doze State (ipg_doze = 1'b1)		Stop State (ipg_stop = 1'b1)
		DOZE bit = 0	DOZE bit = 1	
UART-Clock	ON	ON	ON	OFF
UART Serial/IrDA	ON	ON	OFF	OFF

While in doze state (when *ipg_doze* input pin is set to 1'b1), the UART behavior depends on the DOZE (UCR1[1]) control bit. While the DOZE bit is negated, the UART serial interface is enabled. While the system is in the doze state and the DOZE bit is asserted, the UART is disabled. If the doze state is entered with the DOZE bit asserted while the UART serial interface was receiving or transmitting data, it completes the receive/transmit of the current character and signals to the far-end transmitter/receiver to stop sending/receiving. The control/status/data registers do not change when getting into/out of low power modes.

The following UART interrupts wake the MCU processor from STOP mode:

- RTS (RTSD)
- IrDA asynchronous WAKE (AIRINT)
- Asynchronous WAKE (AWAKE)
- RI (RIDELT in DTE mode only)
- DCD (DCDDELT in DTE mode only)
- DTR (DTRD in DCE mode only)
- DSR (DTRD in DTE mode only)

Setting the UARTEN (UCR1[0]) bit to 0 shuts off the receiver and transmitter logic and the associated clocks.

If the UART is used only in transmit mode, UARTEN and TXEN must be set to 1. If the UART is used only in receive mode, UARTEN and RXEN must be set to 1. Setting TXEN or RXEN to 0 allows to save abundant power.

When an asynchronous WAKE (awake) interrupt exits the MCU from STOP mode, ensure that a dummy character is sent first because the first character might not be received correctly.

31.4.11 UART Operation in System Debug State

The bit UTS [11] controls whether the UART responds to the input signal *ipg_debug*, or if it continues to run as normal.

If the UART is programmed to respond to *ipg_debug* the following occur:

1. The UART halts all operations upon detecting the ipg_debug input.
2. A transfer in progress, either to/from a core (using the IP bus interface) or to/from an external device, completes before halting. This means a single byte/word transfer, not an entire FIFO. Receiving of any further data from an external device will be disabled.
3. Internal registers continue to be writable and readable using the IP bus interface. A read leaves the contents unaffected.
4. The RxFIFO is affected in debug mode in the following way:
   a) All writes into the RxFIFO are prevented.
   b) The bit RXDBG (UTS[9]) is used to select the readability of the RxFIFO during debug mode:

      RXDBG = 0: This holds the read pointer at the location it had upon entering debug mode, and URXD register returns only the data value at that location, no matter how many reads are attempted.

      RXDBG = 1, selectable at any time. This allows reading of the characters received in RxFIFO. It will not be possible to re-read previously read locations, nor will it be possible to readjust the read pointer to the value it had prior to entering debug mode.

31.5 Programming the IrDA Interface

31.5.1 High Speed

As an example, the following sequence can be used to program the IrDA interface to send and receive characters at 115.2 Kbit/s.

Assumptions:

- Input UART clock = 90 MHz.
- Internal clock divider = 3 (divide Input UART clock by 3).
- Baud rate = 115.2 Kbit/s.
- IrDA transceiver is not inverting on both channels: for Tx and Rx, a ‘0’ is represented by a positive pulse, and a ‘1’ is represented by no pulse. (The line stays low.)
- Interrupt: Sent to the MCU when 1 character is received into the RxFIFO (RDR).

Register values and programming orders:

```
UCR1 = 0x0085
UCR1[7] = IREN = 1: Enable IR interface
UCR1[0] = UARTEN = 1: Enable UART

UTS = 0x0000
UFCR = 0x0981
TXTL[5:0] = 0x02: Default value
```

```
RFDIV[2:0] = 0x3: Divide Input UART clock by 3 (resulting internal clock is 30 MHz)
RXTL[5:0] = 0x01: Default value

UBIR = 0x0202
UBMR = 0x20BE Baud rate = 115.2 kbit/s with internal clock = 30 MHz

UCR2 = 0x4027
UCR2[14] = IRTS = 1: Ignore level of RTS input signal
UCR2[5] = WS = 1: Characters are 8-bit length
UCR2[2] = TXEN = 1: Enable Rx path
UCR2 [1] = RXEN = 1: Enable Tx path
UCR2[0] = SRST_B = 1: No software reset

UCR3 = 0x0000
UCR4 = 0x8201
CTSTL[5:0] = 0x20: Default value
UCR4[9] = INVR = 1: Inverted Infrared Reception (because IrDA transceiver is not inverting)
UCR4[1] = DREN = 1: To enable RDR interrupt (sent when one char is received)
```

The UART is ready to send a character as soon as there is a write into the UTXD register. And an interrupt is sent to the MCU when a character is received.

31.5.2 Low Speed

This time, the assumptions are the same as for high speed, but the speed is now 9.6 Kbit/s.This baud rate is below the limit (even with a minimum pulse duration of 2.5 µs; thus, IRSC and REF30 must be set to ‘1’.

Assumptions:

- Input UART clock = 90 MHz.
- Internal clock divider = 3 (divide Input UART clock by 3).
- Baud rate = 9.6 Kbit/s.
- IrDA transceiver is not inverting on both channels: for Tx and Rx, a ‘0’ is represented by a positive pulse, and a ‘1’ is represented by no pulse. (The line stays low.)
- Interrupt: Sent to the MCU when 1 character is received into the RxFIFO (RDR).

Register values and programming orders:

```
UCR1 = 0x0085
UCR1[7] = IREN = 1: Enable IR interface
UCR1[0] = UARTEN = 1: Enable UART

UFCR = 0x0981
UFCR[15:10] = TXTL[5:0] = 0x02: Default value
RFDIV[2:0] = 0x3: Divide Input UART clock by 3 (resulting internal clock is 30 MHz)
UFCR[5:0] = RXTL[5:0] = 0x01: Default value

UBIR = 0x00FF

UBMR = 0xC354 Baud rate = 9.6 kbit/s with internal clock = 30 MHz

UCR2 = 0x4027
UCR2[14] = IRTS = 1: Ignore level of RTS input signal
```

```
UCR2[5] = WS = 1: Characters are 8-bit length
UCR2[2] = TXEN = 1: Enable Rx path
UCR2 [1] = RXEN = 1: Enable Tx path
UCR2[0] = SRST_B = 1: No software reset

UCR3 = 0x0004
UCR3[2] = REF30 = 1: Internal UART clock = 30 MHz

UCR4 = 0x8221
UCR4[15:10] = CTSTL[5:0] = 0x20: Default value
UCR4[9] = INVR = 1: Inverted Infrared Reception (because IrDA transceiver is not inverting)
UCR4[5] = IRSC = 1: Because data rate is below the limit and thus the UART internal clock is
used to measure the pulse duration.
UCR4[1] = DREN = 1: To enable RDR interrupt (sent when one char is received)
```

The UART is now ready to send a character as soon as there is a write into the UTXD register. An interrupt is sent to the MCU when a character is received.

Chapter 32
Universal Serial Bus, On-The-Go (USBOTG)

The Universal Serial Bus, On-The-Go (USBOTG) High-Speed module contains all of the functionality required to support three independent USB ports, compatible with the USB 2.0 specification. In addition to the normal USB functionality, the module also provides support for direct connections to on-board USB peripherals and supports multiple interface types for Serial Transceivers.

A block diagram of the USBOTG module is given in Figure 32-1.

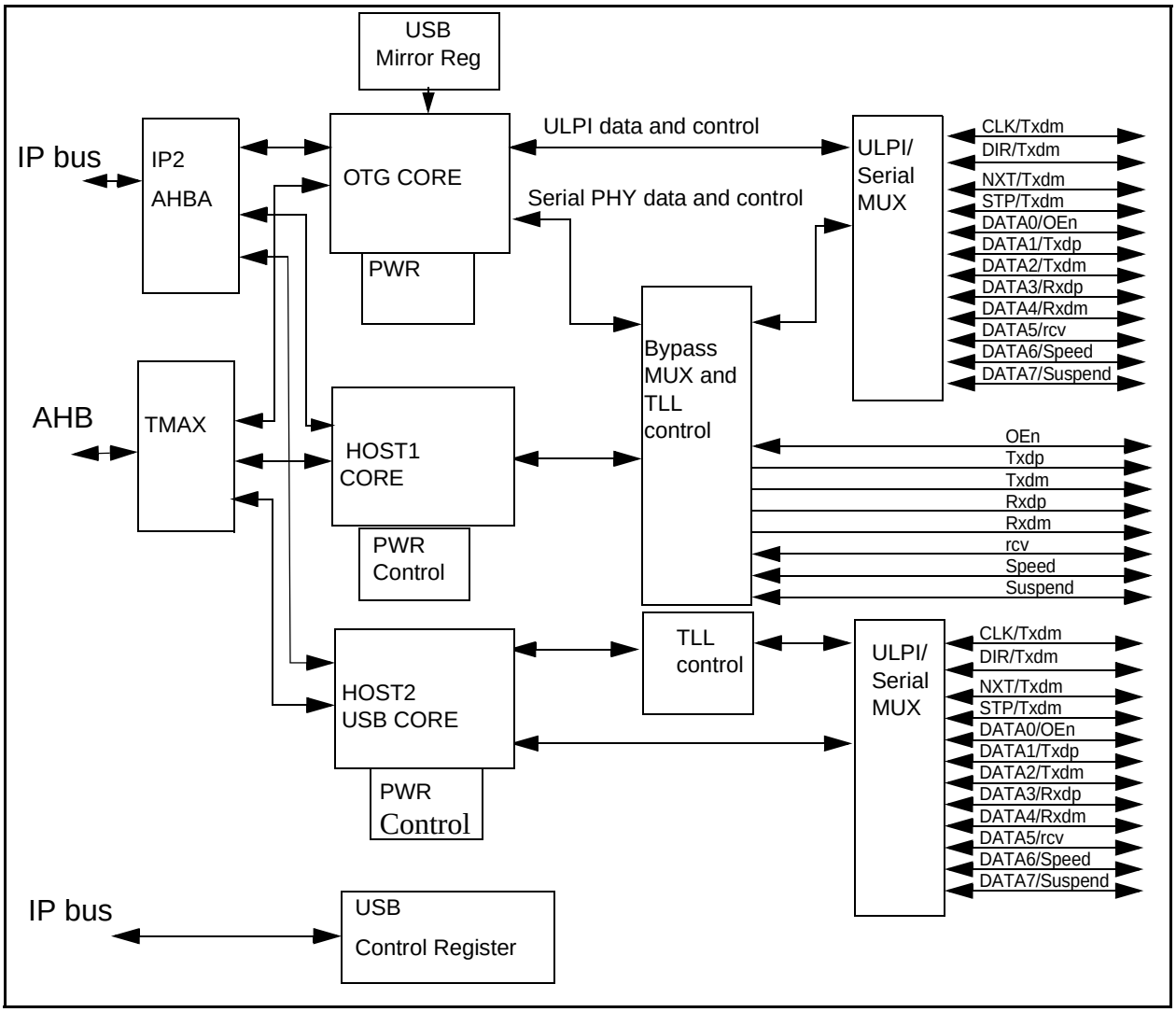


Figure 32-1. USBOTG Block Diagram

32.1 Overview

The USBOTG module provides high performance USB On-The-Go functionality that is compliant with the USB 2.0 specification, the OTG supplement, and the ULPI 1.0 Low Pin Count specification. The module consists of three independent USB cores, each controlling 1 USB port.

In addition to the USB cores, the module provides for a Transceiver-Less Link (TLL) operation on host ports 1 and 2 and allows for routing the OTG transceiver interface to HOST port 1 such that this transceiver can be used to communicate with a USB peripheral connected to host port 1.

The USB module includes the following features:

- Full Speed/Low speed Host only core (HOST 1)
  - Transceiver-Less Link Logic (TLL) for on board connection to a FS/LS USB peripheral
  - Bypass mode to route Host Port 1 signals to OTG I/O port
- High Speed/Full Speed/Low Speed Host Only core (HOST2)
  - High Speed ULPI 1.0 compliant interface
  - Full Speed/Low Speed interface for Serial transceiver
  - TLL function for direct connection to USB peripheral in FS/LS (serial) operation
- High speed OTG core
  - High Speed ULPI 1.0 compliant interface
  - Software configurable for ULPI or serial transceiver interface
  - High Speed (with ULPI transceiver), Full Speed and Low Speed operation in HOST mode
  - High Speed (with ULPI transceiver), and Full Speed operation in Peripheral mode
  - Hardware support for OTG signaling, Session Request Protocol and Host Negotiation Protocol
  - Up to 8 bidirectional endpoints
- Low power modes with local and remote wake-up capability
- Serial PHY interfaces configurable for Bidirectional/Unidirectional and Differential/Single Ended
- Embedded DMA controller.

32.1.1 Modes of Operation

The USB module has two main modes of operation; Normal mode and Bypass mode. Furthermore, the USB interfaces can be configured for High Speed operation (480 Mbps) and/or Full/Low speed operation (12/1.5 Mbps).

This chapter details the configuration options.

32.1.1.1 Operational Modes

32.1.1.1.1 Normal Mode

In normal mode, each USB core controls its corresponding PORT. Each port can work in one or more modes.

- Host Port 1:
  - Full/Low speed only using
  - PHY mode:

    In this mode, an external serial transceiver is connected to the port. This is used for off-board USB connections
  - TLL mode:

    In TLL mode, internal logic is enabled to emulate the functionality of 2 back-to-back connected transceivers. This mode is typically used for on-board USB connections to USB-capable peripherals.
- Host Port 2
  - This port supports ULPI and Serial Transceivers.
  - Serial Interface mode

    PHY mode—for connections using transceivers

    TLL mode—for direct on-board connections to USB peripherals
  - ULPI interface

    ULPI is the low-pin count standard for connecting off-chip High-Speed USB transceivers to a USB device. When the port is configured for ULPI mode, only a ULPI compatible transceiver can be used.
- OTG port:
  - This port requires a transceiver and is intended for off-board USB connections.
  - Serial Interface mode

    In serial mode, a serial OTG transceiver must be connected. The port does not support dedicated signals for OTG signaling. Instead, a transceiver with built-in OTG registers must be used. Typically, the Transceiver registers are accessible over an I2C or SPI interface.
  - ULPI Mode

    It this mode, a ULPI transceiver is connected to the port pins to support High-speed off board USB connections.

32.1.1.1.2 Bypass Mode

Bypass mode affects the operation of the OTG port and HOST port 1. This mode is only available when a serial transceiver is used on the OTG port, and the peripheral device on port 1 is using a TLL connection.

Bypass mode is activated by setting the bypass bit in the USBCONTROL register. In this mode, the USB OTG port connections are internally routed to the USB HOST 1 port, such that the transceiver on the OTG port connects to a peripheral USB device on HOST port 1. The OTG core and the HOST 1 core are disconnected from their ports when bypass is active. The status of the DM and DP inputs for the cores is programmable in the USB Control Register (Figure 32-3).

32.1.1.1.3 Low Power Mode

Each of the three USB cores has an associated power control module that is controlled by the USB core and clocked on a 32 KHz clock. When a USB bus is idle, the transceiver can be placed in low power mode

(suspend), after which the clocks to the USB core can be stopped. The 32 KHz low power clock must remain active as it is needed for wakeup detection.

Either the local CPU or the remote USB Host/Peripheral can initiate a wake-up sequence to resume USB communication.

32.2 Memory Map and Register Definition

The conventions in Figure 32-2 and Table 32-1 serve as a key for the register summary and individual register diagrams.

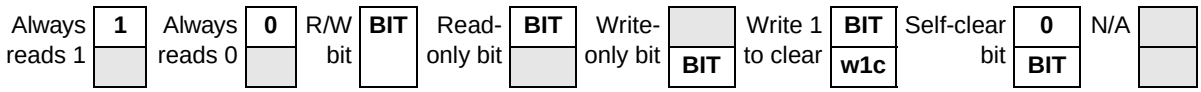


Figure 32-2. Key to Register Fields

Table 32-1 provides a key for register figures and tables and the register summary.

Table 32-1. Register Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
R	Read only. Writing this bit has no effect.
W	Write only.
R/W	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero. (Previously designated slfclr)
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 32-2. USB Module Memory Map

Address	Controller	Register	Access	Reset Value
0x43F8_8000 (ID)	OTG	ID (UOG_ID)	R	0x0000_0000
0x43F8_8004 (HWGENERAL)	OTG	Hardware General (UOG_HWGENERAL)	R	0x0000_0000
0x43F8_8008 (HWHOST)	OTG	Host Hardware Parameters (UOG_HWHOST)	R	0x0000_0000
0x43F8_8010 (HWTXBUF)	OTG	TX Buffer Hardware Parameters (UOG_HWTXBUF)	R	0x0000_0000
0x43F8_8014 (HWRXBUF)	OTG	RX Buffer Hardware Parameters (UOG_HWRXBUF)	R	0x0000_0000
0x43F8_8100 (CAPLENGTH)	OTG	Capability Register Length (UOG_CAPLENGTH)	R	0x40
0x43F8_8100 (CAPLENGTH)	OTG	Host Interface Version (UOG_HCIVERSION)	R	0x0000
0x43F8_8104 (HCSPARAMS)	OTG	Host Control Structural Parameters (UOG_HCSPARAMS)	R	0x0000_0000
0x43F8_8108 (HCCPARAMS)	OTG	Control Capability Parameters (UOG_HCCPARAMS)	R	0x0000_0000
0x43F8_8120 (DCIVERSION)	OTG	Device Interface Version (UOG_DCIVERSION)	R	0x0000
0x43F8_8124 (DCCPARAMS)	OTG	Device Controller Capability Parameters (UOG_DCCPARAMS)	R	0x0000_0000
0x43F8_8140 (USBCMD)	OTG	USB Command Register (UOG_USBCMD)	RW	0x0000_0000
0x43F8_8144 (USBSTS)	OTG	USB Status Register (UOG_USBSTS)	RW	0x0000_0000
0x43F8_8148 (USBINTR)	OTG	Interrupt Enable Register (UOG_USBINTR)	RW	0x0000_0000
0x43F8_814C (FRINDEX)	OTG	USB Frame Index (UOG_FRINDEX)	RW	0x– – – –_– – – –
0x43F8_8154 (PERIODICLISTBASE)	OTG	Host Controller Frame List Base Address (UOG_PERIODICLISTBASE)	RW (32-bit only)	0x0000_0000
0x43F8_8158 (ASYNCLISTADDR)	OTG	Host Controller Next Asynch. Address (UOG_ASYNCLISTADDR)	RW (32-bit only)	0x0000_0000
0x43F8_8160 (BURSTSIZE)	OTG	Host Controller Embedded TT Asynch. Buffer Status (UOG_BURSTSIZE)	RW (32-bit only)	0x0000_0000
0x43F8_8164 (TXFILLTUNING)	OTG	TX FIFO Fill Tuning (UOG_TXFILLTUNING)	RW (32-bit only)	0x0000_0000

Table 32-2. USB Module Memory Map (continued)

Address	Controller	Register	Access	Reset Value
0x43F8_8170 (ULPI)	OTG	ULPI Viewport (ULPIVIEW)	RW	0x0000_0000
	OTG	Config Flag (UOG_CFGFLAG)	R	0x0000_0000
0x43F8_8184 (PORTSCx)	OTG	Port Status and Control (UOG_PORTSC1)	RW	0x0000_0000
0x43F8_81A4 (OTGSC)	OTG	On-The-Go Status and control (UOG_OTGSC)	RW	0x0000_0000
0x43F8_81A8 (USBMODE)	OTG	USB Device Mode (UOG_USBMODE)	RW	0x0000_0000
0x43F8_81AC (ENPTSETUPSTAT)	OTG	Endpoint Setup Status (UOG_ENDPTSETUPSTAT)	RW	0x0000_0000
0x43F8_81B0 (ENDPTPRIME)	OTG	Endpoint Initialization (UOG_ENDPTPRIME)	RW	0x0000_0000
0x43F8_81B4 (ENDPTFLUSH)	OTG	Endpoint De-Initialize (UOG_ENDPTFLUSH)	RW	0x0000_0000
0x43F8_81B8 (ENDPTSTAT)	OTG	Endpoint Status (UOG_ENDPTSTAT)	R	0x0000_0000
0x43F8_81BC (ENDPTCOMPLETE)	OTG	Endpoint Complete (UOG_ENDPTCOMPLETE)	RW	0x0000_0000
0x43F8_81C0 (ENDPTCTRL0)	OTG	Endpoint Control0 (ENDPTCTRL0)	RW	0x0000_0000
0x43F8_81C4 (ENDPTCTRL1)	OTG	Endpoint Control1 (ENDPTCTRL1)	RW	0x0000_0000
0x43F8_81C8 (ENDPTCTRL2)	OTG	Endpoint Control2 (ENDPTCTRL2)	RW	0x0000_0000
0x43F8_81CC (ENDPTCTRL3)	OTG	Endpoint Control3 (ENDPTCTRL3)	RW	0x0000_0000
0x43F8_81D0 (ENDPTCTRL4)	OTG	Endpoint Control4 (ENDPTCTRL4)	RW	0x0000_0000
0x43F8_81D4 (ENDPTCTRL5)	OTG	Endpoint Control5 (ENDPTCTRL5)	RW	0x0000_0000
0x43F8_81D8 (ENDPTCTRL6)	OTG	Endpoint Control6 (ENDPTCTRL6)	RW	0x0000_0000
0x43F8_81DC (ENDPTCTRL7)	OTG	Endpoint Control07(ENDPTCTRL7)	RW	0x0000_0000
0x43F8_8200	Host1	Host 1 ID (UH1_ID)	R	
0x43F8_8204	Host1	Hardware General (UH1_HWGENERAL)	R	
0x43F8_8208	Host1	Host Hardware Parameters (UH1_HWHOST)	R	

Table 32-2. USB Module Memory Map (continued)

Address	Controller	Register	Access	Reset Value
0x43F8_8210	Host1	TX Buffer Hardware Parameters (UH1_HWTXBUF)	R	
0x43F8_8214	Host1	RX Buffer Hardware Parameters (UH1_HWRXBUF)	R	
0x43F8_8300	Host1	Capability Register Length (UH1_CAPLENGTH)	R	
0x43F8_8302	Host1	Host Interface Version (UH1_HCIVERSION)	R	
0x43F8_8304	Host1	Host Control Structural Parameters (UH1_HCSPARAMS)	R	
0x43F8_8308	Host1	Control Capability Parameters (UH1_HCCPARAMS)	R	
0x43F8_8340	Host1	USB Command Register (UH1_USBCMD)	RW	
0x43F8_8344	Host1	USB Status Register (UH1_USBSTS)	RW	
0x43F8_8348	Host1	Interrupt Enable Register (UH1_USBINTR)	RW	
0x43F8_834C	Host1	USB Frame Index (UH1_FRINDEX)	RW	
0x43F8_8354	Host1	Host Controller Frame List Base Address (UH1_PERIODICLISTBASE)	RW (32-bit only)	
0x43F8_8358	Host1	Host Controller Next Asynch. Address (UH1_ASYNCLISTADDR)	RW (32-bit only)	
0x43F8_8360	Host1	Host Controller Embedded TT Asynch. Buffer Status (UH1_BURSTSIZE)	RW (32-bit only)	
0x43F8_8364	Host1	TX FIFO Fill Tuning (UH1_TXFILLTUNING)	RW (32-bit only)	
0x43F8_8380	Host1	Reserved	R	
0x43F8_8384	Host1	Port Status and Control (UH1_PORTSC1)	RW	
0x43F8_83A8	Host1	USB Device Mode (UH1_USBMODE)	RW	
0x43F8_8400	Host2	ID (UH2_ID)	R	
0x43F8_8404	Host2	Hardware General (UH2_HWGENERAL)	R	
0x43F8_8408	Host2	Host Hardware Parameters (UH2_HWHOST)	R	

Table 32-2. USB Module Memory Map (continued)

Address	Controller	Register	Access	Reset Value
0x43F8_8410	Host2	TX Buffer Hardware Parameters (UH2_HWTXBUF)	R	
0x43F8_8414	Host2	RX Buffer Hardware Parameters (UH2_HWRXBUF)	R	
0x43F8_8500	Host2	Capability Register Length (UH2_CAPLENGTH)	R	
0x43F8_8502	Host2	Host Interface Version (UH2_HCIVERSION)	R	
0x43F8_8504	Host2	Host Control Structural Parameters (UH2_HCSPARAMS)	R	
0x43F8_8508	Host2	Control Capability Parameters (UH2_HCCPARAMS)	R	
0x43F8_8540	Host2	USB Command Register (UH2_USBCMD)	RW	
0x43F8_8544	Host2	USB Status Register (UH2_USBSTS)	RW	
0x43F8_8548	Host2	Interrupt Enable Register (UH2_USBINTR)	RW	
0x43F8_854C	Host2	USB Frame Index (UH2_FRINDEX)	RW	
0x43F8_8554	Host2	Host Controller Frame List Base Address (UH2_PERIODICLISTBASE)	RW (32-bit only)	
0x43F8_8558	Host2	Host Controller Next Asynch. Address (UH2_ASYNCLISTADDR)	RW (32-bit only)	
0x43F8_8560	Host2	Host Controller Embedded TT Asynch. Buffer Status (UH2_BURSTSIZE)	RW (32-bit only)	
0x43F8_8564	Host2	TX FIFO Fill Tuning (UH2_TXFILLTUNING)	RW (32-bit only)	
0x43F8_8570	Host2	ULPI Viewport (ULPIVIEW)	RW	
0x43F8_8580	Host2	Reserved	R	
0x43F8_8584	Host2	Port Status and Control (UH2_PORTSC1)	RW	
0x43F8_85A8	Host2	USB Device Mode (UH2_USBMODE)	RW	
0x43F8_8600 (USBCONTROL)		USB Control Register (USB_CTRL)	RW	
0x43F8_8604 (OTGMIRROR)		USB OTG Mirror Register (USB_OTG_MIRROR)	RW	

32.2.1 Register Descriptions

This section describes only the registers that are additional to the USB register set.

32.2.1.1 USBCONTROL—USB Control Register

The USB control register controls the integration specific features of the USB module. These features are not directly related to the USB functionality, but control special features, interfacing on the USB ports, as well as power control and wake-up functionality.

0x43F8_8600 (USBCONTROL) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	OWIR	OSIC		OUIE	OWIE	RxDB		OPM	H2WIR	H2SIC		H2UIE	H2WIE	0	0	H2PM
W																
Reset	—	P	P	0	0	0	0	0	—	P	P	0	0	—	—	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	H1WIR	H1SIC		0	H1WIE	RxDp		H1PM	0	0	H2DT	H1DT	0	0	0	BPE
W																
Reset	—	P	P	—	0	0	0	0	—	—	0	0	—	0	0	0

P = Preset value taken from Preset input signals

Figure 32-3. USB Control Register

Table 32-3. USB Control Register Field Descriptions

Field	Description
31 OWIR	OTG Wake up Interrupt Request. This bit indicates that a wake-up interrupt request is received on the OTG port. This bit is cleared by disabling the wake-up interrupt. 1 Wake-up Interrupt Request received 0 No Wake-up detected
30–29 OSIC	OTG Serial Interface Configuration. Controls the interface type of the OTG port when used with a serial transceiver. This bit field allows for configuring the serial interface for Single Ended or Differential operation combined with Bidirectional or Unidirectional operation. The reset value of OSIC depends on the state of the signals “ADD” and “ADD during reset.” Bit 30 Bit 29 Interface Type 0 0 Differential/Unidirectional (6-wire) 0 1 Differential/Bidirectional (4-wire) 1 0 Single Ended/Unidirectional (6-wire) 1 1 Single Ended/Bidirectional (3-wire)
28 OUIE	OTG ULPI interrupt enable. Controls whether or not interrupts from the ULPI transceiver will trigger the wake-up logic. This bit is only meaningful when a ULPI transceiver is selected. 1 ULPI transceiver interrupts activate the wake-up logic 0 ULPI transceiver interrupts are ignored by the wakeup logic.

Table 32-3. USB Control Register Field Descriptions (continued)

Field	Description
27 OWIE	OTG Wake-up Interrupt Enable. This bit enables or disables the OTG wake-up interrupt. Disabling the interrupt also clears the Interrupt request bit. Wake-up interrupt enable should be turned off after receiving a wake-up interrupt and turned on again prior to going in suspend mode 1 Interrupt Enabled 0 Interrupt Disabled
26–25 OBPVAL	OTG Bypass Value. This field contains the status of the RxDP and RxDM inputs to the OTG core when Bypass mode is enabled. Bit 26 controls RxDB, bit 25 controls RxDm.
24 OPM	OTG Power Mask. The power mask bit controls whether or not the external Vbus Power and OverCurrent detection are active for the OTG port. 1 The USBPWR and OC pins are not used by the OTG core. 0 The USBPWR pin will assert with the OTG core's Vbus power Enable and the assertion of the OC input will be reported to the OTG core.
23 H2WIR	Host 2 Wake-up Interrupt Request. Indicates a pending Wake-up request on Host port 2. This bit is cleared by disabling the interrupt. The interrupt must be disabled for at least 2 clock cycles of the standby clock. 1 Wake-up interrupt received 0 No Wake-up interrupt received
22–21 H2SIC	Host 2 Serial Interface Configuration. Controls the interface type of the Host 2 port when used with a serial transceiver. This bit field allows for configuring the serial interface for Single Ended or Differential operation combined with Bidirectional or Unidirectional operation. Bit 22 Bit 21 Interface Type 0 0 Differential/Unidirectional (6-wire) 0 1 Differential/Bidirectional (4-wire) 1 0 Single Ended/Unidirectional (6-wire) 1 1 Single Ended/Bidirectional (3-wire)
20 H2UIE	Host 2 ULPI interrupt enable. Controls whether or not interrupts from the ULPI transceiver will trigger the wake-up logic. This bit is only meaningful when a ULPI transceiver is selected. 1 ULPI transceiver interrupts activate the wake-up logic 0 ULPI transceiver interrupts are ignored by the wakeup logic.
19 H2WIE	Host 2 Wake-up Interrupt Enable. This bit enables or disables the Host 2 wake-up interrupt. Disabling the interrupt also clears the Interrupt request bit. Wake-up interrupt enable should be turned off after receiving a wake-up interrupt and turned on again prior to going in suspend mode 1 Interrupt Enabled 0 Interrupt Disabled
18–17	Reserved
16 H2PM	Host 2 Power Mask. The power mask bit controls whether or not the external Vbus Power and OverCurrent detection are active for the Host 2 port. 1 The USBPWR and OC pins are not used by the Host 2 core. 0 The USBPWR pin will assert with the Host 2 core's Vbus power Enable and the assertion of the OC input will be reported to the Host 2 core.

Table 32-3. USB Control Register Field Descriptions (continued)

Field	Description
15 H1WIR	Host 1 Wake-up Interrupt Request. Indicates a pending Wake-up request on Host port 1. This bit is cleared by disabling the interrupt. The interrupt must be disabled for at least 2 clock cycles of the standby clock. 1 Wake-up interrupt received 0 No Wake-up interrupt received
14–13 H1SIC	Host 1 Serial Interface Configuration. Controls the interface type of the Host 1 port when used with a serial transceiver. This bit field allows for configuring the serial interface for Single Ended or Differential operation combined with Bidirectional or Unidirectional operation. Bit 14 Bit 13 Interface Type 0 0 Differential/Unidirectional (6-wire) 0 1 Differential/Bidirectional (4-wire) 1 0 Single Ended/Unidirectional (6-wire) 1 1 Single Ended/Bidirectional (3-wire)
12	Reserved
11 H1WIE	Host 1 Wake-up Interrupt Enable. This bit enables or disables the Host 1 wake-up interrupt. Disabling the interrupt also clears the Interrupt request bit. Wake-up interrupt enable should be turned off after receiving a wake-up interrupt and turned on again prior to going in suspend mode 1 Interrupt Enabled 0 Interrupt Disabled
10–9 H1BPVAL	HOST 1 Bypass Value. This field contains the status of the RxDP and RxDM inputs to the HOST1 core when Bypass mode is enabled. Bit 10 controls RxDB, bit 9 controls RxDm.
8 H1PM	Host 1 Power Mask. The power mask bit controls whether or not the external Vbus Power and OverCurrent detection are active for the Host 1 port. 1 The USBPWR and OC pins are not used by the Host 1 core. 0 The USBPWR pin will assert with the Host 1core's Vbus power Enable and the assertion of the OC input will be reported to the Host 1 core.
7–6	Reserved
5 H2DT	Host 2 TLL Disable. This bit controls whether or not the Transceiver less Link Logic is enabled for the serial interface of Host Port 2. 1 TLL is disabled 0 TLL is enabled
4 H1DT	Host 1 TLL disable. This bit controls the TLL logic for Host Port 1 1 TLL is disabled 0 TLL is enabled
3–1	Reserved
0 BPE	Bypass Enable. This bit enables/disables the USB Bypass function. 1 Bypass active — USB signals from Host port 1 are routed to the OTG port. 0 Bypass inactive — Normal mode operation.

32.2.1.2 OTGMIRROR—OTG Port Mirror Register

The OTG port is designed for operation with an external OTG transceiver. When a ULPI transceiver is in use, all OTG signaling is communicated over the ULPI data bus as described in the ULPI specification. However, when a serial transceiver is used, the interface for OTG signaling is not standardized. Most OTG transceivers use a serial interface like I2C or SPI to transfer the OTG signaling back to the CPU and/or USB core. In this case, the USB core has no direct connection the OTG signals in the transceiver.

The OTGMIRROR register provides a soft interface between the OTG signals in the transceiver and the OTG signal inputs to the USB core. The USB driver software is responsible for reading the OTG status registers in the transceiver over the serial interface and set the bits accordingly in the OTGMIRROR register, such that the USB controller knows the status of the transceiver.

The USB driver should be designed such that the latency requirements as defined in the USB 2.0 OTG supplement specification are met. Figure 32-4 shows the register; Table 32-4 provides its field descriptions.

0x43F8_8604 (OTGMIRROR) Access: User Read/Write

	7	6	5	4	3	2	1	0
R	0	0	0	SESEND	VBUSVLD	BSESVLD	ASESVLD	IDDIG
W								
RESET	—	—	—	0	0	0	0	0

Figure 32-4. OTG Mirror Register (OTGMIRROR)

Table 32-4. OTGMIRROR Field Descriptions

Name	Description
7–5	Reserved
4 SESEND	B device Session End. This bit is set by the USB driver when the PHY reports a Session End condition. 1 Session End (0.2V < Vbus < 0.8V) 0 Session Active
3 VBUSVLD	Vbus Valid. The USB driver sets this bit when the transceiver reports Vbus Valid. 1 Vbus is Valid (Vbus > 4.4V) 0 Vbus Invalid (Vbus < 4.4V)
2 BSESVLD	B Session Valid. B session valid should be set when the transceiver reports B-session Valid. 1 B Session is Valid (0.8V < Vbus < 4.0V) 0 B Session is not valid (Vbus < 0.8V)
1 ASESVLD	A session Valid. This bit must be set when a valid ‘A-Session’ level is detected on Vbus. 1 A Session is Valid (0.8V < Vbus < 2.0V) 0 Session is not valid for A-device.
0 IDIDG	OTG ID-pin Status. This bit indicates to the USB core whether it should operate as A-device or as B-device 1 ID pin is High — Operate as B-device 0 ID pin is Low — Operate as A-device

32.3 Functional Description

This section describes the functionality and the topology of the different building blocks of the USB module.

32.3.1 USB HOST Controller 1

Host Controller 1 is configured for Full Speed/Low Speed operation only.

32.3.1.1 Host Controller 1 to Host Port 1 Interface

The Host 1 core USB signals do not connect directly to the HOST1 I/O pins. Instead, the signals pass through the Bypass MUX to allow for additional functionality on Host Port1. See Section 32.3.6, “USB Bypass Mode” for details.

In addition to bypass multiplexing, this MUX also provides interface type conversion for Host core 1. This type conversion is independent of the MUX state.

Depending on the selected interface type, the USB port signals have the functionality as described in Table 32-5.

Table 32-5. Host Port 1 Pin Functions

	Single-Ended Mode			Differential Mode		
Signal	**Unidir**	**Bidir**		**Unidir**	**Bidir**	
OEn	OEn	OEn = 0	OEn = 1	OEn	OEn = 0	OEn = 1
TxDp	DAT			TxDp		
TxDm	SE0			TxDm		
RxDp	RxDp	DATO	DATI	RxDp	TxDp	RxDp
RxDm	RxDm	SE0O	SE0I	RxDm	TxDm	RxDm
RCV	Rcv	—	—	Rcv	—	Rcv

NOTE

If the RCV signal is not available on the external part (for example, when using a direct connection with TLL mode), the RCV input must be tied to the RxVp input.

32.3.2 Host Controller 2

Host controller 2 is configured for operation with a Fs/Ls serial transceiver or a parallel ULPI transceiver (Hs/Fs/Ls). The selection between transceiver type is software programmable. The module defaults to serial mode.

Host port 2 can support either a ULPI transceiver or a Serial Transceiver. Depending on the selected type, signaling for one or the other is available at the top level.

32.3.3 OTG Controller

The OTG controller offers HS/FS/LS capabilities in Host mode and HS/FS in device mode.

32.3.3.1 Host Mode

The controller supports direct connection of a FS/LS device (without external hub). Although there is no separate Transaction Translator block in the system. The transaction translator function normally associated with a USB 2.0 high speed hub has been implemented within the DMA and Protocol engine blocks to support connection to full and low speed devices.

32.3.3.2 Peripheral (Device) Mode

- Up to 8 bidirectional endpoints
- High/full speed operation
- Supports HNP, SRP.
- Remote wakeup capable

32.3.3.3 Special Considerations

The OTG port functions as gateway between the Host 1 Port and the OTG transceiver when in Bypass mode.

32.3.4 USB Power Control Module

The USB module supports suspend and wakeup functionality to place external transceivers in suspend mode, and wake them up either on a local request (CPU initiated) or on remote request by detecting activity on the USB line. The wake-up logic can optionally wake the CPU when it is in sleep mode at the time of the request.

32.3.4.1 Entering Suspend Mode

Suspend mode is always entered under control of driver software by setting the appropriate bit int PORTSC register. Once the controller is suspended, the clocks to the USB block can be stopped.

32.3.4.2 Wake-Up Events

The power control module monitors the USB bus when the USB core is in the suspend state. Depending on whether the core is on Host or Device mode, a number of wakeup conditions are detected. Upon detection of a wakeup condition, an interrupt (asynchronous) is generated on the CPU complex. This interrupt will also re-activate the clocks if these were stopped during the suspend.

32.3.4.2.1 Host Mode Events

The Host controller wakes-up on the following events.

Remote Wakeup Request

A peripheral can request the host to re-activate the bus by driving wake-up signaling on the Dm/Dp lines. The power control module will detect a J-K transition on the Dm/Dp lines and signal the wakeup request to the core.

Wake on Over-Current

If wake on overcurrent is enabled in the PORTSC registers, the power control module will signal a wakeup condition to the USB core.

Wake on Disconnect

The Power Control module detects disconnect events by monitoring the Dp/Dm lines. When a disconnect event is detected (Dm = Dp = 0) and the Wake on Disconnect is enabled in the PORTSC register, the core will be notified.

Wake on Connect

Similar to the Wake on Disconnect, the power control module detects a connect event (Dm or Dp High) and signals this to the USB core by setting the pwrctl_wakeup signal if enabled in the PORTSC register.

32.3.4.2.2 Device Mode Events

When the OTG controller is configured for peripheral operation, the power control module will detected following events.

Detection of Bus Activity

Any non-idle condition on the USB bus will activate the wakeup output of the power control module to notify the USB core of the wakeup event.

32.3.5 TLL Mode

The Transceiver-less Link Logic circuit allows two micro-controllers to use USB for an Inter-processor Communication Link (ICL) without using conventional USB transceivers. The TLL muxes support serial type interfacing only and are available on Host Port 1 and Host Port 2.

32.3.5.1 TLL Functional Description

The TLL logic is a logic representation of 2 serial transceivers connected by a USB cable. The USB bus DM/DP states are modeled internally in the TLL function, such that the USB I/O port acts as if it were a transceiver.

In a regular USB implementation with serial PHY's, the speed selection on the USB bus is done by means of a pull-up resistor on the Peripheral side either on the DM (low speed) or the DP (full Speed) line. This Pull-up pulls one of the USB lines high when the bus is idle.

This option cannot be modeled in logic as the serial USB interface does not provide for such a signal. The TLL MUX is therefore configured for Full Speed only operation.

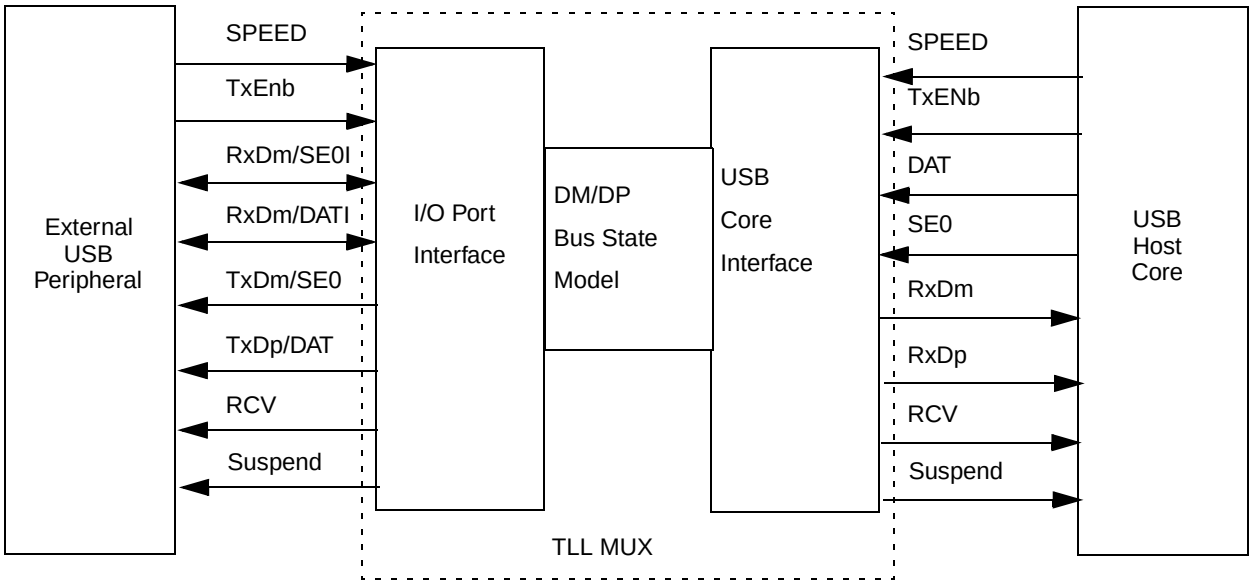


Figure 32-5. TLL MUX Functional Diagram

32.3.5.2 Host Port 1

On Host Port1, the TLL function is integrated with the Bypass function (see Table 32-6) and the Transceiver Type conversion logic. TLL mode is the default mode for this port. It can be disabled by setting bit 4 (Section 32.2.1.2, "OTGMIRROR—OTG Port Mirror Register") in the USBCONTROL register.

Table 32-6. Port 1 TLL and PHY Mode Pin Connections

	TLL mode		PHY Mode	
Pin	**I/O**	**External Device Pin**	**I/O**	**External PHY Pin**
SPEED	I	speed	O	speed
TxEnb	I	TxEnb	O	TxEnb
RxVm	I	TxDm	I	RxVm
RxVp	I	TxDp	I	RxVp
TxDm	O	RxDm	O	TxDm
TxDp	O	RxDp	O	TxDp

Table 32-6. Port 1 TLL and PHY Mode Pin Connections (continued)

	TLL mode		PHY Mode	
Pin	**I/O**	**External Device Pin**	**I/O**	**External PHY Pin**
RCV	O	RCV	I	RCV
Suspend	I	suspend	O	SuspendM

32.3.5.3 Host Port 2

The TLL module on Host Port 2 contains the TLL logic and the Serial Transceiver Type conversion logic. The Interface type conversion is available in both TLL and Non TLL modes.

TLL operation is the default mode. It can be turned off for operation with an external transceiver by setting bit 5 (Section 32.2.1.2, "OTGMIRROR—OTG Port Mirror Register") in the USBCONTROL register. Table 32-7 shows the pin connections for Port 2 TLL and PHY modes.

Table 32-7. Port 2 TLL and PHY Mode Pin Connections

	TLL mode		PHY Mode	
Pin	**I/O**	**External Device Pin**	**I/O**	**External PHY Pin**
SPEED	I	speed	O	speed
TxEnb	I	TxEnb	O	TxEnb
RxVm	I	TxDm	I	RxVm
RxVp/	I	TxDp	O	RxVp
TxDm	O	RxDm	O	TxDm
TxDp	O	RxDp	O	TxDp
RCV	O	RCV	I	RCV
Suspend	I	suspend	O	SuspendM

32.3.6 USB Bypass Mode

The USB Bypass mode is a special mode that allows for the transceiver on the OTG port to be used as transceiver for a USB peripheral device connected to host port 1. This mode is only available for full/low speed serial transceivers.

The Bypass module is a combination of the following:

- the bypass function (Host Port1—OTG port pass through)
- TLL function for Host Port 1
- Serial Transceiver Interface Conversion

32.3.6.1 Bypass Mode Operation

Bypass mode is enabled by writing a '1' to bit 0 (BPE) in the USB Control Register.

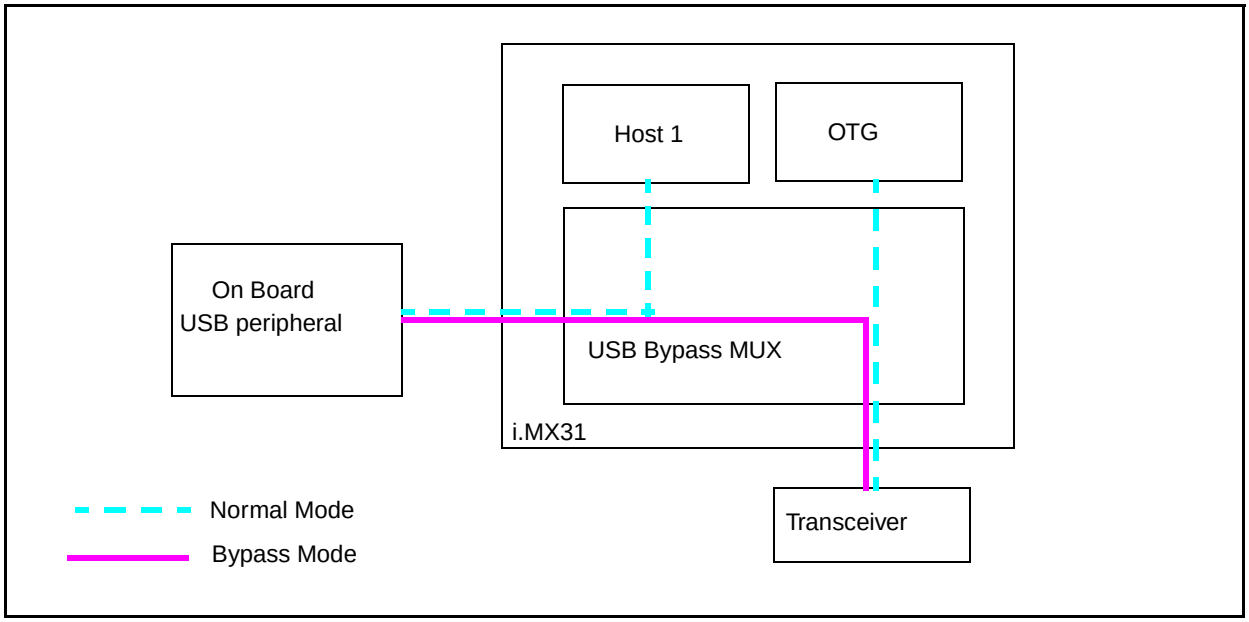


Figure 32-6. USB Bypass MUX Functional Diagram

In bypass mode, the serial interface signals from Host Port 1 are routed to the Serial Interface pins of the OTG port such that an external USB Peripheral device can use the OTG transceiver to connect to an external USB Host. As this function only works with USB peripherals directly connected to Host Port 1, the port is automatically set for TLL mode. Transceiver Interface Type conversion is available in Bypass mode such that the interface type of the OTG transceiver can be different from the Host 1 interface type.

The USB HOST1 core is disconnected from the port and the inputs RxDp, RCV and RxDm from the core are driven by bits 9 and 10 (H1BPVAL) in the USB Control Register.The OTG core is also disconnected from its port and inputs RxDp, RCV and RxDm are driven by bits 25 and 26 (OBPVAL) from the USB Control Register.

32.3.6.2 OTG and Host 1 Pin Functions

Table 32-8 list the pin functions of the Host 1 port when Bypass mode is enabled and the associated OTG pin. The pin functions of the OTG port are not affected by Bypass mode.

Table 32-8. HOST1 Bypass Mode Pin Functions

	Unidirectional				Bidirectional				OTG PORT	
Pin	**I/O**	**Single-Ended**	**I/O**	**Differential**	**I/O**	**Single- Ended**	**I/O**	**Differential**	**I/O**	**Pin**
RxDm	I	RxDm	I	RxDm	I/O	SEOI/SE0O	I/O	RxDm/TxDm	O	TxDm
RxDp	I	RxDp	I	RxDp	I/O	DATI/DATO	I/O	RxDp/TxDP	O	TxDP
RCV	O	RCV	O	RCV		-	O	RCV	I	RCV
TxDm	O	SE0	O	TxDM		-		-	I	RxDM
TxDp	O	DAT	O	TxDP		-		-		RxDp

Table 32-8. HOST1 Bypass Mode Pin Functions (continued)

	Unidirectional				Bidirectional				OTG PORT	
Pin	I/O	Single-Ended	I/O	Differential	I/O	Single- Ended	I/O	Differential	I/O	Pin
OEb	I	OEn	I	OEb	I	OEn	I	OEn		OEb
FS	I	Speed	I	FS	I	Speed	I	Speed		Speed
Suspend	I	Suspend	I	Suspend	I	Suspend	I	Suspend		Suspend

32.3.7 ULPI/Serial MUX

Both Host2 and OTG cores can be configured by software for ULPI or Serial PHY operation. The ULPI/Serial MUX selects between ULPI interface signals and Serial PHY interface signals. The MUX is controlled by the PHY Select signals from the USB core, and is switched when the software selects the interface mode.

The default configuration for the MUX is Serial mode. Switching to ULPI mode is done by writing the Parallel Transceiver Select (PTS) bits in the PORTSC register with 0b10.

32.3.8 Interrupts

32.3.8.1 USB Core Interrupts

Each USB core uses one dedicated vector in the Interrupt Table. The vector numbers associated with each of the cores can be found in the Interrupt section.

With the exception of the wake-up interrupts, all of the interrupt sources are controlled in the USB cores.

32.3.8.2 USB Wake-Up Interrupts

Each USB core has an associated wake-up interrupt. The wake-up interrupts are generated outside the USB cores' but use the same vector as the corresponding cores' interrupt. These interrupt are generated by the Power Control Modules which run on the 32KHz standby clock.The wake-up interrupt is designed to work even when the USB and CPU clocks are disabled, such that a wake-up condition on the USB bus can re-activate the CPU clocks. Because the wake-up interrupt is generated and cleared on a 32 KHz clock, this interrupt request will respond very slowly to clear actions. For this reason, the software must disable the wake-up interrupt to clear the request flag. Disabling the interrupt will mask the request instantaneously as this is clocked by the CPU clock. The software should than wait for at least three 32 KHz clock cycles before re-enabling this interrupt to allow sufficient time for the request flag to clear. As this interrupt is only used during low-power modes of the USB, it is sufficient to enable the wake-up interrupt just prior to enter USB suspend mode.

32.4 USB Interface

32.5 Overview

The Universal Serial Bus (USB) is a cable bus that supports data exchange between a host computer and a wide range of simultaneously accessible peripherals. The attached peripherals share USB bandwidth through a host-scheduled, token-based protocol. The bus allows peripherals to be attached, configured, used, and detached while the host and other peripherals are in operation.

USB software provides a uniform view of the IP system for all application software, hiding implementation details making application software more portable. It manages the dynamic attach and detach of peripherals.

There is only one host in any USB system. The USB interface to the host computer system is referred to as the Host Controller. The Host Controller may be implemented in a combination of hardware, firmware, or software.

There may be multiple USB devices in any system such as hubs, joysticks, speakers, and printers, for example. USB devices present a standard USB interface in terms of comprehension, response, and standard capability.

The host initiates transactions to specific peripherals, while the device responds to control transactions. The device sends and receives data to and from the host using a standard USB data format. The early versions of the USB specification (USB 1.0 and 1.1) operated at 12Mb/s or 1.5 Mbyte/s.

32.5.1 USB 2.0

The USB 2.0 specification supersedes the earlier versions of the USB specification. USB 2.0 offers the user a larger bandwidth increasing data throughput by a factor of 40. All the peripherals used with the previous versions of USB work perfectly with USB 2.0 while also offering a larger choice of higher performance peripherals, such as video-conferencing cameras, next-generation scanners and printers, a fast storage device, for example.

In addition to the 1.5Mbit/s and 12Mbit/s data rates of USB 1.1, the evolution from USB 1.1 to USB 2.0 adds an additional data rate of 480Mbit/s. Existing USB 1.1 connectors and full-speed cables support the higher speeds of the USB 2.0 without any changes. USB 2.0 specifies a microframe, is $1/8^{th}$ of a 1msec frame allowing small buffers even at the high data rate.

USB2.0 system configurations look identical to the end user to configurations based on earlier revisions of the USB specification. However, the end user will need to distinguish between USB 2.0 hubs and USB 1.1 hubs to optimize the placement of USB 2.0 high speed devices. A USB 2.0 hub accepts high-speed transactions at a faster rate but also matches the data rate of the peripherals, while the USB 1.1 hub will only support the lower 12 Mb/s and 1.5 Mb/s data rates.

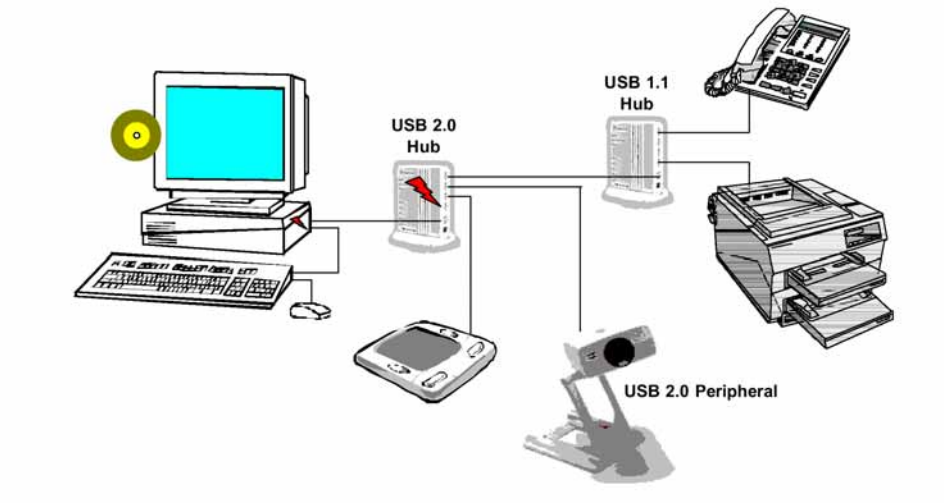


Figure 32-7. Example USB 2.0 System Configuration

The roles of the components of the USB 2.0 system have minor changes from the roles in a USB 1.1 system. However, current peripheral products work with the USB 2.0 system without any changes. Additional performance is not required for many of the human interface devices such as mice, keyboards, and game pads. Both USB 1.1 and USB 2.0 devices will operate in a USB 2.0 system while opening up the possibilities of exciting new higher bandwidth peripherals.

For additional information, refer to USB 2.0 specification USB 2.0 Transceiver Macrocell Interface (UTMI) Specification, Version 1.05, March 2001, Jon Lueker, Steve McGowan (Editor) Ken Oliver, Dean Warren. http://www.intel.com.

32.5.2 USB On-The-Go

USB is a popular standard for connecting peripherals and portable consumer electronic devices such as digital cameras and hand-held computers to host PCs. The On-The-Go (OTG) Supplement to the USB Specification extends USB to peer-to-peer application. Using USB OTG technology consumer electronics, peripherals and portable devices can connect to each other (for example, a digital camera can connect directly to a printer, or a keyboard can connect to a Personal Digital Assistant) to exchange data.

With USB On-The-Go you can develop a fully USB compliant peripheral device that can also assume the role of a USB host. The OTG state machines determine the role of the device based on connector signals, and then initializes the device in the appropriate mode of operation (host or peripheral) based on how it is connected. After connecting the devices can negotiate using the OTG protocols to assume the role of host or peripheral based on the task to be accomplished.

For additional information, refer to the *On-The-Go Supplement to the USB 2.0 Specification* On-The-Go Supplement to the USB 2.0 Specification, Revision 1.0, Dec 2001, On-The-Go Working Group of the USB-IF. http://www.usb.org.

32.6 USBOTG Block Diagram

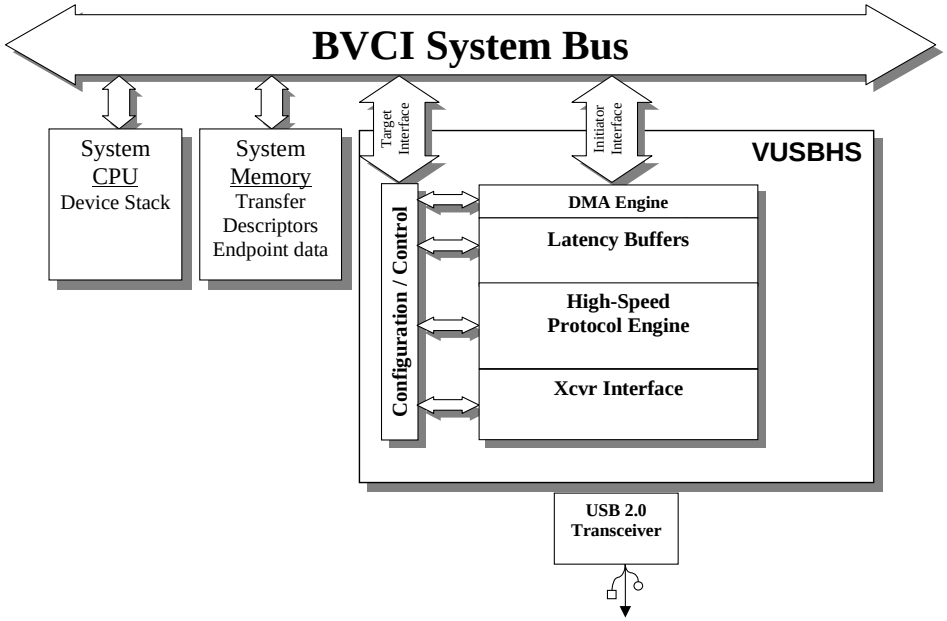


Figure 32-8. USBOTG Block Diagram

32.7 USBOTG Core Features

- The USBOTG dual role USB Host controller or USB Device controller operation using the same hardware.
- Intel™ EHCI host controller. The USB host controller registers and data structures are compliant to Intel ™ EHCI specification. Device controller registers and data structures are implemented as extensions to the EHCI programmers interface.
- BVCI compliant system bus interface OR
- AMBA-AHB compliant system bus interface.
- Direct support for USB Transceiver Macrocell Interface (UTMI), UTMI+ Low Pin Interface (ULPI), or Philips™ interface transceivers.
- The ULPI interface is purchased as a separate option.
- Directly connected USB legacy (USB 1.1) Full and Low speed devices without a companion USB 1.1 host controller or host controller driver software using EHCI standard data structures.
- Configurable dual port RAM buffers isolate memory latency on the system bus from the timing requirements of the USB.
- Fully static and synchronous, synthesizable RTL design available in VHDL or Verilog formats. The synchronous static design simplifies synthesis and test insertion within the tool flow, and provides a framework for low power design.
- Integrated simulation and verification test bench. A verification environment is provided to simulate host and device operations in a standalone or processor integrated environment.

- FPGA prototype available. A Field Programmable Gate Array (FGPA) implementation of the USBOTG core integrated with a 32-bit RISC microprocessor is available to reduce the prototype design time for technology evaluation and software development.

32.8 Functional Description

32.8.1 Device Data Structure

The function of the device operation is to transfer a request in the memory image to and from the Universal Serial Bus. Using a set of linked list transfer descriptors, pointed to by a queue head, the device controller will perform the data transfers.

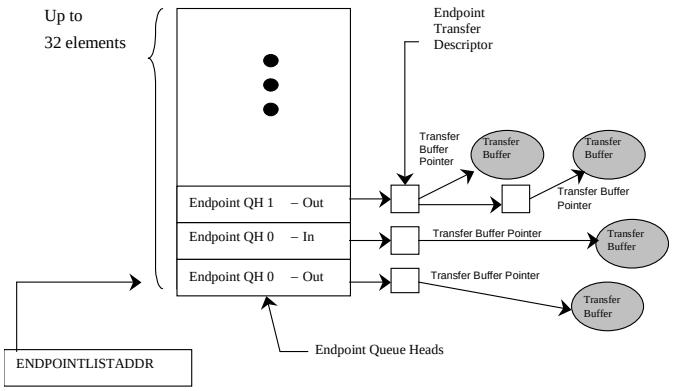


Figure 32-9. End Point Queue Head Organization

The USBOTG Device API incorporates and abstracts for the application developer all of the information contained in the device operational model.

32.8.2 Host Data Structure

The host data structures are used to communicate control, status, and data between software and the Host Controller. The Periodic Frame List is an array of pointers for the periodic schedule. A sliding window on the Periodic Frame List is used. The Asynchronous Transfer List is where all the control and bulk transfers are managed. The USBOTG Host API incorporates and abstracts for the application developer all of the information contained in the host operational model.

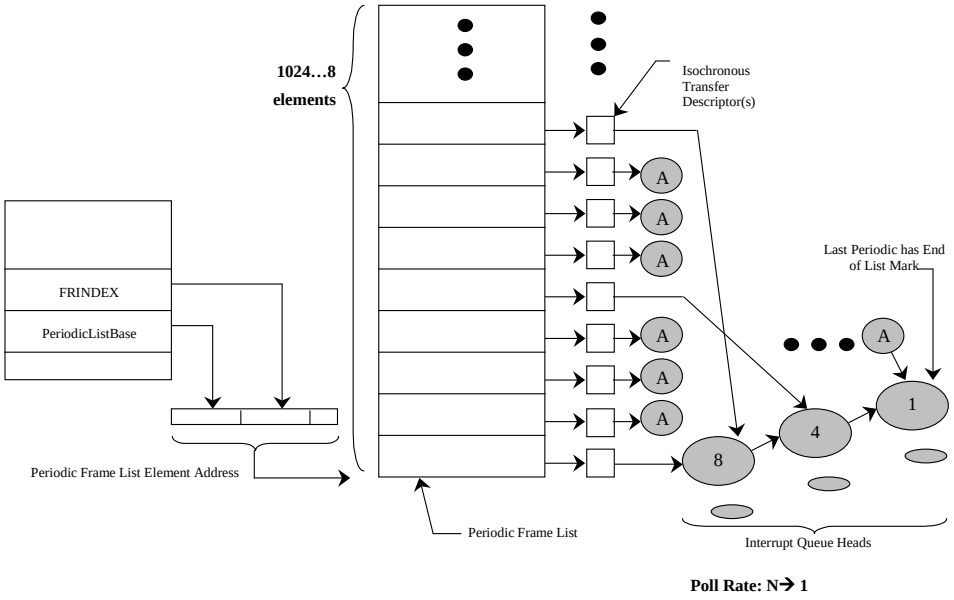


Figure 32-10. Periodic Schedule Organization

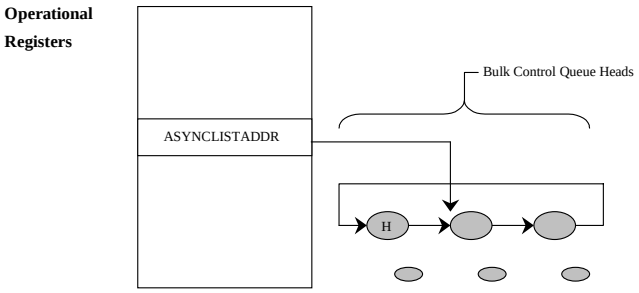


Figure 32-11. Asynchronous Schedule Organization

32.9 Register Interface

Slave accesses from the controlling processor enables access to the configuration, control, and status registers. One function of the system address map is the registers base address, which must begin on a DWord (32-bit) boundary. Register offset definitions are listed in Table 32-9.

Configuration, control and status registers are divided into three categories, identification, capability and operational registers.

- Identification registers are used to declare the slave interface presence along with the complete set of the hardware configuration parameters.
- Static, read only capability registers define the software limits, restrictions, and capabilities of the host/device controller.
- Operational registers are comprised of dynamic control or status registers that may be read only, read/write, or read/write to clear. The following sections define the use of these registers.

EHCI registers are listed alongside device registers to show the complementary nature of host and device control.

NOTE

Host mode EHCI compatibility begins at offset 0x100. If it is necessary to begin the EHCI register set at offset 0x000, the identification registers are disabled from the address map by connecting the upper most address bit of the slave interface to a logic level ‘1’ and adjusting the offsets below accordingly.

Table 32-9. Interface Register Sets

Offset	Register Set	Explanation
000h to 0fch	Identification Registers	Identification registers are used to declare the slave interface presence and include a table of the hardware configuration parameters.
100h to 124h	Capability Registers	Capability registers specify the limits, restrictions, and capabilities of a host/device controller implementation. These values are used as parameters to the host/device controller driver.
140h to 1FCh	Operational Registers	Operational registers are used by the system software to control and monitor the operational state of the host/device controller.

32.9.1 Configuration, Control, and Status Register Set

Table 32-10 shows the device/host capability registers.

Table 32-10. Device/Host Capability Registers

Offset	Size (Bytes)	Mnemonic	Register Name	DEV	OTG	SPH	MPH
000h	4	ID	Identification Register	÷	÷	÷	÷
004h	4	HWGENERAL	General Hardware Parameters	÷	÷	÷	÷
008h	4	HWHOST	Host Hardware Parameters		÷	÷	÷
00Ch	4	HWDEVICE	Device Hardware Parameters	÷	÷		
010h	4	HWTXBUF	TX Buffer Hardware Parameters	÷	÷	÷	÷
014h	4	HWRXBUF	RX Buffer Hardware Parameters	÷	÷	÷	÷

Table 32-10. Device/Host Capability Registers (continued)

Offset	Size (Bytes)	Mnemonic	Register Name	DEV	OTG	SPH	MPH
018h	4	HWTTTXBUF	TT-TX Buffer Hardware Parameters				÷
01Ch	4	HWTTRXBUF	TT-RX Buffer Hardware Parameters				÷
020h-0FCh	232	Reserved	N/A				
100h	1	CAPLENGTH	Capability Register Length	÷	÷	÷	÷
101h	1	Reserved	N/A				
102h	2	HCIVERSION	Host Interface Version Number		÷	÷	÷
104h	4	HCSPARAMS	Host Ctrl. Structural Parameters		÷	÷	÷
108h	4	HCCPARAMS	Host Ctrl. Capability Parameters		÷	÷	÷
10Ch–11Fh	20	Reserved	N/A				
120h	2	DCIVERSION	Dev. Interface Version Number	÷	÷		
122h	2	Reserved	N/A	÷			
124h	4	DCCPARAMS	Device Ctrl. Capability Parameters	÷	÷		
128h–13Ch	24	Reserved	N/A				
140h	4	USBCMD	USB Command	÷	÷	÷	÷
144h	4	USBSTS	USB Status	÷	÷	÷	÷
148h	4	USBINTR	USB Interrupt Enable	÷	÷	÷	÷
14Ch	4	FRINDEX	USB Frame Index	÷	÷	÷	÷
150h	4	Reserved	4G Segment Selector				
154h	4	PERIODICLISTBASE	Frame List Base Address		÷	÷	÷
		Device Addr	USB Device Address	÷	÷		

Table 32-10. Device/Host Capability Registers (continued)

Offset	Size (Bytes)	Mnemonic	Register Name	DEV	OTG	SPH	MPH
158h	4	ASYNCLISTADDR	Next Asynchronous List Address		÷	÷	÷
		Endpointlist Addr	Address at Endpoint list in memory	÷	÷		
15Ch	4	ASYNCTTSTS	Asynchronous Buffer Status For Embedded TT.				÷
160h	4	BURSTSIZE	Programmable Burst Size	÷	÷	÷	÷
164h	4	TXFILLTUNING	Host Transmit Pre-Buffer Packet Tuning		÷	÷	÷
168h	4	TXTTFILLTUNING	Host TT Transmit Pre-Buffer Packet Tuning				÷
16Ch	4	N/A	Reserved				
170-17Ch	16	N/A	Reserved				
180h	4	CONFIGFLAG	Configured Flag Register		÷	÷	÷
184h	4	PORTSC1	Port Status/Control 1	÷	÷	÷	÷
188h	4	PORTSC2	Port Status/Control 2				÷
…	4	PORTSCx	Port Status/Control x				÷
1A0h	4	PORTSC8	Port Status/Control 8				÷
1A4h	4	OTGSC	On-The-Go (OTG) Status and Control		÷		
1A8h	4	USBMODE	USB Device Mode	÷	÷	÷	÷
1ACh	4	ENPDTSETUPSTAT	Endpoint Setup Status	÷	÷		
1B0h	4	ENDPTPRIME	Endpoint Initialization	÷	÷		
1B4h	4	ENDPTFLUSH	Endpoint De-Initialize	÷	÷		
1B8h	4	ENDPTSTATUS	Endpoint Status	÷	÷		

Table 32-10. Device/Host Capability Registers (continued)

Offset	Size (Bytes)	Mnemonic	Register Name	DEV	OTG	SPH	MPH
1BCh	4	ENDPTCOMPLETE	Endpoint Complete	÷	÷		
1C0h	4	ENDPTCTRL0	Endpoint Control 0	÷	÷		
1C4h	4	ENDPTCTRL1	Endpoint Control 1	÷	÷		
…	4	ENDPTCTRLx	Endpoint Control x	÷	÷		
1FCh	4	ENDPTCTRL15	Endpoint Control 15	÷	÷		

NOTE

Italic Text indicates a deviation from EHCI for Device.

32.9.2 Summary of Register Layouts

Table 32-11 shows the USBOTG register summary.

Table 32-11. USBOTG Register Summary

Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h ID									REVISION								NID								ID							
004h HWGENERAL																							SM	PHYM			PHYW		BWT	CLKC		RT
008h HWHOST	TTPER								TTASY																				NPORT			HC
00Ch HWDEVICE																											DEVP					DC
010h HWTXBUF	TXLC								TXCHANADD								TXADD								TXBURST							
014h HWRXBUF																	RXADD								RXBURST							
020h Reserved																																
… Reserved																																
0FCh Reserved																																
100h CAPLENGTH	CAPLENGTH																															

Table 32-11. USBOTG Register Summary (continued)

Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
101h Reserved																																
102h HCIVERSION																	HCIVERSION															
104h HCSPARAMS					N_TT				N_PTT							PI	N_CC				N_PCC							PPC	N_PORTS			
108h HCCPARAMS																	EEC[7:0]								IST[7:4]				R	R	PFL	ADC
10Ch Reserved																																
… Reserved																																
11Fh Reserved																																
120h DCIVERSION																	DCIVERSION															
122h Reserved																																
124h DCCPARAMS																								HC	DC			DEN				
128h Reserved																																
… Reserved																																
13Ch Reserved																																
140h USBCMD									ITC								FS2		SUTW	ATDTW	ASPE		ASP1	ASP0	LR	IAA	ASE	PSE	FS1	FS0	RST	RS
144h USBSTS																	AS	PS	RCL	HCH				SLI	SRI	URI	AAI	SEI	FRI	PCI	UEI	UI
148h USBINTR																								SLE	SRE	URE	AAE	SEE	FRE	PCE	UEE	UE
14Ch FRINDEX																		FRINDEX[13:0]														

Table 32-11. USBOTG Register Summary (continued)

Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
150h Reserved																																
154h PERIODICLISTBASE	PERBASE[31:12]																															
Device Addr	USBADR[31:25]																															
158h ASYNCLISTADDR	ASYBASE[31:5]																															
Endpointlist Addr	EPBASE[31:11]																															
15Ch ASYNCTTSTS																															TTAC	TTAS
160h BURSTSIZE																	TXPBURST								RXPBURST							
164h TXFILLTUNING											TXFIFOTHRES									TXSCHHEALTH					TXSCHOH							
168h TXTTFILLTUNING																				TXTTSCHHEALTH							TXTTSCHOH				
16Ch Reserved																																
170 ULPI Viewport	ULPI Viewport [optional]																															
174 Reserved																																
… Reserved																																
17Ch Reserved																																
180h CONFIGFLAG	set to zero																															1
184h PORTSC1	PTS		STS	PTW	PSPD		PFSC	PHCD	WKOC	WKDS	WKCN	PTC					PIC		PO	PP	LS		HSP	PR	SUSP	FPR	OCC	OCA	PEC	PE	CSC	CCS
188h PORTSC2	PTS		STS	PTW	PSPD		PFSC	PHCD	WKOC	WKDS	WKCN	PTC					PIC		PO	PP	LS		HSP	PR	SUSP	FPR	OCC	OCA	PEC	PE	CSC	CCS

Table 32-11. USBOTG Register Summary (continued)

Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
… PORTSCx	PTS		STS	PTW	PSPD			PFSC	PHCD	WKOC	WKDS	WKCN	PTC				PIC		PO	PP	LS		HSP	PR	SUSP	FPR	OCC	OCA	PEC	PE	CSC	CCS
1A0h PORTSC8	PTS		STS	PTW	PSPD			PFSC	PHCD	WKOC	WKDS	WKCN	PTC				PIC		PO	PP	LS		HSP	PR	SUSP	FPR	OCC	OCA	PEC	PE	CSC	CCS
1A4h OTGSC		DPIE	1msE	BSEIE	BSVIE	ASVIE	AVVIE	IDIE	R	DPIS	1msS	BSEIS	BSVIS	ASVIS	AVVIS	IDIS	R	DPS	1msT	BSE	BSV	ASV	AVV	ID				DP	OT	R		VD
1A8h USBMODE																												SDIS	SLOM			
1ACh ENPDTSETUPSTAT																	ENDPTSETUPSTAT															
1B0h ENDPTPRIME	PETB[15:0]																PERB[15:0]															
1B4h ENDPTFLUSH	FETB[15:0]																FERB[15:0]															
1B8h ENDPTSTATUS	ETBR[15:0]																ERBR[15:0]															
1BCh ENDPTCOMPLETE	ETCE[15:0]																ERCE[15:0]															
1C0h ENDPTCTRL0									TXE				TXT		R	TXS									RXE				RXT			RXS
1C4h ENDPTCTRL1									TXE	TXR	TXI		TXT		TXD	TXS									RXE	RXR	RXI		RXT		RXD	RXS
… ENDPTCTRLx									TXE	TXR	TXI		TXT		TXD	TXS									RXE	RXR	RXI		RXT		RXD	RXS
1FCh ENDPTCTRL15									TXE	TXR	TXI		TXT		TXD	TXS									RXE	RXR	RXI		RXT		RXD	RXS

32.9.3 Identification Registers

Identification registers are used to declare the slave interface presence and include a table of the hardware configuration parameters.

32.9.3.1 ID Register

The Identification register (ID) provides a simple way to determine if the USBOTG 2.0 core is provided in the system. The ID register identifies the USBOTG 2.0 core and its revision.

Figure 32-12 shows the register; Table 32-12 provides its field descriptions.

0x43F8_8000 (ID) Access: User Read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	REVISION							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	1	1	NID						0	0	ID					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-12. ID—Identification Register

Table 32-12. ID Field Descriptions

Field	Description
31–24	Reserved. These bits are reserved and should be set to zero.
23–16 REVISION[7:0]	Revision number of the core
15–14	Reserved. These bits are reserved and should be set to one.
13–8 NID[5:0]	Ones complement version of ID[5:0].
7–6	Reserved. These bits are reserved and should be set to zero.
5–0 ID[5:0]	Configuration number. This number is set to 0x05 and indicates that the peripheral is the USBOTG 2.0 core.

32.9.3.2 HWGENERAL

These are the General hardware parameters, as defined in System Level Issues and Core Configuration.

Figure 32-13 shows the register; Table 32-13 provides its field descriptions.

0x43F8_8004 (HWGENERAL) Access: User Read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	SM	PHYM			PHYW		BWT	CLKC		RT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-13. HWGENERAL—General Hardware Parameters

Table 32-13. HWGENERAL Field Descriptions

Field	Description
31–10	Reserved. These bits are reserved and should be set to zero.
9 SM	VUSB_HS_PHY_SERIAL
8–6 PHYM	VUSB_HS_PHY_TYPE
5–4 PHYW	VUSB_HS_PHY16_8
3 BWT	Reserved for internal testing.
2–1 CLKC	VUSB_HS_CLOCK_CONFIGURATION
0 RT	VUSB_HS_RESET_TYPE

32.9.3.3 HWHOST

These are the Host hardware parameters, as defined in System Level Issues and Core Configuration.

Figure 32-14 shows the register; Table 32-14 provides its field descriptions.

0x43F8_8008 (HWHOST) Access: User Read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TTPER								TTASY							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	NPORT			HC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-14. HWHOST—Host Hardware Parameters

Table 32-14. HWHOST Field Descriptions

Field	Description
31–24 TTPER	VUSB_HS_TT_PERIODIC_CONTEXTS
23–16 TTASY	VUSB_HS_TT_ASYNC_CONTEXTS
15–4	Reserved. These bits are reserved and should be set to zero.
3–1 NPORT	VUSB_HS_NUM_PORT-1
0 HC	VUSB_HS_HOST

32.9.3.4 HWDEVICE

These are the Device hardware parameters, as defined in System Level Issues and Core Configuration.

Figure 32-15 shows the register; Table 32-15 provides its field descriptions.

0x43F8_800C (HWDEVICE) Access: User Read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	DEVEP					DC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-15. HWDEVICE—Device Hardware Parameters

Table 32-15. HWDEVICE Field Descriptions

Field	Description
31–6	Reserved. These bits are reserved and should be set to zero.
5–1 DEVEP	VUSB_HS_DEV_EP
0 DC	device capable; [VUSB_HS_DEV /= 0]

32.9.3.5 HWTXBUF

These are the TX buffer hardware parameters, as defined in System Level Issues and Core Configuration. Figure 32-16 shows the register; Table 32-16 provides its field descriptions.

0x43F8_8010 (HWTXBUF) Access: User Read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TXLCR	0	0	0	0	0	0	0	TXCHANADD							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TXADD								TXBURST							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-16. HWTXBUF—TX Buffer Hardware Parameters

Table 32-16. HWTXBUF Field Descriptions

Field	Description
31 TXLCR	VUSB_HS_TX_LOCAL_CONTEXT_REGISTERS
30–24	Reserved. These bits are reserved and should be set to zero.
23–16 TXCHANADD	VUSB_HS_TX_CHAN_ADD
15–8 TXADD	VUSB_HS_TX_ADD
7–0 TXBURST	VUSB_HS_TX_BURST

32.9.3.6 HWRXBUF

RX buffer hardware parameters as defined in System Level Issues and Core Configuration.

Figure 32-17 shows the register; Table 32-17 provides its field descriptions.

0x43F8_8014 (HWRXBUF) Access: User Read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RXADD								RXBURST							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-17. HWRXBUF—RX Buffer Hardware Parameters

Table 32-17. HWRXBUF Field Descriptions

Field	Description
31–16	Reserved. These bits are reserved and should be set to zero.
15–8 RXADD	VUSB_HS_RX_ADD
7–0 RXBURST	VUSB_HS_RX_BURST

32.9.4 Device/Host Capability Registers

Device/Host Capability registers specify the software limits, restrictions, and capabilities of the host/device controller implementation.

32.9.4.1 CAPLENGTH—EHCI Compliant

Default Value: 40h

This register is used to indicate which offset to add to the register base address at the beginning of the Operational Register.

Figure 32-18 shows the register.

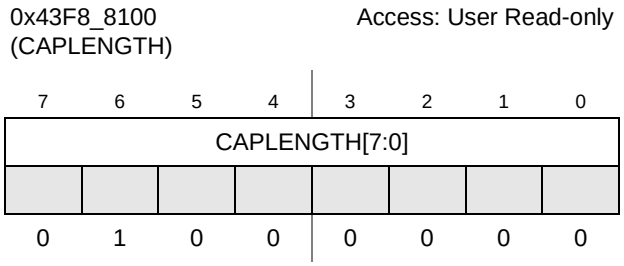


Figure 32-18. CAPLENGTH—Capability Register Length

32.9.4.2 HCIVERSION—EHCI Compliant

Default Value: 0100h

This is a two-byte register containing a BCD encoding of the EHCI revision number supported by this host controller. The most significant byte of this register represents a major revision and the least significant byte is the minor revision.

Figure 32-19 shows the register.

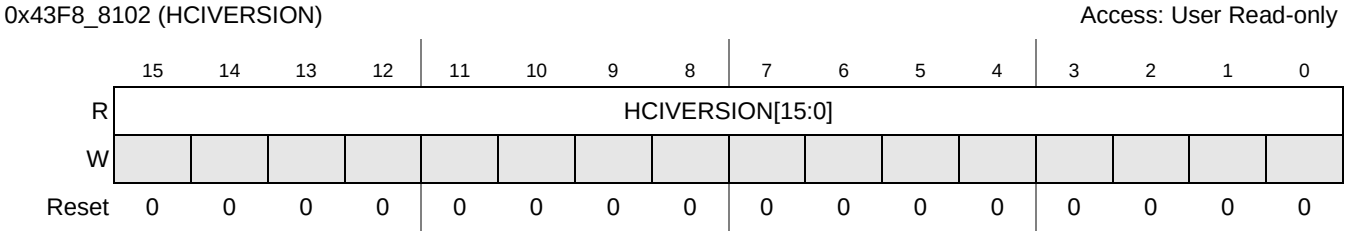


Figure 32-19. HCIVERSION—Host Interface Version Number

32.9.4.3 HCSPARAMS—EHCI Compliant with Extensions

Figure 32-20 shows the register; Table 32-18 provides its field descriptions.

0x43F8_8104 (HCSPARAMS) Access: User Read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	N_TT				N_PTT				0	0	0	PI
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	N_CC				N_PCC				0	0	0	PPC	N_PORTS			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-20. HCSPARAMS—Host Control Structural Parameters

Port steering logic capabilities are described in this register.

Table 32-18. HCSPARAMS Field Descriptions

Field	Description
31–28	Reserved. These bits are reserved and should be set to zero.
27–24 N_TT[3:0]	Number of Transaction Translators (N_TT). This field indicates the number of embedded transaction translators associated with the USB2.0 host controller. For Multi-Port Host this field will always equal “0001”. For all other implementations, N_TT = “0000”. This in a non-EHCI field to support embedded TT.
23–20 N_PTT[3:0]	Number of Ports per Transaction Translator (N_PTT). This field indicates the number of ports assigned to each transaction translator within the USB2.0 host controller. For Multi-Port Host this field will always equal N_PORTS. For all other implementations, N_PTT = “0000”. This in a non-EHCI field to support embedded TT.
19–17	Reserved
PI	Port Indicators (P INDICATOR). This bit indicates whether the ports support port indicator control. When set to one, the port status and control registers include a read/writeable field for controlling the state of the port indicator. This field will always be “1”.
15–12 N_CC[3:0]	Number of Companion Controller (N_CC). This field indicates the number of companion controllers associated with this USB2.0 host controller. A zero in this field indicates there are no internal Companion Controllers. Port-ownership hand-off is not supported. A value larger than zero in this field indicates there are companion USB1.1 host controller(s). Port-ownership hand-offs are supported. High, Full- and Low-speed devices are supported on the host controller root ports. In this implementation this field will always be “0”.

Table 32-18. HCSPARAMS Field Descriptions (continued)

Field	Description
11–8 N_PCC[3:0]	Number of Ports per Companion Controller. This field indicates the number of ports supported per internal Companion Controller. It is used to indicate the port routing configuration to the system software. For example, if N_PORTS has a value of 6 and N_CC has a value of 2 then N_PCC could have a value of 3. The convention is that the first N_PCC ports are assumed to be routed to companion controller 1, the next N_PCC ports to companion controller 2, and so on. In the previous example, the N_PCC could have been 4, where the first 4 are routed to companion controller 1 and the last two are routed to companion controller 2. The number in this field must be consistent with N_PORTS and N_CC. In this implementation this field will always be “0”.
7–5	Reserved
4 PPC	Port Power Control. This field indicates whether the host controller implementation includes port power control. A one indicates the ports have port power switches. A zero indicates the ports do not have port power switches. The value of this field affects the functionality of the Port Power field in each port status and control register. This field will always be “0” for a device only implementation.
3–0 N_PORTS[3:0]	Number of downstream ports. This field specifies the number of physical downstream ports implemented on this host controller. The value of this field determines how many port registers are addressable in the Operational Register. Valid values are in the range of 1h to Fh. A zero in this field is undefined. The number of ports for a host implementation can have set parameters from 1 to 8. This field will always be 1 for device only implementation.

32.9.4.4 HCCPARAMS—EHCI Compliant

Default Value:0006h

This register identifies multiple mode control (time-base bit functionality) addressing capability.

Figure 32-21 shows the register; Table 32-19 provides its field descriptions.

0x43F8_8108 (HCCPARAMS) Access: User Read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	EECP[7:0]								IST[7:4]				0	ASP	PFL	ADC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-21. HCCPARAMS—Host Control Capability Parameters

Table 32-19. HCCPARAMS Field Descriptions

Field	Description
31–16	reserved. These bits are reserved and should be set to zero.
15–8 EECP[7:0]	EHCI Extended Capabilities Pointer. Default = 0. This optional field indicates the existence of a capabilities list. A value of 00h indicates no extended capabilities are implemented. A non-zero value in this register indicates the offset in PCI configuration space of the first EHCI extended capability. The pointer value must be 40h or greater if implemented to maintain the consistency of the PCI header defined for this class of device. For this implementation this field is always “0”.
7–4 IST[7:4]	Isochronous Scheduling Threshold. Default = implementation dependent. This field indicates, relative to the current position of the executing host controller, where software can reliably update the isochronous schedule. When bit [7] is zero, the value of the least significant 3 bits indicates the number of micro-frames a host controller can hold a set of isochronous data structures (one or more) before flushing the state. When bit [7] is a one, then host software assumes the host controller may cache an isochronous data structure for an entire frame. This field will always be “0”.
3	Reserved. These bits are reserved and should be set to zero.
2 ASP	Asynchronous Schedule Park Capability. Default = 1. If this bit is set to a one, then the host controller supports the park feature for high-speed queue heads in the Asynchronous Schedule. The feature can be disabled or enabled and set to a specific level by using the *Asynchronous Schedule Park Mode Enable* and *Asynchronous Schedule Park Mode Count* fields in the USBCMD register. This field will always be “1”

Table 32-19. HCCPARAMS Field Descriptions (continued)

Field	Description
1 PFL	Programmable Frame List Flag. If this bit is set to zero, then the system software must use a frame list length of 1024 elements with this host controller. The USBCMD register Frame List Size field is a read-only register and must be set to zero. If set to a one, then the system software can specify and use a smaller frame list and configure the host controller via the USBCMD register Frame List Size field. The frame list must always be aligned on a 4K-page boundary. This requirement ensures that the frame list is always physically contiguous. This field will always be “1”.
0 ADC	64-bit Addressing Capability. This field will always be “0”. No 64-bit addressing capability is supported.

DCIVERSION (Non-EHCI)

The device controller interface conforms to the two-byte BCD encoding of the interface version number contained in this register.

Figure 32-22 shows the register.

0x43F8_8120 (DCIVERSION) Access: User Read-only

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DCIVERSION[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-22. DCIVERSION—Device Interface Version Number

32.9.4.5 DCCPARAMS (Non-EHCI)

These fields describe the overall host/device capability of the controller.

Figure 32-23 shows the register; Table 32-20 provides its field descriptions.

0x43F8_8124 (DCCPARAMS) Access: User Read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	H	D	0	0	DEN[4:0]				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-23. DCCPARAMS—Device Controller Capability Parameters

Table 32-20. DCCPARAMS Field Descriptions

Field	Description
31–9	Reserved. These bits are reserved and should be set to zero.
8 HC	Host Capable. When this bit is 1, this controller is capable of operating as an EHCI compatible USB 2.0 host controller.
7 DC	Device Capable. When this bit is 1, this controller is capable of operating as a USB 2.0 device.
6–5	Reserved. These bits are reserved and should be set to zero.
4–0 DEN[4:0]	Device Endpoint Number. This field indicates the number of endpoints built into the device controller. If this controller is not device capable, then this field will be zero. Valid values are 0–16.

32.9.5 Device/Host Operational Registers

Operational registers are comprised of dynamic control or status registers that may be read only, read/write, or read/write to clear. The following sections define the use of these registers.

32.9.5.1 USB Command Register (USBCMD)

Default Value:00080B00h (host mode)

00080000h (device mode)

The serial bus host/device controller executes the command indicated in this register.

Figure 32-24 shows the register; Table 32-21 provides its field descriptions.

0x43F8_8140 (USBCMD) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	ITC[7:0]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FS2	0	SUTW	ATDTW	ASPE	0	ASP1	ASP0	LR	IAA	ASE	PSE	FS1	FS0	RST	RS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-24. USB Command Register (USBCMD)

Table 32-21. USBCMD Field Descriptions

Field	Description
31–24	Reserved. These bits are reserved and should be set to zero.
23–16 ITC[7:0]	Interrupt Threshold Control—Read/Write. Default 08h. The system software uses this field to set the maximum rate at which the host/device controller will issue interrupts. ITC contains the maximum interrupt interval measured in micro-frames. Valid values are shown below. 00h Immediate (no threshold) 01h 1 micro-frame 02h 2 micro-frames 04h 4 micro-frames 08h 8 micro-frames 10h 16 micro-frames 20h 32 micro-frames 40h 64 micro-frames
15 FS2	Frame List Size—(Read/Write or Read Only). Default 000b. This field is Read/Write only if Programmable Frame List Flag in the HCCPARAMS registers is set to one. This field specifies the size of the frame list that controls which bits in the Frame Index Register should be used for the Frame List Current index. Note that this field is made up from USBCMD bits 15, 3 and 2. 000 1024 elements (4096 bytes) Default value 001 512 elements (2048 bytes) 010 256 elements (1024 bytes) 011 128 elements (512 bytes) 100 64 elements (256 bytes) 101 32 elements (128 bytes) 110 16 elements (64 bytes) 111 8 elements (32 bytes) Only the host controller uses this field.
14	Reserved

Table 32-21. USBCMD Field Descriptions (continued)

Field	Description
13 SUTW	Setup TripWire—Read/Write. [device mode only] This bit is used as a semaphore to ensure that the setup data payload of 8 bytes is extracted from a QH by the DCD without being corrupted. If the setup lockout mode is off (See USBMODE) then there exists a hazard when new setup data arrives while the DCD is copying the setup data payload from the QH for a previous setup packet. This bit is set and cleared by software and will be cleared by hardware when a hazard exists.
12 ATDTW	Add DTD Tripwire—Read/write. [device mode only] This bit is used as a semaphore to ensure the to proper addition of a new dTD to an active (primed) endpoint's linked list. This bit is set and cleared by software. This bit shall also be cleared by hardware when is state machine is hazard region for which adding a dTD to a primed endpoint may go unrecognized.
11 ASPE	Asynchronous Schedule Park Mode Enable (OPTIONAL)—Read/Write. If the Asynchronous Park Capability bit in the HCCPARAMS register is a one, then this bit defaults to a 1h and is R/W. Otherwise the bit must be a zero and is RO. Software uses this bit to enable or disable Park mode. When this bit is one, Park mode is enabled. When this bit is a zero, Park mode is disabled. This field is set to "1" in this implementation.
10	Reserved
9–8 ASP[1:0]	Asynchronous Schedule Park Mode Count (OPTIONAL)—Read/Write. If the Asynchronous Park Capability bit in the HCCPARAMS register is a one, then this field defaults to 3h and is R/W. Otherwise it defaults to zero and is RO. It contains a count of the number of successive transactions the host controller is allowed to execute from a high-speed queue head on the Asynchronous schedule before continuing traversal of the Asynchronous schedule. Valid values are 1h to 3h. Software must not write a zero to this bit when Park Mode Enable is a one as this results in undefined behavior. This field is set to 3h in this implementation.
7 LR	Light Host/Device Controller Reset (OPTIONAL)—Read Only. Not Implemented. This field will always be "0".
6 IAA	Interrupt on Async Advance Doorbell—Read/Write. This bit is used as a doorbell by software to tell the host controller to issue an interrupt the next time it advances asynchronous schedule. Software must write a 1 to this bit to ring the doorbell. When the host controller has evicted all appropriate cached schedule states, it sets the Interrupt on Async Advance status bit in the USBSTS register. If the Interrupt on Sync Advance Enable bit in the USBINTR register is one, then the host controller will assert an interrupt at the next interrupt threshold. The host controller sets this bit to zero after it has set the Interrupt on Sync Advance status bit in the USBSTS register to one. Software should not write a one to this bit when the asynchronous schedule is inactive. Doing so will yield undefined results. This bit is only used in host mode. Writing a one to this bit when device mode is selected will have undefined results.
5 ASE	Asynchronous Schedule Enable—Read/Write. Default 0b. This bit controls whether the host controller skips processing the Asynchronous Schedule. 0 Do not process the Asynchronous Schedule. 1 Use the ASYNCLISTADDR register to access the Asynchronous Schedule. Only the host controller uses this bit.

Table 32-21. USBCMD Field Descriptions (continued)

Field	Description
4 PSE	Periodic Schedule Enable—Read/Write. Default Ob. This bit controls whether the host controller skips processing the Periodic Schedule. 0 Do not process the Periodic Schedule 1 Use the PERIODICLISTBASE register to access the PeriodicSchedule. Only the host controller uses this bit.
3 FS1	Frame List Size—(Read/Write or Read Only). Default 000b. This field is Read/Write only if Programmable Frame List Flag in the HCCPARAMS registers is set to one. This field specifies the size of the frame list that controls which bits in the Frame Index Register should be used for the Frame List Current index. Note that this field is made up from USBCMD bits 15, 3 and 2. 000 1024 elements (4096 bytes) Default value 001 512 elements (2048 bytes) 010 256 elements (1024 bytes) 011 128 elements (512 bytes) 100 64 elements (256 bytes) 101 32 elements (128 bytes) 110 16 elements (64 bytes) 111 8 elements (32 bytes) Only the host controller uses this field.
2 FS0	

Table 32-21. USBCMD Field Descriptions (continued)

Field	Description
1 RST	Controller Reset (RESET)—Read/Write. Software uses this bit to reset the controller. This bit is set to zero by the Host/Device Controller when the reset process is complete. Software cannot terminate the reset process early by writing a zero to this register. Host Controller: When software writes a one to this bit, the Host Controller resets its internal pipelines, timers, counters, state machines, and so on. to their initial value. Any transaction currently in progress on USB is immediately terminated. A USB reset is not driven on downstream ports. Software should not set this bit to a one when the HCHalted bit in the USBSTS register is a zero. Attempting to reset an actively running host controller will result in undefined behavior. Device Controller: When software writes a one to this bit, the Device Controller resets its internal pipelines, timers, counters, state machines, and so on. to their initial value. Writing a one to this bit when the device is in the attached state is not recommended, since the effect on an attached host is undefined. To ensure that the device is not in an attached state before initiating a device controller reset, all primed endpoints should be flushed and the USBCMD Run/Stop bit should be set to 0.
0 RS	Run/Stop (RS)—Read/Write. Default 0b. 1=Run. 0=Stop. Host Controller: When set to a 1, the Host Controller proceeds with the execution of the schedule. The Host Controller continues execution as long as this bit is set to a one. When this bit is set to 0, the Host Controller completes the current transaction on the USB and then halts. The HC Halted bit in the status register indicates when the Host Controller has finished the transaction and has entered the stopped state. Software should not write a one to this field unless the host controller is in the Halted state (that is, HCHalted in the USBSTS register is a one). Device Controller: Writing a one to this bit will cause the device controller to enable a pull-up on D+ and initiate an attach event. This control bit is not directly connected to the pull-up enable, as the pull-up will become disabled upon transitioning into high-speed mode. Software should use this bit to prevent an attach event before the device controller has been properly initialized. Writing a 0 to this causes a detach event.

32.9.5.2 USBSTS

This register indicates various states of the Host/Device Controller and any pending interrupts. This register does not indicate status resulting from a transaction on the serial bus. Software clears certain bits in this register by writing a 1 to them.

Figure 32-25 shows the register; Table 32-22 provides its field descriptions.

0x43F8_8144 (USBSTS) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	AS	PS	RCL	HCH	0	ULPII	0	SLI	SRI	URI	AAI	SEI	FRI	PCI	UEI	UI
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-25. USBSTS—USB Status

Table 32-22. USBSTS Field Descriptions

Field	Description
31–16	Reserved. These bits are reserved and should be set to zero.
15 AS	Asynchronous Schedule Status—Read Only. 0=Default. This bit reports the current real status of the Asynchronous Schedule. When set to zero the asynchronous schedule status is disabled and if set to one the status is enabled. The Host Controller is not required to immediately disable or enable the Asynchronous Schedule when software transitions the Asynchronous Schedule Enable bit in the USBCMD register. When this bit and the Asynchronous Schedule Enable bit are the same value, the Asynchronous Schedule is either enabled (1) or disabled (0). Only used by the host controller.
14 PS	Periodic Schedule Status—Read Only. 0=Default. This bit reports the current real status of the Periodic Schedule. When set to zero the periodic schedule is disabled, and if set to one the status is enabled. The Host Controller is not required to immediately disable or enable the Periodic Schedule when software transitions the Periodic Schedule Enable bit in the USBCMD register. When this bit and the Periodic Schedule Enable bit are the same value, the Periodic Schedule is either enabled (1) or disabled (0). Only used by the host controller.
13 RCL	Reclamation—Read Only. 0=Default. This is a read-only status bit used to detect an empty asynchronous schedule. Only used by the host controller.
12 HCH	HCHalted—Read Only. 1=Default. This bit is a zero whenever the Run/Stop bit is a one. The Host Controller sets this bit to one after it has stopped executing because of the Run/Stop bit being set to 0, either by software or by the Host Controller hardware (for example, internal error). Only used by the host controller.
11	Reserved. These bits are reserved and should be set to zero.

Table 32-22. USBSTS Field Descriptions (continued)

Field	Description
10 ULPII	ULPI Interrupt—R/WC. 0=Default. When the ULPI Viewport is present in the design, an event completion will set this interrupt. Used by both host and device controller. Only present in designs where configuration constant VUSB_HS_PHY_ULPI = 1.
9	Reserved. These bits are reserved and should be set to zero.
8 SLI	DCSuspend—R/WC. 0=Default. When a device controller enters a suspend state from an active state, this bit will be set to a one. The device controller clears the bit upon exiting from a suspend state. Only used by the device controller.
7 SRI	SOF Received—R/WC. 0=Default. When the device controller detects a Start Of (micro) Frame, this bit will be set to a one. When a SOF is extremely late, the device controller will automatically set this bit to indicate that an SOF was expected. Therefore, this bit will be set roughly every 1ms in device FS mode and every 125ms in HS mode and will be synchronized to the actual SOF that is received. Since the device controller is initialized to FS before connect, this bit will be set at an interval of 1ms during the prelude to connect and chirp. In host mode, this bit will be set every 125us and can be used by host controller driver as a time base. Software writes a 1 to this bit to clear it. This is a non-EHCI status bit.
6 URI	USB Reset Received—R/WC. 0=Default. When the device controller detects a USB Reset and enters the default state, this bit will be set to a one. Software can write a 1 to this bit to clear the USB Reset Received status bit. Only used by the device controller.
5 AAI	Interrupt on Async Advance—R/WC. 0=Default. System software can force the host controller to issue an interrupt the next time the host controller advances the asynchronous schedule by writing a one to the Interrupt on Async Advance Doorbell bit in the USBCMD register. This status bit indicates the assertion of that interrupt source. Only used by the host controller.
4 SEI	System Error— R/WC. This bit is not used in this implementation and will always be set to “0”.
3 FRI	Frame List Rollover—R/WC. The Host Controller sets this bit to a one when the Frame List Index rolls over from its maximum value to zero. The exact value at which the rollover occurs depends on the frame list size. For example. If the frame list size (as programmed in the Frame List Size field of the USBCMD register) is 1024, the Frame Index Register rolls over every time FRINDEX [1 3] toggles. Similarly, if the size is 512, the Host Controller sets this bit to a one every time FHINDEX [12] toggles. Only used by the host controller.

Table 32-22. USBSTS Field Descriptions (continued)

Field	Description
2 PCI	Port Change Detect—R/WC. The Host Controller sets this bit to a one when on any port a Connect Status occurs, a Port Enable/Disable Change occurs, or the Force Port Resume bit is set as the result of a J-K transition on the suspended port. The Device Controller sets this bit to a one when the port controller enters the full or high-speed operational state. When the port controller exits the full or high-speed operation states due to Reset or Suspend events, the notification mechanisms are the USB Reset Received bit and the DCSuspend bits respectively. This bit is not EHCI compatible.
1 UEI	USB Error Interrupt (USBERRINT)—R/WC. When completion of a USB transaction results in an error condition, this bit is set by the Host/Device Controller. This bit is set along with the USBINT bit, if the TD on which the error interrupt occurred also had its interrupt on complete (IOC) bit set See Reference Host Operation Model: Transfer/Transaction Based Interrupt (that is, 4.15.1 in EHCI Enhanced Host Controller Interface Specification for Universal Serial Bus, Revision 0.95, November 2000, Intel Corporation. http://www.intel.com) for a complete list of host error interrupt conditions. The device controller detects resume signaling only.
0 UI	USB Interrupt (USBINT)—R/WC. This bit is set by the Host/Device Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) has an interrupt on complete (IOC) bit set. This bit is also set by the Host/Device Controller when a short packet is detected. A short packet is when the actual number of bytes received was less than the expected number of bytes.

32.9.5.3 USBINTR

The interrupts to software are enabled with this register. An interrupt is generated when a bit is set and the corresponding interrupt is active. The USB Status register (USBSTS) still shows interrupt sources even if they are disabled by the USBINTR register, allowing polling of interrupt events by the software.

Figure 32-26 shows the register; Table 32-23 provides its field descriptions.

0x43F8_8148 (USBINTR) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	ULPIE	0	SLE	SRE	URE	AAE	SEE	FRE	PCE	UEE	UE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-26. USBINTR—USB Interrupt Enable

Table 32-23. USBINTR Field Descriptions

Field	Description	
31–11	Reserved.	These bits are reserved and should be set to zero.
10 ULPIE	ULPI Enable	When this bit is a one, and the ULPI Interrupt bit in the USBSTS register transitions, the controller will issue and interrupt. The interrupt is acknowledged by software writing a one to the ULPI Interrupt bit. Used by both host and device controller. Only present in designs where configuration constant VUSB_HS_PHY_ULPI = 1.
9	Reserved.	These bits are reserved and should be set to zero.
8 SLE	Sleep Enable	When this bit is a one, and the *DCSuspend* bit in the USBSTS register transitions, the device controller will issue an interrupt. The interrupt is acknowledged by software writing a one to the *DCSuspend* bit. Only used by the device controller.
7 SRE	SOF Received Enable	When this bit is a one, and the *SOF Received* bit in the USBSTS register is a one, the device controller will issue an interrupt. The interrupt is acknowledged by software clearing the *SOF Received* bit.
6 URE	USB Reset Enable	When this bit is a one, and the *USB Reset Received* bit in the USBSTS register is a one, the device controller will issue an interrupt. The interrupt is acknowledged by software clearing the *USB Reset Received* bit. Only used by the device controller.
5 AAE	Interrupt on Async Advance Enable	When this bit is a one, and the Interrupt on *Async Advance* bit in the USBSTS register is a one, the host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the Interrupt on *Async Advance* bit. Only used by the host controller.

Table 32-23. USBINTR Field Descriptions (continued)

Field		Description
4 SEE	System Error Enable	When this bit is a one, and the *System Error* bit in the USBSTS register is a one, the host/device controller will issue an interrupt. The interrupt is acknowledged by software clearing the *System Error* bit.
3 FRE	Frame List Rollover Enable	When this bit is a one, and the *Frame List Rollover* bit in the USBSTS register is a one, the host controller will issue an interrupt. The interrupt is acknowledged by software clearing the *Frame List Rollover* bit. Only used by the host controller.
2 PCE	Port Change Detect Enable	When this bit is a one, and the *Port Change Detect* bit in the USBSTS register is a one, the host/device controller will issue an interrupt. The interrupt is acknowledged by software clearing the *Port Change Detect* bit.
1 UEE	USB Error Interrupt Enable	When this bit is a one, and the *USBERRINT* bit in the USBSTS register is a one, the host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the *USBERRINT* bit in the USBSTS register.
0 UE	USB Interrupt Enable	When this bit is a one, and the *USBINT* bit in the USBSTS register is a one, the host/device controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the *USBINT* bit.

32.9.5.4 FRINDEX

Attribute: Read/Write in host mode, Read in device mode

This register is used by the host controller to index the periodic frame list. The register updates every 125 microseconds (once each micro-frame). Bits [N: 3] are used to select a particular entry in the Periodic Frame List during periodic schedule execution. The number of bits used for the index depends on the size of the frame list as set by system software in the Frame List Size field in the USBCMD register.

This register must be written as a DWord. Byte writes produce-undefined results. This register cannot be written unless the Host Controller is in the 'Halted' state as indicated by the HCHalted bit. A write to this register while the Run/Stop hit is set to a one produces undefined results. Writes to this register also affect the SOF value.

In device mode this register is read only and, the device controller updates the FRINDEX [13:3] register from the frame number indicated by the SOF marker. Whenever a SOF is received by the USB bus, FRINDEX [13:3] will be checked against the SOF marker. If FRINDEX [13:3] is different from the SOF marker, FRINDEX [13:3] will be set to the SOF value and FRINDEX [2:0] will be set to zero (that is, SOF for 1 ms frame). If FRINDEX [13:3] is equal to the SOF value, FRINDEX [2:0] will be incremented (that is, SOF for 125 us micro-frame).

Figure 32-27 shows the register; Table 32-24 provides its field descriptions.

0x43F8_814C (FRINDEX) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	—	—	—	—	—	—	—	—	—	—	—	0	—	—	—	—

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	FRINDEX[13:0] [Read/Write in host mode, Read in device mode]													
W			FRINDEX[13:0] [Read/Write in host mode]													
Reset	—	—	—	—	—	—	—	—	—	—	—	0	—	—	—	—

Figure 32-27. FRINDEX—USB Frame Index

Table 32-24. FRINDEX Field Descriptions

Field	Description
31–14	Reserved. These bits are reserved and should be set to zero.
13–0 FRINDEX	Frame Index. The value, in this register, increments at the end of each time frame (for example, micro-frame). Bits [N: 3] are used for the Frame List current index. This means that each location of the frame list is accessed 8 times (frames or micro-frames) before moving to the next index. The following illustrates values of N based on the value of the Frame List Size field in the USBCMD register, when used in host mode. 000b (1024)12 001b (512)11 010b (256)10 011b (128)9 100b (64)8 101b (32)7 110b (16)6 111b (8)5 In device mode the value is the current frame number of the last frame transmitted. It is not used as an index. In either mode bits 2:0 indicate the current microframe.

32.9.5.5 PERIODICLISTBASE; DEVICEADDR

The PERIODICLISTBASE and DEVICEADDR registers are shared between the host controller and the device controller operation.

32.9.5.5.1 Host Controller (PERIODICLISTBASE)

This 32-bit register contains the beginning address of the Periodic Frame List in the system memory. HCD loads this register prior to starting the schedule execution by the Host Controller. The memory structure referenced by this physical memory pointer is assumed to be 4-Kbyte aligned. The contents of this register

are combined with the Frame Index Register (FRINDEX) to enable the Host Controller to step through the Periodic Frame List in sequence. Attribute: Read/Write (Writes must be DWord Writes).

Figure 32-28 shows the register; Table 32-25 provides its field descriptions.

0x43F8_8154 (PERIODICLISTBASE) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PERBASE[31:12]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PERBASE[31:12]				0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-28. PERIODICLISTBASE—Host Controller Frame List Base Address

Table 32-25. PERIODICLISTBASE Field Descriptions

Field	Description
31–12 PERBASE	Base Address (Low). These bits correspond to memory address signals [31:12], respectively. Only used by the host controller.
11–0	Reserved. Must be written as zeros. During run-time, the values of these bits are undefined.

32.9.5.5.2 Device Controller (USB DEVICEADDR)

The upper seven bits of this register represent the device address. After any controller reset or a USB reset, the device address is set to the default address (0). The default address will match all incoming addresses. Software shall reprogram the address after receiving a SET_ADDRESS descriptor.

Figure 32-29 shows the register; Table 32-26 provides its field descriptions.

0x43F8_8154 (DEVICEADDR) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	USBADR[31:25]							0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-29. DEVICEADDR—Device Controller USB Device Address

Table 32-26. DEVICEADDR Field Descriptions

Field	Description
31–25 USBADR	Device Address. These bits correspond to the USB device address
24–0	Reserved. Must be written as zeros. During run-time, the values of these bits are undefined.

32.9.5.6 ASYNCLISTADDR and ENDPOINTLISTADDR

The ASYNCLISTADDR and ENDPOINTLISTADDR registers are shared between the host controller and the device controller operation.

32.9.5.6.1 Host Controller (ASYNCLISTADDR)

This 32-bit register contains the address of the next asynchronous queue head to be executed by the host. Bits [4:0] of this register cannot be modified by the system software and will always return a zero when read. Attribute: Read/Write (Writes must be DWord Writes).

Figure 32-30 shows the register; Table 32-27 provides its field descriptions.

0x43F8_8158 (ASYNCLISTADDR) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ASYBASE[31:5]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ASYBASE[31:5]										0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-30. ASYNCLISTADDR—Host Controller Next Asynchronous Address

Table 32-27. ASYNCLISTADDR Field Descriptions

Field	Description
31–5 ASYBASE[31:5]	Link Pointer Low (LPL). These bits correspond to memory address signals [31:5], respectively. This field may only reference a Queue Head (OH). Only used by the host controller.
4–0	Reserved. These bits are reserved and their value has no effect on operation.

32.9.5.6.2 Device Controller (ENDPOINTLISTADDR)

In device mode, this register contains the address of the top of the endpoint list in system memory. Bits [10:0] of this register cannot be modified by the system software and will always return a zero when read.

The memory structure referenced by this physical memory pointer is assumed 64-byte.

Figure 32-31 shows the register.

0x43F8_8158 (ENDPOINTLISTADDR) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	EPBASE[31:11]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	EPBASE[31:11]					0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-31. ENDPOINTLISTADDR—Device Controller Endpoint List Address

32.9.5.7 BURSTSIZE

This register is used to control dynamically change the burst size used during data movement on the initiator (master) interface. Attribute: Read/Write (Writes must be DWord Writes).

Figure 32-32 shows the register; Table 32-28 provides its field descriptions.

0x43F8_8160 (BURSTSIZE) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TXPBURST
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TXPBURST								RXPBURST							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-32. BURSTSIZE—Host Controller Embedded TT Async. Buffer Status

Table 32-28. BURSTSIZE Field Descriptions

Field	Description
31–17	Reserved. These bits are reserved and their value has no effect on operation.
16–8 TXPBURST	Programmable TX Burst Length. (Read/Write) Default is the constant VUSB_HS_TX_BURST. This register represents the maximum length of a the burst in 32-bit words while moving data from system memory to the USB bus.
7–0 RXPBURST	Programmable RX Burst Length. (Read/Write) Default is the constant VUSB_HS_RX_BURST. This register represents the maximum length of a the burst in 32-bit words while moving data from the USB bus to system memory.

32.9.5.8 TXFILLTUNING

Figure 32-33 shows the TXFILLTUNING register.

0x43F8_8164 (TXFILLTUNING) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	TXFIFOTHR					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	TXSCHEAL					TXSCHOH							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-33. TXFILLTUNING Register

The fields in this register control performance tuning associated with how the host controller posts data to the TX latency FIFO before moving the data onto the USB bus. The specific areas of performance include the how much data to post into the FIFO and an estimate for how long that operation should take in the target system.

Definitions:

T_0 = Standard packet overhead

T_1 = Time to send data payload

T_{ff} = Time to fetch packet into TX FIFO up to specified level.

T_s = Total Packet Flight Time (send-only) packet

$T_s = T_0 + T_1$

T_p = Total Packet Time (fetch and send) packet

$T_p = T_{ff} + T_0 + T_1$

Upon discovery of a transmit (OUT/SETUP) packet in the data structures, host controller checks to ensure T_p remains before the end of the [micro]frame. If so it proceeds to pre-fill the TX FIFO. If at anytime during the pre-fill operation the time remaining the [micro]frame is $< T_s$ then the packet attempt ceases and the packet is tried at a later time. Although this is not an error condition and the host controller will eventually recover, a mark will be made the scheduler health counter to note the occurrence of a “back-off” event. When a back-off event is detected, the partial packet fetched may need to be discarded from the latency buffer to make room for periodic traffic that will begin after the next SOF. Too many back-off events can waste bandwidth and power on the system bus and thus should be minimized (not necessarily eliminated). Back-offs can be minimized with use of the TSCHHEALTH (T_{ff}) described below.

32.9.5.9 ULPI VIEWPORT (Optional)

The register provides indirect access to the ULPI PHY register set. Although the core performs access to the ULPI PHY register set, there may be extraordinary circumstances where software may need direct access.

CAUTION

Writes to the ULPI through the VIEWPORT can substantially harm standard USB operations. Currently no usage model has been defined where software should need to execute writes directly to the ULPI.

NOTE

Executing read operations though the ULPI Viewport should have no harmful side effects to standard USB operations.

The ULPI Viewport is only synthesized in the design if the ULPI option has been purchased and installed and the constant `VUSB_HS_PHY_ULPI` is set to `1`. If the ULPI interface is not enabled, this register will always read zeros.

There are two operations that can be performed with the ULPI Viewport, wakeup and read /write operations. The wakeup operation is used to put the ULPI interface into normal operation mode and re-enable the clock if necessary. A wakeup operation is required before accessing the registers when the ULPI interface is operating in low power mode, serial mode, or car-kit mode. The ULPI state can be determined by reading the sync. state bit (ULPISS). If this bit is a one, then ULPI interface is running in normal operation mode and can accept read/write operations. If the ULPISS indicates a ‘0’ then read/write operations will not be able execute. Undefined behavior will result if ULPISS = 0 and a read or write operation is performed. To execute a wakeup operation, write all 32-bits of the ULPI Viewport where ULPIPORT is constructed appropriately and the ULPIWU bit is a ‘1’ and ULPIRUN bit is a ‘0’. Poll the ULPI Viewport until ULPIWU is zero for the operation to complete.

To execute a read or write operation, write all 32-bits of the ULPI Viewport where ULPIDATWR, ULPIADDR, ULPIPORT, ULPIRW are constructed appropriately and the ULPIRUN bit is a ‘1’. Poll the ULPI Viewport until ULPIRUN is zero for the operation to complete. Once ULPIRUN is zero, the ULPIDATRD will be valid if the operation was a read.

The polling method above could also be replaced and interrupt driven using the ULPI interrupt defined in the USBSTS and USBINTR registers. When a wakeup or read/write operation complete, the ULPI interrupt will be set.

Figure 32-34 shows the register; Table 32-29 provides its field descriptions.

0x43F8_8170 (ULPI) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ULPIWU	ULPIRUN	ULPIRW	0	ULPIS	ULPIPORT			ULPIPORT							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ULPIDATRD								ULPIDATWR							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-34. ULPI VIEWPORT

Table 32-29. ULPI Field Descriptions

Field	Descriptions
31 ULPIWU	ULPI Wakeup—Read/Write. Writing the ‘1’ to this bit will begin the wakeup operation. The bit will automatically transition to 0 after the wakeup is complete. Once this bit is set, the driver cannot set it back to ‘0’. **Note:** The driver must never execute a wakeup and a read/write operation at the same time.
30 ULPIRUN	ULPI Read/Write Run—Read/Write. Writing the ‘1’ to this bit will begin the read/write operation. The bit will automatically transition to 0 after the read/write is complete. Once this bit is set, the driver cannot set it back to ‘0’. **Note:** The driver must never execute a wakeup and a read/write operation at the same time.
29 ULPIRW	ULPI Read/Write Control—Read/Write. (0)—Read; (1)—Write. This bit selects between running a read or write operation.
28	Reserved. This bit is reserved and its value has no effect on operation.
27 ULPISS	ULPI Sync State—Read Only. (1)—Normal Sync. State. (0) In another state (that is, car-kit, serial, low power). This bit represents the state of the ULPI interface. Before reading this bit, the ULPIPORT field should be set accordingly if used with the multi-port host. Otherwise, this field should always remain 0.
26–24 ULPIPORT	ULPI Port Number—Read/Write. For the wakeup or read/write operation to be executed, this value selects the port number the ULPI PHY is attached to in the multi-port host. The range is 0 to 7. This field should always be written as a 0 for the non-multi port products.
23–16 ULPIADDR	ULPI Data Address—Read/Write. When a read or write operation is commanded, the address of the operation is written to this field.
15–8 ULPIDATRD	ULPI Data Read—Read Only. After a read operation completes, the result is placed in this field.
7–0 ULPIDATWR	ULPI Data Write—Read/Write. When a write operation is commanded, the data to be sent is written to this field.

32.9.5.9.1 Host Controller

A host controller must implement one to eight port registers. The number of port registers implemented by a particular instantiation of a host controller is documented in the HCSPARAMs register. Software uses this information as an input parameter to determine how many ports need service.

This register is only reset when power is initially applied or in response to a controller reset. The initial conditions of a port are as follows:

- No device connected
- Port disabled

If the port has port power control, this state remains until software applies power to the port by setting port power to one.

32.9.5.9.2 Device Controller

A device controller must implement only port register one and it does not support power control. Port control in device mode is only used for status port reset, suspend, and current connect status. It is also used to initiate test mode or force signaling and allows software to put the PHY into low power suspend mode and disable the PHY clock.

Figure 32-35 shows the register; Table 32-30 provides its field descriptions.

0x43F8_8184 (PORTSCx) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PTS		STS	PTW	PSPD			0	PFSC	PHCD	WKOC	WKDS	WKCN	PTC[3:0]		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PIC		PO	PP	LS		HSP	PR	SUSP	FPR	OCC	OCA	PEC	PE	CSC	CCS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-35. PORTSCx—Port Status Control[1:8]

Table 32-30. PORTSCx Field Descriptions

Field	Descriptions
31–30 PTS	Parallel Transceiver Select—Read/Write. This register bit pair is used in conjunction with the configuration constant VUSB_HS_PHY_TYPE to control which parallel transceiver interface is selected. If VUSB_HS_PHY_TYPE is set for 0,1,2, or 3 then this bit is read only. If VUSB_HS_PHY_TYPE is 3,4,5, or 6 then this bit is read/write. This field is reset to: “00” if VUSB_HS_PHY_TYPE = 0,4—UTMI/UTMI+ “01” if VUSB_HS_PHY_TYPE = 1,5—Philips Classic “10” if VUSB_HS_PHY_TYPE = 2,6—ULPI “11” if VUSB_HS_PHY_TYPE = 3,7—Serial/1.1 PHY (FS Only) This bit is not defined in the EHCI specification.
29 STS	Serial Transceiver Select—Read/Write. This register bit is used in conjunction with the configuration constant VUSB_HS_PHY_SERIAL to control whether the parallel or serial transceiver interface is selected for FS and LS operation. The Serial Interface Engine can be used in combination with the UTMI+ or ULPI physical interface to provide FS/LS signaling instead of the parallel interface. If VUSB_HS_PHY_SERIAL is set for 0 or 1 then this bit is read only. If VUSB_HS_PHY_SERIAL is 3 or 4 then this bit is read/write. This bit has no effect unless Parallel Transceiver Select is set to UTMI+ or ULPI. The Philips and Serial/1.1 physical interface will use the Serial Interface Engine for FS/LS signaling regardless of this bit value. Note: This bit is reserved for future operation and is a placeholder adding dynamic use of the serial engine in accord with UMTI+ and ULPI characterization logic. This bit is not defined in the EHCI specification.
28 PTW	Parallel Transceiver Width—Read/Write. This register bit is used in conjunction with the configuration constant VUSB_HS_PHY8_16 to control whether the data bus width of the UTMI transceiver interface. If VUSB_HS_PHY8_16 is set for 0 or 1 then this bit is read only. If VUSB_HS_PHY8_16 is 2 or 3 then this bit is read/write. This bit is reset to 1 if VUSB_HS_PHY8_16 selects a default UTMI interface width of 16-bits else it is reset to 0. Writing this bit to 0 selects the 8-bit [60MHz] UTMI interface. Writing this bit to 1 selects the 16-bit [30MHz] UTMI interface. This bit has no effect if the Philips or Serial interfaces are selected. This bit is not defined in the EHCI specification.
27–26 PSPD	Port Speed—Read Only. This register field indicates the speed at which the port is operating. For HS mode operation in the host controller and HS/FS operation in the device controller the port routing steers data to the Protocol engine. For FS and LS mode operation in the host controller, the port routing steers data to the Protocol Engine w/ Embedded Transaction Translator. 00—Full Speed 01—Low Speed 10—High Speed This bit is not defined in the EHCI specification.

Table 32-30. PORTSCx Field Descriptions (continued)

Field	Descriptions
24 PFSC	Port Force Full Speed Connect—Read/Write. Default = 0b. Writing this bit to a 1b will force the port to only connect at Full Speed. It disables the chirp sequence that allows the port to identify itself as High Speed. This is useful for testing FS configurations with a HS host, hub or device. This bit is not defined in the EHCI specification. This bit is for debugging purposes.
23 PHCD	PHY Low Power Suspend—Clock Disable (PLPSCD)—Read/Write. Default = 0b. Writing this bit to a 1b will disable the PHY clock. Writing a 0b enables it. Reading this bit will indicate the status of the PHY clock. NOTE: The PHY clock cannot be disabled if it is being used as the system clock. In device mode, The PHY can be put into Low Power Suspend—Clock Disable when the device is not running (USBCMD Run/Stop=0b) or the host has signaled suspend (PORTSC SUSPEND=1b). Low power suspend will be cleared automatically when the host has signaled resume if using a circuit similar to that in. Before forcing a resume from the device, the device controller driver must clear this bit. In host mode, the PHY can be put into Low Power Suspend—Clock Disable when the downstream device has been put into suspend mode or when no downstream device is connected. Low power suspend is completely under the control of software. This bit is not defined in the EHCI specification.
22 WKOC	Wake on Over-current Enable (WKOC_E)—Read/Write. Default = 0b. Writing this bit to a one enables the port to be sensitive to over-current conditions as wake-up events. This field is zero if *Port Power (PP)* is zero. This bit is output from the controller as signal pwrctl_wake_ovrcurr_en (OTG/host core only) for use by an external power control circuit.
21 WKDC	Wake on Disconnect Enable (WKDSCNNT_E)—Read/Write. Default=0b. Writing this bit to a one enables the port to be sensitive to device disconnects as wake-up events. This field is zero if *Port Power (PP)* is zero or in device mode. This bit is output from the controller as signal pwrctl_wake_dscnnt_en (OTG/host core only) for use by an external power control circuit.
20 WKCN	Wake on Connect Enable (WKCNNT_E)—Read/Write. Default=0b. Writing this bit to a one enables the port to be sensitive to device connects as wake-up events. This field is zero if *Port Power (PP)* is zero or in device mode. This bit is output from the controller as signal pwrctl_wake_dscnnt_en (OTG/host core only) for use by an external power control circuit.

Table 32-30. PORTSCx Field Descriptions (continued)

Field	Descriptions
19–16 PTC[3:0]	Port Test Control—Read/Write. Default = 0000b. Any other value than zero indicates that the port is operating in test mode. 0000b TEST_MODE_DISABLE 0001b J_ STATE 0010b K_STATE 0011b SE0 (host)/NAK (device) 0100b Packet 0101b FORCE_ENABLE_HS 0110b FORCE_ENABLE_FS 0111b FORCE_ENABLE_LS 1000b to 1111b Reserved Refer to Chapter 7 of the USB Specification Revision 2.0 Universal Serial Bus Specification, Revision 2.0, April 2000, Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC, Philips. http://www.usb.org for details on each test mode. The FORCE_ENABLE_FS and FORCE ENABLE_LS are extensions to the test mode support specified in the EHCI specification. Writing the PTC field to any of the FORCE_ENABLE_{HS/FS/LS} values will force the port into the connected and enabled state at the selected speed. Writing the PTC field back to TEST_MODE_DISABLE will allow the port state machines to progress normally from that point. *Note: Low speed operations are not supported as a peripheral device.*
15–14 PIC[1:0]	Port Indicator Control—Read/Write. Default = Ob. Writing to this field has no effect if the P_INDICATOR bit in the HCSPARAMS register is a zero. If P_INDICATOR bit is a one, then the bit is as follows: Bit ValueMeaning 00b Port indicators are off 01b Amber 10b Green 11b Undefined Refer to the USB Specification Revision 2.0 Universal Serial Bus Specification, Revision 2.0, April 2000, Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC, Philips. http://www.usb.org for a description on how these bits are to be used. This field is output from the controller as signals port_ind_ctl_1 and port_ind_ctl_0 for use by an external led driving circuit.
13 PO	Port Owner—Read/Write. Default = 0. This bit unconditionally goes to a 0 when the configured bit in the CONFIGFLAG register makes a 0 to 1 transition. This bit unconditionally goes to 1 whenever the Configured bit is zero System software uses this field to release ownership of the port to a selected host controller (in the event that the attached device is not a high-speed device). Software writes a one to this bit when the attached device is not a high-speed device. A one in this bit means that an internal companion controller owns and controls the port. Port owner handoff is not implemented in this design, therefore this bit will always be 0.

Table 32-30. PORTSCx Field Descriptions (continued)

Field	Descriptions
12 PP	Port Power (PP)—Read/Write or Read Only. The function of this bit depends on the value of the Port Power Switching (PPC) field in the HCSPARAMS register. The behavior is as follows: PPC PP Operation 0b 0b Read Only— A device controller with no OTG capability does not have port power control switches. 1b 1b/0b—RW. Host/OTG controller requires port power control switches. This bit represents the current setting of the switch (0=off, 1=on). When power is not available on a port (that is, PP equals a 0), the port is non-functional and will not report attaches, detaches, and so on. When an over-current condition is detected on a powered port and PPC is a one, the PP bit in each affected port may be transitioned by the host controller driver from a one to a zero (removing power from the port). This feature is implemented in the host/OTG controller (PPC = 1). In a device only implementation port power control is not necessary, thus PPC and PP = 0.
11–10 LS[1:0]	Line Status—Read Only. These bits reflect the current logical levels of the D+ (bit 11) and D- (bit 10) signal lines. The encoding of the bits are as follows: Bits [11:10]Meaning 00b SE0 10b J-state 01b K-state 11b Undefined In host mode, the use of linestate by the host controller driver is not necessary (unlike EHCI), because the port controller state machine and the port routing manage the connection of LS and FS. In device mode, the use of linestate by the device controller driver is not necessary.
9 HSP	High-Speed Port—Read Only. Default = 0b. When the bit is one, the host/device connected to the port is in high-speed mode and if set to zero, the host/device connected to the port is not in a high-speed mode. **Note:** HSP is redundant with PSPD(27:26) but will remain in the design for compatibility. This bit is not defined in the EHCI specification.
8 PR	Port Reset This field is zero if *Port Power (PP)* is zero. In Host Mode: Read/Write. 1=Port is in Reset. 0=Port is not in Reset. Default 0. When software writes a one to this bit the bus-reset sequence as defined in the USB Specification Revision 2.0 is started. *This bit will automatically change to zero after the reset sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a zero after the reset duration is timed in the driver.* In Device Mode: This bit is a read only status bit. Device reset from the USB bus is also indicated in the USBSTS register.

Table 32-30. PORTSCx Field Descriptions (continued)

Field	Descriptions
7 SUSP	Suspend. In Host Mode: Read/Write. 1=Port in suspend state. 0=Port not in suspend state. Default=0. Port Enabled Bit and Suspend bit of this register define the port states as follows: Bits [Port Enabled, Suspend]: Port State 0x Disable 10 Enable 11 Suspend When in suspend state, downstream propagation of data is blocked on this port, except for port reset. The blocking occurs at the end of the current transaction if a transaction was in progress when this bit was written to 1. In the suspend state, the port is sensitive to resume detection. Note that the bit status does not change until the port is suspended and that there may be a delay in suspending a port if there is a transaction currently in progress on the USB. The host controller will unconditionally set this bit to zero when software sets the *Force Port Resume* bit to zero. The host controller ignores a write of zero to this bit. If host software sets this bit to a one when the port is not enabled (that is, *Port enabled* bit is a zero) the results are undefined. This field is zero if *Port Power (PP)* is zero in host mode. In Device Mode: Read Only. 1=Port in suspend state. 0=Port not in suspend state. Default=0. In device mode this bit is a read only status bit.
6 FPR	Force Port Resume —Read/Write. 1= Resume detected/driven on port. 0=No resume (K-state) detected/driven on port. Default = 0. In Host Mode: Software sets this bit to one to drive resume signaling. The Host Controller sets this bit to one if a J-to-K transition is detected while the port is in the Suspend state. When this bit transitions to a one because a J-to-K transition is detected, the *Port Change Detect* bit in the USBSTS register is also set to one. *This bit will automatically change to zero after the resume sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a zero after the resume duration is timed in the driver.* Note that when the Host controller owns the port, the resume sequence follows the defined sequence documented in the USB Specification Revision 2.0. The resume signaling (Full-speed ‘K’) is driven on the port as long as this bit remains a one. This bit will remain a one until the port has switched to the high-speed idle. Writing a zero has no affect because the port controller will time the resume operation clear the bit the port control state switches to HS or FS idle. This field is zero if *Port Power (PP)* is zero in host mode. This bit is not-EHCI compatible. In Device mode: After the device has been in Suspend State for 5ms or more, software must set this bit to one to drive resume signaling before clearing. The Device Controller will set this bit to one if a J-to-K transition is detected while the port is in the Suspend state. The bit will be cleared when the device returns to normal operation. Also, when this bit transitions to a one because a J-to-K transition detected, the *Port Change Detect* bit in the USBSTS register is also set to one.

Table 32-30. PORTSCx Field Descriptions (continued)

Field	Descriptions
5 OCC	Over-current Change—R/WC. Default=0. 1=This bit gets set to one when there is a change to Over-current Active. Software clears this bit by writing a one to this bit position. For host/OTG implementations the user can provide over-current detection to the vbus_pwr_fault input for this condition. For device-only implementations this bit shall always be 0.
4 OCA	Over-current Active—Read Only. Default 0. 1=This port currently has an over-current condition. 0=This port does not have an over-current condition. This bit will automatically transition from one to zero when the over current condition is removed. For host/OTG implementations the user can provide over-current detection to the vbus_pwr_fault input for this condition. For device-only implementations this bit shall always be 0.
3 PEC	Port Enable/Disable Change—R/WC. 1=Port enabled/disabled status has changed. 0=No change. Default = 0. In Host Mode: For the root hub, this bit gets set to a one only when a port is disabled due to disconnect on the port or due to the appropriate conditions existing at the EOF2 point (See Chapter 11 of the USB Specification). Software clears this by writing a one to it. This field is zero if *Port Power (PP)* is zero. In Device mode: The device port is always enabled. (This bit will be zero)
2 PE	Port Enabled/Disabled—Read/Write. 1=Enable. 0=Disable. Default 0. In Host Mode: Ports can only be enabled by the host controller as a part of the reset and enable. Software cannot enable a port by writing a one to this field. Ports can be disabled by either a fault condition (disconnect event or other fault condition) or by the host software. Note that the bit status does not change until the port state actually changes. There may be a delay in disabling or enabling a port due to other host controller and bus events. When the port is disabled, (0b) downstream propagation of data is blocked except for reset. This field is zero if *Port Power (PP)* is zero in host mode. In Device Mode: The device port is always enabled. (This bit will be one)

Table 32-30. PORTSCx Field Descriptions (continued)

Field	Descriptions
1 CSC	Connect Status Change—R/WC. 1 =Change in Current Connect Status. 0=No change. Default 0. In Host Mode: Indicates a change has occurred in the port's Current Connect Status. The host/device controller sets this bit for all changes to the port device connect status, even if system software has not cleared an existing connect status change. For example, the insertion status changes twice before system software has cleared the changed condition, hub hardware will be 'setting' an already-set bit (that is, the bit will remain set). Software clears this bit by writing a one to it. This field is zero if *Port Power (PP)* is zero in host mode. In Device Mode: This bit is undefined in device controller mode.
0 CCS	Current Connect Status—Read Only. In Host Mode: 1=Device is present on port. 0=No device is present. Default = 0. This value reflects the current state of the port, and may not correspond directly to the event that caused the *Connect Status Change* bit (Bit 1) to be set. This field is zero if *Port Power (PP)* is zero in host mode. In Device Mode: 1=Attached. 0=Not Attached. Default=0. A one indicates that the device successfully attached and is operating in either high speed or full speed as indicated by the High Speed Port bit in this register. A zero indicates that the device did not attach successfully or was forcibly disconnected by the software writing a zero to the Run bit in the USBCMD register. It does not state the device being disconnected or suspended.

32.9.5.10 OTGSC

32.9.5.10.1 Host Controller

A host controller implements one On-The-Go (OTG) Status and Control register corresponding to Port 0 of the host controller.

The OTGSC register has four sections:

OTG Interrupt enables (Read/Write)

OTG Interrupt status (Read/Write to Clear)

OTG Status inputs (Read Only)

OTG Controls(Read/Write)

The status inputs are debounced using a 1Msec time constant. Values on the status inputs that do not persist for more than 1Msec will not cause an update of the status input register, or cause an OTG interrupt.

See also USBMODE register.

Figure 32-36 shows the register; Table 32-31 provides its field descriptions.

0x43F8_81A4 (OTGSC) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	DPIE	1msE	BSEIE	BSVIE	ASVIE	AVVIE	IDIE	0	DPIS	1mss	BSEIS	BSVIS	ASVIS	AVVIS	IDIS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	DPS	1msT	BSE	BSV	ASV	AVV	ID	0	0	IDPU	DP	OT	0	VC	VD
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-36. OTGSC—OTG Status Control

Table 32-31. OTGSC Field Descriptions

Field	Description
30 DPIE	Data Pulse Interrupt Enable
29 1msE	1 millisecond timer Interrupt Enable—Read/Write
28 BSEIE	B Session End Interrupt Enable—Read/Write. Setting this bit enables the B session end interrupt.
27 BSVIE	B Session Valid Interrupt Enable—Read/Write. Setting this bit enables the B session valid interrupt.
26 ASVIE	A Session Valid Interrupt Enable—Read/Write. Setting this bit enables the A session valid interrupt.
25 AVVIE	A VBus Valid Interrupt Enable—Read/Write. Setting this bit enables the A VBus valid interrupt.
24 IDIE	USB ID Interrupt Enable—Read/Write. Setting this bit enables the USB ID interrupt.
23	Reserved
22 DPIS	Data Pulse Interrupt Status—Read/Write to Clear. This bit is set when data bus pulsing occurs on DP or DM. Data bus pulsing is only detected when USBMODE.CM = Host (11) and PORTSC(0).PortPower = Off (0). Software must write a one to clear this bit.
21 1msS	1 millisecond timer Interrupt Status—Read/Write to Clear. This bit is set once every millisecond. Software must write a one to clear this bit.
20 BSEIS	B Session End Interrupt Status—Read/Write to Clear. This bit is set when VBus has fallen below the B session end threshold. Software must write a one to clear this bit

Table 32-31. OTGSC Field Descriptions (continued)

Field	Description
19 BSVIS	B Session Valid Interrupt Status—Read/Write to Clear. This bit is set when VBus has either risen above or fallen below the B session valid threshold (0.8 VDC). Software must write a one to clear this bit.
18 ASVIS	A Session Valid Interrupt Status—Read/Write to Clear. This bit is set when VBus has either risen above or fallen below the A session valid threshold (0.8 VDC). Software must write a one to clear this bit.
17 AVVIS	A VBus Valid Interrupt Status—Read/Write to Clear. This bit is set when VBus has either risen above or fallen below the VBus valid threshold (4.4 VDC) on an A device. Software must write a one to clear this bit.
16 IDIS	USB ID Interrupt Status—Read/Write. This bit is set when a change on the ID input has been detected. Software must write a one to clear this bit.
15	Reserved
14 DPS	Data Bus Pulsing Status—Read Only. A ‘1’ indicates data bus pulsing is being detected on the port.
13 1msT	1 millisecond timer toggle—Read Only. This bit toggles once per millisecond.
12 BSE	B Session End—Read Only. Indicates VBus is below the B session end threshold.
11 BSV	B Session Valid—Read Only. Indicates VBus is above the B session valid threshold.
10 ASV	A Session Valid—Read Only. Indicates VBus is above the A session valid threshold.
9 AVV	A VBus Valid—Read Only. Indicates VBus is above the A VBus valid threshold.
8 ID	USB ID—Read Only. 0 = A device 1 = B device
7–6	Reserved
5 IDPU	ID Pull-up—Read/Write. This bit provide control over the ID pull-up resister; 0 = off, 1 = on [default]. When this bit is 0, the ID input will not be sampled.
4 DP	Data Pulsing—Read/Write. Setting this bit causes the pull-up on DP to be asserted for data pulsing during SRP.
3 OT	OTG Termination—Read/Write. This bit must be set when the OTG device is in device mode, this controls the pull-down on DM.
2	Reserved
1 VC	VBUS Charge—Read/Write. Setting this bit causes the VBus line to be charged. This is used for VBus pulsing during SRP.
0 VD	VBUS_Discharge—Read/Write. Setting this bit causes VBus to discharge through a resistor.

32.9.5.11 USBMODE

Figure 32-37 shows the USBMODE register; Table 32-32 provides its field descriptions.

0x43F8_81A8 (USBMODE) Access: User Read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	S	SL	E	CM	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-37. USBMODE—USB Device Mode

Table 32-32. USBMODE Field Descriptions

Field	Description
31–5	Reserved. These bits are reserved and should be set to zero.
4 SDIS	Stream Disable Mode. (0—Inactive [default]; 1—Active) Device Mode: Setting to a ‘1’ disables double priming on both RX and TX for low bandwidth systems. This mode ensures that when the RX and TX buffers are sufficient to contain an entire packet that the standard double buffering scheme is disabled to prevent overruns/underruns in bandwidth limited systems. Note: In High Speed Mode, all packets received will be responded to with a NYET handshake when stream disable is active. Host Mode: Setting to a ‘1’ ensures that overruns/underruns of the latency FIFO are eliminated for low bandwidth systems where the RX and TX buffers are sufficient to contain the entire packet. Enabling stream disable also has the effect of ensuring the TX latency is filled to capacity before the packet is launched onto the USB. Note: Time duration to pre-fill the FIFO becomes significant when stream disable is active. See TXFILLTUNING and TXTTFILLTUNING [MPH Only] to characterize the adjustments needed for the scheduler when using this feature. Note: The use of this feature substantially limits of the overall USB performance that can be achieved.
3 SLOM	Setup Lockout Mode. In device mode, this bit controls behavior of the setup lock mechanism. See Control Endpoint Operation Model. 0—Setup Lockouts On (default); 1—Setup Lockouts Off (DCD requires use of Setup Data Buffer Tripwire in USBCMD)

Table 32-32. USBMODE Field Descriptions (continued)

Field	Description
2 ES	Endian Select—Read/Write. This bit can change the byte alignment of the transfer buffers to match the host microprocessor. The bit fields in the microprocessor interface and the data structures are unaffected by the value of this bit because they are based upon the 32-bit word. 0 Little Endian [Default] 1 Big Endian
1–0 CM[1:0]	Controller Mode—R/WO. Controller mode is defaulted to the proper mode for host only and device only implementations. For those designs that contain both host and device capability, the controller will default to an idle state and will need to be initialized to the desired operating mode after reset. For combination host/device controllers, this register can only be written once after reset. If it is necessary to switch modes, software must reset the controller by writing to the *RESET* bit in the USBCMD register before reprogramming this register. 00 Idle [Default for combination host/device] 01 Reserved 10 Device Controller [Default for device only controller] 11 Host Controller [Default for host only controller]

32.9.5.12 ENDPTSETUPSTAT

Figure 32-38 shows the ENDPTSETUPSTAT register; Table 32-33 provides its field descriptions.

0x43F8_81AC (ENPTSETUPSTAT) Access: Read Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ENDPTSETUPSTAT[15:0]															
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

Figure 32-38. ENDPTSETUPSTAT—Endpoint Setup Status

Table 32-33. ENDPTSETUPSTAT Field Descriptions

Field	Description
31–16	Reserved. These bits are reserved and should be set to zero.
15–0 ENDPTSETUPSTAT	Setup Endpoint Status. For every setup transaction that is received, a corresponding bit in this register is set to one. Software must clear or acknowledge the setup transfer by writing a one to a respective bit after it has read the setup data from Queue head. The response to a setup packet as in the order of operations and total response time is crucial to limit bus time outs while the setup lock our mechanism is engaged. See Managing Endpoints in the Device Operational Model. This register is only used in device mode.

32.9.5.13 ENDPTPRIME

This register is only used in device mode.

Figure 32-39 shows the ENDPTPRIME register; Table 32-34 provides its field descriptions.

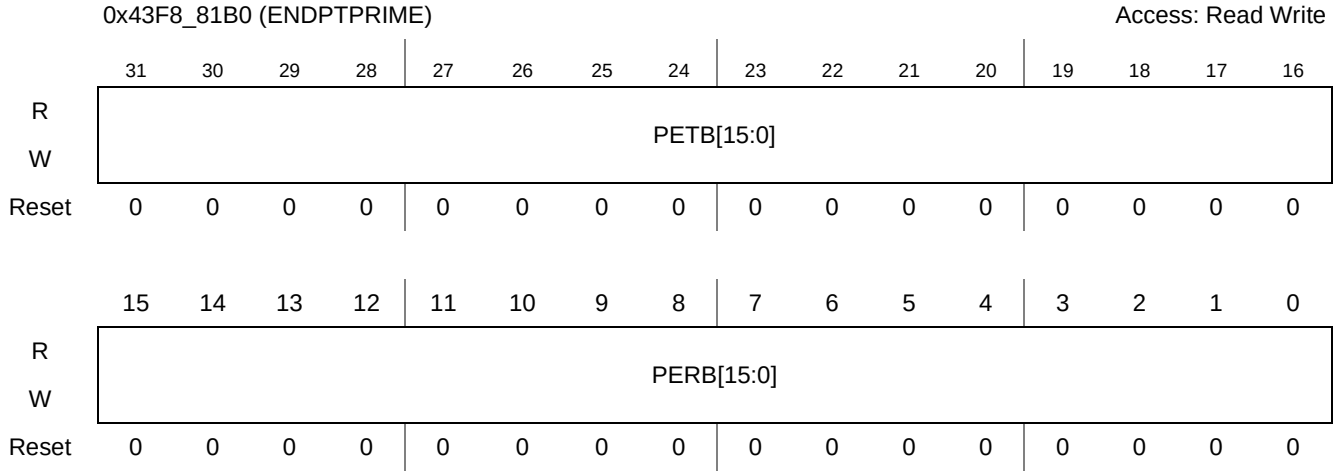


Figure 32-39. ENDPTPRIME—Endpoint Initialization

Table 32-34. ENDPTPRIM Field Descriptions

Field	Description
31–16 PETB [15:0]	Prime Endpoint Transmit Buffer—R/WS. For each endpoint a corresponding bit is used to request that a buffer prepared for a transmit operation to respond to a USB IN/INTERRUPT transaction. Software should write a one to the corresponding bit when posting a new transfer descriptor to an endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when the associated endpoint(s) is (are) successfully primed. Note: These bits will be momentarily set by hardware during hardware re-priming operations when a dTD is retired, and the dQH is updated. PETB[15] – Endpoint #15 PETB[1] – Endpoint #1 PETB[0] – Endpoint #0
15–0 PERB	Prime Endpoint Receive Buffer—R/WS. For each endpoint, a corresponding bit is used to request a buffer prepare for a receive operation for when a USB host initiates a USB OUT transaction. Software should write a one to the corresponding bit whenever posting a new transfer descriptor to an endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when the associated endpoint(s) is (are) successfully primed. Note: These bits will be momentarily set by hardware during hardware re-priming operations when a dTD is retired, and the dQH is updated. Bit 15 – Endpoint #15 Bit 1 – Endpoint #1 Bit 0 – Endpoint #0

32.9.5.14 ENDPTFLUSH

This register is only used in device mode.

Figure 32-40 shows the ENDPTFLUSH register; Table 32-35 provides its field descriptions.

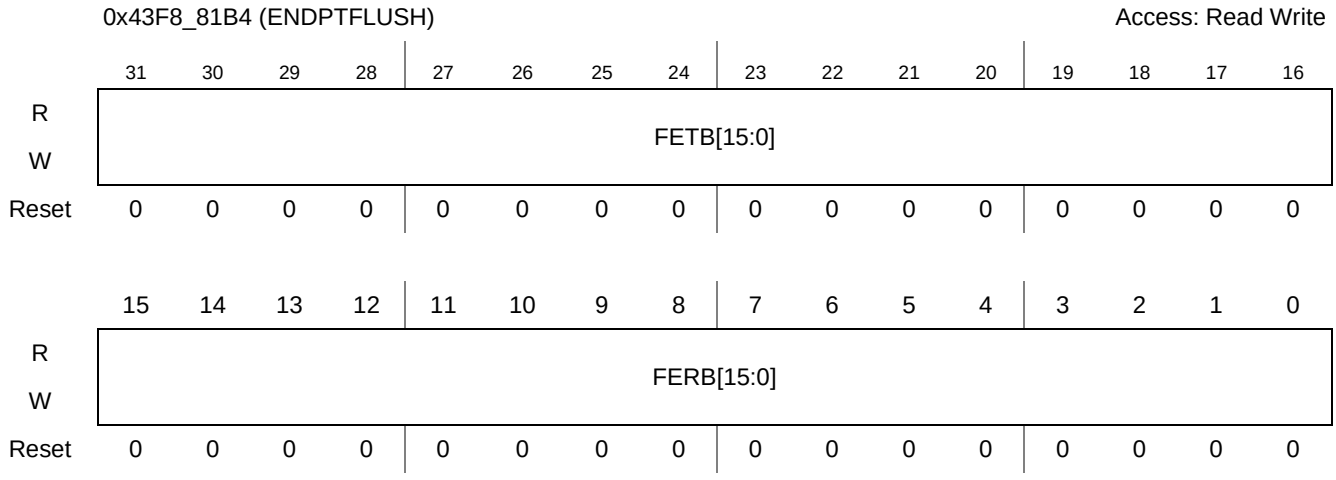


Figure 32-40. ENDPTFLUSH—Endpoint De-Initialize

Table 32-35. ENDPTFLUSH Field Descriptions

Field	Descriptions
31–16 FETB [15:0]	Flush Endpoint Transmit Buffer—R/WS. Writing a one to a bit(s) in this register will cause the associated endpoint(s) to clear any primed buffers. If a packet is in progress for one of the associated endpoints, then that transfer will continue until completion. Hardware will clear this register after the endpoint flush operation is successful. FETB[15] – Endpoint #15 FETB[1] – Endpoint #1 FETB[0] – Endpoint #0
15–0 FERB [15:0]	Flush Endpoint Receive Buffer—R/WS. Writing a one to a bit(s) will cause the associated endpoint(s) to clear any primed buffers. If a packet is in progress for one of the associated endpoints, then that transfer will continue until completion. Hardware will clear this register after the endpoint flush operation is successful. Bit 15 – Endpoint #15 Bit 1 – Endpoint #1 Bit 0 – Endpoint #0

32.9.5.15 ENDPTSTAT

This register is only used in device mode.

Figure 32-41 shows the register; Table 32-36 provides its field descriptions.

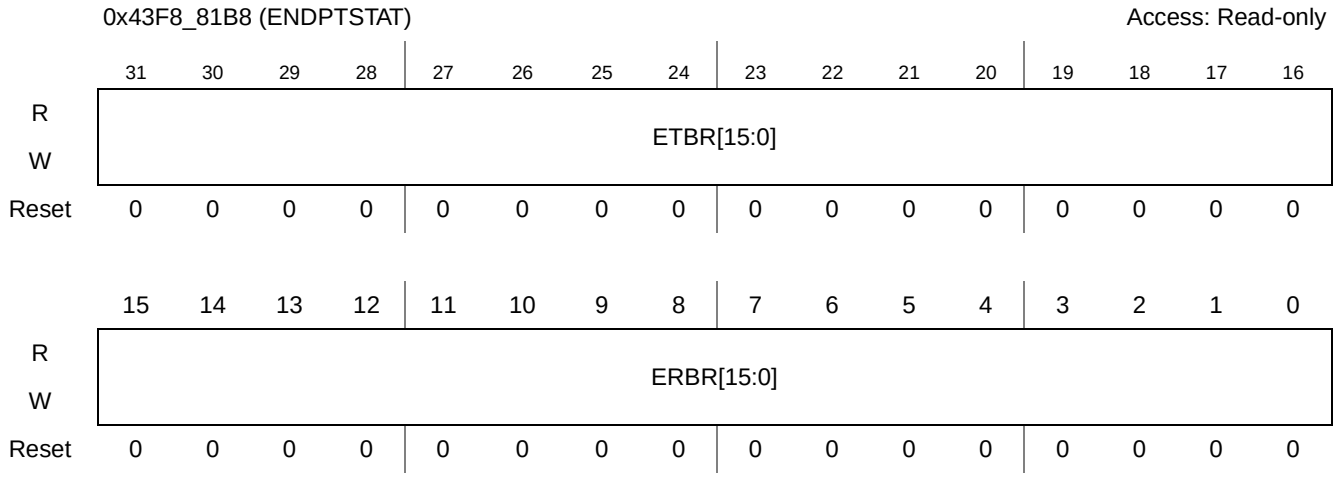


Figure 32-41. ENDPTSTAT—Endpoint Status

Table 32-36. ENDPTSTAT Field Descriptions

Field	Description
ETBR [15:0]	Endpoint Transmit Buffer Ready—Read Only. One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. Note: These bits will be momentarily cleared by hardware during hardware endpoint re-priming operations when a dTD is retired, and the dQH is updated. ETBR[15]– Endpoint #15 ETBR[1]– Endpoint #1 ETBR[0]– Endpoint #0
ERBR [15:0]	Endpoint Receive Buffer Ready—Read Only. One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. Note: These bits will be momentarily cleared by hardware during hardware endpoint re-priming operations when a dTD is retired, and the dQH is updated. ERBR[15]– Endpoint #15 ERBR[1]– Endpoint #1 ERBR[0]– Endpoint #0

32.9.5.16 ENDPTCOMPLETE

This register is only used in device mode.

Figure 32-42 shows the ENDPTCOMPLETE register; Table 32-37 provides its field descriptions.

0x43F8_81BC (ENDPTCOMPLETE) Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ETCE[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ERCE[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-42. ENDPTCOMPLETE—Endpoint Compete

Table 32-37. ENDPTCOMPLETE Field Descriptions

Field	Description
31–16 ETCE [15:0]	Endpoint Transmit Complete Event—R/WC. Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the *USBINT*. Writing a one will clear the corresponding bit in this register. ETCE[15]– Endpoint #15 ETCE[1]– Endpoint #1 ETCE[0]– Endpoint #0
15–0 ERCE [15:0]	Endpoint Receive Complete Event—RW/C. Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the *USBINT*. Writing a one will clear the corresponding bit in this register. ERCE[15]– Endpoint #15 ERCE[1]– Endpoint #1 ERCE[0]– Endpoint #0

32.9.5.17 ENDPTCTRL0

Every Device will implement Endpoint0 as a control endpoint.

Figure 32-43 shows the register; Table 32-38 provides its field descriptions.

0x43F8_81C0 (ENDPTCTRL0) Access: Read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	TXE	0	0	0	TXT		0	TXS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	RXE	0	0	0	RXT		0	RXS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-43. ENDPTCTRL0—Endpoint Control 0

Table 32-38. ENDPTCTRL0 Field Descriptions

Field	Description
31–24	Reserved. These bits are reserved and should be set to zero.
23 TXE	TX Endpoint Enable 1 Enabled Endpoint0 is always enabled.

Table 32-38. ENDPTCTRL0 Field Descriptions (continued)

Field	Description
22–20	Reserved. Bit reserved and should be set to zero.
19–18 TXT[1:0]	TX Endpoint Type—Read/Write 00 Control Endpoint0 is fixed as a Control End Point.
17	Reserved. Bit reserved and should be set to zero.
16 TXS	TX Endpoint Stall—Read/Write 0 End Point OK [Default] 1 End Point Stalled Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue returning STALL until the bit is cleared by software or it will automatically be cleared upon receipt of a new SETUP request.
15–8	Reserved. Bit reserved and should be set to zero.
7 RXE	RX Endpoint Enable 1 Enabled Endpoint0 is always enabled.
6–4	Reserved. Bit reserved and should be set to zero.
3–2 RXT[1:0]	RX Endpoint Type—Read/Write 00 Control Endpoint0 is fixed as a Control End Point.
1	Reserved. Bit reserved and should be set to zero.
0 RXS	RX Endpoint Stall—Read/Write 0 End Point OK. [Default] 1 End Point Stalled Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue returning STALL until the bit is cleared by software or it will automatically be cleared upon receipt of a new SETUP request.

32.9.5.18 ENDPTCTRL0—ENDPTCTRL15

There is an ENDPTCTRLx register for each endpoint in a device.

Figure 32-44 shows the register; Table 32-39 provides its field descriptions.

0x43F8_81C0 (ENDPTCTRL0)
0x43F8_81C4 (ENDPTCTRL1)
0x43F8_81C8 (ENDPTCTRL2)
0x43F8_81CC (ENDPTCTRL3)
0x43F8_81D0 (ENDPTCTRL4)
0x43F8_81D4 (ENDPTCTRL5)
0x43F8_81D8 (ENDPTCTRL6)
0x43F8_81DC (ENDPTCTRL7)

Access: Read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	TXE	TXR	TXI	0	TXT		TXD	TXS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	RXE	RXR	RXI	0	RXT		RXD	RXS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-44. ENDPTCTRL1–ENDPTCTRL15—Endpoint Control 1 to 15

CAUTION

If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (IE. Bulk-type). leaving an unconfigured endpoint control will cause undefined behavior for the data pid tracking on the active endpoint/direction.

Table 32-39. ENDPTCTRL1–ENDPTCTRL15 Field Descriptions

Field	Description
31–24	Reserved. These bits are reserved and should be set to zero.
23 TXE	TX Endpoint Enable 0—Disabled [Default] 1—Enabled An Endpoint should be enabled only after it has been configured.
22 TXR	TX Data Toggle Reset (WS) Write 1—Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit to synchronize the data PID's between the Host and device.

Table 32-39. ENDPTCTRL1–ENDPTCTRL15 Field Descriptions (continued)

Field	Description
21 TXI	TX Data Toggle Inhibit 0—PID Sequencing Enabled. [Default] 1—PID Sequencing Disabled. This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet.
20	Reserved. Bit reserved and should be set to zero.
19–18 TXT[1:0]	TX Endpoint Type—Read/Write 00 Control 01 Isochronous 10 Bulk 11 Interrupt
17 TXD	TX Endpoint Data Source—Read/Write 0—Dual Port Memory Buffer/DMA Engine [DEFAULT] Should always be written as 0.
16 TXS	TX Endpoint Stall—Read/Write 0—End Point OK 1—End Point Stalled This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above.
15–8	Reserved
7 RXE	RX Endpoint Enable 0—Disabled [Default] 1—Enabled An Endpoint should be enabled only after it has been configured.
6 RXR	RX Data Toggle Reset (WS) Write 1—Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit to synchronize the data PID's between the host and device.
5 RXI	RX Data Toggle Inhibit 0—Disabled [Default] 1—Enabled This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID.
4	Reserved
3–2 RXT[1:0]	RX Endpoint Type—Read/Write 00—Control 01—Isochronous 10—Bulk 11—Reserved

Table 32-39. ENDPTCTRL1–ENDPTCTRL15 Field Descriptions (continued)

Field	Description
1 RXD	RX Endpoint Data Sink—Read/Write—TBD 0—Dual Port Memory Buffer/DMA Engine [Default] Should always be written as zero.
0 RXS	RX Endpoint Stall—Read/Write 0—End Point OK. [Default] 1—End Point Stalled This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint, Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above,

32.9.6 OTG Operations

32.9.6.1 Register Bits

In the previous section, the Register interface has behaviors described for device mode and behaviors described for host mode. However, during OTG operations it is necessary to perform tasks independent of the controller mode.

Note also from Figure 32-45 that the only way to transition the controller mode out of host or device mode is with the controller reset bit. Therefore, it is also necessary for the OTG tasks to be performed independent of a controller reset as well as independent of the controller mode.

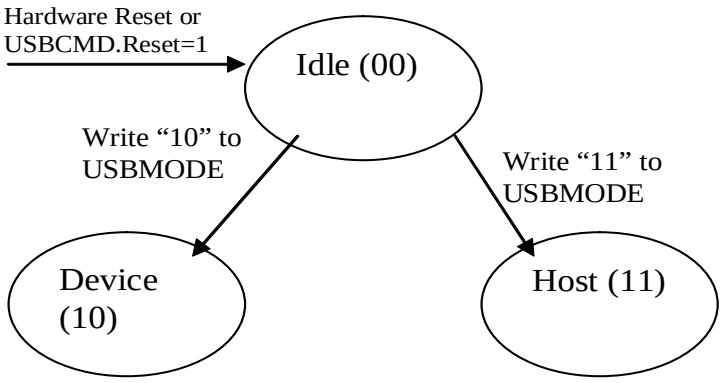


Figure 32-45. Controller Mode

To this end, the listed below are the register bits that are used for OTG operations, which are independent of the controller mode and are also not affected by a write to the reset bit in the USBCMD register:

All Identification Registers

All Device/Host Capability Registers

OTGSC: All bits

PORTSC:

- Physical Interface Select
- Physical Interface Serial Select
- Physical Interface Data Width
- Physical Interface Low Power
- Physical Interface Wake Signals
- Port Indicators
- Port Power

32.10 Host Data Structures

This section defines the interface data structures used to communicate control, status, and data between HCD (software) and the Enhanced Host Controller (hardware). The data structure definitions in this chapter support a 32-bit memory buffer address space. The interface consists of a Periodic Schedule, Periodic Frame List, Asynchronous Schedule, Isochronous Transaction Descriptors, Split-transaction Isochronous Transfer Descriptors, Queue Heads, and Queue Element Transfer Descriptors.

The periodic frame list is the root of all periodic (isochronous and interrupt transfer type) support for the host controller interface. The asynchronous list is the root for all the bulk and control transfer type support. Isochronous data streams are managed using Isochronous Transaction Descriptors. Isochronous split-transaction data streams are managed with Split-transaction Isochronous Transfer Descriptors. All Interrupt, Control, and Bulk data streams are managed via queue heads and Queue Element Transfer Descriptors. These data structures are optimized to reduce the total memory footprint of the schedule and to reduce (on average) the number of memory accesses needed to execute a USB transaction.

Note that software must ensure that no interface data structure reachable by the EHCI host controller spans a 4K-page boundary.

The data structures defined in this section are (from the host controller's perspective) a mix of read-only and read/writeable fields. The host controller must preserve the read-only fields on all data structure writes.

32.10.1 Periodic Frame List

This schedule is for all periodic transfers (isochronous and interrupt). The periodic schedule is referenced from the operational registers space using the *PERIODICLISTBASE* address register and the *FRINDEX* register. The periodic schedule is based on an array of pointers called the Periodic Frame List. The *PERIODICLISTBASE* address register is combined with the *FRINDEX* register to produce a memory pointer into the frame list. The Periodic Frame List implements a *sliding window* of work over time.

Figure 32-46 shows the periodic schedule organization.

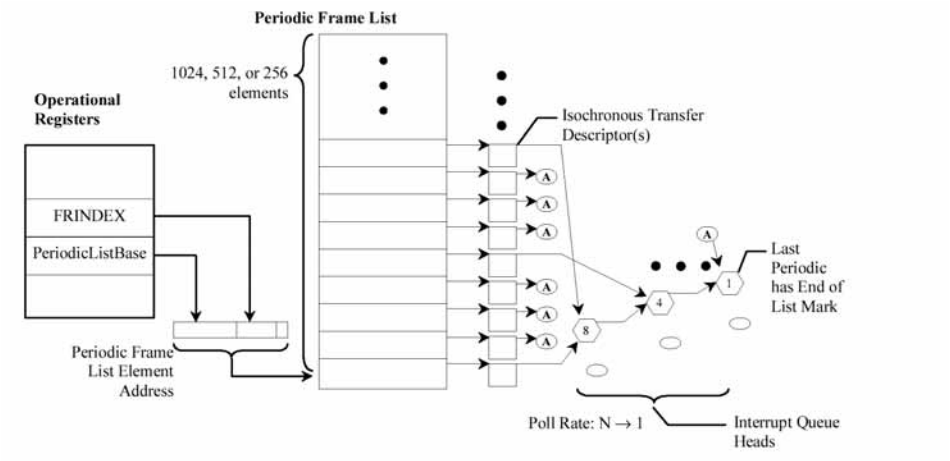


Figure 32-46. Periodic Schedule Organization

[1] Split transaction Interrupt, Bulk and Control are also managed using queue heads and queue element transfer descriptors.

The periodic frame list is a 4K-page aligned array of Frame List Link pointers. The length of the frame list may be programmable. The programmability of the periodic frame list is exported to system software via the HCCPARAMS register. If non-programmable, the length is 1024 elements. If programmable, the length can be selected by system software as one of 256, 512, or 1024 elements. An implementation must support all three sizes. Programming the size (that is, the number of elements) is accomplished by system software writing the appropriate value into *Frame List Size* field in the USBCMD register.

Frame List Link pointers direct the host controller to the first work item in the frame's periodic schedule for the current micro-frame. The link pointers are aligned on DWord boundaries within the Frame List.

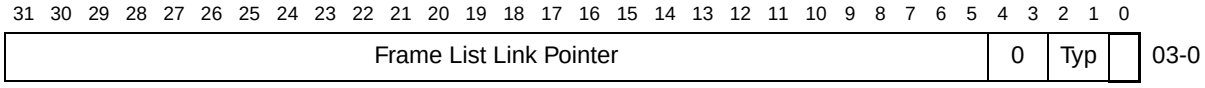


Figure 32-47. Format of Frame List Element Pointer

Frame List Link pointers always reference memory objects that are 32-byte aligned. The referenced object may be an isochronous transfer descriptor for high-speed devices, a split-transaction isochronous transfer descriptor (for full-speed isochronous endpoints), or a queue head (used to support high-, full- and low-speed interrupt). System software should not place non-periodic schedule items into the periodic schedule. The least significant bits in a frame list pointer are used to key the host controller as to the type of object the pointer is referencing.

The least significant bit is the *T-Bit* (bit 0). When this bit is set to a one, the host controller will never use the value of the frame list pointer as a physical memory pointer. The *Typ* field is used to indicate the exact type of data structure being referenced by this pointer. Table 32-40 provides the value encoding.

Table 32-40. Typ Field Value Definitions

Value	Meaning
00b	Isochronous Transfer Descriptor
01b	Queue Head
10b	Split Transaction Isochronous Transfer Descriptor.
11b	Frame Span Traversal Node.

32.10.2 Asynchronous List Queue Head Pointer

The Asynchronous Transfer List (based at the ASYNCLISTADDR register) is where all the control and bulk transfers are managed. Host controllers use this list only when it reaches the end of the periodic list, the periodic list is disabled, or the periodic list is empty. Figure 32-48 shows the asynchronous schedule organization.

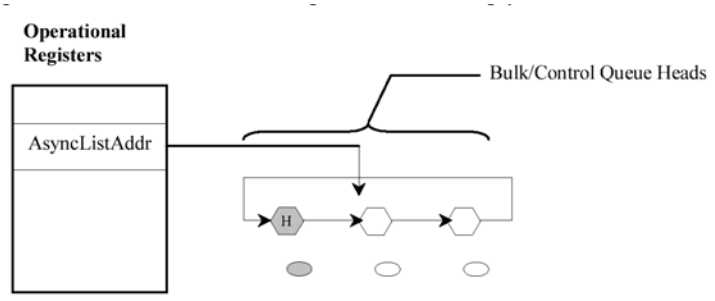


Figure 32-48. Asynchronous Schedule Organization

The Asynchronous list is a simple circular list of queue heads. The ASYNCLISTADDR register is simply pointer to the *next* queue head. This implements a pure round-robin service for all queue heads linked into the asynchronous list.

32.10.3 Isochronous (High-Speed) Transfer Descriptor (iTD)

This structure is used only for high-speed isochronous endpoints. All other transfer types should use queue structures. Isochronous TDs must be aligned on a 32-byte boundary. Table 32-41 shows the iTD register summary. The format of an isochronous transfer descriptor is illustrated in Figure 32-49.

Table 32-41. iTD Register Summary

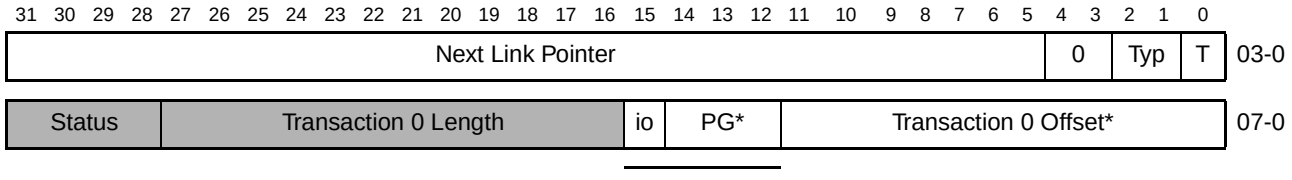


Table 32-41. iTD Register Summary (continued)

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16	15	14 13 12	11 10 9 8 7 6 5 4 3 2 1 0	
Status	Transaction 1 Length	io	PG*	Transaction 1 Offset*	0B-0
Status	Transaction 2 Length	io	PG*	Transaction 2 Offset*	0F-0
Status	Transaction 3 Length	io	PG*	Transaction 3 Offset*	13-1
Status	Transaction 4 Length	io	PG*	Transaction 4 Offset*	17-1
Status	Transaction 5 Length	io	PG*	Transaction 5 Offset*	1B-1
Status	Transaction 6 Length	io	PG*	Transaction 6 Offset*	1F-1
Status	Transaction 7 Length	io	PG*	Transaction 7 Offset*	23-2
Buffer Pointer (Page 0)			EndPt	R	Device Address	27-2
Buffer Pointer (Page 1)			I/	Maximum Packet Size	2B-2
Buffer Pointer (Page 2)			Reserved	Mult	2F-2
Buffer Pointer (Page 3)			Reserved	33-3
Buffer Pointer (Page 4)			Reserved	37-3
Buffer Pointer (Page 5)			Reserved	3B-3
Buffer Pointer (Page 6)			Reserved	3F-3

Figure 32-49. Isochronous Transaction Descriptor (iTD)

	Host Controller Read/Write		Host Controller Read Only. N

NOTE

These fields may be modified by the host controller if the I/O field indicates an OUT.

32.10.3.1 Next Link Pointer

The first DWord of an iTD is a pointer to the next schedule data structure. Table 32-42 shows the next schedule element pointer field descriptions.

Table 32-42. Next Schedule Element Pointer

Field	Description
31–5	Link Pointer (LP). These bits correspond to memory address signals [31:5], respectively. This field points to another Isochronous Transaction Descriptor (iTD/siTD) or Queue Head (QH).
4–3	Reserved. These bits are reserved and their value has no effect on operation. Software should initialize this field to zero.
2–1	QH/(s)iTD Select (Typ). This field indicates to the Host Controller whether the item referenced is an iTD, siTD or a QH. This allows the Host Controller to perform the proper type of processing on the item after it is fetched. Value encodings are as follows: 00b iTD (isochronous transfer descriptor) 01b QH (queue head) 10b siTD (split transaction isochronous transfer descriptor 11b FSTN (frame span traversal node)
0	Terminate (T). 1= Link Pointer field is not valid. 0= Link Pointer field is valid.

32.10.3.2 iTD Transaction Status and Control List

DWords 1 through 8 are eight slots of transaction control and status. Each transaction description includes:

- Status results field
- Transaction length (bytes to send for OUT transactions and bytes received for IN transactions).
- Buffer offset. The *PG* and *Transaction X Offset* fields are used with the buffer pointer list to construct the starting buffer address for the transaction.

The host controller uses the information in each transaction description plus the endpoint information contained in the first three DWords of the Buffer Page Pointer list, to execute a transaction on the USB. Table 32-43 shows the iTD transaction status and control field descriptions.

Table 32-43. iTD Transaction Status and Control

Field	Description	
31–28	Status. This field records the status of the transaction executed by the host controller for this slot. This field is a bit vector with the following encoding:	
	Bit	Definition
	31	Active. Set to one by software to enable the execution of an isochronous transaction by the Host Controller. When the transaction associated with this descriptor is completed, the Host Controller sets this bit to zero indicating that a transaction for this element should not be executed when it is next encountered in the schedule.
	30	Data Buffer Error. Set to a one by the Host Controller during status update to indicate that the Host Controller is unable to keep up with the reception of incoming data (overrun) or is unable to supply data fast enough during transmission (under run). If an overrun condition occurs, no action is necessary.

Table 32-43. iTD Transaction Status and Control (continued)

Field	Description	
	29	Babble Detected. Set to one by the Host Controller during status update when" babble" is detected during the transaction generated by this descriptor.
	28	Transaction Error (XactErr). Set to one by the Host Controller during status update in the case where the host did not receive a valid response from the device (Timeout, CRC, Bad PID, and so on.). This bit may only be set for isochronous IN transactions.
27–16	Transaction X Length. For an OUT, this field is the number of data bytes the host controller will send during the transaction. The host controller is not required to update this field to reflect the actual number of bytes transferred during the transfer. For an IN, the initial value of the endpoint to deliver. During the status update, the host controller writes back the field is the number of bytes the host expects the number of bytes successfully received. The value in this register is the actual byte count (for example, 0→zero length data, 1→one byte, 2→two bytes, and so on). The maximum value this field may contain is 0xC00 (3072).	
15	Interrupt On Complete (IOC). If this bit is set to one, it specifies that when this transaction completes, the Host Controller should issue an interrupt at the next interrupt threshold.	
14–12	Page Select (PG). These bits are set by software to indicate which of the buffer page pointers the offset field in this slot should be concatenated to produce the starting memory address for this transaction. The valid range of values for this field is 0 to 6.	
11–0	Transaction X Offset. This field is a value that is an offset, expressed in bytes, from the beginning of a buffer. This field is concatenated onto the buffer page pointer indicated in the adjacent *PG* field to produce the starting buffer address for this transaction.	

32.10.3.3 iTD Buffer Page Pointer List (Plus)

DWords 9-15 of an isochronous transaction descriptor are nominally page pointers (4K aligned) to the data buffer for this transfer descriptor. This data structure requires the associated data buffer to be contiguous (relative to virtual memory), but allows the physical memory pages to be non-contiguous. Seven page pointers are provided to support the expression of eight isochronous transfers. The seven pointers allow for 3 (transactions) * 1024 (maximum packet size) * 8 (transaction records) (24576 bytes) to be moved with this data structure, regardless of the alignment offset of the first page.

Since each pointer is a 4 Kbyte aligned page pointer, the least significant 12 bits in several of the page pointers are used for other purposes.

Table 32-44 through Table 32-47 show the buffer pointer pages field descriptions.

Table 32-44. iTD Buffer Pointer Page 0 (Plus)

Field	Description
31–12	Buffer Pointer (Page 0). This is a 4 Kbyte aligned pointer to physical memory. Corresponds to memory address bits [31:12].
11–8	Endpoint Number (Endpt). This 4-bit field selects the particular endpoint number on the device serving as the data source or sink.
7	Reserved. Bit reserved for future use and should be initialized by software to zero.
6–0	Device Address. This field selects the specific device serving as the data source or sink.

Table 32-45. iTD Buffer Pointer Page 1 (Plus)

Field	Description
31–12	Buffer Pointer (Page 1). This is a 4K aligned pointer to physical memory. Corresponds to memory address bits [31:12].
11	Direction (I/O). 0 = OUT; 1 = IN. This field encodes whether the high-speed transaction should use an IN or OUT PID.
10–0	Maximum Packet Size. This directly corresponds to the maximum packet size of the associated endpoint (*wMaxPacketSize*). This field is used for high-bandwidth endpoints where more than one transaction is issued per transaction description (for example, per micro-frame). This field is used with the *Multi* field to support high-bandwidth pipes. This field is also used for all IN transfers to detect packet babble. Software should not set a value larger than 1024 (400h). Any value larger yields undefined results.

Table 32-46. iTD Buffer Pointer Page 2 (Plus)

Field	Description
31–12	Buffer Pointer. This is a 4K aligned pointer to physical memory. Corresponds to memory address bits [31:12].
11–2	Reserved. This bit reserved for future use and should be set to zero.
1–0	Multi. This field is used to indicate to the host controller the number of transactions that should be executed per transaction description (for example, per micro-frame). The valid values are as follows: 00b Reserved. A zero in this field yields undefined results. 01b One transaction to be issued for this endpoint per micro-frame 10b Two transactions to be issued for this endpoint per micro-frame 11b Three transactions to be issued for this endpoint per micro-frame

Table 32-47. iTD Buffer Pointer Pages 3–6

Field	Description
31–12	Buffer Pointer. This is a 4K aligned pointer to physical memory. Corresponds to memory address bits [31:12].
11–0	Reserved. These bits reserved for future use and should be set to zero.

32.10.4 Split Transaction Isochronous Transfer Descriptor (siTD)

All Full-speed isochronous transfers through the internal transaction translator are managed using the siTD data structure. This data structure satisfies the operational requirements for managing the split transaction protocol. Table 32-48 shows the siTD register summary.

Table 32-48. siTD Register Summary

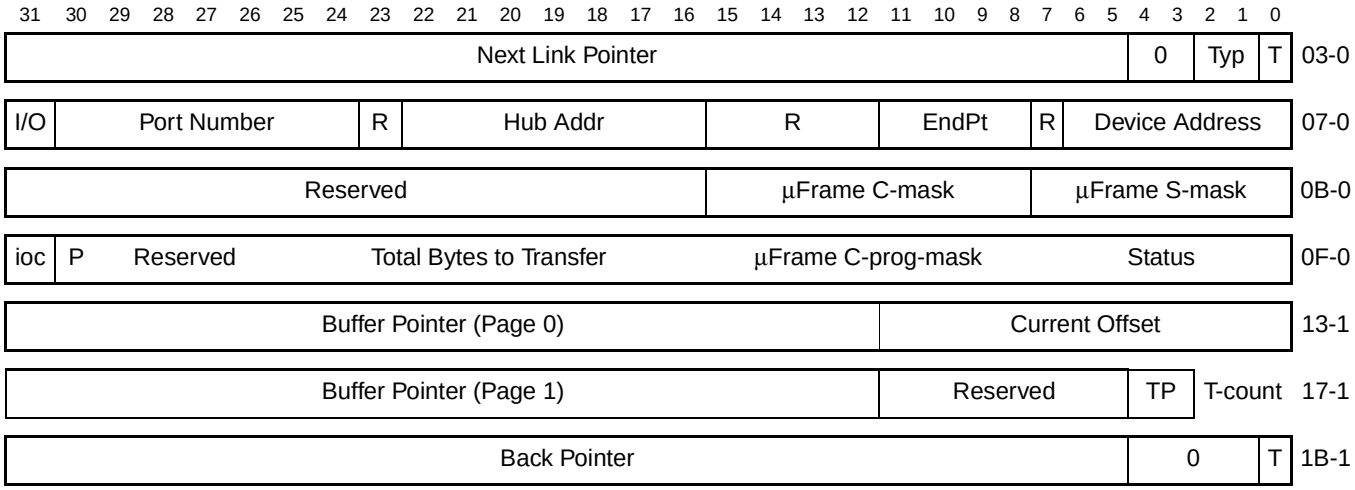


Figure 32-50. Split-transaction Isochronous Transaction Descriptor (siTD)

	Host Controller Read/Write		Host Controller Read Only.

32.10.4.1 Next Link Pointer

DWord0 of a siTD is a pointer to the next schedule data structure. Table 32-49 shows the next link pointer field descriptions.

Table 32-49. Next Link Pointer

Field	Description
31–5	Next Link Pointer (LP). This field contains the address of the next data object to be processed in the periodic list and corresponds to memory address signals [31:5], respectively.
4–3	Reserved. These bits must be written as zeros.
2–1	QH/(s)iTD Select (Typ). This field indicates to the Host Controller whether the item referenced is an iTD/siTD or a QH. This allows the Host Controller to perform the proper type of processing on the item after it is fetched. Value encodings are as follows: 00b iTD (isochronous transfer descriptor) 01b QH (queue head) 10b siTD (split transaction isochronous transfer descriptor 11b FSTN (frame span traversal node)
0	Terminate (T). 1=Link Pointer field is not valid. 0=Link Pointer is valid.

32.10.4.2 siTD Endpoint Capabilities/Characteristics

DWords 1 and 2 specify static information about the full-speed endpoint, the addressing of the parent Companion Controller, and micro-frame scheduling control.

Table 32-50 shows the siTD endpoint and transaction translator characteristics field descriptions.

Table 32-50. Endpoint and Transaction Translator Characteristics

Field	Description
31	Direction (I/O).0 = OUT; 1 = IN. This field encodes whether the full-speed transaction should be an IN or OUT.
30–24	Port Number. This field is the port number of the recipient Transaction Translator.
23	Reserved. Bit reserved and should be set to zero.
22–16	Hub Address. This field holds the device address of the Companion Controllers' hub.
15–12	Reserved. Field reserved and should be set to zero.
11–8	Endpoint Number (Endpt). This 4-bit field selects the particular endpoint number on the device serving as the data source or sink.
7	Reserved. Bit is reserved for future use. It should be set to zero.
6–0	Device Address. This field selects the specific device serving as the data source or sink.

Table 32-51 shows the micro-frame schedule control field descriptions.

Table 32-51. Micro-frame Schedule Control

Field	Description
31–16	Reserved. This field reserved for future use. It should be set to zero.
15–8	Split Completion Mask (µFrame C-Mask). This field (along with the *Active* and *SplitX- state* fields in the *Status* byte) is used to determine during which micro-frames the host controller should execute complete-split transactions. When the criteria for using this field is met, an all zeros value has undefined behavior. The host controller uses the value of the three low-order bits of the FRINDEX register to index into this bit field. If the FRINDEX register value indexes to a position where the *µFrame C-Mask* field is a one, then this siTD is a candidate for transaction execution. There may be more than one bit in this mask set.
7–0	Split Start Mask (µFrame S-mask). This field (along with the *Active* and *SplitX-state* fields in the *Status* byte) is used to determine during which micro-frames the host controller should execute start-split transactions. The host controller uses the value of the three low-order bits of the FRINDEX register to index into this bit field. If the FRINDEX register value indexes to a position where the *µFrame S-mask* field is a one, then this siTD is a candidate for transaction execution. An all zeros value in this field, in combination with existing periodic frame list has undefined results.

32.10.4.3 siTD Transfer State

DWords 3-6 are used to manage the state of the transfer. Table 32-52 shows the siTD transfer status and control field descriptions.

Table 32-52. siTD Transfer Status and Control

Field	Description	
31	Interrupt On Complete (ioc). 0 = Do not interrupt when transaction is complete. 1 = Do interrupt when transaction is complete. When the host controller determines that the split transaction has completed it will assert a hardware interrupt at the next interrupt threshold.	
30	Page Select (P). Used to indicate which data page pointer should be concatenated with the *Current Offset* field to construct a data buffer pointer (0 selects *Page 0* pointer and 1 selects *Page 1*). The host controller is not required to write this field back when the siTD is retired (*Active* bit transitioned from a one to a zero).	
29–26	Reserved. This field reserved for future use and should be set to zero.	
25–16	Total Bytes To Transfer. This field is initialized by software to the total number of bytes expected in this transfer. Maximum value is 1023 (3FFh)	
15–8	µFrame Complete-split Progress Mask (C-prog-Mask). This field is used by the host controller to record which split-completes has been executed.	
7–0	Status. This field records the status of the transaction executed by the host controller for this slot. This field is a bit vector with the following encoding:	
	Bit	**Definition**
	7	Active. Set to one by software to enable the execution of an isochronous split transaction by the Host Controller.
	6	ERR. Set to a one by the Host Controller when an ERR response is received from the Companion Controller.
	5	Data Buffer Error. Set to a one by the Host Controller during status update to indicate that the Host Controller is unable to keep up with the reception of incoming data (overrun) or is unable to supply data fast enough during transmission (under run). In the case of an under run, the Host Controller will transmit an incorrect CRC (thus invalidating the data at the endpoint). If an overrun condition occurs, no action is necessary.
	4	Babble Detected. Set to a one by the Host Controller during status update when" babble" is detected during the transaction generated by this descriptor.
	3	Transaction Error (XactErr). Set to a one by the Host Controller during status update in the case where the host did not receive a valid response from the device (Timeout, CRC, Bad PID, and so on). This bit will only be set for IN transactions.
	2	Missed Micro-Frame. The host controller detected that a host-induced hold-off caused the host controller to miss a required complete-split transaction.

Table 32-52. siTD Transfer Status and Control (continued)

Field	Description	
	1	Split Transaction State (SplitXstate). The bit encodings are as follows: 00b Do Start Split. This value directs the host controller to issue a Start split transaction to the endpoint when a match is encountered in the S-mask. 01b Do Complete Split. This value directs the host controller to issue a Complete split transaction to the endpoint when a match is encountered in the C-mask.
	0	Reserved. Bit reserved for future use and should be set to zero.

32.10.4.4 siTD Buffer Pointer List (Plus)

DWords 4 and 5 are the data buffer page pointers for the transfer. This structure supports one physical page cross. The most significant 20 bits of each DWord in this section are the 4K (page) aligned buffer pointers. The least significant 12 bits of each DWord are used as additional transfer state.

Table 32-53 shows the siTD buffer pointer list (plus) field descriptions.

Table 32-53. Buffer Page Pointer List (Plus)

Field	Description
31–12	Buffer Pointer List. Bits [31:12] of DWords 4 and 5 are 4K paged aligned, physical memory addresses. These bits correspond to physical address bits [31:12] respectively. The lower 12 bits in each pointer are defined and used as specified below. The field *P* specifies the *current* active pointer.
11–0	Page 0: Current Offset. The 12 least significant bits of the Page 0 pointer is the current byte offset for the current page pointer (as selected with the page indicator bit (*P* field)). The host controller is not required to write this field back when the siTD is retired (*Active* bit transitioned from a one to a zero). The least significant bits of Page 1 pointer is split into three sub-fields. Page 1 is as follows:

Table 32-53. Buffer Page Pointer List (Plus) (continued)

Field	Description	
	Bit	**Definition**
	11–5	Reserved
	4–3	Transaction position (TP). This field is used with T-count to determine whether to send *all*, *first*, *middle*, or *last* with each outbound transaction payload. System software must initialize this field with the appropriate starting value. The host controller must correctly manage this state during the lifetime of the transfer. The bit encodings are as follows: 00b All. The entire full-speed transaction data payload is in this transaction (that is, less than or equal to 188 bytes). 01b Begin. This is the first data payload for a full-speed that is greater than 188 bytes.transaction 10B Mid. This is the *middle* payload for a full-speed OUT transaction that is larger than 188 bytes. 11b End. This is the *last* payload for a full-speed OUT transaction that was larger than 188 bytes.
	2–0	Transaction count (T-Count). Software initializes this field with the number of OUT start-splits this transfer requires. Any value larger than 6 is undefined.

32.10.4.5 siTD Back Link Pointer

DWord 6 of a siTD is simply another schedule link pointer. This pointer is always zero, or references a siTD. This pointer cannot reference any other schedule data structure.

Table 32-54 shows the siTD back link field descriptions.

Table 32-54. siTD Back Link Pointer

Field	Description
31–5	siTD Back Pointer. This field is a physical memory pointer to a siTD.
4–1	Reserved. This field is reserved for future use. It should be set to zero.
0	Terminate (T). 1 = siTD Back Pointer field is not valid. 0 = siTD Back Pointer field is valid.

32.10.5 Queue Element Transfer Descriptor (qTD)

This data structure is only used with a queue head. This data structure is used for one or more USB transactions. This data structure is used to transfer up to 20480 (5*4096) bytes. The structure contains two structure pointers used for queue advancement, a DWord of transfer state, and a five-element array of data buffer pointers. This structure is 32 bytes (or one 32-byte cache line). This data structure must be physically contiguous.

The buffer associated with this transfer must be virtually contiguous. The buffer may start on any byte boundary. A separate buffer pointer list element must be used for each physical page in the buffer, regardless of whether the buffer is physically contiguous.

Host controller updates (host controller writes) to stand-alone qTDs only occur during transfer retirement. References in the following bit field definitions of updates to the qTD are to the qTD portion of a queue head.

Table 32-55 shows the qTD register summary.

Table 32-55. qTD Register Summary

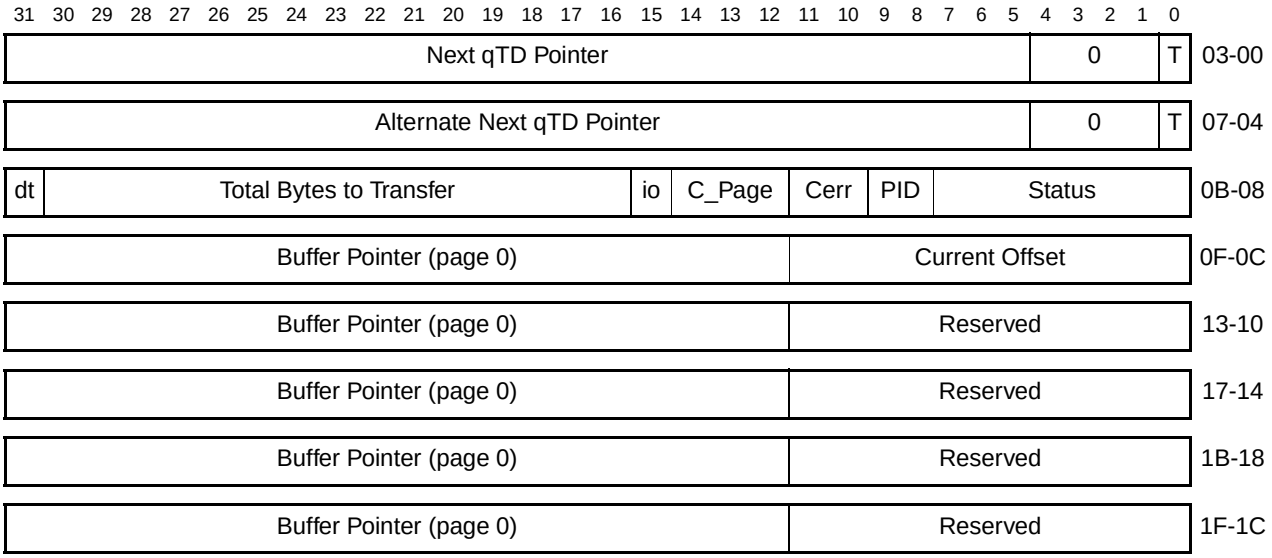


Figure 32-51. Queue Element Transfer Descriptor Block Diagram

	Host Controller Read/Write		Host Controller Read Only.

Queue Element Transfer Descriptors must be aligned on 32-byte boundaries.

32.10.5.1 Next qTD Pointer

The first DWord of an element transfer descriptor is a pointer to another transfer element descriptor.

Table 32-56 shows the qTD pointer field descriptions.

Table 32-56. D Next Element Transfer Pointer (DWord 0)

Field	Description
31–5	Next Transfer Element Pointer. This field contains the physical memory address of the next qTD to be processed. The field corresponds to memory address signals[31:5], respectively.
4–1	Reserved. These bits are reserved and their value has no effect on operation.
0	Terminate (T). 1= pointer is invalid. 0=Pointer is valid (points to a valid Transfer Element Descriptor). This bit indicates to the Host Controller that there are no more valid entries in the queue.

32.10.5.2 Alternate Next qTD Pointer

The second DWord of a queue element transfer descriptor is used to support hardware-only advance of the data stream to the next client buffer on short packet. To be more explicit the host controller will always use this pointer when the current qTD is retired due to short packet.

Table 32-57 shows the TD alternate next element transfer pointer field descriptions.

Table 32-57. TD Alternate Next Element Transfer Pointer (DWord 1)

Field	Description
31–5	Alternate Next Transfer Element Pointer. This field contains the physical memory address of the next qTD to be processed in the event that the current qTD execution encounters a short packet (for an IN transaction). The field corresponds to memory address signals [31:5], respectively.
4–1	Reserved. These bits are reserved and their value has no effect on operation.
0	Terminate (T). 1= pointer is invalid. 0=Pointer is valid (points to a valid Transfer Element Descriptor). This bit indicates to the Host Controller that there are no more valid entries in the queue.

32.10.5.3 qTD Token

The third DWord of a queue element transfer descriptor contains most of the information the host controller requires to execute a USB transaction (the remaining endpoint-addressing information is specified in the queue head).

NOTE

The field descriptions forward reference fields defined in the queue head. Where necessary, these forward references are preceded with a QH notation.

Table 32-58 shows the qTD token field descriptions.

Table 32-58. qTD Token (DWord 2)

Field	Description
31	Data Toggle. This is the data toggle sequence bit. The use of this bit depends on the setting of the *Data Toggle Control* bit in the queue head.
30–16	Total Bytes to Transfer. This field specifies the total number of bytes to be moved with this transfer descriptor. This field is decremented by the number of bytes actually moved during the transaction, only on the successful completion of the transaction. The maximum value software may store in this field is 5 * 4K (5000H). This is the maximum number of bytes 5 page pointers can access. If the value of this field is zero when the host controller fetches this transfer descriptor (and the active bit is set), the host controller executes a zero-length transaction and retires the transfer descriptor. It is not a requirement for OUT transfers that *Total Bytes To Transfer* be an even multiple of QHD.Maximum Packet Length. If software builds such a transfer descriptor for an OUT transfer, the last transaction will always be less than QHD.Maximum Packet Length. Although it is possible to create a transfer up to 20K this assumes the 1st offset into the first page is 0. When the offset cannot be predetermined, crossing past the 5th page can be guaranteed by limiting the total bytes to 16K**. Therefore, the maximum recommended transfer is 16K(4000H).

Table 32-58. qTD Token (DWord 2) (continued)

Field	Description	
15	Interrupt On Complete (IOC). If this bit is set to a one, it specifies that when this qTD is completed, the Host Controller should issue an interrupt at the next interrupt threshold.	
14–12	Current Page (C_Page). This field is used as an index into the qTD buffer pointer list. Valid values are in the range 0H to 4H. The host controller is not required to write this field back when the qTD is retired.	
11–10	Error Counter (CERR). This field is a 2-bit down counter that keeps track of the number of consecutive Errors detected while executing this qTD. If this field is programmed with a non-zero value during set-up, the Host Controller decrements the count and writes it back to the qTD if the transaction fails. If the counter counts from one to zero, the Host Controller marks the qTD inactive, sets the *Halted* bit to a one, and error status bit for the error that caused *CERR* to decrement to zero. An interrupt will be generated if the *USB Error Interrupt Enable* bit in the USBINTR register is set to a one. If HCD programs this field to zero during set-up, the Host Controller will not count errors for this qTD and there will be no limit on the retries of this qTD. Note that write-backs of intermediate execution state are to the queue head overlay area, not the qTD. Error Decrement Counter Transaction Error Yes Data Buffer Error No[3] Stalled No[1] Babble Detected No[1] No Error No[2]	
	Error	**Decrement Counter Error Decrement Counter**
	1	Detection of Babble or Stall automatically halts the queue head. Thus, count is not decremented
	2	If the EPS field indicates a HS device or the queue head is in the Asynchronous Schedule (and *PIDCode* indicates an IN or OUT) and a bus transaction completes and the host controller does not detect a transaction error, then the host controller should reset *CERR* to extend the total number of errors for this transaction. For example, *CERR* should be reset with maximum value (3) on each successful completion of a transaction. The host controller must never reset this field if the value at the start of the transaction is 00b. See section Split Transaction Execution State Machine for Interrupt for CERR adjustment rules when the EPS field indicates a FS or LS device and the queue head is in the Periodic Schedule. See section Asynchronous—Do Complete Split for CERR adjustment rules when the EPS field indicates a FS or LS device, the queue head is in the Asynchronous schedule and the *PIDCode* indicates a SETUP.
	3	Data buffer errors are host problems. They don't count against the device's retries.
	Note: Software must not program CERR to a value of zero when the EPS field is programmed with a value indicating a Full- or Low-speed device. This combination could result in undefined behavior.	
9–8	PID Code. This field is an encoding of the token, which should be used for transactions associated with this transfer descriptor. Encodings are as follows:	

Table 32-58. qTD Token (DWord 2) (continued)

Field	Description	
	00b	OUT Token generates token (E1H)
	01b	IN Token generates token (69H)
	10b	SETUP Token generates token (2DH) (undefined if endpoint is an Interrupt the queue head is non-zero.) transfer type, for example, μ*Frame S-mask* field in
	11b	Reserved
7–0	Status. This field is used by the Host Controller to communicate individual command execution states back to HCD. This field contains the status of the last transaction performed on this qTD. The bit encodings are as follows:	
	Bit	**Status Field Descriptions**
	7	Active. Set to one by software to enable the execution of transactions by the Host Controller.
	6	Halted. Set to a one by the Host Controller during status updates to indicate that a serious error has occurred at the device/endpoint addressed by this qTD. This can be caused by babble, the error counter counting down to zero, or reception of the STALL handshake from the device during a transaction. Any time that a transaction results in the Halted bit being set to a one, the Active bit is also set to zero.
	5	Data Buffer Error. Set to a one by the Host Controller during status update to indicate that the Host Controller is unable to keep up with the reception of incoming data (overrun) or is unable to supply data fast enough during transmission (under run). If an overrun condition occurs, the Host Controller will force a timeout condition on the USB, invalidating the transaction at the source. If the host controller sets this bit to a one, then it remains a one for the duration of the transfer.
	4	Babble Detected. Set to a one by the Host Controller during status update when" babble" is detected during the transaction. In addition to setting this bit, the Host Controller also sets the *Halted* bit to a one. Since "babble" is considered a fatal error for the transfer, setting the Halted bit to a one insures that no more transactions occur because of this descriptor.
	3	Transaction Error (XactErr). Set to a one by the Host Controller during status update in the case where the host did not receive a valid response from the device (Timeout, CRC, Bad PID, and so on). If the host controller sets this bit to a one, then it remains a one for the duration of the transfer.
	2	Missed Micro-Frame. This bit is ignored unless the *QH.EPS* field indicates a full- or low-speed endpoint and the queue head is in the periodic list. This bit is set when the host controller detected that a host-induced hold-off caused the host controller to miss a required complete-split transaction. If the host controller sets this bit to a one, then it remains a one for the duration of the transfer.

Table 32-58. qTD Token (DWord 2) (continued)

Field	Description	
	1	Split Transaction State (SplitXstate). This bit is ignored by the host controller unless the *QH.EPS* field indicates a full- or low-speed endpoint. When a Full- or Low-speed device, the host controller uses this bit to track the state of the split- transaction. The functional requirements of the host controller for managing this state bit and the split transaction protocol depends on whether the endpoint is in the periodic or asynchronous schedule. The bit encodings are as follows: ValueMeaning 0b Do Start Split. This value directs the host controller to issue a Start split transaction to the endpoint. 1b Do Complete Split. This value directs the host controller to issue a Complete split transaction to the endpoint.
	0	Ping State (P)/ERR. If the *QH.EPS* field indicates a High-speed device and the *PID_Code* indicates an OUT endpoint, then this is the state bit for the Ping protocol. The bit encodings are as follows: ValueMeaning 0b Do OUT. This value directs the host controller to issue an OUT PID to the endpoint. 1b Do Ping. This value directs the host controller to issue a PING PID to the endpoint. If the *QH.EPS* field does not indicate a High-speed device, then this field is used as an error indicator bit. It is set to a one by the host controller whenever a periodic split-transaction receives an ERR handshake.

32.10.5.4 qTD Buffer Page Pointer List

The last five DWords of a queue element transfer descriptor is an array of physical memory address pointers. These pointers reference the individual pages of a data buffer.

System software initializes *Current Offset* field to the starting offset into the current page, where current page is selected via the value in the *C_Page* field.

Table 32-59 shows the qTD buffer page pointer field descriptions.

Table 32-59. qTD Buffer Pointer(s) (DWords 3–7)

Field	Description
31–12	Buffer Pointer List. Each element in the list is a 4K page aligned physical memory address. The lower 12 bits in each pointer are reserved (except for the first one), as each memory pointer must reference the start of a 4K page. The field C_Page specifies the current active pointer. When the transfer element descriptor is fetched, the starting buffer address is selected using C_Page (similar to an array index to select an array element). If a transaction spans a 4K buffer boundary, the host controller must detect the page-span boundary in the data stream, increment C_Page and advance to the next buffer pointer in the list, and conclude the transaction via the new buffer pointer.
11–0	Current Offset (Reserved). This field is reserved in all pointers except the first one (for example, Page 0). The host controller should ignore all reserved bits. For the page 0 current offset interpretation, this field is the byte offset into the current page (as selected by C_Page). The host controller is not required to write this field back when the qTD is retired. Software should ensure the Reserved fields are initialized to zeros.

32.10.6 Queue Head

Table 32-60 shows the Queue Head register summary.

Table 32-60. Queue Head Register Summary

31–16	15	14–13	12	11–8	7	6–0	
Queue Head Horizontal Link Pointer			0	Typ		T	03-00H
RL	C	Maximum Packet Length	H	dt	EP	EndPt	I	Device Address	07-04H
Mult	Port Number*	Hub Addr*	µFrame C-mask*			µFrame S-mask*			0B-08H
Current qTD Pointer						0			0F-0CH
Next qTD Pointer						0		T	13-10H
Alternate Next qTD pointer						NakCnt		T	17-14H
d	Total Bytes to Transfer		io	C_Page	Cerr	PID	Status				1B-18H
Buffer Pointer (Page 0)						Current Offset					1F-1CH
Buffer Pointer (Page 1)						Reserved		C-prog-mask*			23-20H
Buffer Pointer (Page 2)						S-bytes*				FrameTag*	27-24H
Buffer Pointer (Page 3)						Reserved					2B-28H
Buffer Pointer (Page 4)						Reserved					2F-2CH

Transfer OverlayTransfer Results
Static Endpoint State
*These fields are used exclusively to support split transactions to USB 2.0 Hubs

Figure 32-52. Queue Head Structure Layout

	Host Controller Read/Write		Host Controller Read Only.

32.10.7 Queue Head Horizontal Link Pointer

The first DWord of a Queue Head contains a link pointer to the next data object to be processed after any required processing in this queue has been completed, as well as the control bits defined below.

This pointer may reference a queue head or one of the isochronous transfer descriptors. It must not reference a queue element transfer descriptor.

Table 32-61 shows the Queue Head DWord 0 field descriptions.

Table 32-61. Queue Head DWord 0

Field	Description
31–5	Queue Head Horizontal Link Pointer (QHLP). This field contains the address of the next data object to be processed in the horizontal list and corresponds to memory address signals [31:5], respectively.
4–3	Reserved. These bits must be written as zeros.
2–1	QH/(s)iTD Select (Typ). This field indicates to the hardware whether the item referenced by the link pointer is an iTD, siTD or a QH. This allows the Host Controller to perform the proper type of processing on the item after it is fetched. Value encodings are as follows: ValueMeaning 00b iTD (isochronous transfer descriptor) 01b QH (queue head) 10b siTD (split transaction isochronous transfer descriptor) 11b FSTN (frame span traversal node)
0	Terminate (T). 1=Last QH (pointer is invalid). 0=Pointer is valid. If the queue head is in the context of the periodic list, a one bit in this field indicates to the host controller that this is the end of the periodic list. This bit is ignored by the host controller when the queue head is in the Asynchronous schedule. Software must ensure that queue heads reachable by the host controller always have valid horizontal link pointers.

32.10.7.1 Endpoint Capabilities/Characteristics

The second and third DWords of a Queue Head specifies static information about the endpoint. This information does not change over the lifetime of the endpoint. There are three types of information in this region:

- Endpoint characteristics. These are the USB endpoint characteristics including addressing, maximum packet size, and endpoint speed.

- Endpoint capabilities. These are adjustable parameters of the endpoint. They effect how the endpoint data stream is managed by the host controller.
- Split transaction characteristics. This data structure is used to manage full- and low-speed data streams for bulk, control, and interrupt via split transactions to USB2.0 Hub Transaction Translator. There are additional fields used for addressing the hub and scheduling the protocol transactions (for periodic).

The host controller must not modify the bits in this region.

Table 32-62 shows the endpoint characteristics: Queue Head DWord 1 field descriptions.

Table 32-62. Endpoint Characteristics: Queue Head DWord 1

Field	Description
31–28	NAK Count Reload (RL). This field contains a value, which is used by the host controller to reload the NAK Counter field.
27	Control Endpoint Flag (C). If the *QH.EPS* field indicates the endpoint is not a high-speed device, and the endpoint is a control endpoint, then software must set this bit to a one. Otherwise, it should always set this bit to a zero.
26–16	Maximum Packet Length. This directly corresponds to the maximum packet size of the associated endpoint (*wMaxPacketSize*). The maximum value this field may contain is 0x400 (1024).
15	Head of Reclamation List Flag (H). This bit is set by System Software to mark a queue head as being the head of the reclamation list.
14	Data Toggle Control (DTC). This bit specifies where the host controller should get the initial data toggle on an overlay transition. 0b Ignore DT bit from incoming qTD. Host controller preserves DT bit in the queue head. 1b Initial data toggle comes from incoming qTD DT bit. Host controller replaces DT bit in the queue head from the DT bit in the qTD.
13–12	Endpoint Speed (EPS). This is the speed of the associated endpoint. Bit combinations are as follows: 00b Full-Speed (12Mbs) 01b Low-Speed (1.5Mbs) 10b High-Speed (480 Mb/s) 11b Reserved This field must not be modified by the host controller.
11–8	Endpoint Number (Endpt). This 4-bit field selects the particular endpoint number on the device serving as the data source or sink.

Table 32-62. Endpoint Characteristics: Queue Head DWord 1 (continued)

Field	Description
7	Inactivate on Next Transaction (I). This bit is used by system software to request that the host controller set the Active bit to zero. This field is only valid when the queue head is in the Periodic Schedule and the EPS field indicates a Full or Low-speed endpoint. Setting this bit to a one when the queue head is in the Asynchronous Schedule or the EPS field indicates a high-speed device yields undefined results.
6–0	Device Address. This field selects the specific device serving as the data source or sink.

Table 32-63 shows the endpoint characteristics: Queue Head DWord 2 field descriptions.

Table 32-63. Endpoint Capabilities: Queue Head DWord 2

Field	Description
31–30	High-Bandwidth Pipe Multiplier (Mult). This field is a multiplier used to key the host controller as the number of successive packets the host controller may submit to the endpoint in the current execution. The host controller makes the simplifying assumption that software properly initializes this field (regardless of location of queue head in the schedules or other run-time parameters). The valid values are as follows: 00b Reserved. A zero in this field yields undefined results. 01b One transaction to be issued for this endpoint per micro-frame 10b Two transactions to be issued for this endpoint per micro-frame 11b Three transactions to be issued for this endpoint per micro-frame
29–23	Port Number. This field is ignored by the host controller unless the *EPS* field indicates a full- or low-speed device. The value is the port number identifier on the USB 2.0 Hub (for hub at device address *Hub Addr* below), below which the full- or low-speed device associated with this endpoint is attached. This information is used in the split-transaction protocol.
22–16	Hub Addr. This field is ignored by the host controller unless the *EPS* field indicates a full-or low-speed device. The value is the USB device address of the USB 2.0 Hub below which the full- or low-speed device associated with this endpoint is attached. This field is used in the split-transaction protocol.

Table 32-63 shows the endpoint characteristics: Queue Head DWord 2 field descriptions.

Table 32-63. Endpoint Capabilities: Queue Head DWord 2 (continued)

Field	Description
15–8	Split Completion Mask (µFrame C-Mask). This field is ignored by the host controller unless the *EPS* field indicates this device is a low- or full-speed device and this queue head is in the periodic list. This field (along with the *Active* and *SplitX-state* fields) is used to determine during which micro-frames the host controller should execute a complete-split transaction. When the criteria for using this field are met, a zero value in this field has undefined behavior. This field is used by the host controller to match against the three low-order bits of the FRINDEX register. If the FRINDEX register bits decode to a position where the *µFrame C- Mask* field is a one, then this queue head is a candidate for transaction execution. There may be more than one bit in this mask set.
7–0	Interrupt Schedule Mask (µFrame S-mask). This field is used for all endpoint speeds. Software should set this field to a zero when the queue head is on the asynchronous schedule. A non-zero value in this field indicates an interrupt endpoint. The host controller uses the value of the three low-order bits of the FRINDEX register as an index into a bit position in this bit vector. If the *µFrame S-mask* field has a one at the indexed bit position then this queue head is a candidate for transaction execution. If the *EPS* field indicates the endpoint is a high-speed endpoint, then the transaction executed is determined by the *PID_Code* field contained in the execution area. This field is also used to support split transaction types: Interrupt (IN/OUT). This condition is true when this field is non-zero and the *EPS* field indicates this is either a full- or low-speed device. A zero value in this field, in combination with existing in the periodic frame list has undefined results.

32.10.7.2 Transfer Overlay

The nine DWords in this area represent a *transaction working space* for the host controller. The general operational model is that the host controller can detect whether the overlay area contains a description of an active transfer. If it does not contain an active transfer, then it follows the *Queue Head Horizontal Link Pointer* to the next queue head. The host controller will never follow the *Next Transfer Queue Element or Alternate Queue Element* pointers unless it is actively attempting to advance the queue. For the duration of the transfer, the host controller keeps the incremental status of the transfer in the overlay area. When the transfer is complete, the results are written back to the original queue element.

The DWord3 of a Queue Head contains a pointer to the source qTD currently associated with the overlay. The host controller uses this pointer to write back the overlay area into the source qTD after the transfer is complete.

Table 32-64 shows the current qTD link pointer field descriptions.

Table 32-64. Current qTD Link Pointer

Field	Description
31–5	Current Element Transaction Descriptor Link Pointer. This field contains the address Of the current transaction being processed in this queue and corresponds to memory address signals [31:5], respectively.
4–0	Reserved (R). These bits are ignored by the host controller when using the value as an address to write data. The actual value may vary depending on the usage.

The DWords 4-11 of a queue head are the transaction overlay area. This area has the same base structure as a Queue Element Transfer Descriptor. The queue head utilizes the reserved fields of the page pointers to implement tracking the state of split transactions.

This area is characterized as an *overlay* because when the queue is advanced to the next queue element, the source queue element is *merged* onto this area. This area serves an execution cache for the transfer.

Table 32-65. Host-Controller Rules for Bits in Overlay (DWords 5, 6, 8, and 9)

DWord	Field	Description
5	4–1	NAK Counter (NakCnt)µRW. This field is a counter the host controller decrements whenever a transaction for the endpoint associated with this queue head results in a NAK or NYET response. This counter is reloaded from *RL* before a transaction is executed during the first pass of the reclamation list (relative to an Asynchronous List Restart condition). It is also loaded from RL during an overlay.
6	31	Data Toggle. The *Data Toggle Control* controls whether the host controller preserves this bit when an overlay operation is performed.
6	15	Interrupt On Complete (IOC). The IOC control bit is always inherited from the source qTD when the overlay operation is performed.
6	11–10	Error Counter (C_ERR). This two-bit field is copied from the qTD during the overlay and written back during queue advancement.
6	0	Ping State (P)/ERR. If the *EPS* field indicates a high-speed endpoint, then this field should be preserved during the overlay operation.
8	7–0	Split-transaction Complete-split Progress (C-prog-mask). This field is initialized to zero during any overlay. This field is used to track the progress of an interrupt split-transaction.
9	4–0	Split-transaction Frame Tag (Frame Tag). This field is initialized to zero during any overlay. This field is used to track the progress of an interrupt split-transaction.
9	11–5	S-bytes. Software must ensure that the *S-bytes* field in a *qTD* is zero before activating the *qTD*. This field is used to keep track of the number of bytes sent or received during an IN or OUT split transaction.

32.10.8 Periodic Frame Span Traversal Node (FSTN)

This data structure is to be used only for managing Full- and Low-speed transactions that span a Host-frame boundary. See section Host Controller Operational Model for FSTNs for full operational

details. Software must not use an FSTN in the Asynchronous Schedule. An FSTN in the Asynchronous schedule results in undefined behavior. Software must not use the FSTN feature with a host controller whose HCIVERSION register indicates a revision implementation below 0096h. FSTNs are not defined for implementations before 0.96 and their use will yield undefined results.

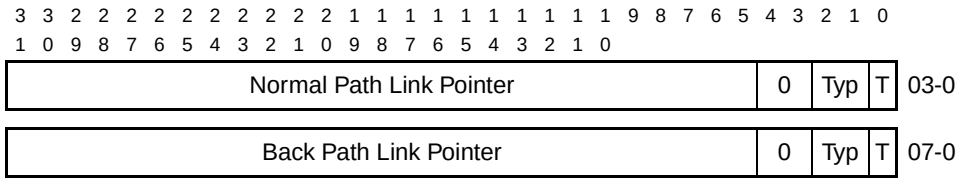
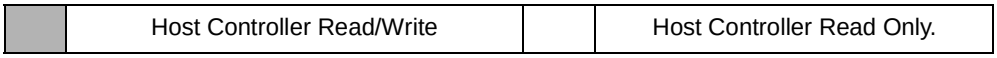


Figure 32-53. FSTN Register

Figure 32-54. Frame Span Traversal Node Structure Layout



32.10.8.1 FSTN Normal Path Pointer

The first DWord of an FSTN contains a link pointer to the next schedule object. This object can be of any valid periodic schedule data type.

Table 32-66. FSTN Normal Path Pointer Field Descriptions

Field	Description
31–5	Normal Path Link Pointer (NPLP). This field contains the address of the next data object to be processed in the periodic list and corresponds to memory address signals [31:5], respectively.
4–3	Reserved. These bits must be written as 0s.
2–1	QH/(s)iTD/FSTN Select (Typ). This field indicates to the Host Controller whether the item referenced is a iTD/siTD, a QH or an FSTN. This allows the Host Controller to perform the proper type of processing on the item after it is fetched. Value encodings are as follows: 00b iTD (isochronous transfer descriptor) 01b QH (queue head) 10b siTD (split transaction isochronous transfer descriptor) 11b FSTN (Frame Span Traversal Node)
0	Terminate (T). 1=Link Pointer field is not valid. 0=Link Pointer is valid.

32.10.8.2 FSTN Back Path Link Pointer

The second DWord of an FTSN node contains a link pointer to a queue head. If the *T-bit* in this pointer is a zero, then this FSTN is a *Save-Place* indicator. Its *Typ* field must be set by software to indicate the target data structure is a queue head. If the *T-bit* in this pointer is set to a one, then this FSTN is the *Restore* indicator. When the *T-bit* is a one, the host controller ignores the *Typ* field. Table 32-67 shows the FSTN Back Path Link Pointer field descriptions.

Table 32-67. FSTN Back Path Link Pointer Field Descriptions

Field	Description
31–5	Back Path Link Pointer (BPLP). This field contains the address of a Queue Head. This field corresponds to memory address signals [31:5], respectively.
4–3	Reserved. These bits must be written as 0s.
2–1	Typ. Software must ensure this field is set to indicate the target data structure is a Queue Head. Any other value in this field yields undefined results.
0	Terminate (T). 1=Link Pointer field is not valid (that is, the host controller must not use bits [31:5] (in combination with the CTRLDSSEGMENT register if applicable) as a valid memory address). This value also indicates that this FSTN is a Restore indicator. 0=Link Pointer is valid (that is, the host controller may use bits [31:5] (in combination with the CTRLDSSEGMENT register if applicable) as a valid memory address). This value also indicates that this FSTN is a Save-Place indicator.

32.11 Host Operational Model

The general operational model is for the enhanced interface host controller hardware and enhanced interface host controller driver (generally referred to as system software). Each significant operational feature of the EHCI host controller is discussed in a separate section. Each section presents the operational model requirements for the host controller hardware. Where appropriate, recommended system software operational models for features are also presented.

32.11.1 Host Controller Initialization

When the system boots, the host controller is enumerated, assigned a base address for the register space and BIOS sets the FLADJ register to a system-specific value. After initial power-on or *HCReset* (hardware or via *HCReset* bit in the USBCMD register), all of the operational registers will be at their default values, as illustrated in Table 32-68. After a hardware reset, only the operational registers not contained in the Auxiliary power well will be at their default values.

Table 32-68. Default Values of Operational Register Space

Operational Register	Default Value (after Reset)
USBCMD	00080000h (00080B00h if *Asynchronous Schedule Park Capability is a one*)
USBSTS	00001000h
USBINTR	00000000h
FRINDEX	00000000h
CTRLDSSEGMENT	00000000h
PERIODICLISTBASE	Undefined
ASYNCLISTADDR	Undefined

Table 32-68. Default Values of Operational Register Space (continued)

CONFIGFLAG	00000000h
PORTSC	00002000h (w/*PPC* set to one); 00003000h (w/*PPC* set to a zero)

To initialize the host controller, software should perform the following steps

- Program the CTRLDSSEGMENT register with 4-Gigabyte segment where all of the interface data structures are allocated.
- Write the appropriate value to the USBINTR register to enable the appropriate interrupts.
- Write the base address of the Periodic Frame List to the PERIODICLIST BASE register. If there are no work items in the periodic schedule, all elements of the Periodic Frame List should have their *T-Bits* set to a one.
- Write the USBCMD register to set the desired interrupt threshold, frame list size (if applicable) and turn the host controller *ON* via setting the *Run/Stop* bit.
- Write a 1 to CONFIGFLAG register to route all ports to the EHCI controller.

At this point, the host controller is up and running and the port registers will begin reporting device connects, among others. System software can enumerate a port through the reset process (where the port is in the enabled state). At this point, the port is active with SOFs occurring down the enabled port enabled High-speed ports, but the schedules have not yet been enabled. The EHCI Host controller will not transmit SOFs to enabled Full- or Low-speed ports. To communicate with devices via the asynchronous schedule, system software must write the ASYNDLISTADDR register with the address of a control or bulk queue head. Software must then enable the asynchronous schedule by writing a one to the *Asynchronous Schedule Enable* bit in the USBCMD register. To communicate with devices via the periodic schedule, system software must enable the periodic schedule by writing a one to the *Periodic Schedule Enable* bit in the USBCMD register. Note that the schedules can be turned on before the first port is reset (and enabled).

Any time the USBCMD register is written, system software must ensure the appropriate bits are preserved, depending on the intended operation.

32.11.2 Port Routing and Control

A USB 2.0 Host controller is comprised of one high-speed host controller, which implements the EHCI programming interface and 0 to N USB 1.1 companion host controllers. Companion host controllers (cHCs) may be implementations of either Universal or Open host controller specifications. This configuration is used to deliver the required full USB 2.0-defined port capability; for example, Low-, Full-, and High-Speed capability for every port. Figure 32-55 illustrates a simple block diagram of the port routing logic and its relationship to the high-speed and companion host controllers within a USB 2.0 host controller.

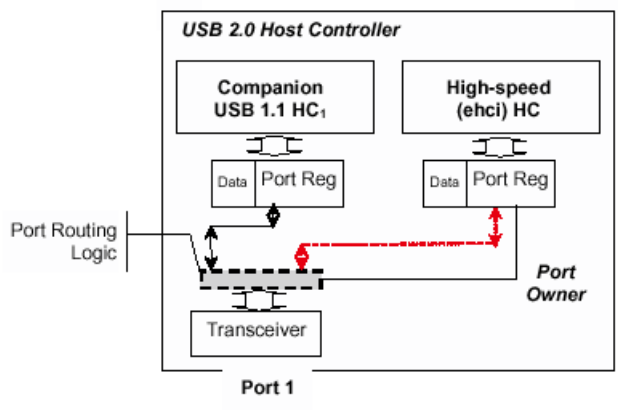


Figure 32-55. Example USB 2.0 Host Controller Port Routing Block Diagram

There exists one transceiver per physical port and each host controller module has its own port status and control registers. The EHCI controller has port status and control registers for every port. Each companion host controller has only the port control and status registers it is required to operate. Each transceiver can be controlled by either the EHCI host controller or one companion host controller. Routing logic lies between the transceiver and the port status and control registers.[1]

The port routing logic is controlled from signals originating in the EHCI host controller. The EHCI host controller has a *global* routing policy control field and per-port *ownership* control fields. The *Configured Flag* (*CF*) bit (defined in section BURSTSIZE) is the global routing policy control. At power-on or reset, the default routing policy is to the companion controllers (if they exist). If the system does not include a driver for the EHCI host controller and the host controller includes Companion Controllers, then the ports will still work in Full- and Low-speed mode (assuming the system includes a driver for the companion controllers). In general, when the EHCI owns the ports, the companion host controllers' port registers do not see a connect indication from the transceiver. Similarly, when a companion host controller owns a port, the EHCI controller's port registers do not see a connect indication from the transceiver. The details on the rules for the port routing logic are described in the following sections. The USB 2.0 host controller must be implemented as a multi-function PCI device if the implementation includes companion controllers. The companion host controllers' function numbers must be less than the EHCI host controller function number. The EHCI host controller must be a larger function number with respect to the companion host controllers associated with this EHCI host controller. If a PCI device implementation contains only an EHCI controller (that is, no companion controllers or other PCI functions), then the EHCI host controller must be function zero, in accordance with the PCI Specification. The *N_CC* field in the Structural Parameter register (*HCSPARAMS*) indicates whether the controller implementation includes companion host controllers. When *N_CC* has a non-zero value there exists companion host controllers. If N_CC has a value of zero, then the host controller implementation does not include companion host controllers. If the host controller root ports are exposed to attachment of full- or low-speed devices, the ports will always fail the high-speed chirp during reset and the ports will not be enabled. System software can notify the user of the illegal condition. This type of implementation requires a USB 2.0 hub be connected to a root port to provide full and low-speed device connectivity.

1.The routing logic should not be implemented in the 480 MHz clock domain of the transceiver.

System software uses information in the host controller capability registers to determine how the ports are routed to the companion host controllers. See Section 32.9.4.3, "HCSPARAMS—EHCI Compliant with Extensions."[1]

32.11.2.1 Port Routing Control via EHCI *Configured Flag* (*CF*) Bit

Each port in the USB 2.0 host controller can be routed either to a single companion host controller or to the EHCI host controller. The port routing logic is controlled by two mechanisms in the EHCI HC: a host controller global flag and per-port control. The *Configured Flag* (*CF*) bit (defined in Section BURSTSIZE), is used to globally set the policy of the routing logic. Each port register has a *Port Owner* control bit which allows the EHCI Driver to explicitly control the routing of individual ports. Whenever the *CF bit* transitions from a zero to a one (this transition is only available under program control) the port routing unconditionally routes all of the port registers to the EHCI HC (all *Port Owner* bits go to zero). While the *CF-bit* is a one, the EHCI Driver can control individual ports' routing via the *Port Owner* control bit. Likewise, whenever the *CF bit* transitions from a one to a zero (as a result of Aux power application, *HCRESET*, or software writing a zero to *CF-bit*), the port routing unconditionally routes all of the port registers to the appropriate companion HC. The default value for the EHCI HC's *CF bit* (after Aux power application or *HCRESET*) is zero. Table 32-69 summarizes the default routing for all the ports, based on the value of the EHCI HC's *CF bit*.

The *view* of the port depends on the current owner. A Universal or Open companion host controller will see port register bits consistent with the appropriate specification. Port bit definitions that are required for EHCI host controllers are not visible to companion host controllers.

Table 32-69. Default Port Routing Depending on EHCI HC CF Bit

HS CF Bit	Default Port Ownership	Explanation
0B	Companion HCs	The companion host controllers own the ports and only Full- and Low-speed devices are supported in the system. The exact port assignments are implementation dependent. The ports behave only as Full- and Low-speed ports in this configuration
1B	EHCI HC	The EHCI host controller has default ownership over all of the ports. The routing logic inhibits device connect events from reaching the companion HCs' port status and control registers when the port owner is the EHCI HC.The EHCI HC has access to the additional port status and control bits defined in this chapter (see Figure 32-35, PortSCx). The EHCI HC can temporarily release control of the port to a companion HC by setting the *PortOwner* bit in the PORTSC register to a one.

1.If an implementation includes more than one set of companion and EHCI host controllers, they are organized as groups of companion host controllers with intermixed EHCI controllers.

32.11.2.2 Port Routing Control via *PortOwner* and Disconnect Event

Manipulating the port routing via the *CF-bit* is an extreme process and not intended to be used during normal operation. The normal mode of port ownership transferal is on the granularity of individual ports using the *Port Owner* bit in the EHCI HC's PORTSC register (for hand-offs from EHCI to companion host controllers). Individual port ownership is returned to the EHCI controller when the port registers a device disconnect. When the disconnect is detected, the port routing logic immediately returns the port ownership to the EHCI controller. The companion host controller port register detects the device disconnect and operates normally.

Under normal operating conditions (assuming all HC drivers loaded and operational and the EHCI *CF-bit* is set to a one), the typical port enumeration sequence proceeds as illustrated below:

- Initial condition is that EHCI is port owner. A device is connected causing the port to detect a connect, set the port connect change bit and issue a port-change interrupt (if enabled).
- EHCI Driver identifies the port with the new connect change bit asserted and sends a change report to the hub driver. Hub driver issues a GetPortStatus() request and identifies the connect change. It then issues a request to clear the connect change, followed by a request to reset and enable the port.
- When the EHCI Driver receives the request to reset and enable the port, it first checks the value reported by the *LineStatus* bits in the PORTSC register. If they indicate the attached device is a full-speed device (for example, D+ is asserted), then the EHCI Driver sets the *PortReset* control bit to a one (and sets the *PortEnable* bit to a zero) which begins the reset-process. Software times the duration of the reset, then terminates reset signaling by writing a zero to the port reset bit. The reset process is actually complete when software reads a zero in the *PortReset* bit. The EHCI Driver checks the *PortEnable* bit in the PORTSC register. If set to a one, the connected device is a high-speed device and EHCI Driver (root hub emulator) issues a change report to the hub driver and the hub driver continues to enumerate the attached device.
- At the time the EHCI Driver receives the port reset and enable request the *LineStatus* bits might indicate a low-speed device. Additionally, when the port reset process is complete, the *PortEnable* field may indicate that a full-speed device is attached. In either case the EHCI driver sets the *PortOwner* bit in the PORTSC register to a one to release port ownership to a companion host controller.
- When the EHCI Driver sets *PortOwner* bit to a one, the port routing logic makes the connection state of the transceiver available to the companion host controller port register and removes the connection state from the EHCI HC port. The EHCI PORTSC register observes and reports a disconnect event via the disconnect change bit. The EHCI Driver detects the connection status change (either by polling or by port change interrupt) and then sends a change report to the hub driver. When the hub driver requests that port-state, the EHCI Driver responds with a reset complete change set to a one, a connect change set to a one and a connect status set to a zero. This information is derived directly from the EHCI port register. This allows the hub driver to assume the device was disconnected during reset. It will acknowledge the change bits and wait for the next change event. While the EHCI controller does not own the port, it simply remains in a state where the port reports no device connected.
  The device-connect evaluation circuitry of the companion HC activates and detects the device, the companion Driver detects the connection and enumerates the port.

When a port is routed to a companion HC, it remains under the control of the companion HC until the device is disconnected from the root port (ignoring for now the scenario where EHCI's *CF-bit* transitions from a 1b to a 0b). When a disconnect occurs, the disconnect event is detected by both the companion HC port control and the EHCI port ownership control. On the event, the port ownership is returned immediately to the EHCI controller. The companion HC stack detects the disconnect and acknowledges as it would in an ordinary standalone implementation. Subsequent connects will be detected by the EHCI port register and the process will repeat.

32.11.2.3 Example Port Routing State Machine

Figure 32-56 illustrates an example of how the port ownership should be managed. The following sections describe the entry conditions to each state.

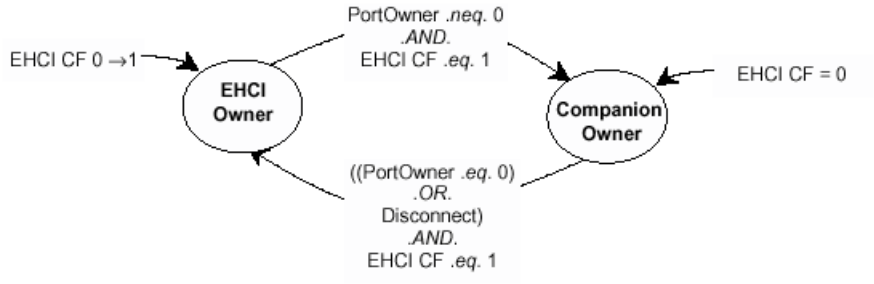


Figure 32-56. Port Owner Handoff State Machine

32.11.2.3.1 EHCI HC Owner

Entry to this state occurs whenever one of the following events occur:

- When the EHCI HC's *Configure Flag* (*CF*) bit in the CONFIGFLAG register transitions from a zero to a one. This signals the fact that the system has a host controller driver for the EHCI HC and that all ports in the USB 2.0 host controller must default route to the EHCI controller.
- When the port is owned by a companion HC and the device is disconnected from the port. The EHCI port routing control logic is notified of the disconnect, and returns port routing to the EHCI controller. The connection state of the companion HC goes immediately to the disconnected state (with appropriate side effect to connect change, enable and enable change). The companion HC driver will acknowledge the disconnect by setting the connect status change bit to a zero. This allows the companion HC's driver to interact with the port completely through the disconnect process.
- When system software writes a zero to the *PortOwner* bit in the PORTSC register. This allows software to take ownership of a port from a companion host controller. When this occurs, the routing logic to the companion HC effectively signals a disconnect to the companion HC's port status and control register.

32.11.2.3.2 Companion HC Owner

Entry to this state occurs whenever one of the following events occur:

- When the *Port Owner* field transitions from a zero to a one.

- When the HS-mode HC's *Configure Flag* (*CF*) is equal to zero.

On entry to this state, the routing logic allows the companion HC port register to detect a device connect. Normal port enumeration proceeds.

32.11.2.4 Port Power

The *Port Power Control* (*PPC*) bit in the HCSPARAMS register indicates whether the USB 2.0 host controller has port power control (See section Section 32.9.4.3, "HCSPARAMS—EHCI Compliant with Extensions"). When this bit is a zero, then the host controller does not support software control of port power switches. When in this configuration, the port power is always available and the companion host controllers must implement functionality consistent with port power always on. When the *PPC* bit is a one, then the host controller implementation includes port power switches. Each available switch has an output enable, which is referred to in this discussion as *PortPowerOutputEnable* (*PPE*). *PPE* is controlled based on the state of the combination bits *PPC* bit, EHCI *Configured* (*CF*)-bit and individual *Port Power* (*PP*) bits. Table 32-70 Port Power Enable Control Rules illustrates the summary behavioral model.

Table 32-70. Port Power Enable Control Rules

CF	CHC[2] (PP)	EHC[3] (PP)	Owner	PPE[1]	Description
0	0	X	CHC	0	When the EHCI controller has not been configured, the port is owned by the companion host controller. When the companion HC's port power select is off, then the port power is off.
0	1	X	CHC	1	Similar to previous entry. When the companion HC's port power select is on, then the port power is on.
1	0	0	CHC	0	Port owner has port power turned off, the power to port is off.
1	0	0	EHC	0	Port owner has port power turned off, the power to port is off.
1	0	1	EHC	1	Port owner has port power on, so power to port is on.
1	0	1	CHC	1	If either HC has port power turned on, the power to the port is on.
1	1	0	EHC	1	If either HC has port power turned on, the power to the port is on.
1	1	0	CHC	1	Port owner has port power on, so power to port is on.
1	1	1	CHC	1	Port owner has port power on, so power to port is on.
1	1	1	EHC	1	Port owner has port power on, so power to port is on.

a) [1]PPE (Port Power Enable). This bit actually turns on the port power switch (if one exists).

a) [2]CHC (Companion Host Controller).

a) [3]EHC (EHCI Host Controller).

32.11.2.5 Port Reporting Over-Current

Host controllers are by definition power providers on USB. Whether the ports are considered high- or low-powered is a platform implementation issue. Each EHCI PORTSC register has an over-current status and over-current change bit. The functionality of these bits is specified in the USB Specification Revision 2.0.

The over current detection and limiting logic usually resides outside the host controller logic. This logic may be associated with one or more ports. When this logic detects an over-current condition it is made available to both the companion and EHCI ports. The effect of an over-current status on a companion host controller port is beyond the scope of this document. The over-current condition effects the following bits in the PORTSC register on the EHCI port:

- *Over-current Active* bits are set to a one. When the over-current condition goes away, the *Over-current Active* bit will transition from a one to a zero.
- *Over-current Change* bits are set to a one. On every transition of the *Over-current Active* bit the host controller will set the *Over-current Change* bit to a one. Software sets the *Over-current Change* bit to a zero by writing a one to this bit.
- *Port Enabled/Disabled* bit is set to a zero. When this change bit gets set to a one, then the *Port Change Detect* bit in the USBSTS register is set to a one.
- *Port Power (PP)* bits may optionally be set to a zero. There is no requirement in USB that a power provider shut off power in an over current condition. It is sufficient to *limit* the current and leave power applied. When the *Over-current Change* bit transitions from a zero to a one, the host controller also sets the *Port Change Detect* bit in the USBSTS register to a one. In addition, if the *Port Change Interrupt Enable* bit in the USBINTR register is a one, then the host controller will issue an interrupt to the system. Refer to Table 32-71 for summary behavior for over-current detection when the host controller is halted (suspended from a device component point of view).

32.11.3 Suspend/Resume

The EHCI host controller provides an equivalent suspend and resume model as that defined for individual ports in a USB 2.0 Hub. Control mechanisms are provided to allow system software to suspend and resume individual ports. The mechanisms allow the individual ports to be resumed completely via software initiation. Other control mechanisms are provided to parameterize the host controller's response (or sensitivity) to external resume events. In this discussion, host-initiated, or software initiated resumes are called Resume Events/Actions. Bus-initiated resume events are called wake-up events. The classes of wakeup events are as follows:

- Remote-wakeup enabled device asserts resume signaling. In similar kind to USB 2.0 Hubs, EHCI controllers must always respond to explicit device resume signaling and wake up the system (if necessary).
- Port connect and disconnect and over-current events. Sensitivity to these events can be turned on or off by using the per-port control bits in the PORTSC registers.

Selective suspend is a feature supported by every PORTSC register. It is used to place specific ports into a suspend mode. This feature is used as a functional component for implementing the appropriate power management policy implemented in a particular operating system. When system software intends to suspend the entire bus, it should selectively suspend all enabled ports, then shut off the host controller by setting the *Run/Stop* bit in the USBCMD register to a zero. The EHCI module can then be placed into a lower device state via the PCI power management interface (See Appendix A, Enhanced Host Controller Interface Specification for Universal Serial Bus, Revision 0.95, November 2000, Intel Corporation. http://www.intel.com).

When a wake event occurs the system will resume operation and system software will eventually set the *Run/Stop* bit to a one and resume the suspended ports. Software must not set the *Run/Stop* bit to a one until it is confirmed that the clock to the host controller is stable. This is usually confirmed in a system implementation in that all of the clocks in the system are stable before the CPU is restarted. So, by definition, if software is running, clocks in the system are stable and the *Run/Stop* bit in the USBCMD register can be set to a one. There are also minimum system software delays defined in the *PCI Power Management Specification*. Refer to this specification for more information.

32.11.3.1 Port Suspend/Resume

System software places individual ports into suspend mode by writing a one into the appropriate PORTSC *Suspend* bit. Software must only set the *Suspend* bit when the port is in the enabled state (*Port Enabled* bit is a one) and the EHCI is the port owner (*Port Owner* bit is a zero).

The host controller may evaluate the *Suspend* bit immediately or wait until a micro-frame or frame boundary occurs. If evaluated immediately, the port is not suspended until the current transaction (if one is executing) completes. Therefore, there may be several micro-frames of activity on the port until the host controller evaluates the *Suspend* bit. The host controller must evaluate the *Suspend* bit at least every frame boundary.

System software can initiate a resume on a selectively suspended port by writing a one to the *Force Port Resume* bit. Software should not attempt to resume a port unless the port reports that it is in the suspended state (see Figure 32-35, PORTSCx). If system software sets *Force Port Resume* bit to a one when the port is not in the suspended state, the resulting behavior is undefined. To assure proper USB device operation, software must wait for at least 10 milliseconds after a port indicates that it is suspended (*Suspend* bit is a one) before initiating a port resume via the *Force Port Resume* bit. When *Force Port Resume* bit is a one, the host controller sends resume signaling down the port. System software times the duration of the resume (nominally 20 milliseconds) then sets the *Force Port Resume* bit to a zero. When the host controller receives the write to transition *Force Port Resume* to zero, it completes the resume sequence as defined in the USB specification, and sets both the *Force Port Resume* and *Suspend* bits to zero. Software-initiated port resumes do not affect the *Port Change Detect* bit in the USBSTS register nor do they cause an interrupt if the *Port Change Interrupt Enable* bit in the USBINTR register is a one. An external USB event may also initiate a resume. The wake events are defined above. When a wake event occurs on a suspended port, the resume signaling is detected by the port and the resume is reflected downstream within 100 µsec. The port's *Force Port Resume* bit is set to a one and the *Port Change Detect* bit in the USBSTS register is set to a one. If the *Port Change Interrupt Enable* bit in the USBINTR register is a one the host controller will issue a hardware interrupt.

System software observes the resume event on the port, delays a port resume time (nominally 20 ms), then terminates the resume sequence by writing zero to the *Force Port Resume* bit in the port. The host controller receives the write of zero to *Force Port Resume*, terminates the resume sequence and sets *Force Port Resume* and *Suspend* port bits to zero. Software can determine that the port is enabled (not suspended) by sampling the PORTSC register and observing that the *Suspend* and *Force Port Resume* bits are zero. Software must ensure that the host controller is running (that is, *HCHalted* bit in the USBSTS register is a zero), before terminating a resume by writing a zero to a port's *Force Port Resume* bit. If *HCHalted* is a one when *Force Port Resume* is set to a zero, then SOFs will not occur down the enabled port and the device will return to suspend mode in a maximum of 10 milliseconds.

Table 32-71 summarizes the wake-up events. Whenever a resume event is detected, the *Port Change Detect* bit in the USBSTS register is set to a one. If the *Port Change Interrupt Enable* bit is a one in the USBINTR register, the host controller will also generate an interrupt on the resume event. Software acknowledges the resume event interrupt by clearing the *Port Change Detect* status bit in the USBSTS register.

Table 32-71. Behavior During Wake-up Events

Port Status and Signaling Type	Signaled Port Response	Device State	
		D0	not D0
Port disabled, resume K-State received	No Effect	N/A	N/A
Port suspended, Resume K-State received	Resume reflected downstream on signaled port. Force Port Resume status bit in PORTSC register is set to a one. Port Change Detect bit in USBSTS register set to a one.	[1], [2]	[2]
Port is enabled, disabled or suspended, and the port's WKDSCNNT_E bit is a one. A disconnect is detected.	Depending in the initial port state, the PORTSC Connect and Enable status bits are set to zero, and the Connect Change status bit is set to a one. Port Change Detect bit in the USBSTS register is set to a one.	[1], [2]	[2]
Port is enabled, disabled or suspended, and the port's WKDSCNNT_E bit is a zero. A disconnect is detected.	Depending on the initial port state, the PORTSC Connect and Enable status bits are set to zero, and the Connect Change status bit is set to a one. Port Change Detect bit in the USBSTS register is set to a one.	[1], [3]	[3]
Port is not connected and the port's WKCNNT_E bit is a one. A connect is detected.	PORTSC Connect Status and Connect Status Change bits are set to a one. Port Change Detect bit in the USBSTS register is set to a one.	[1], [2]	[2]
Port is not connected and the port's WKCNNT_E bit is a zero. A connect is detected.	PORTSC Connect Status and Connect Status Change bits are set to a one. Port Change Detect bit in the USBSTS register is set to a one.	[1], [3]	[3]
Port is connected and the port's WKOC_E bit is a one. An over-current condition occurs.	PORTSC Over-current Active, Over-current Change bits are set to a one. If Port Enable/Disable bit is a one, it is set to a zero. Port Change Detect bit in the USBSTS register is set to a one	[1], [2]	[2]
Port is connected and the port's WKOC_E bit is a zero. An over-current condition occurs.	PORTSC Over-current Active, Over-current Change bits are set to a one. If Port Enable/Disable bit is a one, it is set to a zero. Port Change Detect bit in the USBSTS register is set to a one.	[1], [3]	[3]

[1] Hardware interrupt issued if Port Change Interrupt Enable bit in the USBINTR register is a one.

[2] PME# asserted if enabled (Note: PME Status must always be set to a one).

[3] PME# not asserted.

32.11.4 Schedule Traversal Rules

The host controller executes transactions for devices using a simple, shared-memory schedule. The schedule is comprised of a few data structures, organized into two distinct lists. The data structures are designed to provide the maximum flexibility required by USB, minimize memory traffic and hardware/software complexity.

System software maintains two schedules for the host controller: a periodic schedule and an asynchronous schedule. The root of the periodic schedule is the *PERIODICLISTBASE* register (see Section PERIODICLISTBASE; DEVICEADDR). The *PERIODICLISTBASE* register is the physical memory base address of the periodic frame list. The periodic frame list is an array of physical memory pointers. The objects referenced from the frame list must be valid schedule data structures as defined in Host Data Structures. In each micro-frame, if the periodic schedule is enabled (see Section Periodic Schedule) then the host controller must execute from the periodic schedule before executing from the asynchronous schedule. It will only execute from the asynchronous schedule after it encounters the end of the periodic schedule. The host controller traverses the periodic schedule by constructing an array offset reference from the *PERIODICLISTBASE* and the *FRINDEX* registers (see Figure 32-57). It fetches the element and begins traversing the graph of linked schedule data structures.

The end of the periodic schedule is identified by a *next* link pointer of a schedule data structure having its *T-bit* set to a one. When the host controller encounters a *T-Bit* set to a one during a horizontal traversal of the periodic list, it interprets this as an End-Of-Periodic-List mark. This causes the host controller to cease working on the periodic schedule and transitions immediately to traversing the asynchronous schedule. Once this transition is made, the host controller executes from the asynchronous schedule until the end of the micro-frame.

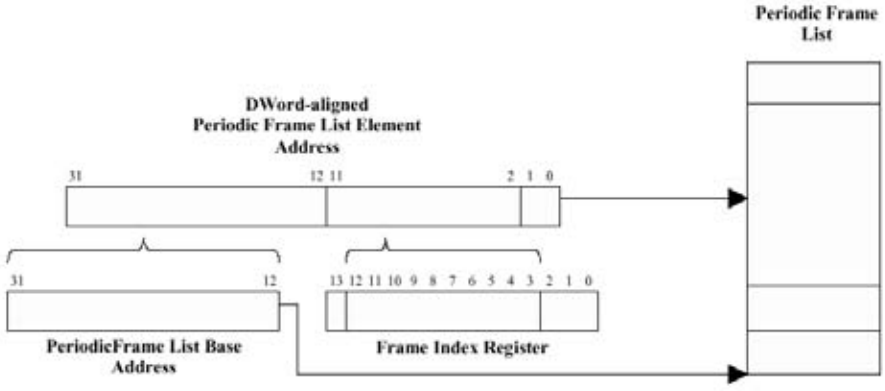


Figure 32-57. Derivation of Pointer into Frame List Array

When the host controller determines that it is time to execute from the asynchronous list, it uses the operational register *ASYNCLISTADDR* to access the asynchronous schedule, see Figure 32-58.

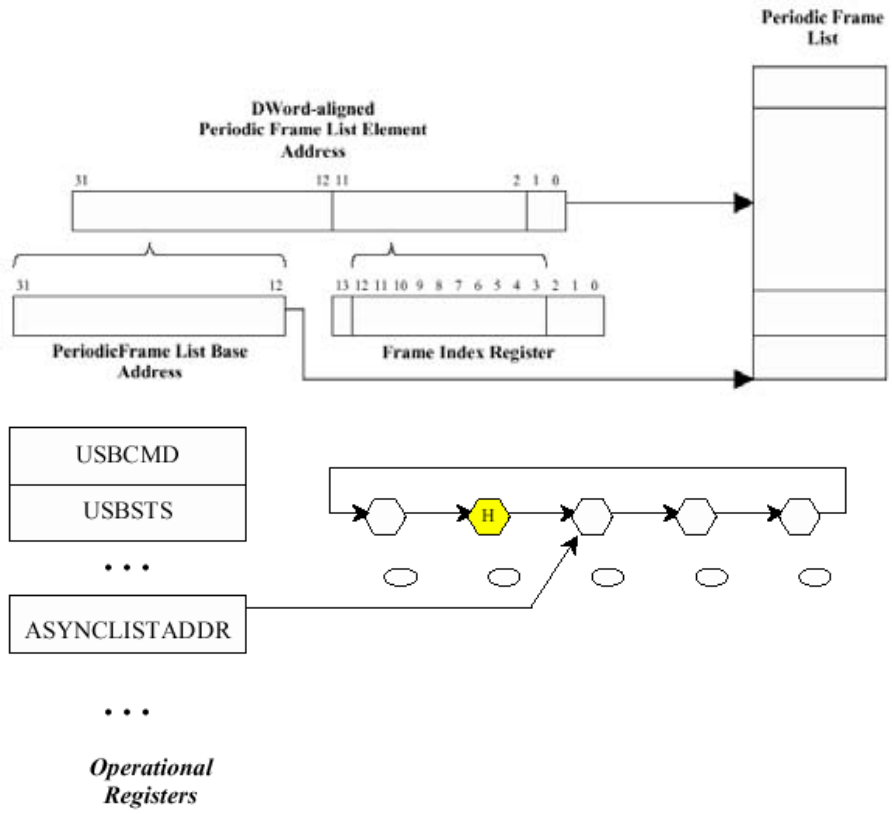


Figure 32-58. General Format of Asynchronous Schedule List

The *ASYNCLISTADDR* register contains a physical memory pointer to the *next* queue head. When the host controller makes a transition to executing the asynchronous schedule, it begins by reading the queue head referenced by the *ASYNCLISTADDR* register. Software must set queue head *horizontal* pointer *T-bits* to a zero for queue heads in the asynchronous schedule. See Section Asynchronous Schedule for complete operational details.

32.11.4.1 Example: Preserving Micro-Frame Integrity

One of the requirements of a USB host controller is to maintain *Frame Integrity.* This means that the HC must preserve the micro-frame boundaries. For example: SOF packets must be generated on time (within the specified allowable jitter), and High-speed EOF1,2 thresholds must be enforced. The end of micro-frame timing points EOF1 and EOF2 are clearly defined in the USB Specification Revision 2.0. One implication of this responsibility is that the HC must ensure that it does not start transactions that will not be completed before the end of the micro-frame. More precisely, no transactions should be started by the host controller, which cannot be completed in their entirety before the EOF1 point. To enforce this rule, the host controller must check each transaction before it starts to ensure that it will complete before the end of the micro-frame.

So, what exactly needs to be involved in this check? Fundamentally, the transaction data payload, plus bit stuffing, plus transaction overhead must be taken into consideration. It is possible to be extremely accurate on how much time the next transaction will take. Take OUTs for an example. The host controller must fetch

all of the OUT data from memory to send it onto the USB bus. A host controller implementation could pre-fetch *all* of the OUT data, and pre-compute the actual number of bits in the token and data packets. In addition, the system knows the depth of the target endpoint, so it could closely estimate turnaround time for handshake. In addition, the host controller knows the size of a handshake packet. Pre-computing effects of bit stuffing and summing up the other overhead numbers could allow the host controller to know exactly whether there was enough bus time, before EOF1 to complete the OUT transaction. To accomplish this particular approach takes an inordinate amount of time and hardware complexity.

The alternative is to make a reasonable guess whether the next transaction can be started. An example approximation algorithm is described below. This example algorithm relies on the EHCI policy that periodic transactions are scheduled first in the micro-frame. It is a reasonable assumption that software will never over-commit the micro-frame to periodic transactions greater than the specification allowable 80%. In the available remaining 20% bandwidth, the host controller has some ability (in this example) to decide whether or not to execute a transaction. The result of this algorithm is that sometimes, under some circumstances a transaction will not be executed that *could* have been executed. However, under all circumstances, a transaction will never be started unless there is enough time in the frame to complete the transaction.

Transaction Fit—A Best-Fit Approximation Algorithm

A curve is calculated which represents the *latest* start time for every packet size, at which software will schedule the start of a periodic transaction. This curve is the 80% bandwidth curve. Another curve is calculated which is the absolute, latest permitted start time for every packet size. This curve represents the absolute latest time, that a transaction of each packet size can be started and completed, in the micro-frame. A plot of these two curves is illustrated in Figure 32-59. The plot Y-axis represents the number of byte-times left in a frame.

The space between the *80%* and the *Last Start* plots is bandwidth reclamation area. In this algorithm the host controller may skip transactions during this time if it is prudent.

The Best-Fit Approximation method plots a function ($f(x)$) between the 80% and *Last Start* curves. The function $f(x)$ adds a constant to every transaction's maximum packet size and the result compared with the number of bytes left in the frame. The constant represents an approximation of the effects of bit stuffing and protocol overhead. The host controller starts transactions whose results land *above* the function curve. The host controller will not start transactions whose results land below the function curve.

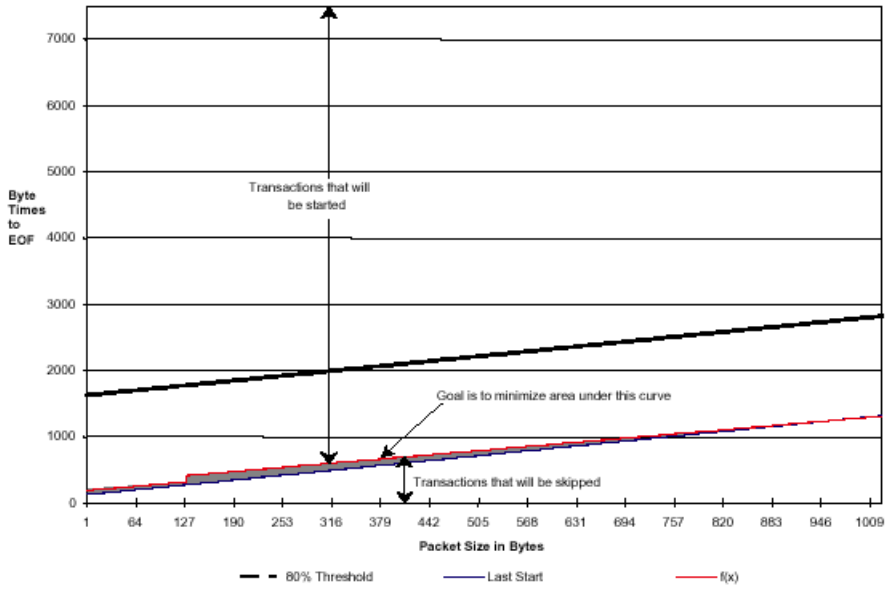


Figure 32-59. Best Fit Approximation

The *LastStart* line was calculated in this example to assume the absolute worst-case bus overhead per transaction. The particular transaction used was a start-split, zero-length OUT transaction with a handshake. Summaries of the component parts are listed in Table 32-72. The component times were derived from the protocol timings defined in the USB Specification Revision 2.0.

Table 32-72. Example Worse-case Transaction Timing Components

Component	Bit time	Byte Time	Explanation
Split Token	76	9.5	Split token as defined in USB core specification. Includes sync, token, eop, and so on.
Host 2 Host IPG	88	11	Number of bit times required between consecutive host packets.
Token	67	8.375	Token as defined in USB core specification. Includes sync, token, eop, and so on.
Host 2 Host IPG	88	11	Same as above
Data Packet (0 data bytes)	66.7	8.34	Zero-length data packet. Includes sync, PID, crc16, eop, and so on.
Turnaround time	721	90.125	Time for packet initiator (Host) to see the beginning of a response to a transmitted packet.
Handshake packet	48	6	Handshake packet as defined in USB core specification. Includes sync, PID, eop, and so on.
		144	Total

The exact details of the function (f(x)) are up to the particular implementation. However, it should be obvious that the goal is to minimize the area under the curve between the approximation function and the

Last Start curve, without dipping below the *LastStart* line, while at the same time keeping the check as simple as possible for hardware implementation. The f(x) in Figure 32-59 was constructed using the following pseudo-code test on each transaction size data point. This algorithm assumes that the host controller keeps track of the remaining bits in the frame.

```
Algorithm CheckTransactionWillFit (MaximumPacketSize, HC_BytesLeftInFrame)
Begin
        Local Temp = MaximumPacketSize + 192
        Local rvalue = TRUE

        If MaximumPacketSize >= 128 then
                Temp += 128
        End If

        If Temp > HC_BytesLeftInFrame then
                Rvalue = FALSE
        End If
        Return rvalue
End
```

This algorithm takes two inputs, the current maximum packet size of the transaction and a hardware counter of the number of bytes left in the current micro-frame. It unconditionally adds a simple constant of 192 to the maximum packet size to account for a first-order effect of transaction overhead and bit stuffing. If the transaction size is greater than or equal to 128 bytes, then an additional constant of 128 is added to the running sum to account for the additional worst-case bit stuffing of payloads larger than 128. An inflection point was inserted at 128 because the *f*(*x*) plot was getting *close* to the *LastStart* line.

32.11.5 Periodic Schedule Frame Boundaries Versus Bus Frame Boundaries

The USB Specification Revision 2.0 requires that the frame boundaries (SOF frame number changes) of the high-speed bus and the full- and low-speed bus(s) below USB 2.0 Hubs be strictly aligned. Super-imposed on this requirement is that USB 2.0 Hubs manage full- and low-speed transactions via a micro-frame pipeline (see start- (SS) and complete- (CS) splits illustrated in Figure 32-60). A simple, direct projection of the frame boundary model into the host controller interface schedule architecture creates tension (complexity for both hardware and software) between the frame boundaries and the scheduling mechanisms required to service the full- and low-speed transaction translator periodic pipelines.

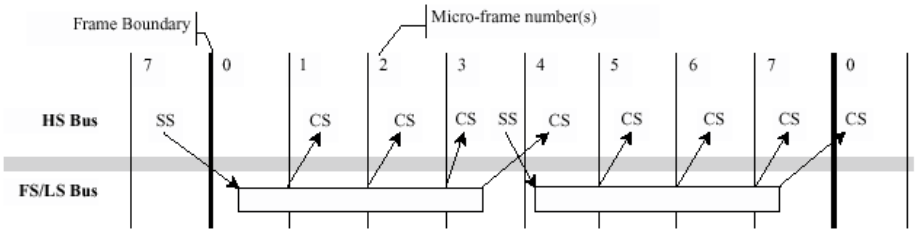


Figure 32-60. Frame Boundary Relationship Between HS bus and FS/LS Bus

The simple projection, as Figure 32-60 illustrates, introduces frame-boundary wrap conditions for scheduling on both the beginning and end of a frame. To reduce the complexity for hardware and software, the host controller is required to implement a one micro-frame phase shift for its view of frame boundaries. The phase shift eliminates the beginning of frame and frame-wrap scheduling boundary conditions.

The implementation of this phase shift requires that the host controller use one register value for accessing the periodic frame list and another value for the frame number value included in the SOF token. These two values are separate, but tightly coupled. The periodic frame list is accessed via the Frame List Index Register (FRINDEX) documented in Section FRINDEX and initially illustrated in Section Schedule Traversal Rules . Bits FRINDEX[2:0], represent the micro-frame number. The SOF value is coupled to the value of FRINDEX[13:3]. Both FRINDEX[13:3] and the SOF value are incremented based on FRINDEX[2:0]. It is required that the SOF value be delayed from the FRINDEX value by one micro-frame. The one micro-frame delay yields host controller periodic schedule and bus frame boundary relationship as illustrated in Figure 32-61. This adjustment allows software to trivially schedule the periodic start and complete-split transactions for full-and low-speed periodic endpoints, using the natural alignment of the periodic schedule interface. The reasons for selecting this phase-shift are beyond the scope of this module chapter.

Figure 32-61 illustrates how periodic schedule data structures relate to schedule frame boundaries and bus frame boundaries. To aid the presentation, two terms are defined. The host controller's view of the 1-millisecond boundaries is called *H-Frames*. The high-speed bus's view of the 1-millisecond boundaries is called *B-Frames*.

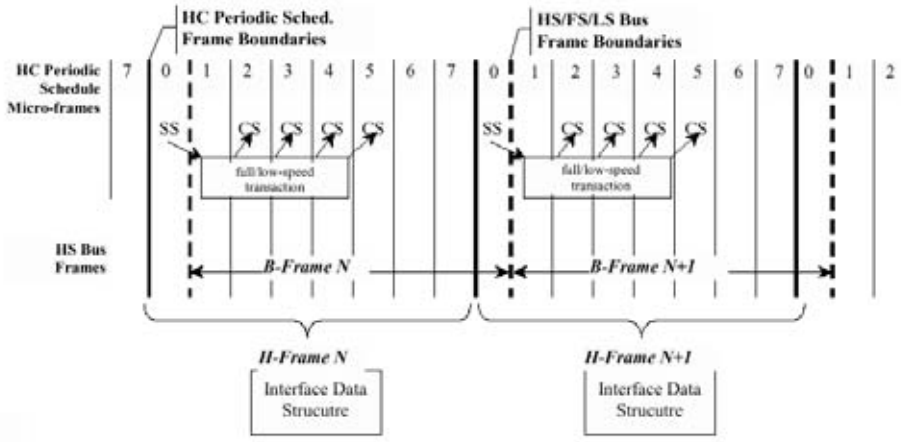


Figure 32-61. Relationship of Periodic Schedule Frame Boundaries to Bus Frame Boundaries

H-Frame boundaries for the host controller correspond to increments of FRINDEX[13:3]. Micro-frame numbers for the *H-Frame*

are tracked by FRINDEX[2:0]. *B-Frame* boundaries are visible on the high-speed bus via changes in the SOF token's frame number. Micro-frame numbers on the high-speed bus are only derived from the SOF token's frame number (that is, the high-speed bus will see eight SOFs with the same frame number value). *H-Frames*

and *B-Frames* have the fixed relationship (that is, *B-Frames* lag *H-Frames* by one micro-frame time) illustrated in Figure 32-61. The host controller's periodic schedule is naturally aligned to *H-Frames*.

Software schedules transactions for full- and low-speed periodic endpoints relative the *H-Frames*. The result is these transactions execute on the high-speed bus at exactly the right time for the USB 2.0 Hub periodic pipeline. As described in Section FRINDEX, the SOF Value can be implemented as a shadow register (in this example, called SOFV), which lags the FRINDEX register bits [13:3] by one micro-frame count. Table 32-73 illustrates the required relationship between the value of FRINDEX and the value of SOFV. This lag behavior can be accomplished by incrementing FRINDEX[13:3] based on carry-out on the 7 to 0 increment of FRINDEX[2:0] and incrementing SOFV based on the transition of 0 to 1 of FRINDEX[2:0].

Software is allowed to write to FRINDEX. Section FRINDEX provides the requirements that software should adhere when writing a new value in FRINDEX.

Table 32-73. Operation of FRINDEX and SOFV (SOF Value Register)

	Current			Next	
FRINDEX[F]	SOFV	FRINDEX[mF]	FRINDEX[F]	SOFV	FRINDEX[mF]
N	N	111b	N+1	N	000b
N+1	N	000b	N+1	N+1	001b
N+1	N+1	001b	N+1	N+1	010b
N+1	N+1	010b	N+1	N+1	011b
N+1	N+1	011b	N+1	N+1	100b
N+1	N+1	100b	N+1	N+1	101b
N+1	N+1	101b	N+1	N+1	110b
N+1	N+1	110b	N+1	N+1	111b

Where [F] = [13:3]; [μF] = [2:0]

32.11.6 Periodic Schedule

The periodic schedule traversal is enabled or disabled via the *Periodic Schedule Enable* bit in the USBCMD register. If the *Periodic Schedule Enable* bit is set to a zero, then the host controller simply does not try to access the periodic frame list via the *PERIODICLISTBASE* register. Likewise, when the *Periodic Schedule Enable* bit is a one, then the host controller does use the *PERIODICLISTBASE* register to traverse the periodic schedule. The host controller will not react to modifications to the *Periodic Schedule Enable* immediately. To eliminate conflicts with split transactions, the host controller evaluates the *Periodic Schedule Enable* bit only when FRINDEX[2:0] is zero. System software must not disable the periodic schedule if the schedule contains an active split transaction work item that spans the 000b micro-frame. These work items must be removed from the schedule before the *Periodic Schedule Enable* bit is written to a zero. The *Periodic Schedule Status* bit in the USBSTS register indicates status of the periodic schedule. System software enables (or disables) the periodic schedule by writing a one (or zero) to the *Periodic Schedule Enable* bit in the USBCMD register. Software then can poll the *Periodic Schedule Status* bit to determine when the periodic schedule has made the desired transition. Software must not modify the *Periodic Schedule Enable* bit unless the value of the *Periodic Schedule Enable* bit equals that of the *Periodic Schedule Status* bit.

The periodic schedule is used to manage all isochronous and interrupt transfer streams. The base of the periodic schedule is the periodic frame list. Software links schedule data structures to the periodic frame list to produce a graph of scheduled data structures. The graph represents an appropriate sequence of transactions on the USB. <Blue Cross-Reference>Figure 32-62. illustrates isochronous transfers (using iTDs and siTDs) with a period of one are linked directly to the periodic frame list. Interrupt transfers (are managed with queue heads) and isochronous streams with periods other than one are linked following the period-one iTD/siTDs. Interrupt queue heads are linked into the frame list ordered by poll rate. Longer poll rates are linked first (for example, closest to the periodic frame list), followed by shorter poll rates, with queue heads with a poll rate of one, on the very end.

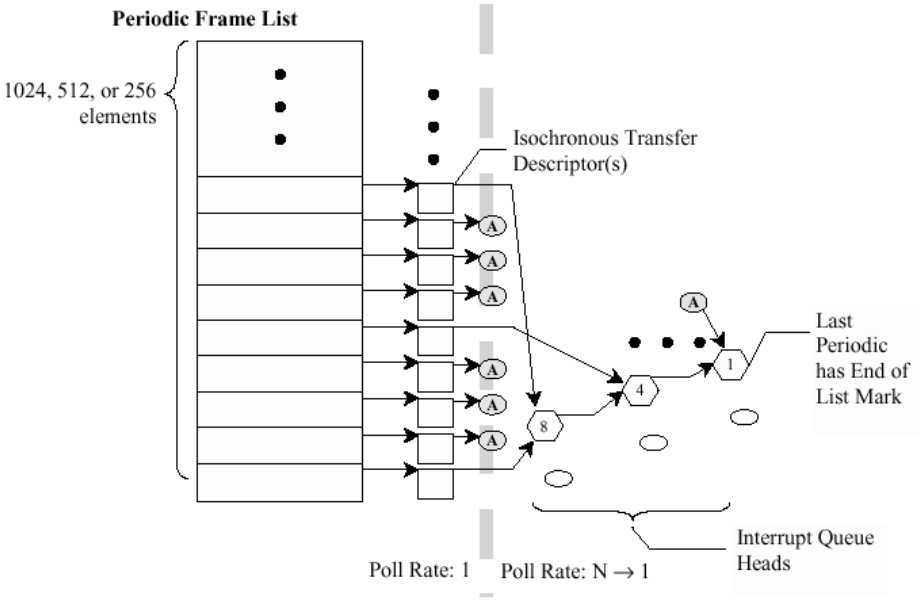


Figure 32-62. Example Periodic Schedule

32.11.7 Managing Isochronous Transfers Using iTDs

The structure of an iTD is presented in Isochronous (High-Speed) Transfer Descriptor (iTD). There are four distinct sections to an iTD:

- The first field is the *Next Link Pointer.* This field is for schedule linkage purposes only;
- Transaction description array. This area is an eight-element array. Each element represents control and status information for one micro-frame's worth of transactions for a single high-speed isochronous endpoint.
- The buffer page pointer array is a 7-element array of physical memory pointers to data buffers. These are 4K aligned pointers to physical memory.
- Endpoint capabilities. This area utilizes the unused low-order 12 bits of the buffer page pointer array. The fields in this area are used across all transactions executed for this iTD, including endpoint addressing, transfer direction, maximum packet size and high-bandwidth multiplier.

32.11.7.1 Host Controller Operational Model for iTDs

The host controller uses FRINDEX register bits [12:3] to index into the periodic frame list. This means that the host controller visits each frame list element eight consecutive times before incrementing to the next periodic frame list element. Each iTD contains eight transaction descriptions, which map directly to FRINDEX register bits [2:0]. Each iTD can span 8 micro-frames worth of transactions. When the host controller fetches an iTD, it uses FRINDEX register bits [2:0] to index into the transaction description array. If the *active* bit in the *Status* field of the indexed transaction description is set to zero, the host controller ignores the iTD and follows the *Next* pointer to the next schedule data structure.

When the indexed *active* bit is a one the host controller continues to parse the iTD. It stores the indexed transaction description and the general endpoint information (device address, endpoint number, maximum packet size, and so on.). It also uses the *Page Select (PG)* field to index the *buffer pointer* array, storing the selected *buffer pointer* and the next sequential *buffer pointer.* For example, if *PG* field is a 0, then the host controller will store Page 0 and Page 1.

The host controller constructs a physical data buffer address by concatenating the current buffer pointer (as selected using the current transaction description's *PG* field) and the transaction description's *Transaction Offset* field. The host controller uses the endpoint addressing information and *I/O-bit* to execute a transaction to the appropriate endpoint. When the transaction is complete, the host controller clears the active bit and writes back any additional status information to the *Status* field in the currently selected transaction description.

The data buffer associated with the iTD must be virtually contiguous memory. Seven page pointers are provided to support eight high-bandwidth transactions regardless of the starting packet’s offset alignment into the first page. A starting buffer pointer (physical memory address) is constructed by concatenating the page pointer (example: page 0 pointer) selected by the active transaction descriptions’ *PG* (example value: 00B) field with the transaction offset field. As the transaction moves data, the host controller must detect when an increment of the current buffer pointer will cross a page boundary. When this occurs the host controller simply replaces the current buffer pointer’s page portion with the next page pointer (example: page 1 pointer) and continues to move data. The size of each bus transaction is determined by the value in the *Maximum Packet Size* field. An iTD supports high-bandwidth pipes via the *Mult* (multiplier) field. When the *Mult* field is 1, 2, or 3, the host controller executes the specified number of *Maximum Packet* sized bus transactions for the endpoint in the current micro-frame. In other words, the *Mult* field represents a transaction count for the endpoint in the current micro-frame. If the *Mult* field is zero, the operation of the host controller is undefined. The transfer description is used to service all transactions indicated by the *Mult* field.

For OUT transfers, the value of the *Transaction X Length* field represents the total bytes to be sent during the micro-frame. The *Mult* field must be set by software to be consistent with *Transaction X Length* and *Maximum Packet Size.* The host controller will send the bytes in *Maximum Packet Size*’d portions. After each transaction, the host controller decrements it's local copy of *Transaction X Length* by *Maximum Packet Size*. The number of bytes the host controller sends is always *Maximum Packet Size* or *Transaction X Length,* whichever is less. The host controller advances the transfer state in the transfer description, updates the appropriate record in the iTD and moves to the next schedule data structure. The maximum sized transaction supported is 3 x 1024 bytes.

For IN transfers, the host controller issues *Mult* transactions. It is assumed that software has properly initialized the iTD to accommodate all of the possible data. During each IN transaction, the host controller must use *Maximum Packet Size* to detect packet babble errors. The host controller keeps the sum of bytes received in the *Transaction X Length* field. After all transactions for the endpoint have completed for the micro-frame, *Transaction X Length* contains the total bytes received. If the final value of *Transaction X Length* is less than the value of *Maximum Packet Size*, then less data than was allowed for was received from the associated endpoint. This *short packet* condition does not set the *USBINT* bit in the USBSTS register to a one. The host controller will not detect this condition. If the device sends more than *Transaction X Length* or *Maximum Packet Size* bytes (whichever is less), then the host controller will set the *Babble Detected* bit to a one and set the *Active* bit to a zero. Note, that the host controller is not required to update the iTD field *Transaction X Length* in this error scenario. If the *Mult* field is greater than one, then the host controller will automatically execute the value of *Mult* transactions. The host controller will not execute all *Mult* transactions if:

- The endpoint is an OUT and *Transaction X Length* goes to zero before all the *Mult* transactions have executed (ran out of data), or
- The endpoint is an IN and the endpoint delivers a short packet, or an error occurs on a transaction before *Mult* transactions have been executed. The end of micro-frame may occur before all of the transaction opportunities have been executed. When this happens, the transfer state of the transfer description is advanced to reflect the progress that was made, the result written back to the iTD and the host controller proceeds to processing the next micro-frame. Refer to Appendix D for a table summary of the host controller required behavior for all the high-bandwidth transaction cases.

32.11.7.2 Software Operational Model for iTDs

A client buffer request to an isochronous endpoint may span 1 to N micro-frames. When N is larger than one, system software may have to use multiple iTDs to read or write data with the buffer (if N is larger than eight, it must use more than one iTD).

Figure 32-63 illustrates the simple model of how a client buffer is mapped by system software to the periodic schedule (that is, the periodic frame list and a set of iTDs). On the right is the client description of its request. The description includes a buffer base address plus additional annotations to identify which portions of the buffer should be used with each bus transaction. In the middle is the iTD data structures used by the system software to service the client request. Each iTD can be initialized to service up to 24 transactions, organized into eight groups of up to three transactions each. Each group maps to one micro-frame's worth of transactions. The EHCI controller does not provide per-transaction results within a micro-frame. It treats the per-micro-frame transactions as a single logical transfer. On the left is the host controller’s frame list. System software establishes references from the appropriate locations in the frame list to each of the appropriate iTDs. If the buffer is large, then system software can use a small set of iTDs to service the entire buffer. System software can activate the transaction description records (contained in each iTD) in any pattern required for the particular data stream.

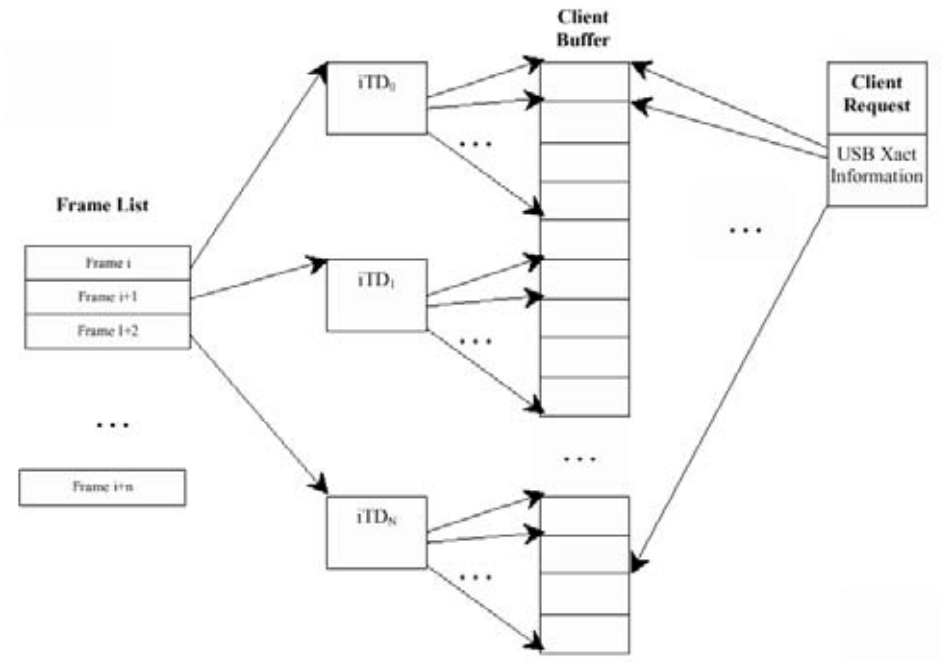


Figure 32-63. Example Association of iTDs to Client Request Buffer

As noted above, the client request includes a pointer to the base of the buffer and offsets into the buffer to annotate which buffer sections are to be used on each bus transaction that occurs on this endpoint. System software must initialize each transaction description in an iTD to ensure it uses the correct portion of the client buffer. For example, for each transaction description, the *PG* field is set to index the correct physical buffer page pointer and the *Transaction Offset* field is set relative to the correct buffer pointer page (for example, the same one referenced by the *PG* field). When the host controller executes a transaction it selects a transaction description record based on FRINDEX[2:0]. It then uses the current *Page Buffer Pointer* (as selected by the *PG* field) and concatenates to the *transaction offset* field. The result is a starting buffer address for the transaction. As the host controller moves data for the transaction, it must watch for a page wrap condition and properly advance to the next available *Page Buffer Pointer.* System software must not use the Page 6 buffer pointer in a transaction description where the length of the transfer will wrap a page boundary. Doing so will yield undefined behavior. The host controller hardware is not required to 'alias' the page selector to page zero. USB 2.0 isochronous endpoints can specify a period greater than one. Software can achieve the appropriate scheduling by linking iTDs into the appropriate frames (relative to the frame list) and by setting appropriate transaction description elements active bits to a one.

32.11.8 Periodic Scheduling Threshold

The *Isochronous Scheduling Threshold* field in the HCCPARAMS capability register is an indicator to system software as to how the host controller pre-fetches and effectively caches schedule data structures. It is used by system software when adding isochronous work items to the periodic schedule. The value of this field indicates to system software the minimum distance it can update isochronous data (relative to the current location of the host controller execution in the periodic list) and still have the host controller process them.

The iTD and siTD data structures each describe 8 micro-frames worth of transactions. The host controller is allowed to cache one (or more) of these data structures to reduce memory traffic. There are three basic caching models that account for the fact the isochronous data structures span 8 micro-frames. The three caching models are as follows: no caching, micro-frame caching and frame caching.

When software is adding new isochronous transactions to the schedule, it always performs a read of the FRINDEX register to determine the current frame and micro-frame the host controller is currently executing. Of course, there is no information about where in the micro-frame the host controller is, so a constant uncertainty-factor of one micro-frame has to be assumed. Combining the knowledge of where the host controller is executing with the knowledge of the caching model allows the definition of simple algorithms for how closely software can reliably work to the executing host controller.

No caching is indicated with a value of zero in the *Isochronous Scheduling Threshold* field. The host controller may pre-fetch data structures during a periodic schedule traversal (per micro-frame) but will always dump any accumulated schedule state at the end of the micro-frame. At the appropriate time relative to the beginning of every micro-frame, the host controller always begins schedule traversal from the frame list. Software can use the value of the FRINDEX register (plus the constant 1 uncertainty-factor) to determine the approximate position of the executing host controller. When no caching is selected, software can add an isochronous transaction as near as 2 micro-frames in front of the current executing position of the host controller.

Frame caching is indicated with a non-zero value in bit [7] of the *Isochronous Scheduling Threshold* field. In the frame-caching model, system software assumes that the host controller caches one (or more) isochronous data structures for an entire frame (8 micro-frames). Software uses the value of the FRINDEX register (plus the constant 1 uncertainty) to determine the *current* micro-frame/frame (assume modulo 8 arithmetic in adding the constant 1 to the micro-frame number). For any current frame N, if the current micro-frame is 0 to 6, then software can safely add isochronous transactions to Frame N + 1. If the current micro-frame is 7, then software can add isochronous transactions to Frame N + 2.

Micro-frame caching is indicated with a non-zero value in the least-significant 3 bits of the *Isochronous Scheduling Threshold* field. System software assumes the host controller caches one or more periodic data structures for the number of micro-frames indicated in the *Isochronous Scheduling Threshold* field. For example, if the count value were 2, then the host controller keeps a *window* of 2 micro-frames worth of state (current micro-frame, plus the next) on-chip. On each micro-frame boundary, the host controller releases the current micro-frame state and begins accumulating the next micro-frame state.

32.11.9 Asynchronous Schedule

The Asynchronous schedule traversal is enabled or disabled via the *Asynchronous Schedule Enable* bit in the USBCMD register. If the *Asynchronous Schedule Enable* bit is set to a zero, then the host controller simply does not try to access the asynchronous schedule via the *ASYNCLISTADDR* register. Likewise, when the *Asynchronous Schedule Enable* bit is a one, then the host controller does use the *ASYNCLISTADDR* register to traverse the asynchronous schedule. Modifications to the *Asynchronous Schedule Enable* bit are not necessarily immediate. Rather the new value of the bit will only be taken into consideration the next time the host controller needs to use the value of the *ASYNCLISTADDR* register to get the next queue head.

The *Asynchronous Schedule Status* bit in the USBSTS register indicates status of the asynchronous schedule. System software enables (or disables) the asynchronous schedule by writing a one (or zero) to the *Asynchronous Schedule Enable* bit in the USBCMD register. Software then can poll the *Asynchronous Schedule Status* bit to determine when the asynchronous schedule has made the desired transition. Software must not modify the *Asynchronous Schedule Enable* bit unless the value of the *Asynchronous Schedule Enable* bit equals that of the *Asynchronous Schedule Status* bit.

The asynchronous schedule is used to manage all Control and Bulk transfers. Control and Bulk transfers are managed using queue head data structures. The asynchronous schedule is based at the *ASYNCLISTADDR* register. The default value of the *ASYNCLISTADDR* register after reset is undefined and the schedule is disabled when the *Asynchronous Schedule Enable* bit is a zero.

Software may only write this register with defined results when the schedule is disabled (for example, *Asynchronous Schedule Enable* bit in the USBCMD and the *Asynchronous Schedule Status* bit in the USBSTS register are zero). System software enables execution from the asynchronous schedule by writing a valid memory address (of a queue head) into this register. Then software enables the asynchronous schedule by setting the *Asynchronous Schedule Enable* bit is set to one. The asynchronous schedule is actually enabled when the *Asynchronous Schedule Status* bit is a one.

When the host controller begins servicing the asynchronous schedule, it begins by using the value of the *ASYNCLISTADDR* register. It reads the first referenced data structure and begins executing transactions and traversing the linked list as appropriate. When the host controller “completes” processing the asynchronous schedule, it retains the value of the last accessed queue head's horizontal pointer in the *ASYNCLISTADDR* register. Next time the asynchronous schedule is accessed, this is the first data structure that will be serviced. This provides round-robin fairness for processing the asynchronous schedule.

A host controller “completes” processing the asynchronous schedule when one of the following events occur:

- The end of a micro-frame occurs.
- The host controller detects an empty list condition (that is, see Section 32.11.10.2, “Empty Asynchronous Schedule Detection”).
- The schedule has been disabled via the *Asynchronous Schedule Enable* bit in the USBCMD register.

The queue heads in the asynchronous list are linked into a simple circular list as shown in Figure 32-58. Queue head data structures are the only valid data structures that may be linked into the asynchronous schedule. An isochronous transfer descriptor (iTD or siTD) in the asynchronous schedule yields undefined results.

The maximum packet size field in a queue head is sized to accommodate the use of this data structure for all non-isochronous transfer types. The USB Specification, Revision 2.0 specifies the maximum packet sizes for all transfer types and transfer speeds. System software should always parameterize the queue head data structures according to the core specification requirements.

This is a software requirement section. There are two independent events for adding queue heads to the asynchronous schedule. The first is the initial activation of the asynchronous list. The second is inserting a new queue head into an activated asynchronous list.

32.11.10 Adding Queue Heads to Asynchronous Schedule

Activation of the list is simple. System software writes the physical memory address of a queue head into the *ASYNCLISTADDR* register, then enables the list by setting the *Asynchronous Schedule Enable* bit in the USBCMD register to a one.

When inserting a queue head into an active list, software must ensure that the schedule is always coherent from the host controllers' point of view. This means that the system software must ensure that all queue head pointer fields are valid. For example, qTD pointers have *T-Bit*s set to a one or reference valid qTDs and the *Horizontal Pointer* references a valid queue head data structure. The following algorithm represents the functional requirements:

```
InsertQueueHead (pQHeadCurrent, pQueueHeadNew)
        --
        -- Requirement: all inputs must be properly initialized.
        --
        -- pQHeadCurrent is a pointer to a queue head that is
        -- already in the active list
        -- pQHeadNew is a pointer to the queue head to be added
        --
        -- This algorithm links a new queue head into a existing
        -- list
        --
        pQueueHeadNew.HorizontalPointer = pQueueHeadCurrent.HorizontalPointer

        pQueueHeadCurrent.HorizontalPointer = physicalAddressOf(pQueueHeadNew)

End InsertQueueHead
```

32.11.10.1 Removing Queue Heads from Asynchronous Schedule

This is a software requirement section. There are two independent events for removing queue heads from the asynchronous schedule. The first is shutting down (deactivating) the asynchronous list. The second is extracting a single queue head from an activated list. Software deactivates the asynchronous schedule by setting the *Asynchronous Schedule Enable* bit in the USBCMD register to a zero. Software can determine when the list is idle when the *Asynchronous Schedule Status* bit in the USBSTS register is a zero. The normal mode of operation is that software removes queue heads from the asynchronous schedule without shutting it down. Software must not remove an active queue head from the schedule. Software should first deactivate all active qTDs, wait for the queue head to go inactive, then remove the queue head from the asynchronous list. Software removes a queue head from the asynchronous list via the following algorithm. As illustrated, the unlinking is quite easy. Software merely must ensure all of the link pointers reachable by the host controller are kept consistent.

```
UnlinkQueueHead (pQHeadPrevious, pQueueHeadToUnlink, pQHeadNext)
        --
        -- Requirement: all inputs must be properly initialized.
        --
        -- pQHeadPrevious is a pointer to a queue head that
        -- references the queue head to remove
        -- pQHeadToUnlink is a pointer to the queue head to be
        -- removed
        -- pQheadNext is a pointer to a queue head still in the
```

```
        -- schedule. Software provides this pointer with the
        -- following strict rules:
        --      if the host software is one queue head, then
        --      pQHeadNext must be the same as
        --      QueueheadToUnlink.HorizontalPointer. If the host
        --      software is unlinking a consecutive series of
        --      queue heads, QHeadNext must be set by software to
        --      the queue head remaining in the schedule.
        --
        -- This algorithm unlinks a queue head from a circular list
        --
        pQueueHeadPrevious.HorizontalPointer = pQueueHeadToUnlink.HorizontalPointer
pQueueHeadToUnlink.HorizontalPointer = pQHeadNext
End UnlinkQueueHead
```

If software removes the queue head with the *H-bit* set to a one, it must select another queue head still linked into the schedule and set its *H-bit* to a one. This should be completed before removing the queue head. The requirement is that software keep one queue head in the asynchronous schedule, with its *H-bit* set to a one. At the point software has removed one or more queue heads from the asynchronous schedule, it is unknown whether the host controller has a cached pointer to them. Similarly, it is unknown how long the host controller might retain the cached information, as it is implementation dependent and may be affected by the actual dynamics of the schedule load. Therefore, once software has removed a queue head from the asynchronous list, it must retain the coherency of the queue head (link pointers, and so on.). It cannot disturb the removed queue heads until it knows that the host controller does not have a local copy of a pointer to any of the removed data structures.

The method software uses to determine when it is safe to modify a removed queue head is to handshake with the host controller. The handshake mechanism allows software to remove items from the asynchronous schedule, then execute a simple, lightweight handshake that is used by software as a key that it can free (or reuse) the memory associated the data structures it has removed from the asynchronous schedule.

The handshake is implemented with three bits in the host controller. The first bit is a command bit (*Interrupt on Async Advance Doorbell* bit in the USBCMD register) that allows software to inform the host controller that something has been removed from its asynchronous schedule. The second bit is a status bit (*Interrupt on Async Advance* bit in the USBSTS register) that the host controller sets after it has released all on-chip state that may potentially reference one of the data structures just removed. When the host controller sets this status bit to a one, it also sets the command bit to a zero. The third bit is an interrupt enable (*Interrupt on Async Advance* bit in the USBINTR register) that is matched with the status bit. If the status bit is a one and the interrupt enable bit is a one, then the host controller will assert a hardware interrupt.

Figure 32-64 illustrates a general example. In this example, consecutive queue heads (B and C) are unlinked from the schedule using the algorithm above. Before the unlink operation, the host controller has a copy of queue head A.

The unlink algorithm requires that as software unlinks each queue head, the unlinked queue head is loaded with the address of a queue head that will remain in the asynchronous schedule.

When the host controller observes that doorbell bit being set to a one, it makes a note of the local reachable schedule information. In this example, the local reachable schedule information includes both queue heads (A and B). It is sufficient that the host controller can set the status bit (and clear the doorbell bit) as soon as it has traversed beyond current reachable schedule information (that is, traversed beyond queue head (B) in this example).

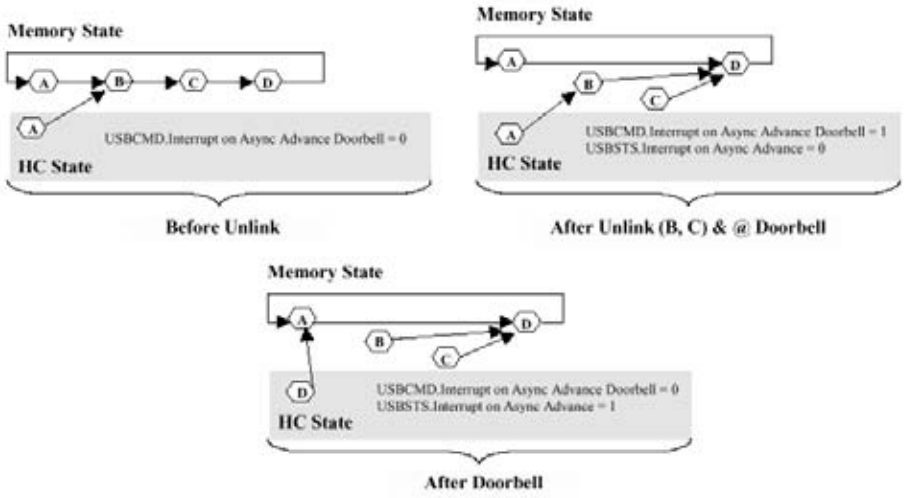


Figure 32-64. Generic Queue Head Unlink Scenario

Alternatively, a host controller implementation is allowed to traverse the entire asynchronous schedule list (for example, observed the head of the queue [twice]) before setting the *Advance on Async* status bit to a one.

Software may re-use the memory associated with the removed queue heads after it observes the *Interrupt on Async Advance* status bit is set to a one, following assertion of the doorbell. Software should acknowledge the *Interrupt on Async Advance* status as indicated in the USBSTS register, before using the doorbell handshake again.

32.11.10.2 Empty Asynchronous Schedule Detection

The Enhanced Host Controller Interface uses two bits to detect when the asynchronous schedule is empty. The queue head data structure (see <Blue Cross-Reference>Figure 32-52.) defines an *H-bit* in the queue head, which allows software to mark a queue head as being the *head* of the reclaim list. The Enhanced Host Controller Interface also keeps a 1-bit flag in the USBSTS register (*Reclamation*) that is set to a zero when the Enhanced Interface Host Controller observes a queue head with the H-bit set to a one. The reclamation flag in the status register is set to one when any USB transaction from the asynchronous schedule is executed (or whenever the asynchronous schedule starts, see Section 32.11.10.4, “Asynchronous Schedule Traversal: Start Event”).

If the Enhanced Host Controller Interface ever encounters an *H-bit* of one and a *Reclamation* bit of zero, the EHCI controller simply stops traversal of the asynchronous schedule.

An example illustrating the H-bit in a schedule is illustrated in Figure 32-65.

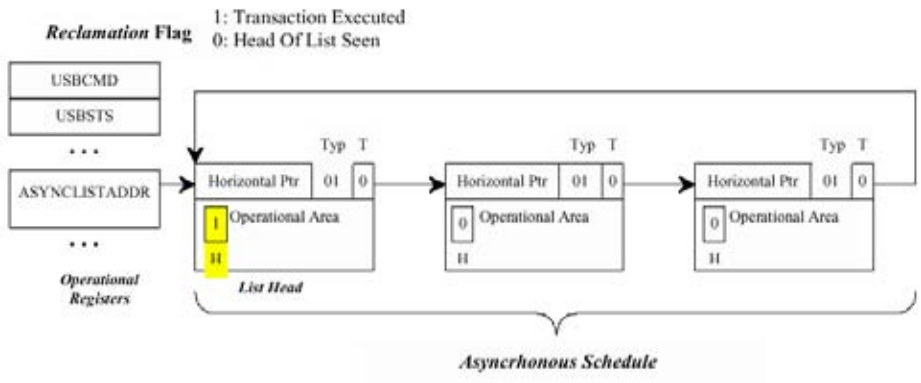


Figure 32-65. Asynchronous Schedule List w/Annotation to Mark Head of List

Software must ensure there is at most one queue head with the *H-bit* set to a one, and that it is always coherent with respect to the schedule.

32.11.10.3 Restarting Asynchronous Schedule Before EOF

There are many situations where the host controller will detect an empty list *long* before the end of the micro-frame. It is important to remember that under many circumstances the schedule traversal has stopped due to NAK/NYET responses from all endpoints.

An example of particular interest is when a start-split for a bulk endpoint occurs early in the micro-frame. Given the EHCI simple traversal rules, the complete-split for that transaction may NAK/NYET out very quickly. If it is the only item in the schedule, then the host controller will cease traversal of the Asynchronous schedule very early in the micro-frame. To provide reasonable service to this endpoint, the host controller should issue the complete-split before the end of the current micro-frame, instead of waiting until the next micro-frame. When the reason for host controller idling asynchronous schedule traversal is because of empty list detection, it is mandatory the host controller implement a 'waking' method to resume traversal of the asynchronous schedule. An example method is described below.

Example Method for Restarting Asynchronous Schedule Traversal

The reason for idling the host controller when the list is empty is to keep the host controller from unnecessarily occupying too much memory bandwidth. The question is as follows: *how long should the host controller stay idle before restarting*?

The answer in this example is based on deriving a manifest constant, which is the amount of time the host controller will stay idle before restarting traversal. In this example, the manifest constant is called *AsyncSchedSleepTime*, and has a value of 10 µsec. The value is derived based on the analysis in Section Example Derivation for AsyncSchedSleepTime. The traversal algorithm is simple:

- Traverse the Asynchronous schedule until the either an End-Of-micro-Frame event occurs, or an empty list is detected. If the event is an End-of-micro-Frame, go attempt to traverse the Periodic schedule. If the event is an empty list, then set a sleep timer and go to a *schedule sleep* state.

- When the sleep timer expires, set working context to the Asynchronous Schedule start condition and go to *schedule active* state. The start context allows the HC to reload *NAKcnt* fields, and so on. so the HC has a chance to run for more than one iteration through the schedule.

This process simply repeats itself each micro-frame. Figure 32-66 illustrates a sample state machine to manage the active and sleep states of the Asynchronous Schedule traversal policy. There are three states: Actively traversing the Asynchronous schedule, Sleeping, and Not Active. The last two are similar in terms of interaction with the Asynchronous schedule, but the Not Active state means that the host controller is busy with the Periodic schedule or the Asynchronous schedule is not enabled. The Sleeping state is specifically a special state where the host controller is just waiting for a period of time before resuming execution of the Asynchronous schedule.

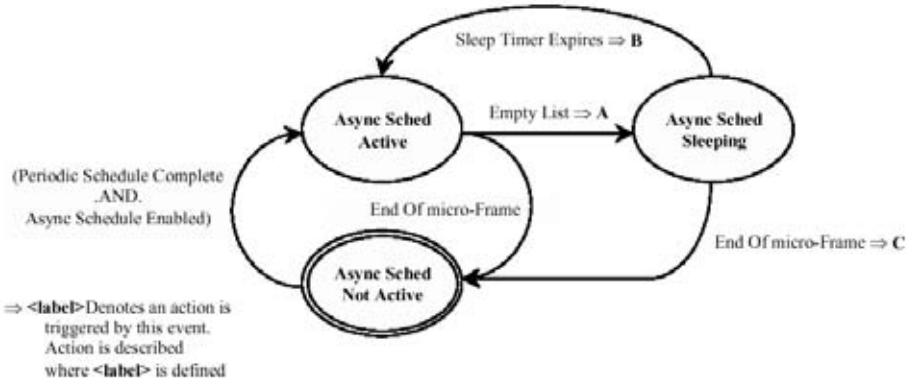


Figure 32-66. Example State Machine for Managing Asynchronous Schedule Traversal

The actions referred to in Figure 32-66 are defined in Table 32-74.

Table 32-74. Asynchronous Schedule SM Transition Actions

Action	Action Description Label
A	On detection of the empty list, the host controller sets the *AsynchronousTraversalSleepTimer* to *AsyncSchedSleepTime*.
B	When the *AsynchronousTraversalSleepTimer* expires, the host controller sets the *Reclamation* bit in the USBSTS register to a one and moves the NAK Counter reload state machine to WaitForListHead (see Section 32.11.11, "Operational Model for NAK Counter").
C	The host controller cancels the sleep timer (*AsynchronousTraversalSleepTimer*).

Async Sched Not Active

This is the initial state of the traversal state machine after a host controller reset. The traversal state machine will not leave this state when the *Asynchronous Schedule Enable* bit in the USBCMD register is a zero.

This state is entered from Async Sched Active or Async Sched Sleeping states when the end-of-micro-frame event is detected.

Async Sched Active

This state is entered from the Async Sched Not Active state when the periodic schedule is not active. It is also entered from the Async Sched Sleeping states when the *AsyncrhonousTraversalSleepTimer* expires. On every transition into this state, the host controller sets the *Reclamation* bit in the USBSTS register to a one.

While in this state, the host controller will continually traverse the asynchronous schedule until either the end of micro-frame or an empty list condition is detected.

Async Sched Sleeping

The state is entered from the Async Sched Active state when a schedule empty condition is detected. On entry to this state, the host controller sets the *AsynchronousTraversalSleepTimer* to *AsyncSchedSleepTime*.

Example Derivation for *AsyncSchedSleepTime*

The derivation is based on analysis of what work the host controller could be doing next. It assumes the host controller does not keep any state about what work is possibly pending in the asynchronous schedule. The schedule could contain any mix of the possible combinations of high- full- or low-speed control and bulk requests. Table 32-75 summarizes some of the typical 'next transactions' that could be in the schedule, and the amount of time (for example, *footprint* or *wall clock*) the transaction will take to complete.

Table 32-75. Typical Low-/Full-speed Transaction Times

Transaction Attributes		Footprint (time)	Description
Speed	HS	11.9 ms	Maximum foot print for a worst-case, full-sized bulk data transaction.
Size	512	9.45 ms	Maximum footprint for an approximate best-case, full-sized bulk data transaction.
Type	Bulk		
Speed	FS	~50 ms	Approximate typical for full-sized bulk data. An 8-byte low-speed is about 2x, or between 90 and 100 ms.
Size	64		
Type	Bulk		
Speed	FS	~12 ms	Approximate typical for 8-byte bulk/control (that is, setup)
Size	8		
Type	Cntrl		

A *AsyncSchedSleepTime* value of 10 µs provides a reasonable relaxation of the system memory load and still provides a good level of service for the various transfer types and payload sizes. For example, say we detect an empty list after issuing a start-split for a 64-byte full-speed bulk request. Assuming this is the only thing in the list, the host controller will get the results of the full-speed transaction from the hub during the fifth complete-split request. If the full-speed transaction was an IN and it NAK'd, the 10 µs sleep period would allow the host controller to get the NAK results on the first complete-split.

32.11.10.4 Asynchronous Schedule Traversal: Start Event

Once the HC has *idled* itself via the empty schedule detection (Section 0), it will naturally *activate* and begin processing from the Periodic Schedule at the beginning of each micro-frame. In addition, it may have idled itself early in a micro-frame. When this occurs (idles early in the micro-frame) the HC must occasionally *re-activate* during the micro-frame and traverse the asynchronous schedule to determine whether any progress can be made. The requirements and method for this restart are described in Section 32.11.10.3, "Restarting Asynchronous Schedule Before EOF." Asynchronous schedule *Start Event*s are defined to be:

- Whenever the host controller transitions from the periodic schedule to the asynchronous schedule. If the periodic schedule is disabled and the asynchronous schedule is enabled, then the beginning of the micro-frame is equivalent to the transition from the periodic schedule, or
- The asynchronous schedule traversal restarts from a sleeping state (see Section 32.11.10.3, "Restarting Asynchronous Schedule Before EOF").

32.11.10.5 Reclamation Status Bit (USBSTS Register)

The operation of the empty asynchronous schedule detection feature (Section 32.11.10.2, "Empty Asynchronous Schedule Detection") depends on the proper management of the *Reclamation* bit in the USBSTS register. The host controller tests for an empty schedule just after it fetches a new queue head while traversing the asynchronous schedule (See Section 32.11.12.1, "Fetch Queue Head").

It is required that the host controller sets the *Reclamation* bit to a one whenever an asynchronous schedule traversal *Start Event*, as documented in Section 32.11.10.4, "Asynchronous Schedule Traversal: Start Event," occurs. The *Reclamation* bit is also set to a one whenever the host controller executes a transaction while traversing the asynchronous schedule (see Section 32.11.12.3, "Execute Transaction"). The host controller sets the *Reclamation* bit to a zero whenever it finds a queue head with its *H-bit* set to a one. Software should only set a queue head's *H-bit* if the queue head is in the asynchronous schedule. If software sets the *H-bit* in an interrupt queue head to a one, the resulting behavior is undefined. The host controller may set the *Reclamation* bit to a zero when executing from the periodic schedule.

32.11.11 Operational Model for NAK Counter

This section describes the operational model for the *NakCnt* field defined in a queue head (see Section Queue Head). Software should not use this feature for interrupt queue heads. This rule is not required to be enforced by the host controller.

USB protocol has built-in flow control via the NAK response by a device. There are several scenarios, beyond the Ping feature, where an endpoint may naturally NAK or NYET the majority of the time. An example is the host controller management of the split transaction protocol for control and bulk endpoints. All bulk endpoints (High- or Full-speed) are serviced via the same asynchronous schedule. The time between the *Start-split* transaction and the first *Complete-split* transaction could be very short (that is, like when the endpoint is the only one in the asynchronous schedule). The hub NYETs (effectively NAKs) the *Complete-split* transaction until the classic transaction is complete. This could result in the host controller thrashing memory, repeatedly fetching the queue head and executing the transaction to the Hub, which will not complete until after the transaction on the classic bus completes.

There are two component fields in a queue head to support the throttling feature: a counter field (*NakCnt*), and a counter reload field (*RL*). *NakCnt* is used by the host controller as one of the criteria to determine whether or not to execute a transaction to the endpoint. There are two operational modes associated with this counter:

- Not Used. This mode is set when the *RL* field is zero. The host controller ignores the *NakCnt* field for any execution of transactions through a queue head with an *RL* field of zero. Software must use this selection for interrupt endpoints.
- NAK Throttle Mode. This mode is selected when the *RL* field is non-zero. In this mode, the value in the *NakCnt* field represents the maximum number of NAK or NYET responses the host controller will tolerate on each endpoint. In this mode, the HC will decrement the *NakCnt* field based on the token/handshake criteria listed in Table 32-76. The host controller must reload *NakCnt* when the endpoint successfully moves data (for example, policy to reward device for moving data).

Table 32-76. NakCnt Field Adjustment Rules

Token	Handshake	
	Handshake NAK	NYET
IN/PING	decrement *NakCnt*	N/A (protocol error)
OUT	decrement *NakCnt*	No Action[1]Start
Split	decrement *NakCnt*	N/A (protocol error)
Complete Split	No Action	Decrement *NakCnt*

[1] Recommended behavior on this response is to reload *NakCnt*

In summary, system software enables the counter by setting the reload field (*RL*) to a non-zero value. The host controller may execute a transaction if *NakCnt* is non-zero. The host controller will not execute a transaction if *NakCnt* is zero. The reload mechanism is described in detail in Section NAK Count Reload Control .

Note that when all queue heads in the Asynchronous Schedule either exhausts all transfers or all NakCnt's go to zero, then the host controller will detect an empty Asynchronous Schedule and idle schedule traversal (see Section 32.11.10.2, “Empty Asynchronous Schedule Detection”).

Any time the host controller begins a new traversal of the Asynchronous Schedule, a *Start Event* is assumed, see Section 32.11.10.4, “Asynchronous Schedule Traversal: Start Event.” Every time a Start-Event occurs, the NAK Count reload procedure is enabled.

32.11.11.1 NAK Count Reload Control

When the host controller reaches the *Execute Transaction* state for a queue head (meaning that it has an active operational state), it checks to determine whether the *NakCnt* field should be reloaded from *RL* (see Section 32.11.12.3, “Execute Transaction”). If the answer is yes, then *RL* is copied into *NakCnt*. After the reload or if the reload is not active, the host controller evaluates whether to execute the transaction.

The host controller must reload NAK counters (*NakCnt* see Figure 32-52) in queue heads during the first pass through the reclamation list after an asynchronous schedule Start Event (see Section 32.11.10.4, "Asynchronous Schedule Traversal: Start Event" for the definition of the Start Event). The Asynchronous Schedule should have at most one queue head marked as the head (see Figure 32-66). Figure 32-73 illustrates an example state machine that satisfies the operational requirements of the host controller detecting the first pass through the Asynchronous Schedule. This state machine is maintained internal to the host controller and is only used to gate reloading of the NAK counter during the queue head traversal state: Execute Transaction (Figure 32-68). The host controller does not perform the NAK counter reload operation if the RL field (see Figure 32-52) is set to zero.

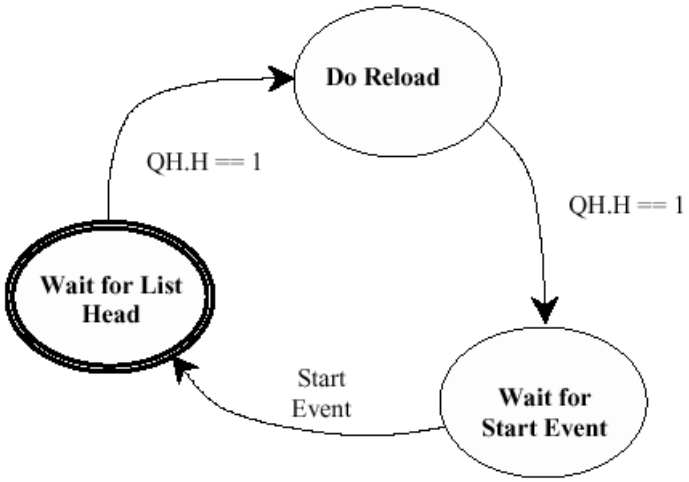


Figure 32-67. Example HC State Machine for Controlling NAK Counter Reloads

Wait for List Head

This is the initial state. The state machine enters this state from Wait for Start Event when a start event as defined in Section 32.11.10.4, "Asynchronous Schedule Traversal: Start Event" occurs. The purpose of this state is to wait for the first observation of the head of the Asynchronous Schedule. This occurs when the host controller fetches a queue head whose *H-bit* is set to a one.

Do Reload

This state is entered from the Wait for List Head state when the host controller fetches a queue head with the *H-bit* set to a one. While in this state, the host controller will perform NAK counter reloads for every queue head visited that has a non-zero NAK reload value (*RL*) field.

Wait for Start Event

This state is entered from the *Do Reload* state when a queue head with the *H-bit* set to a one is fetched. While in this state, the host controller will not perform NAK counter reloads.

32.11.12 Managing Control/Bulk/Interrupt Transfers via Queue Heads

This section presents an overview of how the host controller interacts with queuing data structures.

Queue heads use the Queue Element Transfer Descriptor (qTD) structure defined in Section Queue Element Transfer Descriptor (qTD). One queue head is used to manage the data stream for one endpoint. The queue head structure contains static endpoint characteristics and capabilities. It also contains a working area from where individual bus transactions for an endpoint are executed (see Overlay area defined in Figure 32-52). Each qTD represents one or more bus transactions, which is defined in the context of this specification as a *transfer.*

The general processing model for the host controller's use of a queue head is simple:

- read a queue head
- execute a transaction from the overlay area,
- write back the results of the transaction to the overlay area
- move to the next queue head

If the host controller encounters errors during a transaction, the host controller will set one (or more) of the error reporting bits in the queue head's *Status* field. The *Status* field accumulates all errors encountered during the execution of a qTD (for example, the error bits in the queue head *Status* field are 'sticky' until the transfer (qTD) has completed). This state is always written back to the source qTD when the transfer is complete. On transfer (for example, buffer or halt conditions) boundaries, the host controller must auto-advance (without software intervention) to the next qTD. Additionally, the hardware must be able to halt the queue so no additional bus transactions will occur for the endpoint and the host controller will not advance the queue.

An example host controller operational state machine of a queue head traversal is illustrated in Figure 32-68. This state machine is a model for how a host controller should traverse a queue head. The host controller must be able to advance the queue from the *Fetch QH* state to avoid all hardware/software race conditions. This simple mechanism allows software to simply link qTDs to the queue head and *activate* them, then the host controller will always *find* them if/when they are reachable.

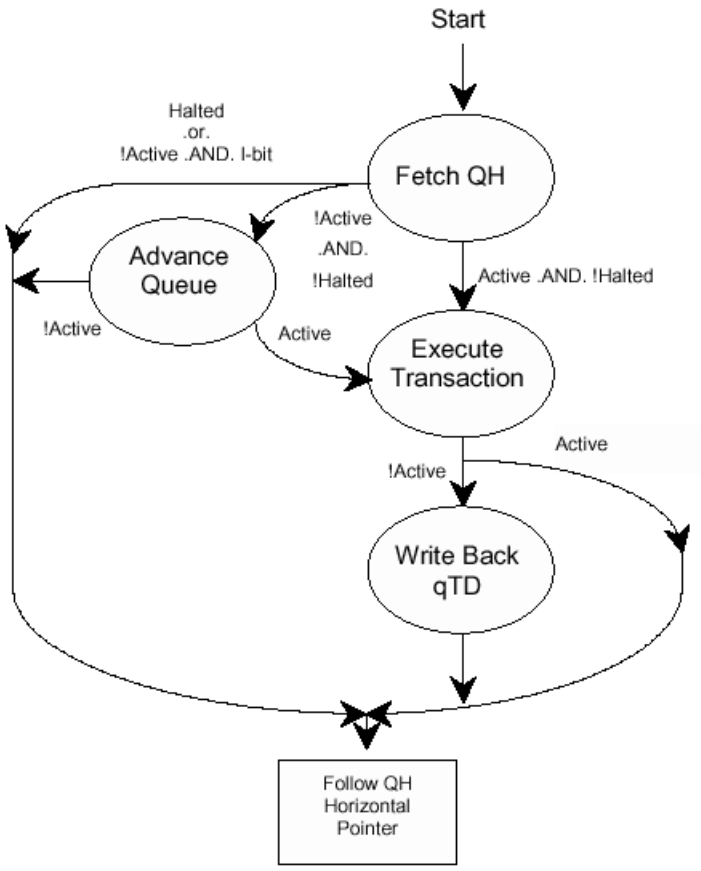


Figure 32-68. Host Controller Queue Head Traversal State Machine

This traversal state machine applies to all queue heads, regardless of transfer type or whether split transactions are required. The following sections describe each state. Each state description describes the entry criteria. The Execute Transaction state (Section 32.11.12.3, “Execute Transaction”) describes the basic requirements for all endpoints. Sections Split Transactions for Asynchronous Transfers and Split Transaction Interrupt describe details of the required extensions to the Execute Transaction state for endpoints requiring split transactions.

Note: Prior to software placing a queue head into either the periodic or asynchronous list, software must ensure the queue head is properly initialized. Minimally, the queue head should be initialized to the following (see Section Queue Head for layout of a queue head):

- Valid static endpoint state
- For the very first use of a queue head, software may zero-out the queue head transfer overlay, then set the *Next qTD Pointer* field value to reference a valid qTD.

32.11.12.1 Fetch Queue Head

A queue head can be referenced from the physical address stored in the ASYNCLISTADDR Register (Section ASYNCLISTADDR; ENDPOINTLISTADDR). Additionally, it may be referenced from the *Next*

Link Pointer field of an iTD, siTD, FSTN or another Queue Head. If the referencing link pointer has the *Typ* field set to indicate a queue head, it is assumed to reference a queue head structure as defined in Figure 32-68.

While in this state, the host controller performs operations to implement empty schedule detection (Section 32.11.10.2, "Empty Asynchronous Schedule Detection") and NAK Counter reloads (Section 32.11.11, "Operational Model for NAK Counter"). After the queue head has been fetched, the host controller conducts the following queries for empty schedule detection:

- If queue head is not an interrupt queue head (that is, *S-mask* is a zero), and
- The *H-bit* is a one, and
- The *Reclamation* bit in the USBSTS register is a zero.

When these criteria are met, the host controller will stop traversing the asynchronous list (as described in Section 32.11.10.2, "Empty Asynchronous Schedule Detection"). When the criteria are not met, the host controller continues schedule traversal. If the queue head is not an interrupt and the *H-bit* is a one and the *Reclamation* bit is a one, then the host controller sets the *Reclamation* bit in the USBSTS register to a zero before completing this state. The operations for reloading of the NAK Counter are described in detail in Section 32.11.11, "Operational Model for NAK Counter."

This state is complete when the queue head has been read on-chip.

32.11.12.2 Advance Queue

To advance the queue, the host controller must find the next qTD, adjust pointers, perform the overlay and write back the results to the queue head.

This state is entered from the FetchQHD state if the overlay *Active* and *Halt* bits are set to zero. On entry to this state, the host controller determines which next pointer to use to fetch a qTD, fetches a qTD and determines whether or not to perform an overlay. Note that if the *I-bit* is a one and the *Active* bit is a zero, the host controller immediately skips processing of this queue head, exits this state and uses the horizontal pointer to the next schedule data structure. If the field *Bytes to Transfer* is not zero and the *T-bit* in the *Alternate Next qTD Pointer* is set to zero, then the host controller uses the *Alternate Next qTD Pointer.* Otherwise, the host controller uses the *Next qTD Pointer.* If *Next qTD Pointer*'s *T-bit* is set to a one, then the host controller exits this state and uses the horizontal pointer to the next schedule data structure.

Using the selected pointer the host controller fetches the referenced qTD. If the fetched qTD has its *Active* bit set to a one, the host controller moves the pointer value used to reach the qTD (*Next* or *Alternate Next*) to the *Current qTD Pointer* field, then performs the overlay. If the fetched qTD has its *Active* bit set to a zero, the host controller aborts the queue advance and follows the queue head's horizontal pointer to the next schedule data structure. The host controller performs the overlay based on the following rules:

- The value of the data toggle (*dt*) field in the overlay area depends on the value of the *data toggle control* (*dtc*) bit (see Table 32-91).
- If the *EPS* field indicates the endpoint is a high-speed endpoint, the *Ping* state field is preserved by the host controller. The value of this field is not changed as a result of the overlay.
- *C-prog-mask* field is set to zero (field from incoming qTD is ignored, as is the current contents of the overlay area).

- *Frame Tag* field is set to zero (field from incoming qTD is ignored, as is the current contents of the overlay area).
- *NakCnt* field in the overlay area is loaded from the *RL* field in the queue head's Static Endpoint State.
- All other areas of the overlay are set by the incoming qTD.

The host controller exits this state when it has committed the write to the queue head.

32.11.12.3 Execute Transaction

The host controller enters this state from the Fetch Queue Head state only if the *Active* bit in *Status* field of the queue head is set to a one.

On entry to this state, the host controller executes a few pre-operations, then checks some pre-condition criteria before committing to executing a transaction for the queue head.

The pre-operations performed and the pre-condition criteria depend on whether the queue head is an interrupt endpoint. The host controller can determine that a queue head is an interrupt queue head when the queue head's *S-mask* field contains a non-zero value. It is the responsibility of software to ensure the *S-mask* field is appropriately initialized based on the transfer type. There are other criteria that must be met if the *EPS* field indicates that the endpoint is a low- or full-speed endpoint, see Sections Split Transactions for Asynchronous Transfers and Split Transaction Interrupt.

32.11.12.3.1 Interrupt Transfer Pre-condition Criteria

If the queue head is for an interrupt endpoint (for example, non-zero *S-mask* field), then the FRINDEX[2:0] field must identify a bit in the *S-mask* field that has a one in it. For example, an *S-mask* value of 00100000b would evaluate to true only when FRINDEX[2:0] is equal to 101b. If this condition is met then the host controller considers this queue head for a transaction**.**

32.11.12.3.2 Asynchronous Transfer Pre-Operations and Pre-Condition Criteria

If the queue head is not for an interrupt endpoint (for example, a zero *S-mask* field), then the host controller performs one pre-operation and then evaluates one pre-condition criteria: The pre-operation is as follows:

Checks the NAK counter reload state (Section 32.11.11, "Operational Model for NAK Counter"). It may be necessary for the host controller to reload the NAK Counter field. The reload is performed at this time.

The pre-condition evaluated is as follows:

- Whether or not the *NakCnt* field has been reloaded, the host controller checks the value of the *NakCnt* field in the queue head. If *NakCnt* is non-zero, or if the *Reload Nak Counter* field is zero, then the host controller considers this queue head for a transaction.

32.11.12.3.3 Transfer Type Independent Pre-Operations

Regardless of the transfer type, the host controller always performs at least one pre-operation and evaluates one pre-condition. The pre-operation is as follows:

- A host controller internal transaction (down) counter *qHTransactionCounter* is loaded from the queue head's *Mult* field. A host controller implementation is allowed to ignore this for queue heads on the asynchronous list. It is mandatory for interrupt queue heads. Software should ensure that the *Mult* field is set appropriately for the transfer type.

The pre-conditions evaluated are as follows:

- The host controller determines whether there is enough time in the micro-frame to complete this transaction (see Transaction Fit—A Best-Fit Approximation Algorithm for an example evaluation method). If there is not enough time to complete the transaction, the host controller exits this state.
- If the value of *qHTransactionCounter* for an interrupt endpoint is zero, then the host controller exits this state.

When the pre-operations are complete and pre-conditions are met, the host controller sets the *Reclamation* bit in the USBSTS register to a one and then begins executing one or more transactions using the endpoint information in the queue head. The host controller iterates *qHTransactionCounter* times in this state executing transactions. After each transaction is executed, *qHTransactionCounter* is decremented by one. The host controller will exit this state when one of the following events occurs:

- The *qHTransactionCounter* decrements to zero, or
- The endpoint responds to the transaction with any handshake other than an ACK,[4] or
- The transaction experiences a transaction error, or
- The *Active* bit in the queue head goes to a zero, or
- There is not enough time in the micro-frame left to execute the next transaction (see Transaction Fit—A Best-Fit Approximation Algorithm for an example method for implementing the frame boundary test).

[4] Note that for a high-bandwidth interrupt OUT endpoint, the host controller may optionally immediately retry the transaction if it fails.

The results of each transaction is recorded in the on-chip overlay area. If data was successfully moved during the transaction, the transfer state in the overlay area is advanced. To advance queue head's transfer state, the *Total Bytes to Transfer* field is decremented by the number of bytes moved in the transaction, the data toggle bit (*dt*) is toggled, the current page offset is advanced to the next appropriate value (for example, advanced by the number of bytes successfully moved), and the *C_Page* field is updated to the appropriate value (if necessary). See Section Buffer Pointer List Use for Data Streaming with qTDs .

Note that the *Total Bytes To Transfer* field may be zero when all the other criteria for executing a transaction are met. When this occurs, the host controller will execute a zero-length transaction to the endpoint. If the *PID_Code* field indicates an IN transaction and the device delivers data, the host controller will detect a packet babble condition, set the *babble* and *halted* bits in the *Status* field, set the *Active* bit to a zero, write back the results to the source qTD, then exit this state.

In the event an IN token receives a data PID mismatch response, the host controller must ignore the received data (for example, not advance the transfer state for the bytes received). Additionally, if the

endpoint is an interrupt IN, then the host controller must record that the transaction occurred (for example, decrement *qHTransactionCounter*). It is recommended (but not required) the host controller continue executing transactions for this endpoint if the resultant value of *qHTransactionCounter* is greater than one.

If the response to the IN bus transaction is a NAK (or NYET) and *RL* is non-zero, *NakCnt* is decremented by one. If *RL* is zero, then no write-back by the host controller is required (for a transaction receiving a NAK or NYET response and the value of *CErr* did not change). Software should set the *RL* field to zero if the queue head is an interrupt endpoint. Host controller hardware is not required to enforce this rule or operation.

After the transaction has finished and the host controller has completed the post processing of the results (advancing the transfer state and possibly *NakCnt*, the host controller writes back the results of the transaction to the queue head's overlay area in main memory.

The number of bytes moved during an IN transaction depends on how much data the device endpoint delivers. The maximum number of bytes a device can send is *Maximum Packet Size*. The number of bytes moved during an OUT transaction is either *Maximum Packet Length* bytes or *Total Bytes to Transfer*, whichever is less.

If there was a transaction error during the transaction, the transfer state (as defined above) is not advanced by the host controller. The *CErr* field is decremented by one and the status field is updated to reflect the type of error observed. Transaction errors are summarized in Section Transaction Error .

The following events will cause the host controller to clear the *Active* bit in the queue head's overlay status field. When the *Active* bit transitions from a one to a zero, the transfer in the overlay is considered complete. The reason for the transfer completion (clearing the *Active* bit) determines the next state.

- *CErr* field decrements to zero. When this occurs the *Halted* bit is set to a one and *Active* is set to a zero. This results in the hardware not advancing the queue and the pipe halts. Software must intercede to recover.
- The device responds to the transaction with a STALL PID. When this occurs, the *Halted* bit is set to a one and the *Active* bit is set to a zero. This results in the hardware not advancing the queue and the pipe halts. Software must intercede to recover.
- The *Total Bytes to Transfer* field is zero after the transaction completes. Note that for a zero length transaction, it was zero before the transaction was started. When this condition occurs, the *Active* bit is set to zero.
- The PID code is an IN, and the number of bytes moved during the transaction is less than the *Maximum Packet Length*. When this occurs, the *Active* bit is set to zero and a short packet condition exists. The short-packet condition is detected during the Advance Queue state. Refer to Section Split Transactions for additional rules for managing low- and full-speed transactions.

With the exception of a NAK response (when *RL* field is zero), the host controller always writes the results of the transaction back to the overlay area in main memory. This includes when the transfer completes. For a high-speed endpoint, the queue head information written back includes minimally the following fields: The *PID Code* field indicates an IN and the device sends more than the expected number of bytes (for example, *Maximum Packet Length* or *Total Bytes to Transfer* bytes, whichever is less) (for example, a packet babble). This results in the host controller setting the *Halted* bit to a one.

- NakCnt, dt, Total Bytes to Transfer, C_Page, Status, CERR, and Current Offset

For a low- or full-speed device the queue head information written back also includes the fields:

- C-prog-mask, FrameTag and S-bytes.

The duration of this state depends on the time it takes to complete the transaction(s) and the status write to the overlay is committed.

32.11.12.3.4 Halting a Queue Head

A halted endpoint is defined only for the transfer types that are managed via queue heads (control, bulk and interrupt). The following events indicate that the endpoint has reached a condition where no more activity can occur without intervention from the driver:

- An endpoint may return a STALL handshake during a transaction
- A transaction had three consecutive error conditions
- A Packet Babble error occurs on the endpoint

When any of these events occur (for a queue head) the Host Controller halts the queue head and set the USBERRINT status bit in the USBSTS register to a one. To halt the queue head, the *Active* bit is set to a zero and the *Halted* bit is set to a one. There may be other error status bits that are set when a queue is halted. The host controller always writes back the overlay area to the source qTD when the transfer is complete, regardless of the reason (normal completion, short packet or halt). The host controller will not advance the transfer state on a transaction that results in a *Halt* condition (for example, no updates necessary for *Total Bytes to Transfer, C_Page, Current Offset,* and *dt*). The host controller must update *CErr* as appropriate. When a queue head is halted, the *USB Error Interrupt* bit in the USBSTS register is set to a one. If the *USB Error Interrupt Enable* bit in the USBINTR register is set to a one, a hardware interrupt is generated at the next interrupt threshold.

Asynchronous Schedule Park Mode

Asynchronous Schedule Park mode is a special execution mode that can be enabled by system software, where the host controller is permitted to execute more than one bus transaction from a high-speed queue head in the Asynchronous schedule before continuing horizontal traversal of the Asynchronous schedule. This feature has no effect on queue heads or other data structures in the Periodic schedule. This feature is similar in intent as the *Mult* feature that is used in the Periodic schedule. Where-as the *Mult* feature is a characteristic that is tunable for each endpoint; park-mode is a policy that is applied to all high-speed queue heads in the asynchronous schedule. It is essentially the specification of an iterator for consecutive bus transactions to the same endpoint. All of the rules for managing bus transactions and the results of those as defined in Section Execute Transaction apply. This feature merely specifies how many consecutive times the host controller is permitted to execute from the same queue head before moving to the next queue head in the Asynchronous List. This feature should allow the host controller to attain better bus utilization for those devices that are capable of moving data at maximum rate, while at the same time providing a fair service to all endpoints.

A host controller exports its capability to support this feature to system software by setting the *Asynchronous Schedule Park Capability* bit in the HCCPARAMs register to a one. This information keys system software that the *Asynchronous Schedule Park Mode Enable* and *Asynchronous Schedule Park Mode Count* fields in the USBCMD register are modifiable. System software enables the feature by writing a one to the *Asynchronous Schedule Park Mode Enable* bit.

When park-mode is not enabled (for example, *Asynchronous Schedule Park Mode Enable* bit in the USBCMD register is a zero), the host controller must not execute more than one bus transaction per high-speed queue head, per traversal of the asynchronous schedule. When park-mode is enabled, the host controller must not apply the feature to a queue head whose *EPS* field indicates a Low/Full-speed device (that is, only one bus transaction is allowed from each Low/Full-speed queue head per traversal of the asynchronous schedule). Park-mode may only be applied to queue heads in the Asynchronous schedule whose *EPS* field indicates that it is a high-speed device.

The host controller must apply park mode to queue heads whose *EPS* field indicates a high-speed endpoint. The maximum number of consecutive bus transactions a host controller may execute on a high-speed queue head is determined by the value in the *Asynchronous Schedule Park Mode Count* field in the USBCMD register. Software must not set *Asynchronous Schedule Park Mode Enable* bit to a one and also set *Asynchronous Schedule Park Mode Count* field to a zero. The resulting behavior is not defined. An example behavioral example describes the operational requirements for the host controller implementing park-mode. This feature does not affect how the host controller handles the bus transaction as defined in Section Execute Transaction . It only effects how many consecutive bus transactions for the current queue head can be executed. All boundary conditions, error detection and reporting applies as usual. This feature is similar in concept to the use of the *Mult* field for high-bandwidth Interrupt for queue heads in the Periodic Schedule.

The host controller effectively loads an internal down-counter *PM-Count* from *Asynchronous Schedule Park Mode Count* when *Asyncrhonous Schedule Park Mode Enable* bit is a one, and a high-speed queue head is first fetched and meets all the criteria for executing a bus transaction. After the bus transaction, *PM-Count* is decremented. The host controller may continue to execute bus transactions from the current queue head until *PM-Count* goes to zero, an error is detected, the buffer for the current transfer is exhausted or the endpoint responds with a flow-control or STALL handshake. Table 34 Actions for Park Mode, based on Endpoint Response and Residual Transfer State34 summarizes the responses that effect whether the host controller continues with another bus transaction for the current queue head.

Table 32-77. Actions for Park Mode, Based on Endpoint Response and Residual Transfer State

PID	Endpoint Response	Transfer State after Transaction		Action
		PM-Count	Bytes to Transfer	
IN	DATA[0,1] w/Maximum Packet sized data	Not zero	Not Zero	Allowed to perform another bus transaction.[1,2]
		Not zero	Zero	Retire qTD and move to next QH
		Zero	Don't care	Move to next QH.
	DATA[0,1] w/short packet	Don't care	Don't care	Retire qTD and move to next QH.
	NAK	Don't care	Don't care	Move to next QH.
	STALL, XactErr	Don't care	Don't care	Move to next QH.

Table 32-77. Actions for Park Mode, Based on Endpoint Response and Residual Transfer State (continued)

PID	Endpoint Response	Transfer State after Transaction		Action
		PM-Count	Bytes to Transfer	
OUT	ACK	Not zero	Not Zero	Allowed to perform another bus transaction. [2]
		Not zero	Zero	Retire qTD and move to next QH
		Zero	Don't' care	Move to next QH.
	NYET, NAK	Don't care	Don't care	Move to next QH.
	STALL, XactErr	Don't care	Don't care	Move to next QH
PING	ACK	Not Zero	Not Zero	Allowed to perform another bus transaction. [2]
	NAK	Don't care	Don't care	Move to next QH
	STALL, XactErr	Don't care	Don't care	Move to next QH

[1] Note, the host controller may continue to execute bus transactions from the current high-speed queue head (if PM-Count is not equal to zero), if a PID mismatch is detected (for example, expected DATA1 and received DATA0, or visa-versa).,

[2] Note, this specification does not *require* that the host controller execute another bus transaction when *PM-Count* is non-zero. Implementations are encouraged to make appropriate complexity and performance trade-offs.

32.11.12.4 Write Back qTD

This state is entered from the Execute Transaction state when the *Active* bit is set to a zero. The source data for the write-back is the transfer results area of the queue head overlay area (see Figure 50 Queue Head Structure Layout50). The host controller uses the *Current qTD Pointer* field as the target address for the qTD. The queue head transfer result area is written back to the transfer result area of the target qTD. This state is also referred to as: qTD retirement. The fields that must be written back to the source qTD include *Total Bytes to Transfer*, *Cerr*, and *Status*.

The duration of this state depends on when the qTD write-back has been committed.

32.11.12.5 Follow Queue Head Horizontal Pointer

The host controller must use the horizontal pointer in the queue head to the next schedule data structure when any of the following conditions exist:

- If the *Active* bit is a one on exit from the Execute Transaction state, or
- When the host controller exits the Write Back qTD state, or
- If the Advance Queue state fails to advance the queue because the target qTD is not active, or
- If the *Halted* bit is a one on exit from the Fetch QH state.

There is no functional requirement that the host controller wait until the current transaction is complete before using the horizontal pointer to read the next linked data structure. However, it must wait until the current transaction is complete before executing the next data structure.

32.11.12.6 Buffer Pointer List Use for Data Streaming with qTDs

A qTD has an array of buffer pointers, which is used to reference the data buffer for a transfer. This specification requires that the buffer associated with the transfer be *virtually contiguous*. This means: if the buffer spans more than one physical page, it must obey the following rules (Figure 66 Example Mapping of qTD Buffer Pointers to Buffer Pages66 illustrates an example):

- The first portion of the buffer must begin at some offset in a page and extend through the end of the page.
- The remaining buffer cannot be allocated in small chunks scattered around memory. For each 4K chunk beyond the first page, each buffer portion matches to a full 4K page. The final portion, which may only be large enough to occupy a portion of a page, must start at the top of the page and be contiguous within that page.

The buffer pointer list in the qTD is long enough to support a maximum transfer size of 20K bytes. This case occurs when all five buffer pointers are used and the first offset is zero. A qTD handles a 16Kbyte buffer with any starting buffer alignment.

The host controller uses the field *C_Page* field as an index value to determine which buffer pointer in the list should be used to start the current transaction. The host controller uses a different buffer pointer for each physical page of the buffer. This is always true, even if the buffer is physically contiguous.

The host controller must detect when the current transaction will span a page boundary and automatically move to the next available buffer pointer in the page pointer list. The next available pointer is reached by incrementing *C_Page* and pulling the next page pointer from the list. Software must ensure there are sufficient buffer pointers to move the amount of data specified in the *Bytes to Transfer* field.

Figure 32-69 Example Mapping of qTD Buffer Pointers to Buffer Pages66 illustrates a nominal example of how System software would initialize the buffer pointers list and the *C_Page* field for a transfer size of 16383 bytes. *C_Page* is set to zero. The upper 20-bits of Page 0 references the start of the physical page. *Current Offset* (the lower 12-bits of queue head Dword 7) holds the offset in the page for example, 2049 (for example, 4096-2047). The remaining page pointers are set to reference the beginning of each subsequent 4K page.

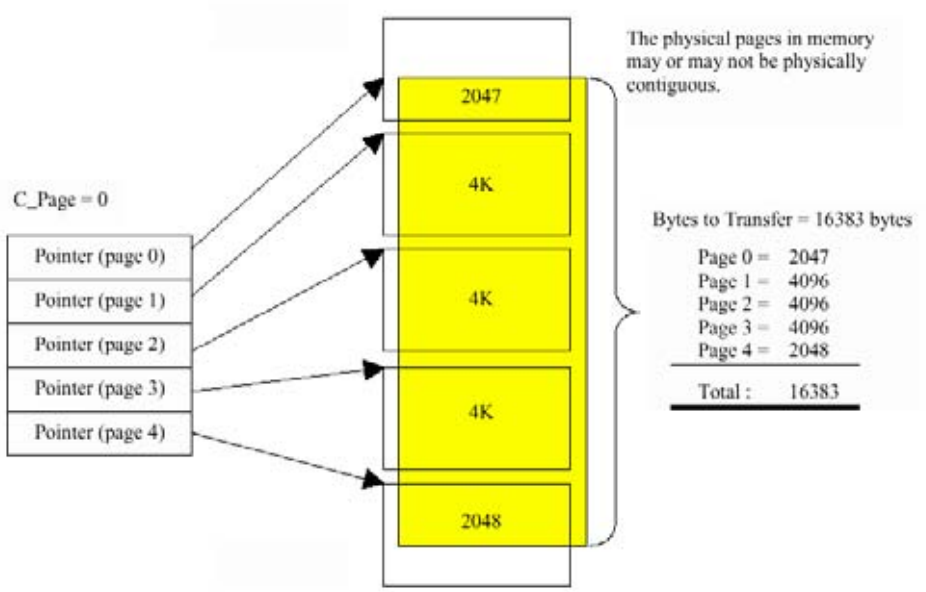


Figure 32-69. Example Mapping of qTD Buffer Pointers to Buffer Pages

For the first transaction on the qTD (assuming a 512-byte transaction), the host controller uses the first buffer pointer (page 0 because *C_Page* is set to zero) and concatenates the *Current Offset* field. The 512 bytes are moved during the transaction, the *Current Offset* and *Total Bytes to Transfer* are adjusted by 512 and written back to the queue head working area.

During the 4th transaction, the host controller needs 511 bytes in page 0 and one byte in page 1. The host controller will increment *C_Page* (to 1) and use the page 1 pointer to move the final byte of the transaction. After the 4th transaction, the active page pointer is the page 1 pointer and *Current Offset* has rolled to one, and both are written back to the overlay area. The transactions continue for the rest of the buffer, with the host controller automatically moving to the next page pointer (that is, *C_Page*) when necessary. There are three conditions for how the host controller handles *C_Page*.

- The current transaction does not span a page boundary. The value of *C_Page* is not adjusted by the host controller.
- The current transaction does span a page boundary. The host controller must detect the page cross condition and advance to the next buffer while streaming data to/from the USB.
- The current transaction completes on a page boundary (that is, the last byte moved for the current transaction is the last byte in the page for the current page pointer). The host controller must increment *C_Page* before writing back status for the transaction.

Note that the only valid adjustment the host controller may make to *C_Page* is to increment by one.

32.11.12.7 Adding Interrupt Queue Heads to the Periodic Schedule

The link path(s) from the periodic frame list to a queue head establishes in which frames a transaction can be executed for the queue head. Queue heads are linked into the periodic schedule so they are polled at the appropriate rate. System software sets a bit in a queue head's *S-Mask* to indicate which micro-frame with-in a 1 millisecond period a transaction should be executed for the queue head. Software must ensure

that all queue heads in the periodic schedule have *S-Mask* set to a non-zero value. An *S-mask* with a zero value in the context of the periodic schedule yields undefined results.

If the desired poll rate is greater than one frame, system software can use a combination of queue head linking and *S-Mask* values to spread interrupts of equal poll rates through the schedule so that the periodic bandwidth is allocated and managed in the most efficient manner possible. Some examples are illustrated in Table 32-78.

Table 32-78. Example Periodic Reference Patterns for Interrupt Transfers with 2ms Poll Rate

Frame # Reference Sequence	Description
0, 2, 4, 6, 8, and so on *S-Mask* = 01h	A queue head for the *bInterval* of 2 milliseconds (16 micro-frames) is linked into the periodic schedule so that it is reachable from the periodic frame list locations indicated in the previous column. In addition, the *S-Mask* field in the queue head is set to 01h, indicating that the transaction for the endpoint should be executed on the bus during micro-frame 0 of the frame.
0, 2, 4, 6, 8, and so on *S-Mask* = 02h	Another example of a queue head with a *bInterval* of 2 milliseconds is linked into the periodic frame list at exactly the same interval as the previous example. However, the *S-Mask* is set to 02h indicating that the transaction for the endpoint should be executed on the bus during micro-frame 1 of the frame.

32.11.12.8 Managing Transfer Complete Interrupts from Queue Heads

The host controller will set an interrupt to be signaled at the next interrupt threshold when the completed transfer (qTD) has an *Interrupt on Complete* (*IOC*) bit set to a one, or whenever a transfer (qTD) completes with a short packet. If system software needs multiple qTDs to complete a client request (that is, like a control transfer) the intermediate qTDs do not require interrupts. System software may only need a single interrupt to notify it that the complete buffer has been transferred. System software may set IOC's to occur more frequently. A motivation for this may be that it wants early notification so that interface data structures can be re-used in a timely manner.

32.11.13 Ping Control

USB 2.0 defines an addition to the protocol for high-speed devices called Ping. Ping is required for all USB 2.0 High-speed bulk and control endpoints. Ping is not allowed for a split-transaction stream. This extension to the protocol eliminates the bad side-effects of Naking OUT endpoints. The *Status* field has a *Ping State* bit, which the host controller uses to determine the *next* actual PID it will use in the next transaction to the endpoint (see Table 18 qTD Token (DWord 2)18). The Ping State bit is only managed by the host controller for queue heads that meet the following criteria:

- Queue head is not an interrupt and
- *EPS* field equals High-Speed and
- *PIDCode* field equals OUT

Table 36 Ping Control State Transition Table36 illustrates the state transition table for the host controller's responsibility for maintaining the PING protocol. Refer to Chapter 8 in the USB Specification Revision 2.0 for detailed description on the Ping protocol.

Table 32-79. Ping Control State Transition Table

	Event		
Current	**Host**	**Device**	**Next**
Do Ping	PING	Nak	Do Ping
Do Ping	PING	Ack	Do OUT
Do Ping	PING	XactErr[1]	Do Ping
Do Ping	PING	Stall	N/C[2] Do
OUT	OUT	Nak	Do Ping
Do OUT	OUT	Nyet	Do Ping
Do OUT	OUT	Ack	Do OUT
Do OUT	OUT	XactErr[1]	Do Ping
Do OUT	OUT	Stall	N/C[2]

1 Transaction Error (XactErr) is any time the host misses the handshake.

2 No transition change required for the Ping State bit. The Stall handshake results in the endpoint being halted (for example, Active set to zero and Halt set to a one). Software intervention is required to restart queue. 3 A Nyet response to an OUT means that the device has accepted the data, but cannot receive any more at this time. Host must advance the transfer state and additionally, transition the Ping State bit to Do Ping. The Ping State bit has the following encoding:

Table 32-80. Ping State Encoding

Value	Meaning
0B	Do OUT The host controller will use an OUT PID during the next bus transaction to this endpoint.
1B	Do Ping The host controller will use a PING PID during the next bus transaction to this endpoint.

The defined ping protocol (see USB 2.0 Specification, Chapter 8) allows the host to be *imprecise* on the initialization of the ping protocol (that is, start in *Do OUT* when we don't know whether there is space on the device or not). The host controller manages the *Ping State* bit. System software sets the initial value in the queue head when it initializes a queue head. The host controller preserves the *Ping State* bit across all queue advancements. This means that when a new qTD is written into the queue head overlay area, the previous value of the *Ping State* bit is preserved.

32.11.14 Split Transactions

USB 2.0 defines extensions to the bus protocol for managing USB 1.x data streams through USB 2.0 Hubs. This section describes how the host controller uses the interface data structures to manage data streams

with full- and low-speed devices, connected below USB 2.0 hub, utilizing the split transaction protocol. Refer to USB 2.0 Specification for the complete definition of the split transaction protocol. Full- and Low-speed devices are enumerated identically as high-speed devices, but the transactions to the Full- and Low-speed endpoints use the split-transaction protocol on the high-speed bus. The split transaction protocol is an encapsulation of (or wrapper around) the Full- or Low-speed transaction. The high-speed wrapper portion of the protocol is addressed to the USB 2.0 Hub and Transaction Translator below which the Full- or Low-speed device is attached.

The EHCI interface uses dedicated data structures for managing full-speed isochronous data streams (see Section Split Transaction Isochronous Transfer Descriptor (siTD)). Control, Bulk and Interrupt are managed using the queuing data structures (see Sections Queue Head). The interface data structures need to be programmed with the device address and the Transaction Translator number of the USB 2.0 Hub operating as the Low-/Full-speed host controller for this link. The following sections describe the details of how the host controller must process and manage the split transaction protocol.

32.11.14.1 Split Transactions for Asynchronous Transfers

A queue head in the asynchronous schedule with an *EPS* field indicating a full-or low-speed device indicates to the host controller that it must use split transactions to stream data for this queue head. All full-speed bulk and full-, low-speed control are managed via queue heads in the asynchronous schedule.

Software must initialize the queue head with the appropriate device address and port number for the transaction translator that is serving as the full/low-speed host controller for the links connecting the endpoint. Software must also initialize the split transaction state bit (*SplitXState*) to Do-Start-Split. Finally, if the endpoint is a control endpoint, then system software must set the *Control Transfer Type* (*C*) bit in the queue head to a one. If this is not a control transfer type endpoint, the *C* bit must be initialized by software to be a zero. This information is used by the host controller to properly set the Endpoint Type (ET) field in the split transaction bus token. When the *C* bit is a zero, the split transaction token's ET field is set to indicate a bulk endpoint. When the *C* bit is a one, the split transaction token's ET field is set to indicate a control endpoint. Refer to Chapter 8 of USB Specification Revision 2.0 for details.

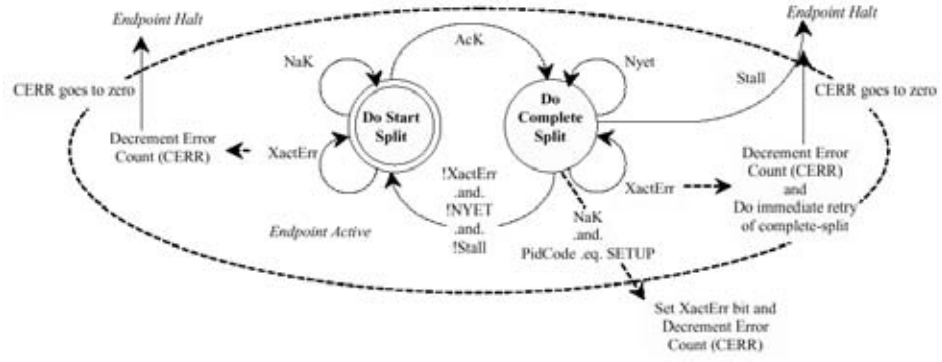


Figure 32-70. Host Controller Asynchronous Schedule Split-Transaction State Machine

32.11.14.1.1 Asynchronous—Do Start Split

This is the state which software must initialize a full- or low-speed asynchronous queue head. This state is entered from the Do Complete Split state only after a complete-split transaction receives a valid response from the transaction translator that is not a Nyet handshake.

For queue heads in this state, the host controller will execute a start-split transaction to the appropriate transaction translator. If the bus transaction completes without an error and *PidCode* indicates an IN or OUT transaction, then the host controller will reload the error counter (*CErr*). If it is a successful bus transaction and the *PidCode* indicates a SETUP, the host controller will not reload the error counter. If the transaction translator responds with a Nak, the queue head is left in this state, and the host controller proceeds to the next queue head in the asynchronous schedule.

If the host controller times out the transaction (no response, or bad response) the host controller decrements *Cerr* and proceeds to the next queue head in the asynchronous schedule.

32.11.14.1.2 Asynchronous—Do Complete Split

This state is entered from the Do Start Split state only after a start-split transaction receives an Ack handshake from the transaction translator.

For queue heads in this state, the host controller will execute a complete-split transaction to the appropriate transaction translator. If the transaction translator responds with a Nyet handshake, the queue head is left in this state, the error counter is reset and the host controller proceeds to the next queue head in the asynchronous schedule. When a Nyet handshake is received for a bus transaction where the queue head's *PidCode* indicates an IN or OUT, the host controller will reload the error counter (*CErr*). When a Nyet handshake is received for a complete-split bus transaction where the queue head's *PidCode* indicates a SETUP, the host controller must not adjust the value of *CErr.*

Independent of *PIDCode*, the following responses have the effects:

- Transaction Error (XactErr). Timeout or data CRC failure, and so on. The error counter (*Cerr*) is decremented by one and the complete split transaction is *immediately* retried (if possible). If there is not enough time in the micro-frame to execute the retry, the host controller MUST ensure that the next time the host controller begins executing from the Asynchronous schedule, it must begin executing from this queue head. If another start-split (for some other endpoint) is sent to the transaction translator before the complete-split is really completed, the transaction translator could dump the results (which were never delivered to the host). This is why the core specification states the retries must be immediate. A method to accomplish this behavior is to not advance the asynchronous schedule. When the host controller returns to the asynchronous schedule in the next micro-frame, the first transaction from the schedule will be the retry for this endpoint. If *Cerr* went to zero, the host controller must halt the queue.
- NAK. The target endpoint Nak'd the full- or low-speed transaction. The state of the transfer is not advanced and the state is exited. If the *PidCode* is a SETUP, then the Nak response is a protocol error. The *XactErr* status bit is set to a one and the *CErr* field is decremented.
- STALL. The target endpoint responded with a STALL handshake. The host controller sets the *halt* bit in the status byte, retires the qTD but does not attempt to advance the queue.
- If the *PidCode* indicates an IN, then any of following responses are expected:

- DATA0/1. On reception of data, the host controller ensures the PID matches the expected data toggle and checks CRC. If the packet is *good*, the host controller will advance the state of the transfer, for example, move the data pointer by the number of bytes received, decrement *BytesToTransfer* field by the number of bytes received, and toggle the *dt* bit. The host controller will then exit this state. The response and advancement of transfer may trigger other processing events, such as retirement of the qTD and advancement of the queue.
- If the data sequence PID does not match the expected, the data is ignored, the transfer state is not advanced and this state is exited. If the *PidCode* indicates an OUT/SETUP, then any of following responses are expected:
- ACK. The target endpoint accepted the data, so the host controller must advance the state of the transfer. The *Current Offset* field is incremented by *Maximum Packet Length* or *Bytes to Transfer*, whichever is less. The field *Bytes To Transfer* is decremented by the same amount and the data toggle bit (*dt*) is toggled. The host controller will then exit this state.
- Advancing the transfer state may cause other processing events such as retirement of the qTD and advancement of the queue (see Section 32.11.12, “Managing Control/Bulk/Interrupt Transfers via Queue Heads”).

32.11.14.2 Split Transaction Interrupt

Split-transaction Interrupt-IN/OUT endpoints are managed via the same data structures used for high-speed interrupt endpoints. They both co-exist in the periodic schedule. Queue heads/qTDs offer the set of features required for reliable data delivery, which is characteristic to interrupt transfer types. The split-transaction protocol is managed completely within this defined functional transfer framework. For example, for a high-speed endpoint, the host controller will visit a queue head, execute a high-speed transaction (if criteria are met) and advance the transfer state (or not) depending on the results of the entire transaction. For low- and full-speed endpoints, the details of the *execution* phase are different (that is, takes more than one bus transaction to complete), but the remainder of the operational framework is intact. This means that the transfer advancement, and so on occurs as defined in Section 32.11.12, “Managing Control/Bulk/Interrupt Transfers via Queue Heads” but only occurs on the completion of a split transaction.

32.11.14.2.1 Split Transaction Scheduling Mechanisms for Interrupt

Full- and low-speed Interrupt queue heads have an *EPS* field indicating full- or low-speed and have a non-zero *S-mask* field. The host controller can detect this combination of parameters and assume the endpoint is a periodic endpoint. Low- and full-speed interrupt queue heads require the use of the split transaction protocol. The host controller sets the Endpoint Type (ET) field in the split token to indicate the transaction is an interrupt. These transactions are managed through a transaction translator's periodic pipeline. Software should not set these fields to indicate the queue head is an interrupt unless the queue head is used in the periodic schedule.

System software manages the per/transaction translator periodic pipeline by budgeting and scheduling exactly during which micro-frames the start-splits and complete-splits for each endpoint will occur. The characteristics of the transaction translator are such that the high-speed transaction protocol must execute during explicit micro-frames, or the data or response information in the pipeline is lost. Figure 68 Split Transaction, Interrupt Scheduling Boundary Conditions68 illustrates the general scheduling boundary

conditions that are supported by the EHCI periodic schedule and queue head data structure. The **S** and C_x labels indicate micro-frames where software can schedule start-splits and complete splits (respectively).

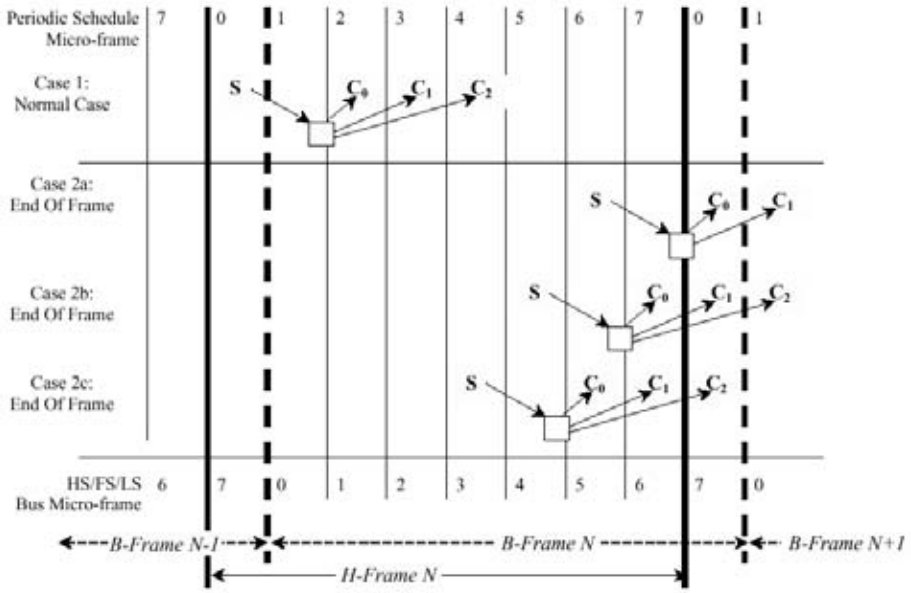


Figure 32-71. Split Transaction, Interrupt Scheduling Boundary Conditions

The scheduling cases are:

- Case 1: The normal scheduling case is where the entire split transaction is completely bounded by a frame (*H-Frame* in this case).
- Case 2a through Case 2c: The USB 2.0 Hub pipeline rules states clearly, when and how many complete-splits must be scheduled to account for earliest to latest execution on the full/low-speed link. The complete-splits may span the *H-Frame* boundary when the start-split is in micro-frame 4 or later. When this occurs, the *H-Frame* to *B-Frame* alignment requires that the queue head be reachable from consecutive periodic frame list locations. System software cannot build an efficient schedule that satisfies this requirement unless it uses FSTNs. Figure 69 General Structure of EHCI Periodic Schedule Utilizing Interrupt Spreading69 illustrates the general layout of the periodic schedule.

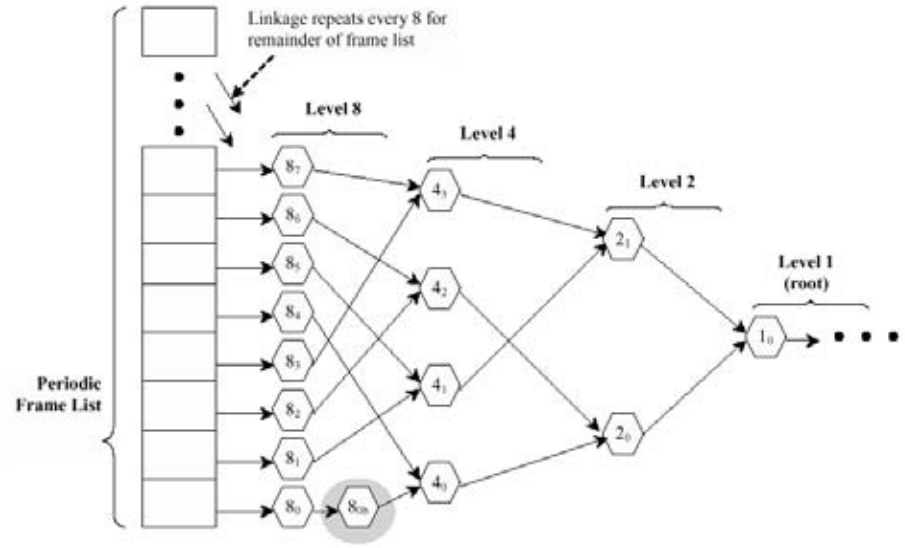


Figure 32-72. General Structure of EHCI Periodic Schedule Utilizing Interrupt Spreading

The periodic frame list is effectively the leaf level a binary tree, which is always traversed leaf to root. Each level in the tree corresponds to a 2^N poll rate. Software can efficiently manage periodic bandwidth on the USB by *spreading* interrupt queue heads that have the same poll rate requirement across all the available paths from the frame list. For example, system software can schedule eight poll rate 8 queue heads and account for them once in the high-speed bus bandwidth allocation.

When an endpoint is allocated an execution footprint that spans a frame boundary, the queue head for the endpoint must be reachable from consecutive locations in the frame list. An example would be if 8_{0b} where such an endpoint. Without additional support on the interface, to get 8_{0b} reachable at the correct time, software would have to link 8_1 to 8_{0b}. It would then have to move 4_1 and everything linked after into the same path as 4_0. This upsets the integrity of the binary tree and disallows the use of the spreading technique.

FSTN data structures are used to preserve the integrity of the binary-tree structure and enable the use of the spreading technique. Section Host Controller Operational Model for FSTNs defines the hardware and software operational model requirements for using FSTNs.

The following queue head fields are initialized by system software to instruct the host controller when to execute portions of the split-transaction protocol.

- *SplitXState*. This is a single bit residing in the *Status* field of a queue head (see Table 18 qTD Token (DWord 2)18). This bit is used to track the current state of the split transaction.
- *Frame S-mask*. This is a bit-field where-in system software sets a bit corresponding to the micro-frame (within an *H-Frame*) that the host controller should execute a start-split transaction. This is always qualified by the value of the *SplitXState* bit in the *Status* field of the queue head. For example, referring to Figure 68 Split Transaction, Interrupt Scheduling Boundary Conditions68, case one, the *S-mask* would have a value of 00000001b indicating that if the queue head is traversed by the host controller, and the *SplitXState* indicates Do_Start, and the current micro-frame as indicated by FRINDEX[2:0] is 0, then execute a start-split transaction.

- *Frame C-mask*. This is a bit-field where system software sets one or more bits corresponding to the micro-frames (within an *H-Frame*) that the host controller should execute complete-split transactions. The interpretation of this field is always qualified by the value of the *SplitXState* bit in the *Status* field of the queue head. For example, referring to Figure 68 Split Transaction, Interrupt Scheduling Boundary Conditions68, case one, the *C-mask* would have a value of 00011100b indicating that if the queue head is traversed by the host controller, and the *SplitXState* indicates Do_Complete, and the current micro-frame as indicated by FRINDEX[2:0] is 2, 3, or 4, then execute a complete-split transaction. It is software's responsibility to ensure that the translation between *H-Frames* and *B-Frames* is correctly performed when setting bits in *S-mask* and *C-mask*

32.11.14.2.2 Host Controller Operational Model for FSTNs

The FSTN data structure is used to manage Low/Full-speed interrupt queue heads that need to be reached from consecutive frame list locations (that is, boundary cases 2a through 2c). An FSTN is essentially a *back pointer*, similar in intent to the back pointer field in the siTD data structure (see Section siTD Back Link Pointer).

This feature provides software a simple primitive to save a schedule position, redirect the host controller to traverse the necessary queue heads in the previous frame, then restore the original schedule position and complete normal traversal.

There are four components to the use of FSTNs:

- FSTN data structure, defined in Section Section 32.10.8, “Periodic Frame Span Traversal Node (FSTN) .”
- A *Save Place* indicator. This is always an FSTN with its *Back Path Link Pointer. T-bit* set to zero.
- A *Restore* indicator. This is always an FSTN with its *Back Path Link Pointer.T-bit* set to a one.
- Host controller FSTN traversal rules.

Host Controller Operational Model for FSTNs

When the host controller encounters an FSTN during micro-frames 2 through 7 it simply follows the node’s *Normal Path Link Pointer* to access the next schedule data structure. Note that the FSTN’s *Normal Path Link Pointer.T-bit* may set to a one, which the host controller must interpret as the end of periodic list mark.

When the host controller encounters a *Save-Place* FSTN in micro-frames 0 or 1, it will save the value of the *Normal Path Link Pointer* and set an internal flag indicating that it is executing in *Recovery Path* mode. *Recovery Path* mode modifies the host controller’s rules for how it traverses the schedule and limits which data structures will be considered for execution of bus transactions. The host controller continues executing in *Recovery Path* mode until it encounters a *Restore* FSTN or it determines that it has reached the end of the micro-frame (see details in the list below).

The rules for schedule traversal and limited execution while in *Recovery Path* mode are:

- Always follow the *Normal Path Link Pointer* when it encounters an FSTN that is a *Save-Place* indicator. The host controller must not recursively follow *Save-Place* FSTNs. Therefore, while executing in *Recovery Path* mode, it must never follow an FSTN’s *Back Path Link Pointer.*

- Do not process an siTD or, iTD data structure. Simply follow its *Next Link Pointer.*
- Do not process a QH (Queue Head) whose *EPS* field indicates a high-speed device. Simply follow its *Horizontal Link Pointer.*
- When a QH's *EPS* field indicates a Full/Low-speed device, the host controller will only consider it for execution if its *SplitXState* is DoComplete (note: this applies whether the *PID Code* indicates an IN or an OUT). See Sections Execute Transaction and Tracking Split Transaction Progress for Interrupt Transfers for a complete list of additional conditions that must be met in general for the host controller to issue a bus transaction. Note that the host controller must not execute a Start-split transaction while executing in *Recovery Path* mode. See Periodic Interrupt—Do Complete Split for special handling when in *Recovery Path* mode.
- Stop traversing the *recovery path* when it encounters an FSTN that is a *Restore* indicator. The host controller unconditionally uses the saved value of the *Save-Place* FSTN's *Normal Path Link Pointer* when returning to the normal path traversal. The host controller must clear the context of executing a *Recovery Path* when it restores schedule traversal to the *Save-Place* FSTN's *Normal Path Link Pointer.*
- If the host controller determines that there is not enough time left in the micro-frame to complete processing of the periodic schedule, it abandons traversal of the recovery path, and clears the context of executing a recovery path. The result is that at the start of the next consecutive micro-frame, the host controller starts traversal at the frame list.

An example traversal of a periodic schedule that includes FSTNs is illustrated in Figure 70 Example Host Controller Traversal of Recovery Path via FSTNs70.

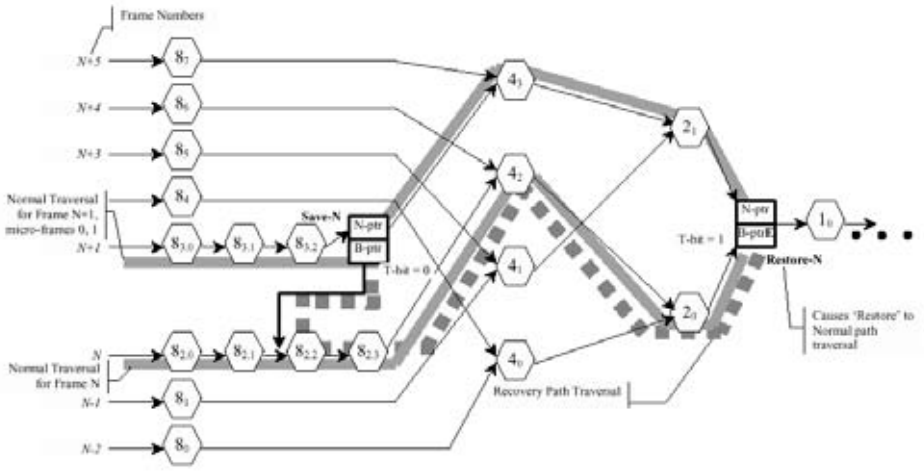


Figure 32-73. Example Host Controller Traversal of Recovery Path via FSTNs

In frame N+1 (micro-frames 0 and 1), when the host controller encounters Save-Path FSTN (Save-N), it observes that Save-N.Back Path Link Pointer.T-bit is zero (definition of a Save-Path indicator). The host controller saves the value of Save-N.Normal Path Link Pointer and follows Save-N.Back Path Link Pointer. At the same time, it sets an internal flag indicating that it is now in *Recovery Path* mode (the recovery path is annotated in Figure 70 Example Host Controller Traversal of Recovery Path via FSTNs70 with a large dashed line). The host controller continues traversing data structures on the recovery path and executing only those bus transactions as noted above, on the recovery path until it reaches Restore FSTN

(Restore-N). Restore-N.Back Path Link Pointer.T-bit is set to a one (definition of a Restore indicator), so the host controller exits *Recovery Path* mode by clearing the internal *Recovery Path* mode flag and commences (restores) schedule traversal using the saved value of the *Save-Place* FSTN's *Normal Path Link Pointer* (for example, Save-N.Normal Path Link Pointer). The nodes traversed during these micro-frames include: {$8_{3.0}$, $8_{3.1}$, $8_{3.2}$, Save-A, $8_{2.2}$, $8_{2.3}$, 4_2, 2_0, Restore-N, 4_3, 2_1, Restore-N, 1_0 …}. The nodes on the recovery-path are bolded.In frame N (micro-frames 0-7), for this example, the host controller will traverse all of the schedule data structures utilizing the *Normal Path Link Pointers* in any FSTNs it encounters. This is because the host controller has not yet encountered a *Save-Place* FSTN so it not executing in *Recovery Path* mode. When it encounters the *Restore* FSTN, (Restore-N), during micro-frames 0 and 1, it uses Restore-N.Normal Path Link Pointer to traverse to the next data structure (that is, normal schedule traversal). This is because the host controller must use a Restore FSTN's *Normal Path Link Pointer* when not executing in a *Recovery-Path* mode. The nodes traversed during frame N include: {$8_{2.0}$, $8_{2.1}$, $8_{2.2}$, $8_{2.3}$, 4_2, 2_0, Restore-N, 1_0 …}.

In frame N+1 (micro-frames 2-7), when the host controller encounters Save-Path FSTN Save-N, it will unconditionally follow Save-N.Normal Path Link Pointer. The nodes traversed during these micro-frames include: {$8_{3.0}$, $8_{3.1}$, $8_{3.2}$, Save-A, 4_3, 2_1, Restore-N, 1_0 …}.

32.11.14.3 Software Operational Model for FSTNs

Software must create a consistent, coherent schedule for the host controller to traverse. When using FSTNs, system software must adhere to the following rules:

- Each *Save-Place* indicator requires a matching *Restore* indicator.
  - The *Save-Place* indicator is an FSTN with a valid *Back Path Link Pointer and T-bit* equal to zero. Note that *Back Path Link Pointer.Typ* field must be set to indicate the referenced data structure is a queue head. The *Restore* indicator is an FSTN with its *Back Path Link Pointer.T-bit* set to a one.
  - A *Restore* FSTN may be matched to one or more *Save-Place* FSTNs. For example, if the schedule includes a poll-rate 1 level, then system software only needs to place a *Restore* FSTN at the beginning of this list to match all possible *Save-Place* FSTNs.
- If the schedule does not have elements linked at a poll-rate level of one, and one or more *Save-Place* FSTNs are used, then System Software must ensure the *Restore* FSTN's *Normal Path Link Pointer*'s *T-bit* is set to a one, as this will be use to mark the end of the periodic list.
- When the schedule does have elements linked at a poll rate level of one, a *Restore* FSTN must be the first data structure on the poll rate one list. All traversal paths from the frame list converge on the poll-rate one list. System software must ensure that *Recovery Path* mode is exited before the host controller is allowed to traverse the poll rate level one list.
- A *Save-Place* FSTN's *Back Path Link Pointer* must reference a queue head data structure. The referenced queue head must be reachable from the previous frame list location. In other words, if the *Save-Place* FSTN is reachable from frame list offset N, then the FSTN's *Back Path Link Pointer* must reference a queue head that is reachable from frame list offset N-1.

Software should make the schedule as efficient as possible. What this means in this context is that software should have no more than one *Save-Place* FSTN reachable in any single frame. Note there will be times when two (or more, depending on the implementation) could exist as full/low-speed footprints change with

bandwidth adjustments. This could occur, for example when a bandwidth rebalance causes system software to move the *Save-Place* FSTN from one poll rate level to another. During the transition, software must preserve the integrity of the previous schedule until the new schedule is in place.

32.11.14.3.1 Tracking Split Transaction Progress for Interrupt Transfers

To correctly maintain the data stream, the host controller must be able to detect and report errors where data is lost. For interrupt-IN transfers, data is lost when it makes it into the USB 2.0 hub, but the USB 2.0 host system is unable to get it from the USB 2.0 Hub and into the system before it expires from the transaction translator pipeline. When a lost data condition is detected, the queue must be halted, thus signaling system software to recover from the error. A data-loss condition exists whenever a start-split is issued, accepted and successfully executed by the USB 2.0 Hub, but the complete-splits get unrecoverable errors on the high-speed link, or the complete-splits do not occur at the correct times. One reason complete-splits might not occur at the right time would be due to host-induced system hold-offs that cause the host controller to miss bus transactions because it cannot get timely access to the schedule in system memory.

The same condition can occur for an interrupt-OUT, but the result is not an endpoint halt condition, but rather effects only the progress of the transfer. The queue head has the following fields to track the progress of each split transaction. These fields are used to keep incremental state about which (and when) portions have been executed.

- *C-prog-mask*. This is an eight-bit bit-vector where the host controller keeps track of which complete-splits have been executed. Due to the nature of the Transaction Translator periodic pipeline, the complete-splits need to be executed in-order. The host controller needs to detect when the complete-splits have not been executed in order. This can only occur due to system hold-offs where the host controller cannot get to the memory-based schedule. *C-prog-mask* is a simple bit-vector that the host controller sets one of the *C-prog*-mask bits for each complete-split executed. The bit position is determined by the micro-frame number in which the complete-split was executed. The host controller always checks *C-prog-mask* before executing a complete-split transaction. If the previous complete-splits have not been executed then it means one (or more) have been skipped and data has potentially been lost.
- *FrameTag*. This field is used by the host controller during the complete-split portion of the split transaction to tag the queue head with the frame number (*H-Frame* number) when the next complete split must be executed.
- *S-bytes*. This field can be used to store the number of data payload bytes sent during the start-split (if the transaction was an OUT). The *S-bytes* field must be used to accumulate the data payload bytes received during the complete-splits (for an IN).

32.11.14.3.2 Split Transaction Execution State Machine for Interrupt

In the following presentation, all references to micro-frame are in the context of a micro-frame within an *H-Frame*.

As with asynchronous Full- and Low-speed endpoints, a split-transaction state machine is used to manage the split transaction sequence. Aside from the fields defined in the queue head for scheduling and tracking

the split transaction, the host controller calculates one internal mechanism that is also used to manage the split transaction. The internal calculated mechanism is:

- *cMicroFrameBit*. This is a single-bit encoding of the current micro-frame number. It is an eight-bit value calculated by the host controller at the beginning of every micro-frame. It is calculated from the three least significant bits of the *FRINDEX* register (that is, *cMicroFrameBit* = (1 shifted-left(*FRINDEX*[2:0]))). The *cMicroFrameBit* has at most one bit asserted, which always corresponds to the current micro-frame number. For example, if the current micro-frame is 0, then *cMicroFrameBit* will equal 00000001b. The variable *cMicroFrameBit* is used to compare against the *S-mask* and *C-mask* fields to determine whether the queue head is marked for a start- or complete-splt transaction for the current micro-frame.

Figure 32-74 illustrates the state machine for managing a complete interrupt split transaction. There are two phases to each split transaction. The first is a single start-split transaction, which occurs when the *SplitXState* is at Do_Start and the single bit in *cMicroFrameBit* has a corresponding bit active in *QH.S-mask*. The transaction translator does not acknowledge the receipt of the periodic start-split, so the host controller unconditionally transitions the state to Do_Complete. Due to the available jitter in the transaction translator pipeline, there will be more than one complete-split transaction scheduled by software for the Do_Complete state. This translates simply to the fact that there are multiple bits set to a one in the *QH.C-mask* field.

The host controller keeps the queue head in the Do_Complete state until the split transaction is complete (see definition below), or an error condition triggers the *three-strikes-rule* (for example, after the host tries the same transaction three times, and each encounters an error, the host controller will stop retrying the bus transaction and halt the endpoint, thus requiring system software to detect the condition and perform system-dependent recovery).

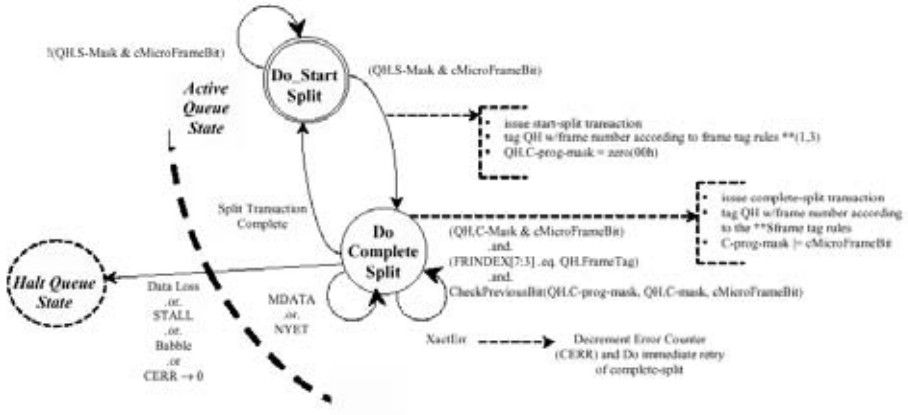


Figure 32-74. Split Transaction State Machine for Interrupt

See Managing QH.FrameTag Field for the frame tag management rules.

Periodic Interrupt—Do Start Split

This is the state software must initialize a full- or low-speed interrupt queue head *StartXState* bit. This state is entered from the Do_Complete Split state only after the split transaction is complete. This occurs when

one of the following events occur: The transaction translator responds to a complete-split transaction with one of the following:

- NAK. A NAK response is a propagation of the full- or low-speed endpoint's NAK response.
- ACK. An ACK response is a propagation of the full- or low-speed endpoint's ACK response. Only occurs on an OUT endpoint.
- DATA 0/1. Only occurs for INs. Indicates that this is the last of the data from the endpoint for this split transaction.
- ERR. The transaction on the low-/full-speed link below the transaction translator had a failure (for example, timeout, bad CRC, and so on).
- NYET (and Last). The host controller issued the last complete-split and the transaction translator responded with a NYET handshake. This means that the start-split was not correctly received by the transaction translator, so it never executed a transaction to the full- or low-speed endpoint, see Periodic Interrupt—Do Complete Split for the definition of ‘Last’.

Each time the host controller visits a queue head in this state (once within the Execute Transaction state), it performs the following test to determine whether to execute a start-split.

- *QH.S-mask* is bit-wise anded with *cMicroFrameBit*.

If the result is non-zero, then the host controller will issue a start-split transaction. If the *PIDCode* field indicates an IN transaction, the host controller must zero-out the *QH.S-bytes* field. After the split-transaction has been executed, the host controller sets up state in the queue head to track the progress of the complete-split phase of the split transaction. Specifically, it records the expected frame number into *QH.FrameTag* field (see Section Managing QH.FrameTag Field), set *C-prog-mask* to zero (00h), and exits this state. Note that the host controller must not adjust the value of *CErr* as a result of completion of a start-split transaction.

Periodic Interrupt—Do Complete Split

This state is entered unconditionally from the Do Start Split state after a start-split transaction is executed on the bus. Each time the host controller visits a queue head in this state (once within the Execute Transaction state), it checks to determine whether a complete-split transaction should be executed now.

There are four tests to determine whether a complete-split transaction should be executed.

- Test A. cMicroFrameBit is bit-wise anded with QH.C-mask field. A non-zero result indicates that software scheduled a complete-split for this endpoint, during this micro-frame.
- Test B. QH.FrameTag is compared with the current contents of FRIN*DEX*[7:3]. An equal indicates a match.
- Test C. The complete-split progress bit vector is checked to determine whether the previous bit is set, indicating that the previous complete-split was appropriately executed. An example algorithm for this test is provided below:

```
Algorithm Boolean CheckPreviousBit(QH.C-prog-mask, QH.C-mask, cMicroFrameBit)
Begin
-- Return values:
-- TRUE - no error
-- FALSE - error
--
```

```
Boolean rvalue = TRUE;
previousBit = cMicroframeBit logical-rotate-right(1)
-- Bit-wise anding previousBit with C-mask indicates
-- whether there was an intent
-- to send a complete split in the previous micro-frame. So,
-- if the
-- 'previous bit' is set in C-mask, check C-prog-mask to
-- make sure it
-- happened.
If (previousBit bitAND QH.C-mask)then
                          If not(previousBit bitAND QH.C-prog-mask) then
                  rvalue = FALSE;
         End if
End If
-- If the C-prog-mask already has a one in this bit position,
-- then an aliasing
-- error has occurred. It will probably get caught by the
-- FrameTag Test, but
-- at any rate it is an error condition that as detectable here
-- should not allow
-- a transaction to be executed.
If (cMicroFrameBit bitAND QH.C-prog-mask) then
 rvalue = FALSE;
End if
return (rvalue)
End Algorithm
```

- Test D. Check to see if a start-split should be executed in this micro-frame. Note this is the same test performed in the Do Start Split state (see Periodic Interrupt—Do Complete Split). Whenever it evaluates to TRUE and the controller is NOT processing in the context of a *Recovery Path* mode, it means a start-split should occur in this micro-frame. Test D and Test A evaluating to TRUE at the same time is a system software error. Behavior is undefined.

If (A .and. B .and. C .and. not(D)) then the host controller will execute a complete-split transaction. When the host controller commits to executing the complete-split transaction, it updates *QH.C-prog-mask* by bit-ORing with *cMicroFrameBit*. On completion of the complete-split transaction, the host controller records the result of the transaction in the queue head and sets *QH.FrameTag* to the expected *H-Frame* number (see Section Managing QH.FrameTag Field). The effect to the state of the queue head and thus the state of the transfer depends on the response by the transaction translator to the complete-split transaction. The following responses have the effects (note that any responses that result in decrementing of the *CErr* will result in the queue head being halted by the host controller if the result of the decrement is zero):

- NYET (and Last). On each NYET response, the host controller checks to determine whether this is the last complete-split for this split transaction. Last is defined in this context as the condition where all of the scheduled complete-splits have been executed. If it is the last complete-split (with a NYET response), then the transfer state of the queue head is not advanced (never received any data) and this state exited. The transaction translator must have responded to all the clompete-splits with NYETs, meaning that the start-split issued by the host controller was not received. The start-split should be retried at the next poll period.

- The test for whether this is the Last complete split can be performed by XOR *QH.C-mask* with *QH.C-prog-mask*. If the result is all zeros then all complete-splits have been executed. When this condition occurs, the *XactErr* status bit is set to a one and the *CErr* field is decremented.
- NYET (and not Last). See above description for testing for Last. The complete-split transaction received a NYET response from the transaction translator. Do not update any transfer state (except for *C-prog-mask* and *FrameTag*) and stay in this state. The host controller must not adjust *CErr* on this response.
- Transaction Error (XactErr). Timeout, data CRC failure, and so on. The *CErr* field is decremented and the *XactErr* bit in the *Status* field is set to a one. The complete split transaction is *immediately* retried (if *Cerr* is non-zero).If there is not enough time in the micro-frame to complete the retry and the endpoint is an IN, or *CErr* is decremented to a zero from a one, the queue is halted. If there is not enough time in the micro-frame to complete the retry and the endpoint is an OUT and *CErr* is not zero, then this state is exited (that is, return to Do Start Split). This results in a retry of the entire OUT split transaction, at the next poll period. Refer to Chapter 11 Hubs (specifically the section full- and low-speed Interrupts) in the USB Specification Revision 2.0 for detailed requirements on why these errors must be immediately retried.
- ACK. This can only occur if the target endpoint is an OUT. The target endpoint ACK'd the data and this response is a propagation of the endpoint ACK up to the host controller. The host controller must advance the state of the transfer. The *Current Offset* field is incremented by *Maximum Packet Length* or *Bytes to Transfer*, whichever is less. The field *Bytes To Transfer* is decremented by the same amount. And the data toggle bit (*dt*) is toggled. The host controller will then exit this state for this queue head. The host controller must reload *CErr* with maximum value on this response. Advancing the transfer state may cause other process events such as retirement of the qTD and advancement of the queue (see Section 32.11.12, “Managing Control/Bulk/Interrupt Transfers via Queue Heads”).
- MDATA. This response will only occur for an IN endpoint. The transaction translator responded with zero or more bytes of data and an MDATA PID. The incremental number of bytes received is accumulated in *QH.S-bytes*. The host controller must not adjust *CErr* on this response.
- DATA0/1. This response may only occur for an IN endpoint. The number of bytes received is added to the accumulated byte count in *QH.S-bytes*. The state of the transfer is advanced by the result and the host controller will exit this state for this queue head.
- Advancing the transfer state may cause other processing events such as retirement of the qTD and advancement of the queue (see Section Section 32.11.12, “Managing Control/Bulk/Interrupt Transfers via Queue Heads).
- If the data sequence PID does not match the expected, the entirety of the data received in this split transaction is ignored, the transfer state is not advanced and this state is exited.
- NAK. The target endpoint Nak'd the full- or low-speed transaction. The state of the transfer is not advanced, and this state is exited. The host controller must reload *CErr* with maximum value on this response.
- ERR. There was an error during the full- or low-speed transaction. The ERR status bit is set to a one, *Cerr* is decremented, the state of the transfer is not advanced, and this state is exited.
- STALL. The queue is halted (an exit condition of the Execute Transaction state). The status field bits: *Active* bit is set to zero and the *Halted* bit is set to a one and the qTD is retired. Responses

which are not enumerated in the list or which are received out of sequence are illegal and may result in undefined host controller behavior. The other possible combinations of tests A, B, C, and D may indicate that data or response was lost. Table 32-81 lists the possible combinations and the appropriate action.

Table 32-81. Interrupt IN/OUT Do Complete Split State Execution Criteria

Condition	Action	Description
not(A) not(D)	Ignore QHD	Neither a start nor complete-split is scheduled for the current micro-frame.Host controller should continue walking the schedule.
A not(C)	If PIDCode = IN Halt QHD If PIDCode = OUT Retry start-split	Progress bit check failed. These means a complete-split has been missed. There is the possibility of lost data. If *PIDCode* is an IN, then the Queue head must be halted. If *PIDCode* is an OUT, then the transfer state is not advanced and the state exited (for example, start-split is retried). This is a host-induced error and does not effect *CER*R. In either case, set the *Missed Micro-frame* bit in the status field to a one.
A not(B) C	If PIDCode = IN Halt QHD If PIDCode = OUT Retry start-split	*QH.FrameTag* test failed. This means that exactly one or more *H-Frames* have been skipped. This means complete-splits and have missed. There is the possibility of lost data. If *PIDCode* is an IN, then the Queue head must be halted. If *PIDCode* is an OUT, then the transfer state is not advanced and the state exited (for example, start-split is retried). This is a host-induced error and does not effect *CER*R. In either case, set the *Missed Micro-frame* bit in the status field to a one.
A B C not(D)	Execute complete-split	This is the non-error case where the host controller executes a complete-split transaction.
D	If PIDCode = IN Halt QHD If PIDCode = OUT Retry start-split	This is a degenerate case where the start-split was issued, but all of the complete-splits were skipped and all possible intervening opportunities to detect the missed data failed to fire. If *PIDCode* is an IN, then the Queue head must be halted. If *PIDCode* is an OUT, then the transfer state is not advanced and the state exited (for example, start-split is retried). This is a host-induced error and does not effect *CER*R. In either case, set the *Missed Micro-frame* bit in the status field to a one. Note: When executing in the context of a *Recovery Path* mode, the host controller is allowed to process the queue head and take the actions indicated above, or it may wait until the queue head is visited in the normal processing mode. Regardless, the host controller must not execute a start-split in the context of a executing in a *Recovery Path* mode.

Managing QH.FrameTag Field

The *QH.FrameTag* field in a queue head is completely managed by the host controller. The rules for setting *QH.FrameTag* are simple:

- Rule 1: If transitioning from Do Start Split to Do Complete Split and the current value of FRINDEX[2:0] is 6 QH.FrameTag is set to FRINDEX[7:3] + 1. This accommodates split transactions whose start-split and complete-splits are in different H-Frames (case 2a, see Figure 68 Split Transaction, Interrupt Scheduling Boundary Conditions68).
- Rule 2: If the current value of FRINDEX[2:0] is 7, QH.FrameTag is set to FRINDEX[7:3] + 1. This accommodates staying in Do Complete Split for cases 2a, 2b, and 2c (Figure 68 Split Transaction, Interrupt Scheduling Boundary Conditions68).
- Rule 3: If transitioning from Do_Start Split to Do Complete Split and the current value of FRINDEX[2:0] is not 6, or currently in Do Complete Split and the current value of (FRINDEX[2:0]) is not 7, FrameTag is set to FRINDEX[7:3]. This accommodates all other cases (Figure 68 Split Transaction, Interrupt Scheduling Boundary Conditions68).

32.11.14.3.3 Rebalancing the Periodic Schedule

System software must occasionally adjust a periodic queue head's S-mask and C-mask fields during operation. This need occurs when adjustments to the periodic schedule create a new bandwidth budget and one or more queue head's are assigned new execution footprints (that is, new S-mask and C-mask values).

It is imperative that System software must not update these masks to new values in the midst of a split transaction. To avoid any race conditions with the update, the EHCI host controller provides a simple assist to system software. System software sets the *Inactivate-on-next-Transaction* (*I*) bit to a one to signal the host controller that it intends to update the S-mask and C-mask on this queue head. System software will then wait for the host controller to observe the *I-bit* is a one and transition the *Active* bit to a zero. The rules for how and when the host controller sets the *Active* bit to zero are enumerated below:

- If the *Active* bit is a zero, no action is taken. The host controller does not attempt to advance the queue when the *I-bit* is a one.
- If the *Active* bit is a one and the *SplitXState* is DoStart (regardless of the value of *S-mask*), the host controller will simply set *Active* bit to a zero. The host controller is not required to write the transfer state back to the *current* qTD. Note that if the *S-mask* indicates that a start-split is scheduled for the current micro-frame, the host controller must not issue the start-split bus transaction. It must set the *Active* bit to zero.

System software must save transfer state before setting the *I-bit* to a one. This is required so that it can correctly determine what transfer progress (if any) occurred after the *I-bit* was set to a one and the host controller executed it's final bus-transaction and set *Active* to a zero.

After system software has updated the S-mask and C-mask, it must then reactivate the queue head. Since the *Active* bit and the *I-bit* cannot be updated with the same write, system software needs to use the following algorithm to coherently re-activate a queue head that has been stopped via the *I-bit*.

5. Set the *Halted* bit to a one, then
6. Set the *I-bit* to a zero, then
7. Set the *Active* bit to a one and the *Halted* bit to a zero in the same write.

Setting the *Halted* bit to a one inhibits the host controller from attempting to advance the queue between the time the *I-bit* goes to a zero and the *Active* bit goes to a one.

32.11.14.4 Split Transaction Isochronous

Full-speed isochronous transfers are managed using the split-transaction protocol through a USB 2.0 transaction translator in a USB2.0 Hub. The EHCI controller utilizes siTD data structure to support the special requirements of isochronous split-transactions. This data structure uses the scheduling model of isochronous TDs (iTD, Section Isochronous (High-Speed) Transfer Descriptor (iTD)) (see Section Managing Isochronous Transfers Using iTDs for the operational model of iTDs) with the contiguous data feature provided by queue heads. This simple arrangement allows a single isochronous scheduling model and adds the additional feature that all data received from the endpoint (per split transaction) must land into a contiguous buffer.

32.11.14.4.1 Split Transaction Scheduling Mechanisms for Isochronous

Full-speed isochronous transactions are managed through a transaction translator's periodic pipeline. As with full- and low-speed interrupt, system software manages each transaction translator's periodic pipeline by budgeting and scheduling exactly during which micro-frames the start-splits and complete-splits for each full-speed isochronous endpoint occur. The requirements described in Section Split Transaction Scheduling Mechanisms for Interrupt apply. Figure 72 Split Transaction, Isochronous Scheduling Boundary Conditions72 illustrates the general scheduling boundary conditions that are supported by the EHCI periodic schedule. The ^{s}x and ^{c}x labels indicate micro-frames where software can schedule start- and complete-splits (respectively). The *H-Frame* boundaries are marked with a large, solid bold vertical line. The *B-Frame* boundaries are marked with a large, bold, dashed line. The bottom of the figure illustrates the relationship of an siTD to the *H-Frame*.

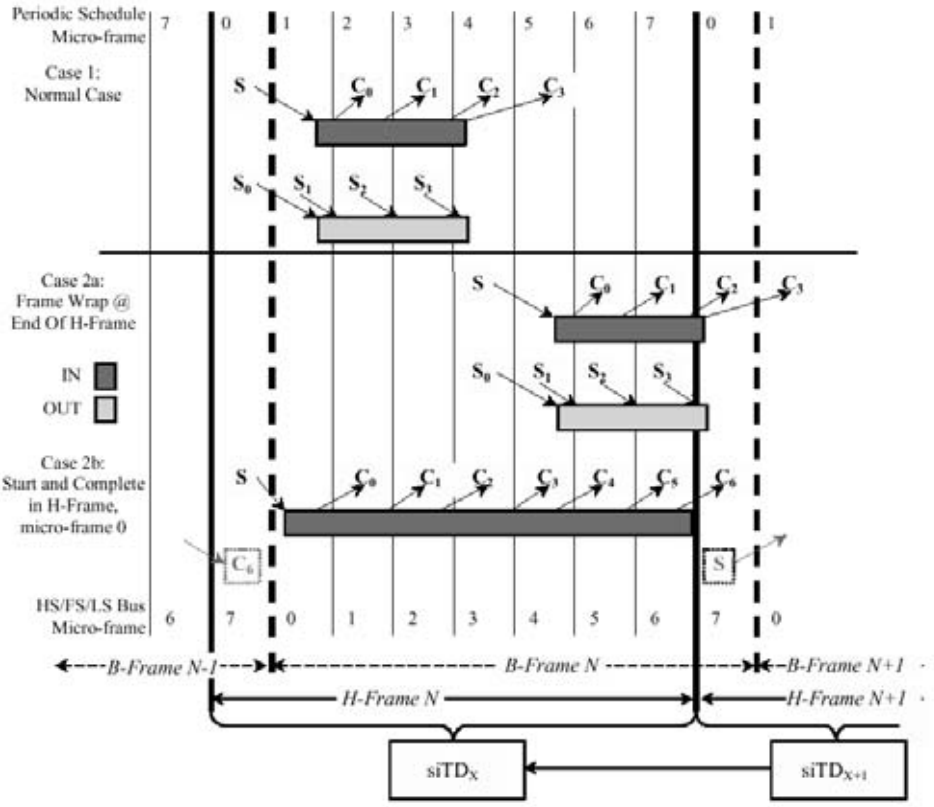


Figure 32-75. Split Transaction, Isochronous Scheduling Boundary Conditions

When the endpoint is an isochronous OUT, there are only start-splits, and no complete-splits. When the endpoint is an isochronous IN, there is at most one start-split and one to *N* complete-splits. The scheduling boundary cases are:

- *Case 1*: The entire split transaction is completely bounded by an *H-Frame*. For example: the start-splits and complete-splits are all scheduled to occur in the same *H-Frame*.
- *Case 2a*: This boundary case is where one or more (at most two) complete-splits of a split transaction IN are scheduled across an *H-Frame* boundary. This can only occur when the split transaction has the possibility of moving data in *B-Frame*, micro-frames 6 or 7 (*H-Frame* micro-frame 7 or 0). When an *H-Frame* boundary wrap condition occurs, the scheduling of the split transaction spans more than one location in the periodic list.(for example, it takes two siTDs in adjacent periodic frame list locations to fully describe the scheduling for the split transaction).
- Although the scheduling of the split transaction may take two data structures, all of the complete-splits for each full-speed IN isochronous transaction must use only one data pointer. For this reason, siTDs contain a back pointer, the use of which is described below.
- Software must never schedule full-speed isochronous OUTs across an *H-Frame* boundary.
- *Case 2b*: This case can only occur for a very large isochronous IN. It is the only allowed scenario where a start-split and complete-split for the same endpoint can occur in the same micro-frame. Software must enforce this rule by scheduling the large transaction first. Large is defined to be anything larger than 579 byte maximum packet size.

A subset of the same mechanisms employed by full- and low-speed interrupt queue heads are employed in siTDs to schedule and track the portions of isochronous split transactions. The following fields are initialized by system software to instruct the host controller when to execute portions of the split transaction protocol.

- *SplitXState*. This is a single bit residing in the *Status* field of an siTD (see Table 13 siTD Transfer Status and Control13). This bit is used to track the current state of the split transaction. The rules for managing this bit are described in Section In the following presentation, all references to micro-frame are in the context of a micro-frame within an H-Frame.Split Transaction Execution State Machine for Isochronous .
- *Frame S-mask*. This is a bit-field where-in system software sets a bit corresponding to the micro-frame (within an *H-Frame*) that the host controller should execute a start-split transaction. This is always qualified by the value of the *SplitXState* bit. For example, referring to the IN example in Figure 72 Split Transaction, Isochronous Scheduling Boundary Conditions72, case one, the *S-mask* would have a value of 00000001b indicating that if the siTD is traversed by the host controller, and the *SplitXState* indicates Do Start Split, and the current micro-frame as indicated by FRINDEX[2:0] is 0, then execute a start-split transaction.
- *Frame C-mask*. This is a bit-field where system software sets one or more bits corresponding to the micro-frames (within an *H-Frame*) that the host controller should execute complete-split transactions. The interpretation of this field is always qualified by the value of the *SplitXState* bit. For example, referring to the IN example in Figure 72 Split Transaction, Isochronous Scheduling Boundary Conditions72, case one, the *C-mask* would have a value of 00111100b indicating that if the siTD is traversed by the host controller, and the *SplitXState* indicates **Do Complete Split**, and the current micro-frame as indicated by *FRINDEX*[2:0] is 2, 3, 4, or 5, then execute a complete-split transaction.
- *Back Pointer*. This field in a siTD is used to complete an IN split-transaction using the previous *H-Frame*'s siTD. This is only used when the scheduling of the complete-splits span an *H-Frame* boundary.

There exists a one-to-one relationship between a high-speed isochronous split transaction (including all start- and complete-splits) and one full-speed isochronous transaction. An siTD contains (amongst other things) buffer state and split transaction scheduling information. An siTD's buffer state always maps to one full-speed isochronous data payload. This means that for any full-speed transaction payload, a single siTD's data buffer must be used. This rule applies to both IN an OUTs. An siTD's scheduling information usually also maps to one high-speed isochronous split transaction. The exception to this rule is the *H-Frame* boundary wrap cases mentioned above.

The siTD data structure describes at most, one frame's worth of high-speed transactions and that description is strictly bounded within a frame boundary. Figure 73 siTD Scheduling Boundary Examples73 illustrates some examples. On the top are examples of the full-speed transaction footprints for the boundary scheduling cases described above. In the middle are time-frame references for both the *B-Frames* (HS/FS/LS Bus) and the *H-Frames*. On the bottom is illustrated the relationship between the scope of an siTD description and the time references. Each *H-Frame* corresponds to a single location in the periodic frame list. The implication is that each siTD is reachable from a single periodic frame list location at a time.

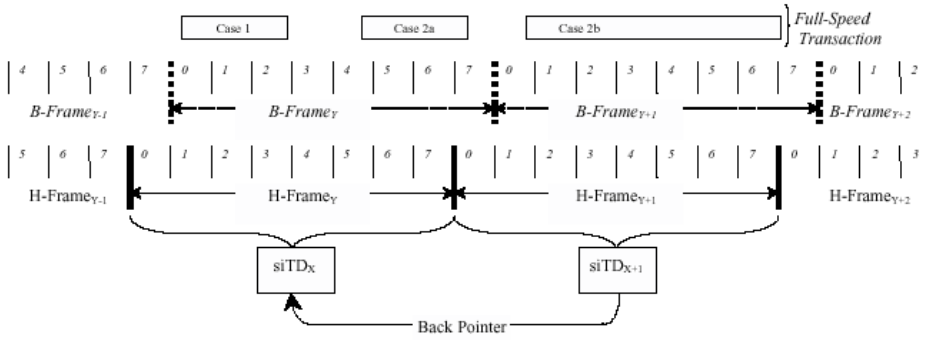


Figure 32-76. siTD Scheduling Boundary Examples

Each case is described below:

- *Case 1*: One siTD is sufficient to describe and complete the isochronous split transaction because the whole isochronous split transaction is tightly contained within a single *H-Frame*.
- *Case 2a, 2b*: Although both INs and OUTs can have these footprints, OUTs always take only one siTD to schedule. However, INs (for these boundary cases) require two siTDs to complete the scheduling of the isochronous split transaction. $siTD_X$ is used to always issue the start-split and the first *N* complete-splits. The full-speed transaction (for these cases) can deliver data on the full-speed bus segment during micro-frame 7 of *H-Frame*$_{Y+1}$, or micro-frame 0 of *H-Frame*$_{Y+2}$. The complete splits are scheduled using $siTD_{X+2}$ (not shown). The complete-splits to extract this data must use the buffer pointer from $siTD_{X+1}$. The only way for the host controller to reach $siTD_{X+1}$ from *H-Frame*$_{Y+2}$ is to use $siTD_{X+2}$'s back pointer. The host controller rules for when to use the back pointer are described is Section Periodic Isochronous - Do Complete Split .

Software must apply the following rules when calculating the schedule and linking the schedule data structures into the periodic schedule:

- Software must ensure that an isochronous split-transaction is started so that it will complete before the end of the *B-Frame*.
- Software must ensure that for a single full-speed isochronous endpoint, there is never a start-split and complete-split in *H-Frame, micro-frame 1*. This is mandated as a rule so that case 2a and case 2b can be discriminated. According to the core USB specification, the long isochronous transaction illustrated in Case 2b, could be scheduled so that the start-split was in micro-frame 1 of *H-Frame* N and the last complete-split would need to occur in micro-frame 1 of H-Frame N+1. However, it is impossible to discriminate between cases 2a and case 2b, which has significant impact on the complexity of the host controller.

32.11.14.4.2 Tracking Split Transaction Progress for Isochronous Transfers

To correctly maintain the data stream, the host controller must be able to detect and report errors where device to host data is lost. Isochronous endpoints do not employ the concept of a halt on error, however the host is required to identify and report per-packet errors observed in the data stream. This includes schedule traversal problems (skipped micro-frames), timeouts and corrupted data received.

In similar kind to interrupt split-transactions, the portions of the split transaction protocol must execute in the micro-frames they are scheduled. The queue head data structure used to manage full- and low-speed interrupt has several mechanisms for tracking when portions of a transaction have occurred. Isochronous transfers use siTDs, for their transfers, and the data structures are only reachable via the schedule in the exact micro-frame in which they are required (so all the mechanism employed for tracking in queue heads is not required for siTDs). Software has the option of reusing siTD several times in the complete periodic schedule. However, it must ensure that the results of split transaction N are consumed and the siTD reinitialized (activated) before the host controller gets back to the siTD (in a future micro-frame).

Split-transaction isochronous OUTs utilize a low-level protocol to indicate which portions of the split transaction data have arrived. Control over the low-level protocol is exposed in an siTD via the fields *Transaction Position* (*TP*) and *Transaction Count* (*T-count*). If the entire data payload for the OUT split transaction is larger than 188 bytes, there will be more than one start-split transaction, each of which require proper annotation. If host hold-offs occur, then the sequence of annotations received from the host will not be complete, which is detected and handled by the transaction translator. See Section Periodic Isochronous - Do Start Split for a description on how these fields are used during a sequence of start-split transactions.

The fields *siTD.T-Count* and *siTD.TP* are used by the host controller to drive and sequence the transaction position annotations. It is the responsibility of system software to properly initialize these fields in each siTD. Once the budget for a split-transaction isochronous endpoint is established, *S-mask*, *T-Count*, and *TP* initialization values for all the siTD associated with the endpoint are constant. They remain constant until the budget for the endpoint is recalculated by software and the periodic schedule adjusted.

For IN-endpoints, the transaction translator simply annotates the response data packets with enough information to allow the host controller to identify the last data. As with split transaction Interrupt, it is the host controller's responsibility to detect when it has missed an opportunity to execute a complete-split. The following field in the siTD is used to track and detect errors in the execution of a split transaction for an IN isochronous endpoint.

- *C-prog-mask*. This is an eight-bit bit-vector where the host controller keeps track of which complete-splits have been executed. Due to the nature of the Transaction Translator periodic pipeline, the complete-splits need to be executed in-order. The host controller needs to detect when the complete-splits have not been executed in order. This can only occur due to system hold-offs where the host controller cannot get to the memory-based schedule. *C-prog-mask* is a simple bit-vector that the host controller sets a bit for each complete-split executed. The bit position is determined by the micro-frame (FRINDEX[2:0]) number in which the complete-split was executed. The host controller always checks *C-prog-mask* before executing a complete-split transaction. If the previous complete-splits have not been executed, then it means one (or more) have been skipped and data has potentially been lost. System software is required to initialize this field to zero before setting an siTD's *Active* bit to a one.

If a transaction translator returns with the final data before all of the complete-splits have been executed, the state of the transfer is advanced so that the remaining complete-splits are not executed. Refer to Section Periodic Isochronous - Do Complete Split for a description on how the state of the transfer is advanced. It is important to note that an IN siTD is retired based solely on the responses from the Transaction Translator to the complete-split transactions. This means, for example, that it is possible for a transaction translator to respond to a complete-split with an MDATA PID. The number of bytes in the MDATA's data

payload could cause the siTD field *Total Bytes to Transfer* to decrement to zero. This response can occur, before all of the scheduled complete-splits have been executed. In other interface, data structures (for example, high-speed data streams through queue heads), the transition of *Total Bytes to Transfer* to zero signals the end of the transfer and results in setting of the *Active* bit to zero. However, in this case, the result has not been delivered by the Transaction Translator and the host must continue with the next complete-split transaction to extract the residual transaction state. This scenario occurs because of the pipeline rules for a Transaction Translator (see Chapter 11 of the Universal Serial Bus Revision 2.0). In summary the periodic pipeline rules require that on a micro-frame boundary, the Transaction Translator will hold the final two bytes received (if it has not seen an End Of Packet (EOP)) in the full-speed bus pipe stage and give the remaining bytes to the high-speed pipeline stage. At the micro-frame boundary, the Transaction Translator could have received the entire packet (including both CRC bytes) but not received the packet EOP. In the next micro-frame, the Transaction Translator will respond with an MDATA and send all of the data bytes (with the two CRC bytes being held in the full-speed pipeline stage). This could cause the siTD to decrement it's *Total Bytes to Transfer* field to zero, indicating it has received all expected data. The host must still execute one more (scheduled) complete-split transaction in order to extract the results of the full-speed transaction from the Transaction Translator (for example, the Transaction Translator may have detected a CRC failure, and this result must be forwarded to the host).

If the host experiences hold-offs that cause the host controller to skip one or more (but not all) scheduled split transactions for an isochronous OUT, then the protocol to the transaction translator will not be consistent and the transaction translator will detect and react to the problem. Likewise, for host hold-offs that cause the host controller to skip one or more (but not all) scheduled split transactions for an isochronous IN, the *C-prog-mask* is used by the host controller to detect errors. However, if the host experiences a hold-off that causes it to skip all of an siTD, or an siTD expires during a host hold off (for example, a hold-off occurs and the siTD is no longer reachable by the host controller in order for it to report the hold-off event), then system software must detect that the siTDs have not been processed by the host controller (for example, state not advanced) and report the appropriate error to the client driver.

In the following presentation, all references to micro-frame are in the context of a micro-frame within an *H-Frame*.

Split Transaction Execution State Machine for Isochronous

If the *Active* bit in the *Status* byte is a zero, the host controller will ignore the siTD and continue traversing the periodic schedule. Otherwise the host controller will process the siTD as specified below. A split transaction state machine is used to manage the split-transaction protocol sequence. The host controller uses the fields defined in Section Tracking Split Transaction Progress for Isochronous Transfers , plus the variable *cMicroFrameBit* defined in Section Split Transaction Execution State Machine for Interrupt to track the progress of an isochronous split transaction. Figure 74 Split Transaction State Machine for Isochronous74 illustrates the state machine for managing an siTD through an isochronous split transaction. Bold, dotted circles denote the state of the *Active* bit in the *Status* field of a siTD. The Bold, dotted arcs denote the transitions between these states. Solid circles denote the states of the split transaction state machine and the solid arcs denote the transitions between these states. Dotted arcs and boxes reference actions that take place either as a result of a transition or from being in a state.

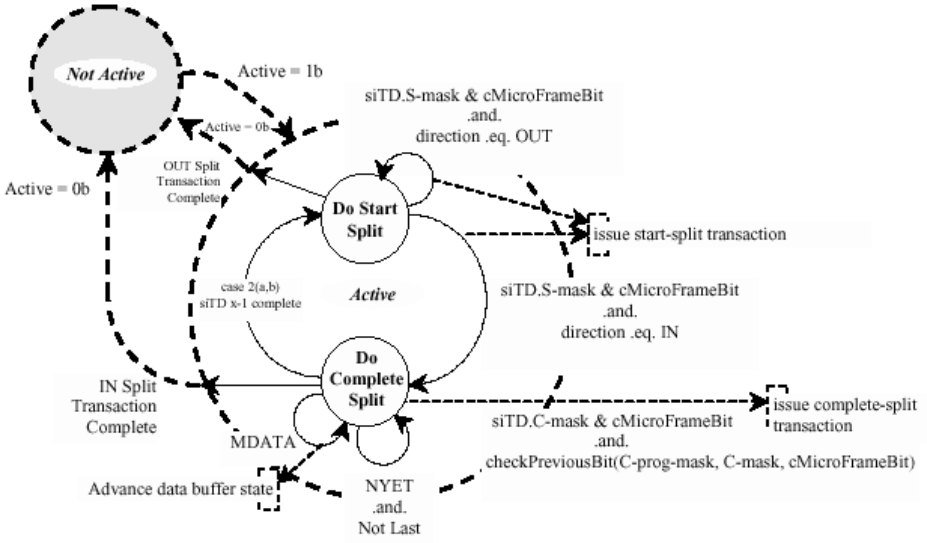


Figure 32-77. Split Transaction State Machine for Isochronous

Periodic Isochronous - Do Start Split

Isochronous split transaction OUTs use only this state. An siTD for a split-transaction isochronous IN is either initialized to this state, or the siTD transitions to this state from Do Complete Split when a case 2a (IN) or 2b scheduling boundary isochronous split-transaction completes.

Each time the host controller reaches an active siTD in this state, it checks the *siTD.S-mask* against *cMicroFrameBit*. If there is a one in the appropriate position, the siTD will execute a start-split transaction. By definition, the host controller cannot *reach* an siTD at the wrong time. If the *I/O* field indicates an IN, then the start-split transaction includes only the extended token plus the full-speed token. Software must initialize the *siTD.Total Bytes To Transfer* field to the number of bytes expected. This is usually the maximum packet size for the full-speed endpoint. The host controller exits this state when the start-split transaction is complete.

The remainder of this section is specific to an isochronous OUT endpoint (that is, the *I/O* field indicates an OUT). When the host controller executes a start-split transaction for an isochronous OUT it includes a data payload in the start-split transaction. The memory buffer address for the data payload is constructed by concatenating *siTD.Current Offset* with the page pointer indicated by the page selector field (*siTD.P*). A zero in this field selects Page 0 and a 1 selects Page 1. During the start-split for an OUT, if the data transfer crosses a page boundary during the transaction, the host controller must detect the page cross, update the *siTD.P*-bit from a zero to a one, and begin using the *siTD.Page 1* with *siTD.Current Offset* as the memory address pointer. The field *siTD.TP* is used to annotate each start-split transaction with the indication of which part of the split-transaction data the current payload represents (ALL, BEGIN, MID, END). In all cases the host controller simply uses the value in *siTD.TP* to mark the start-split with the correct transaction position code.

T-Count is always initialized to the number of start-splits for the current frame. *TP* is always initialized to the first required transaction position identifier. The scheduling boundary case (see Figure 73 siTD Scheduling Boundary Examples73) is used to determine the initial value of *TP*. The initial cases are summarized in Table 38 Initial Conditions for OUT siTD's TP and T-count Fields38.

Table 32-82. Table 38 Initial Conditions for OUT siTD's TP and T-count Fields

Case	T-count	TP	Description
1, 2a	=1	ALL	When the OUT data payload is less than (or equal to) 188 bytes, only one start-split is required to move the data. The one start-split must be marked with an ALL.
1, 2a	!=1	BEGIN	When the OUT data payload is greater than 188 bytes more than one start-split must be used to move the data. The initial start-split must be marked with a BEGIN.

After each start-split transaction is complete, the host controller updates *T-Count* and *TP* appropriately so that the next start-split is correctly annotated. Table 39 Transaction Position (TP)/Transaction Count (T-Count) Transition Table39 illustrates all of the *TP* and *T-count* transitions, which must be accomplished by the host controller.

Table 32-83. Transaction Position (TP)/Transaction Count (T-Count) Transition Table

TP	T-count next	TP next	Description
ALL	0	N/A	Transition from ALL, to done.
BEGIN	1	END	Transition from BEGIN to END. Occurs when *T-count* starts at 2.
BEGIN	!=1	MID	Transition from BEGIN to MID. Occurs when *T-count* starts at greater than 2.
MID	!=1	MID	*TP* stays at MID while *T-count* is not equal to 1 (for example, greater than 1). This case can occur for any of the scheduling boundary cases where the *T-count* starts greater than 3.
MID	1	END	Transition from MID to END. This case can occur for any of the scheduling boundary cases where the *T-count* starts greater than 2.

The start-split transactions do not receive a handshake from the transaction translator, so the host controller always advances the transfer state in the siTD after the bus transaction is complete. To advance the transfer state the following operations take place:

- The *siTD.Total Bytes To Transfer* and the *siTD.Current Offset* fields are adjusted to reflect the number of bytes transferred.
- The *siTD.P* (page selector) bit is updated appropriately.
- The *siTD.TP* and *siTD.T-count* fields are updated appropriately as defined in Table 39 Transaction Position (TP)/Transaction Count (T-Count) Transition Table39.

These fields are then written back to the memory based siTD. The *S-mask* is fixed for the life of the current budget. As mentioned above, *TP* and *T-count* are set specifically in each siTD to reflect the data to be sent from this siTD. Therefore, regardless of the value of *S-mask*, the actual number of start-split transactions depends on *T-count* (or equivalently, *Total Bytes to Transfer*). The host controller must set the *Active* bit to a zero when it detects that all of the schedule data has been sent to the bus. The preferred method is to detect when *T-Count* decrements to zero as a result of a start-split bus transaction. Equivalently, the host

controller can detect when *Total Bytes to Transfer* decrements to zero. Either implementation must ensure that if the initial condition is *Total Bytes to Transfer* equal to zero and *T-count* is equal to a one, then the host controller will issue a single start-split, with a zero-length data payload. Software must ensure that *TP*, *T-count* and *Total Bytes to Transfer* are set to deliver the appropriate number of bus transactions from each siTD. An inconsistent combination will yield undefined behavior.

If the host experiences hold-offs that cause the host controller to skip start-split transactions for an OUT transfer, the state of the transfer will not progress appropriately. The transaction translator will observe protocol violations in the arrival of the start-splits for the OUT endpoint (for example, the transaction position annotation will be incorrect as received by the transaction translator).

Example scenarios are described in Section Split Transaction for Isochronous - Processing Examples .

A host controller implementation can optionally track the progress of an OUT split transaction by setting appropriate bits in the *siTD.C-prog-mask* as it executes each scheduled start-split. The *checkPreviousBit*() algorithm defined in Section Periodic Isochronous - Do Complete Split can be used prior to executing each start-split to determine whether start-splits were skipped. The host controller can use this mechanism to detect missed micro-frames. It can then set the siTD's *Active* bit to zero and stop execution of this siTD. This saves on both memory and high-speed bus bandwidth.

Periodic Isochronous - Do Complete Split

This state is only used by a split-transaction isochronous IN endpoint. This state is entered unconditionally from the Do Start State after a start-split transaction is executed for an IN endpoint. Each time the host controller visits an siTD in this state, it conducts a number of tests to determine whether it should execute a complete-split transaction. The individual tests are listed below. The sequence they are applied depends on which micro-frame the host controller is currently executing which means that the tests might not be applied until after the siTD referenced from the back pointer has been fetched.

- Test A. *cMicroFrameBit* is bit-wise anded with *siTD.C-mask* field. A non-zero result indicates that software scheduled a complete-split for this endpoint, during this micro-frame. This test is always applied to a newly fetched siTD that is in this state.
- Test B. The *siTD.C-prog-mask* bit vector is checked to determine whether the previous complete splits have been executed. An example algorithm is below (this is slightly different than the algorithm used in Periodic Interrupt—Do Complete Split). The sequence in which this test is applied depends on the current value of FRINDEX[2:0]. If FRINDEX[2:0] is 0 or 1, it is not applied until the back pointer has been used. Otherwise it is applied immediately.

```
Algorithm Boolean CheckPreviousBit(siTD.C-prog-mask, siTD.C-mask, cMicroFrameBit)
Begin
        Boolean rvalue = TRUE;
        previousBit = cMicroFrameBit rotate-right(1)
        -- Bit-wise anding previousBit with C-mask indicates whether there was an intent
        -- to send a complete split in the previous micro-frame. So, if the
        -- 'previous bit' is set in C-mask, check C-prog-mask to make sure it
        -- happened.
        if previousBit bitAND siTD.C-mask then
                if not (previousBit bitAND siTD.C-prog-mask) then
                        rvalue = FALSE
                End if
        End if
        Return rvalue
```

```
End Algorithm
```

If Test A is true and FRINDEX[2:0] is zero or one, then this is a case 2a or 2b scheduling boundary (see Figure 72 Split Transaction, Isochronous Scheduling Boundary Conditions72). See Section Periodic Isochronous - Do Complete Split for details in handling this condition.

If Test A and Test B evaluate to true, then the host controller will execute a complete-split transaction using the transfer state of the current siTD. When the host controller commits to executing the complete-split transaction, it updates *QH.C-prog-mask* by bit-ORing with *cMicroFrameBit*. The transfer state is advanced based on the completion status of the complete-split transaction. To advance the transfer state of an IN siTD, the host controller must:

- Decrement the number of bytes received from *siTD.Total Bytes To Transfer,*
- Adjust *siTD.Current Offset* by the number of bytes received,
- Adjust *siTD.P* (page selector) field if the transfer caused the host controller to use the next page pointer, and
- Set any appropriate bits in the *siTD.Status* field, depending on the results of the transaction.

Note that if the host controller encounters a condition where *siTD.Total Bytes To Transfer* is zero, and it receives more data, the host controller must not write the additional data to memory. The *siTD.Status.Active* bit must be set to zero and the *siTD.Status.Babble Detected* bit must be set to a one. The fields *siTD.Total Bytes To Transfer*, *siTD.Current Offset*, and *siTD.P* (page selector) are not required to be updated as a result of this transaction attempt.

The host controller must accept (assuming good data packet CRC and sufficient room in the buffer as indicated by the value of *siTD.Total Bytes To Transfer*) MDATA and DATA0/1 data payloads up to and including 192 bytes. A host controller implementation may optionally set *siTD.Status Active* to a zero and *siTD.Status.Babble Detected* to a one when it receives and MDATA or DATA0/1 with a data payload of more than 192 bytes. The following responses have the noted effects:

- ERR. The full-speed transaction completed with a time-out or bad CRC and this is a reflection of that error to the host. The host controller sets the *ERR* bit in the *siTD.Status* field and sets the *Active* bit to a zero.
- Transaction Error (XactErr). The complete-split transaction encounters a Timeout, CRC16 failure, and so on. The *siTD.Status* field *XactErr* field is set to a one and the complete-split transaction must be retried immediately. The host controller must use an internal error counter to count the number of retries as a counter field is not provided in the siTD data structure. The host controller will not retry more than two times. If the host controller exhausts the retries or the end of the micro-frame occurs, the *Active* bit is set to zero.
- DATAx (0 or 1). This response signals that the final data for the split transaction has arrived. The transfer state of the siTD is advanced and the *Active* bit is set to a zero. If the *Bytes To Transfer* field has not decremented to zero (including the reception of the data payload in the DATAx response), then less data than was expected, or allowed for was actually received. This *short packet* event does not set the USBINT status bit in the USBSTS register to a one. The host controller will not detect this condition.

- NYET (and Last). On each NYET response, the host controller also checks to determine whether this is the last complete-split for this split transaction. Last was defined in Periodic Interrupt—Do Complete Split. If it is the last complete-split (with a NYET response), then the transfer state of the siTD is not advanced (never received any data) and the *Active* bit is set to a zero. No bits are set in the *Status* field because this is essentially a skipped transaction. The transaction translator must have responded to all the scheduled clompete-splits with NYETs, meaning that the start-split issued by the host controller was not received. This result should be interpreted by system software as if the transaction was completely skipped. The test for whether this is the last complete split can be performed by XORing C-mask with C-prog-mask. A zero result indicates that all complete-splits have been executed.
- MDATA (and Last). See above description for testing for Last. This can only occur when there is an error condition. Either there has been a babble condition on the full-speed link, which delayed the completion of the full-speed transaction, or software set up the *S-mask* and/or *C-mask*s incorrectly. The host controller must set *XactErr* bit to a one and the *Active* bit is set to a zero.
- NYET (and not Last). See above description for testing for Last. The complete-split transaction received a NYET response from the transaction translator. Do not update any transfer state (except for *C-prog-mask*) and stay in this state.
- MDATA (and not Last). The transaction translator responds with an MDATA when it has partial data for the split transaction. For example, the full-speed transaction data payload spans from micro-frame X to X+1 and during micro-frame X, the transaction translator will respond with an MDATA and the data accumulated up to the end of micro-frame X. The host controller advances the transfer state to reflect the number of bytes received.

If Test A succeeds, but Test B fails, it means that one or more of the complete-splits have been skipped. The host controller sets the *Missed Micro-Frame* status bit and sets the *Active* bit to a zero.

Complete-Split for Scheduling Boundary Cases 2a, 2b

Boundary cases 2a and 2b (INs only) (see Figure 72 Split Transaction, Isochronous Scheduling Boundary Conditions72) require that the host controller use the transaction state context of the previous siTD to finish the split transaction. Table 40 Summary siTD Split Transaction State40 enumerates the transaction state fields.

Table 32-84. Table 40 Summary siTD Split Transaction State

Buffer State	Status	Execution Progress
Total Bytes To Transfer P (page select) Current Offset TP (transaction position) T-count (transaction count)	All bits in the status field	C-prog-mask

NOTE

TP and T-*count* are used only for Host to Device (OUT) endpoints.

If software has budgeted the schedule of this data stream with a frame wrap case, then it must initialize the *siTD.Back Pointer* field to reference a valid siTD and will have the *siTD.Back Pointer.T-bit* in the *siTD.Back Pointer*

field set to a zero. Otherwise, software must set the *siTD.Back Pointer.T-bit* in the *siTD.Back Pointer* field to a one. The host controller's rules for interpreting when to use the *siTD.Back Pointer* field are listed below. These rules apply only when the siTD's *Active* bit is a one and the *SplitXState* is Do Complete Split.

- When cMicroFrameBit is a 1h and the siTDX.Back Pointer.T-bit is a zero, or
- If cMicroFrameBit is a 2h and siTDX.S-mask[0] is a zero

When either of these conditions apply, then the host controller must use the transaction state from $siTD_{X-1}$.

To access $siTD_{X-1}$, the host controller reads on-chip the siTD referenced from $siTD_X$*.Back Pointer.*

The host controller must save the entire state from $siTD_X$ while processing $siTD_{X-1}$. This is to accommodate for case 2b processing. The host controller must not recursively walk the list of *siTD.Back Pointers*.

If $siTD_{X-1}$ is active (*Active* bit is a one and *SplitXStat* is Do Complete Split), then both Test A and Test B are applied as described above. If these criteria to execute a complete-split are met, the host controller executes the complete split and evaluates the results as described above. The transaction state (see Table 40 Summary siTD Split Transaction State40) of $siTD_{X-1}$ is appropriately advanced based on the results and written back to memory. If the resultant state of $siTD_{X-1}$'s *Active* bit is a one, then the host controller returns to the context of $siTD_X$, and follows its next pointer to the next schedule item. No updates to $siTD_X$ are necessary.

If $siTD_{X-1}$ is active (*Active* bit is a one and *SplitXStat* is Do Start Split), then the host controller must set *Active* bit to a zero and *Missed Micro-Frame* status bit to a one and the resultant status written back to memory.

If $siTD_{X-1}$'s *Active* bit is a zero, (because it was zero when the host controller first visited $siTD_{X-1}$ via $siTD_X$'s back pointer, it transitioned to zero as a result of a detected error, or the results of $siTD_{X-1}$'s complete-split transaction transitioned it to zero), then the host controller returns to the context of $siTD_X$ and transitions its *SplitXState* to Do Start Split. The host controller then determines whether the case 2b start split boundary condition exists (for example, if *cMicroframeBit* is a 1b and $siTD_X$*.S-mask*[0] is a 1b). If this criterion is met the host controller immediately executes a start-split transaction and appropriately advances the transaction state of $siTD_X$, then follows $siTD_X$*.Next Pointer* to the next schedule item. If the criterion is not met, the host controller simply follows $siTD_X$*.Next Pointer* to the next schedule item. Note that in the case of a 2b boundary case, the split-transaction of $siTD_{X-1}$ will have its *Active* bit set to zero when the host controller returns to the context of $siTD_X$. Also, note that software should not initialize an siTD with *C-mask* bits 0 and 1 set to a one and an *S-mask* with bit zero set to a one. This scheduling combination is not supported and the behavior of the host controller is undefined.

Split Transaction for Isochronous-Processing Examples

There is an important difference between how the hardware/software manages the isochronous split transaction state machine and how it manages the asynchronous and interrupt split transaction state machines. The asynchronous and interrupt split transaction state machines are encapsulated within a single queue head. The progress of the data stream depends on the progress of each split transaction. In some

respects, the split-transaction state machine is sequenced via the Execute Transaction queue head traversal state machine (see Figure 32-68).

Isochronous is a pure time-oriented transaction/data stream. The interface data structures are optimized to efficiently describe transactions that need to occur at specific times. The isochronous split-transaction state machine must be managed across these time-oriented data structures. This means that system software must correctly describe the scheduling of split-transactions across more than one data structure.

Then the host controller must make the appropriate state transitions at the appropriate times, in the correct data structures.

For example, Table 32-85 illustrates a couple of frames worth of scheduling required to schedule a case 2a full-speed isochronous data stream.

Table 32-85. Example Case 2a—Software Scheduling siTDs for an IN Endpoint

siTDx		Micro-Frames								Initial SplitXState
#	Masks	0	1	2	3	4	5	6	7	
X	S-Mask					1				Do Start Split
	C-Mask	1	1					1	1	
X+1	S-Mask					1				Do Complete Split
	C-Mask	1	1					1	1	
X+2	S-Mask					1				Do Complete Split
	C-Mask	1	1					1	1	
X+3	S-Mask	Repeats previous pattern								Do Complete Split
	C-Mask									

This example shows the first three siTDs for the transaction stream. Since this is the case-2a frame-wrap case, *S-mask*s of all siTDs for this endpoint have a value of 10h (a one bit in micro-frame 4) and *C-mask* value of C3h (one-bits in micro-frames 0,1, 6 and 7). Additionally, sofware ensures that the *Back Pointer* field of each siTD references the appropriate siTD data structure (and the *Back Pointer T-bits* are set to zero).

The initial *SplitXState* of the first siTD is Do Start Split. The host controller will visit the first siTD eight times during frame X. The C-mask bits in micro-frames 0 and 1 are ignored because the state is Do Start Split. During micro-frame 4, the host controller determines that it can run a start-split (and does) and changes *SplitXState* to Do Complete Split. During micro-frames 6 and 7, the host controller executes complete-splits. Notice the siTD for frame X+1 has it's *SplitXState* initialized to Do Complete Split. As the host controller continues to traverse the schedule during *H-Frame* X+1, it will visit the second siTD eight times. During micro-frames 0 and 1 it will detect that it must execute complete-splits.

During *H-Frame* X+1, micro-frame 0, the host controller detects that $siTD_{X+1}$'s *Back Pointer.T-bit* is a zero, saves the state of $siTD_{X+1}$ and fetches $siTD_X$. It executes the complete split transaction using the transaction state of $siTD_X$. If the $siTD_X$ split transaction is complete, siTD's *Active* bit is set to zero and results written back to $siTD_X$. The host controller retains the fact that $siTD_X$ is retired and transitions the *SplitXState* in the $siTD_{X+1}$ to Do Start Split. At this point, the host controller is prepared to execute the

start-split for $siTD_{X+1}$ when it reaches micro-frame 4. If the split-transaction completes early (transaction-complete is defined in Section Periodic Isochronous - Do Complete Split), for example, before all the scheduled complete-splits have been executed, the host controller will transition *$siTD_X$.SplitXState* to Do Start Split early and naturally skip the remaining scheduled complete-split transactions. For this example, $siTD_{X+1}$ does not receive a DATA0 response until *H-Frame* X+2, micro-frame 1.

During *H-Frame X+2*, micro-frame 0, the host controller detects that $siTD_{X+2}$'s *Back Pointer.T-bit* is a zero, saves the state of $siTD_{X+2}$ and fetches $siTD_{X+1}$. As described above, it executes another split transaction, receives an MDATA response, updates the transfer state, but does not modify the *Active* bit. The host controller returns to the context of $siTD_{X+2}$, and traverses it's next pointer without any state change updates to $siTD_{X+2}$. S

During *H-Frame X+2*, micro-frame 1, the host controller detects $siTD_{X+2}$'s *S-mask[0]* is a zero, saves the state of $siTD_{X+2}$ and fetches $siTD_{X+1}$. It executes another complete-split transaction, receives a DATA0 response, updates the transfer state and sets the *Active* bit to a zero. It returns to the state of $siTD_{X+2}$ and changes its *SplitXState* to Do Start Split. At this point, the host controller is prepared to execute start-splits for $siTD_{X+2}$ when it reaches micro-frame 4. <TBD… describe how software detects that there was missing micro-frames (don't think we care about missing out micro-frames. There is enough residual state to identify than not all transactions were executed.).

32.11.15 Host Controller Pause

When the host controller's *HCHalted* bit in the USBSTS register is a zero, the host controller is sending SOF (Start OF Frame) packets down all enabled ports. When the schedules are enabled, the EHCI host controller will access the schedules in main memory each micro-frame. This constant pinging of main memory is known to create CPU power management problems for mobile systems. Specifically, mobile systems aggressively manage the state of the CPU, based on recent history usage. In the more aggressive power saving modes, the CPU can disable its caches. Current PC architectures assume that bus-master accesses to main memory must be cache-coherent. So, when bus masters are busy touching memory, the CPU power management software can detect this activity over time and inhibit the transition of the CPU into its lowest power savings mode. USB controllers are bus-masters and the frequency at which they access their memory-based schedules keeps the CPU power management software from placing the CPU into its lowest power savings state.

USB Host controllers don't access main memory when they are suspended. However, there are a variety of reasons why placing the USB controllers into suspend won't work, but they are beyond the scope of this document. The base requirement is that the USB controller needs to be kept out of main memory, while at the same time, the USB bus is kept from going into suspend.

EHCI controllers provide a large-grained mechanism that can be manipulated by system software to change the memory access pattern of the host controller. System software can manipulate the schedule enable bits in the USBCMD register to turn on/off the scheduling traversal. A software heuristic can be applied to implement an on/off duty cycle that allows the USB to make reasonable progress and allow the CPU power management to get the CPU into its lowest power state. This method is not intended to be applied at all times to throttle USB, but should only be applied in very specific configurations and usage loads. For example, when only a keyboard or mouse is attached to the USB, the heuristic could detect times

when the USB is attempting to move data only very infrequently and can adjust the duty cycle to allow the CPU to reach it's low power state for longer periods of time. Similarly, it could detect increases in the USB load and adjust the duty cycle appropriately, even to the point where the schedules are never disabled. The assumption here is that the USB is moving data and the CPU will be required to process the data streams.

It is suggested that in order to provide a complete solution for the system, the companion host controllers should also provide a similar method to allow system software to inhibit the companion host controller from accessing it's shared memory based data structures (schedule lists or otherwise).

32.11.16 Port Test Modes

EHCI host controllers must implement the port test modes Test J_State, Test K_State, Test_Packet, Test Force_Enable, and Test SE0_NAK as described in the USB Specification Revision 2.0. The system is only allowed to test ports that are owned by the EHCI controller (for example, *CF-bit* is a one and *PortOwner* bit is a zero). System software is allowed to have at most one port in test mode at a time. Placing more than one port in test mode will yield undefined results. The required, per port test sequence is (assuming the *CF-bit* in the CONFIGFLAG register is a one):

- Disable the periodic and asynchronous schedules by setting the *Asynchronous Schedule Enable* and *Periodic Schedule Enable* bits in the USBCMD register to a zero.
- Place all enabled root ports into the suspended state by setting the *Suspend* bit in each appropriate PORTSC register to a one.
- Set the *Run/Stop* bit in the USBCMD register to a zero and wait for the *HCHalted* bit in the USBSTS register, to transition to a one. Note that an EHCI host controller implementation may optionally allow port testing with the *Run/Stop* bit set to a one. However, all host controllers must support port testing with *Run/Stop* set to a zero and *HCHalted* set to a one.
- Set the *Port Test Control* field in the port under test PORTSC register to the value corresponding to the desired test mode. If the selected test is Test_Force_Enable, then the *Run/Stop* bit in the USBCMD register must then be transitioned back to one to enable transmission of SOFs out of the port under test.
- When the test is complete, system software must ensure the host controller is halted (*HCHalted* bit is a one) then it terminates and exits test mode by setting *HCReset* to a one.

32.11.17 Interrupts

The EHCI Host Controller hardware provides interrupt capability based on a number of sources. There are several general groups of interrupt sources:

- Interrupts as a result of executing transactions from the schedule (success and error conditions)
- Host controller events (Port change events, and so on)
- Host Controller error events

All transaction-based sources are maskable through the Host Controller’s Interrupt Enable register (USBINTR, see Section USBINTR). Additionally, individual transfer descriptors can be marked to generate an interrupt on completion. This section describes each interrupt source and the processing that occurs in response to the interrupt.

During normal operation, interrupts may be immediate or deferred until the next interrupt threshold occurs. The interrupt threshold is a tunable parameter via the *Interrupt Threshold Control* field in the USBCMD register. The value of this register controls when the host controller will generate an interrupt on behalf of normal transaction execution. When a transaction completes during an interrupt interval period, the interrupt signaling the completion of the transfer will not occur until the interrupt threshold occurs. For example, the default value is eight micro-frames. This means that the host controller will not generate interrupts any more frequently than once every eight micro-frames.

Section Host System Error details effects of a host system error.

If an interrupt has been scheduled to be generated for the current interrupt threshold interval, the interrupt is not signaled until after the status for the last complete transaction in the interval has been written back to host memory. This may sometimes result in the interrupt not being signaled until the next interrupt threshold.

Initial interrupt processing is the same, regardless of the reason for the interrupt. When an interrupt is signaled by the hardware, CPU control is transferred to host controller's USB interrupt handler. The precise mechanism to accomplish the transfer is OS specific. For this discussion it is just assumed that control is received. When the interrupt handler receives control, its first action is to reads the USBSTS (USB Status Register). It then acknowledges the interrupt by clearing all of the interrupt status bits by writing ones to these bit positions. The handler then determines whether the interrupt is due to schedule processing or some other event. After acknowledging the interrupt, the handler (via an OS-specific mechanism), schedules a deferred procedure call (DPC) which will execute later. The DPC routine processes the results of the schedule execution. The precise mechanisms used are beyond the scope of this document.

Note: the host controller is not required to de-assert a currently active interrupt condition when software sets the interrupt enables (in the USBINR register, see Section USBINTR) to a zero. The only reliable method software should use for acknowledging an interrupt is by transitioning the appropriate status bits in the USBSTS register (Section USBSTS) from a one to a zero.

32.11.17.1 Transfer/Transaction Based Interrupts

These interrupt sources are associated with transfer and transaction progress. They are all dependent on the next interrupt threshold.

32.11.17.1.1 Transaction Error

A transaction error is any error that caused the host controller to think that the transfer did not complete successfully. Table 42 Summary of Transaction Errors42 lists the events/responses that the host can observe as a result of a transaction. The effects of the error counter and interrupt status are summarized in the following paragraphs. Most of these errors set the *XactErr* status bit in the appropriate interface data structure.

There is a small set of protocol errors that relate only when executing a queue head and fit under the umbrella of a WRONG PID error that are significant to explicitly identify. When these errors occur, the *XactErr* status bit in the queue head is set and the *CErr* field is decremented. When the *PIDCode* indicates a SETUP, the following responses are protocol errors and result in *XactErr* bit being set to a one and the *CErr* field being decremented.

- *EPS* field indicates a high-speed device and it returns a Nak handshake to a SETUP.
- *EPS* field indicates a high-speed device and it returns a Nyet handshake to a SETUP.
- *EPS* field indicates a low- or full-speed device and the complete-split receives a Nak handshake.

Table 32-86. Summary of Transaction Errors

Event / Result	Queue Head/qTD/iTD/siTD Side-effects		USB Status Register (USBSTS)
	Cerr	Status Field	USBERRINT
CRC	-1	XactErr set to a one.	1[1]
Timeout	-1	XactErr set to a one.	1[1]
Bad PID[2]	-1	XactErr set to a one.	1[1]
Babble	N/A	Section Serial Bus Babble	1
Buffer Error	N/A	Section Data Buffer Error	

1 If occurs in a queue head, then *USBERRINT* is asserted only when *CErr* counts down from a one to a zero. In addition the queue is halted, see Section Halting a Queue Head .

2 The host controller received a response from the device, but it could not recognize the PID as a valid PID.

32.11.17.1.2 Serial Bus Babble

When a device transmits more data on the USB than the host controller is expecting for this transaction, it is defined to be babbling. In general, this is called a *Packet Babble*. When a device sends more data than the *Maximum Length* number of bytes, the host controller sets the *Babble Detected* bit to a one and halts the endpoint if it is using a queue head (see Section Halting a Queue Head). *Maximum Length* is defined as the minimum of *Total Bytes to Transfer* and *Maximum Packet Size*. The *CErr* field is not decremented for a packet babble condition (only applies to queue heads). A babble condition also exists if IN transaction is in progress at High-speed EOF2 point. This is called a frame babble. A frame babble condition is recorded into the appropriate schedule data structure. In addition, the host controller must disable the port to which the frame babble is detected.

The *USBERRINT* bit in the USBSTS register is set to a one and if the *USB Error Interrupt Enable* bit in the USBINTR register is a one, then a hardware interrupt is signaled to the system at the next interrupt threshold. The host controller must never start an OUT transaction that will babble across a micro-frame EOF.

NOTE

When a host controller detects a data PID mismatch, it must either: disable the packet babble checking for the duration of the bus transaction or do packet babble checking based solely on *Maximum Packet Size*. The USB core specification defines the requirements on a data receiver when it receives a data PID mismatch (for example, expects a DATA0 and gets a DATA1 or visa-versa). In summary, it must ignore the received data and respond with an ACK handshake to advance the transmitter's data sequence.

The EHCI interface allows System software to provide buffers for a Control, Bulk or Interrupt IN endpoint that are not an even multiple of the maximum packet size specified by the device. Whenever a device misses an ACK for an IN endpoint, the host and device are out of synchronization with respect to the progress of the data transfer. The host controller may have advanced the transfer to a buffer that is less than maximum packet size. The device will re-send its maximum packet size data packet, with the original data PID, in response to the next IN token. To properly manage the bus protocol, the host controller must disable the packet babble check when it observes the data PID mismatch.

32.11.17.1.3 Data Buffer Error

This event indicates that an overrun of incoming data or a underrun of outgoing data has occurred for this transaction. This would generally be caused by the host controller not being able to access required data buffers in memory within necessary latency requirements. These conditions are not considered transaction errors, and do not effect the error count in the queue head. When these errors do occur, the host controller records the fact the error occurred by setting the *Data Buffer Error* bit in the queue head, iTD or siTD.

If the data buffer error occurs on a non-isochronous IN, the host controller will not issue a handshake to the endpoint. This will force the endpoint to resend the same data (and data toggle) in response to the next IN to the endpoint.

If the data buffer error occurs on an OUT, the host controller must corrupt the end of the packet so that it cannot be interpreted by the device as a good data packet. Simply truncating the packet is not considered acceptable. An acceptable implementation option is to 1's complement the CRC bytes and send them. There are other options suggested in the Transaction Translator section of the USB Specification Revision 2.0.

32.11.17.1.4 USB Interrupt (Interrupt on Completion (IOC)

Transfer Descriptors (iTDs, siTDs, and queue heads (qTDs)) contain a bit that can be set to cause an interrupt on their completion. The completion of the transfer associated with that schedule item causes the USB Interrupt (USBINT) bit in the USBSTS register to be set to a one. In addition, if a short packet is encountered on an IN transaction associated with a queue head, then this event also causes USBINT to be set to a one. If the USB Interrupt Enable bit in the USBINTR register is set to a one, a hardware interrupt is signaled to the system at the next interrupt threshold. If the completion is because of errors, the *USBERRINT* bit in the USBSTS register is also set to a one.

32.11.17.1.5 Short Packet

Reception of a data packet that is less than the endpoint's Max Packet size during Control, Bulk or Interrupt transfers signals the completion of the transfer. Whenever a short packet completion occurs during a queue head execution, the *USBINT* bit in the USBSTS register is set to a one. If the *USB Interrupt Enable* bit is set in the USBINTR register, a hardware interrupt is signaled to the system at the next interrupt threshold.

32.11.17.2 Host Controller Event Interrupts

These interrupt sources are independent of the interrupt threshold (with the one exception being the Interrupt on Async Advance, see Section Interrupt on Async Advance).

32.11.17.2.1 Port Change Events

Port registers contain status and status change bits. When the status change bits are set to a one, the host controller sets the *Port Change Detect* bit in the USBSTS register to a one. If the *Port Change Interrupt Enable* bit in the USBINTR register is a one, then the host controller will issue a hardware interrupt. The port status change bits include:

- Connect Status Change
- Port Enable/Disable Change
- Over-current Change
- Force Port Resume

32.11.17.2.2 Frame List Rollover

This event indicates that the host controller has wrapped the frame list. The current programmed size of the frame list effects how often this interrupt occurs. If the frame list size is 1024, then the interrupt will occur every 1024 milliseconds, if it is 512, then it will occur every 512 milliseconds, and so on. When a frame list rollover is detected, the host controller sets the *Frame List Rollover* bit in the USBSTS register to a one. If the *Frame List Rollover Enable* bit in the USBINTR register is set to a one, the host controller issues a hardware interrupt. This interrupt is not delayed to the next interrupt threshold.

32.11.17.2.3 Interrupt on Async Advance

This event is used for deterministic removal of queue heads from the asynchronous schedule. Whenever the host controller advances the on-chip context of the asynchronous schedule, it evaluates the value of the *Interrupt on Async Advance Doorbell* bit in the USBCMD register. If it is a one, it sets the *Interrupt on Async Advance* bit in the USBSTS register to a one. If the *Interrupt on Async Advance Enable* bit in the USBINTR register is a one, the host controller issues a hardware interrupt at the next interrupt threshold. A detailed explanation of this feature is described in Section Removing Queue Heads from Asynchronous Schedule .

32.11.17.2.4 Host System Error

The host controller is a bus master and any interaction between the host controller and the system may experience errors. The type of host error may be catastrophic to the host controller (such as a Master Abort) making it impossible for the host controller to continue in a coherent fashion. In the presence of non-catastrophic host errors, such as parity errors, the host controller could potentially continue operation. The recommended behavior for these types of errors is to escalate it to a catastrophic error and halt the host controller. Host-based error must result in the following actions:

- The *Run/Stop* bit in the USBCMD register is set to a zero.
- The following bits in the USBSTS register are set:
  - *Host System Error* bit is to a one.

— *HCHalted* bit is set to a one.

- If the *Host System Error Enable* bit in the USBINTR register is a one, then the host controller will issue a hardware interrupt. This interrupt is not delayed to the next interrupt threshold. Table 43 Summary Behavior of EHCI Host Controller on Host System Errors43 summarizes the required actions taken on the various host errors.

Table 32-87. Table 43 Summary Behavior of EHCI Host Controller on Host System Errors

Cycle Type	Master Abort	Target Abort	Data Phase Parity
Frame list pointer fetch (read)	Fatal	Fatal	Fatal [o]
siTD fetch (read)	Fatal	Fatal	Fatal [o]
siTD status write-back (write)	Fatal [o]	Fatal [o]	Fatal [o]
iTD fetch (read)	Fatal	Fatal	Fatal [o]
iTD status write-back (write)	Fatal [o]	Fatal [o]	Fatal [o]
qTD fetch (read)	Fatal	Fatal	Fatal [o]
qHD status write-back (write)	Fatal [o]	Fatal [o]	Fatal [o]
Data write	Fatal [o]	Fatal [o]	Fatal [o]
Data read	Fatal	Fatal	Fatal [o]

[o] Potentially, a host controller implementation could continue operation without a halt. However, the recommended behavior is to halt the host controller.

NOTE

After a *Host System Error*, Software must reset the host controller via *HCReset* in the USBCMD register before re-initializing and restarting the host controller.

32.12 EHCI Deviation

For the purposes a dual-role Host/Device controller with support for On-The-Go applications, it is necessary to deviate from the EHCI specification Enhanced Host Controller Interface Specification for Universal Serial Bus, Revision 0.95, November 2000, Intel Corporation. http://www.intel.com. Device operation & On-The-Go operation is not specified in the EHCI and thus the implementation supported in this core is proprietary. The host mode operation of the core is near EHCI compatible with few minor differences documented in this section.

The particulars of the deviations occur in the areas summarized here:

- Embedded Transaction Translator—Allows direct attachment of FS and LS devices in host mode without the need for a companion controller.

- Device operation - In host mode the device operational registers are generally disabled and thus device mode is mostly transparent when in host mode. However, there are a couple exceptions documented in the following sections.
- Embedded design interface - This core does not a PCI Interface and therefore the PCI configuration registers described in the EHCI specification are not applicable.
- On-The-Go Operation - This design includes an On-The-Go controller for Port #1.

32.12.1 Embedded Transaction Translator Function

The ARC USB High-Speed On-The-Go controller supports directly connected full and low speed devices without requiring a companion controller by including the capabilities of a USB 2.0 high speed hub transaction trasnslator. Although there is no separate Tranaction Translator block in the system, the transaction translator function normally associated with a high speed hub has been implemented within the DMA and Protocol engine blocks. The embedded transaction translator fuction is an extension to EHCI interface, but makes use of the standard data structures and operational models that exist in the EHCI specification to support full and low speed devices.

32.12.1.1 Capability Registers

The following additions have been added to the capability registers to support the embedded Transaction Translator Function:

- N_TT added to HCSPARAMS—Host Control Structural Parameters
- N_PTT added to HCSPARAMS—Host Control Structural Parameters

See Section 32.9.4.3, “HCSPARAMS—EHCI Compliant with Extensions,” with extensions for usage information.

32.12.1.2 Operational Registers

The following additions have been added to the operational registers to support the embedded TT:

- It is a new register.
- It is an addition of a two-bit Port Speed (PSPD) to the PORTSCx register.

32.12.1.3 Discovery

In a standard EHCI controller design, the EHCI host controller driver detects a Full speed (FS) or Low speed (LS) device by noting if the port enable bit is set after the port reset operation. The port enable will only be set in a standard EHCI controller implementation after the port reset operation and when the host and device negotiate a High-Speed connection (for example, Chirp completes successfully).

Since this controller has an embedded Transaction Translator, the port enable will always be set after the port reset operation regardless of the result of the host device chirp result and the resulting port speed will be indicated by the PSPD field in PORTSCx.

Therefore, the standard EHCI host controller driver requires an alteration to handle directly connected Full and Low speed devices or hubs.

The change is a fundamental one in that is summarized in this table:

Table 32-88. Speed Detection Translation

Standard EHCI	EHCI with Embedded Transaction Translator
After port enable bit is set following a connection and reset sequence, the device/hub is assumed to be HS.	After port enable bit is set following a connection and reset sequence, the device/hub speed is noted from PORTSCx.
FS and LS devices are assumed to be downstream from a HS hub thus, all port-level control is performed through the Hub Class to the nearest Hub.	FS and LS device can be either downstream from a HS hub or directly attached. When the FS/LS device is downstream from a HS hub, then port-level control is done using the Hub Class through the nearest Hub. When a FS/LS device is directly attached, then port-level control is accomplished using PORTSCx.
FS and LS devices are assumed to be downstream from a HS hub with HubAddr=X. [where HubAddr > 0 and HubAddr is the address of the Hub where the bus transitions from HS to FS/LS (ie. Split target hub)]	FS and LS device can be either downstream from a HS hub with HubAddr = X [HubAddr > 0] or directly attached [where HubAddr = 0 and HubAddr is the address of the Root Hub where the bus transitions from HS to FS/LS (ie. Split target hub is the root hub)]

32.12.1.4 Data Structures

The same data structures used for FS/LS transactions though a HS hub are also used for transactions through the Root Hub with sm embedded Transaction Translator. Here it is demonstrated how the Hub Address and Endpoint Speed fields should be set for directly attached FS/LS devices and hubs:

1. QH (for direct attach FS/LS)—Async. (Bulk/Control Endpoints) Periodic (Interrupt)
   - — Hub Address = 0
   - — Transactions to direct attached device/hub.
     - – QH.EPS = Port Speed
   - — Transactions to a device downstream from direct attached FS hub.
     - – QH.EPS = Downstream Device Speed

     NOTE

     When QH.EPS = 01 (LS) and PORTSCx.PSPD = 00 (FS), a LS-pre-pid will be sent before the transmitting LS traffic.

   - — Maximum Packet Size must be less than or equal 64 or undefined behaviour may result.
2. siTD (for direct attach FS)—Periodic (ISO Endpoint)
   - — All FS ISO transactions:
     - – Hub Address = 0
     - – siTD.EPS = 00 (full speed)
       - Maximum Packet Size must less than or equal to 1023 or undefined behaviour may result.

32.12.1.5 Operational Model

The operational models are well defined for the behavior of the Transaction Translator (see USB 2.0 specification Universal Serial Bus Specification, Revision 2.0, April 2000, Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC, Philips. http://www.usb.org) and for the EHCI controller moving packets between system memory and a USB-HS hub. Since the embedded Transaction Translator exists within the host controller there is no physical bus between EHCI host controller driver and the USB FS/LS bus. These sections will briefly discuss the operational model for how the EHCI and Transaction Translator operational models are combined without the physical bus between. The following sections assume the reader is familiar with both the EHCI and USB 2.0 Transaction Translator operational models.

32.12.1.5.1 Micro-Frame Pipeline

The EHCI operational model uses the concept of H-frames and B-frames to describe the pipeline between the Host (H) and the Bus (B). The embedded Transaction Translator shall use the same pipeline algorithms specified in the USB 2.0 specification for a Hub-based Transaction Translator.

All periodic transfers always begin at B-frame 0 (after SOF) and continue until the stored periodic transfers are complete. As an example of the micro-frame pipeline implemented in the embedded Transaction Translator, all periodic transfers that are tagged in EHCI to execute in H-frame 0 will be ready to execute on the bus in B-frame 0.

It is important to note that when programming the S-mask and C-masks in the EHCI data structures to schedule periodic transfers for the embedded Transaction Translator, the EHCI host controller driver must follow the same rules specified in EHCI for programming the S-mask and C-mask for downstream Hub-based Transaction Translators.

Once periodic transfers are exhausted, any stored asynchronous transfer will be moved. Asynchronous transfers are opportunistic in that they shall execute whenever possible and their operation is not tied to H-frame and B-frame boundaries with the exception that an asynchronous transfer can not babble through the SOF (start of B-frame 0.)

32.12.1.5.2 Split State Machines

The start and complete split operational model differs from EHCI slightly because there is no bus medium between the EHCI controller and the embedded Transaction Translator. Where a start or complete-split operation would occur by requesting the split to the HS hub, the start/complete split operation is simple an internal operation to the embedded Transaction Translator. The following table summarizes the conditions where handshakes are emulated from internal state instead of actual handshakes to HS split bus traffic.

Table 32-89. Split State Handshake Conditions

Condition	Emulate TT Response
Start-Split: All asynchronous buffers full.	NAK
Start-Split: All periodic buffers full.	ERR
Start-Split: Success for start of Async. Transaction.	ACK

Table 32-89. Split State Handshake Conditions (continued)

Condition	Emulate TT Response
Start-Split: Start Periodic Transaction.	No Handshake (Ok)
Complete-Split: Failed to find transaction in queue.	Bus Time Out
Complete-Split: Transaction in Queue is Busy.	NYET
Complete-Split: Transaction in Queue is Complete.	[Actual Handshake from LS/FS device]

Note: The un-shaded cells represent Start-Splits and the shaded cells represent Complete-Splits.

32.12.1.5.3 Asynchronous Transaction Scheduling and Buffer Management

The following USB 2.0 specification items are implemented in the embedded Transaction Translator:

USB 2.0—11.17.3

- Sequencing is provided & a packet length estimator ensures no full-speed/low-speed packet babbles into SOF time.

USB 2.0—11.17.4

- Transaction tracking for 2 data pipes.

USB 2.0—11.17.5

- Clear_TT_Buffer capability provided though the use of the register.

32.12.1.5.4 Periodic Transaction Scheduling and Buffer Management

The following USB 2.0 specification items are implemented in the embedded Transaction Translator:

USB 2.0—11.18.6.[1-2]

- Abort of pending start-splits
  - — EOF (and not started in micro-frames 6)
  - — Idle for more than 4 micro-frames
- Abort of pending complete-splits
  - — EOF
  - — Idle for more than 4 micro-frames

USB 2.0—11.18.[7-8]

- Transaction tracking for up to 16 data pipes.
- Some applications may not require transaction tracking up to a maximum of 16 periodic data pipes. The option to limit the tracking to only 4 periodic data pipes exists in the by changing the configuration constant VUSB_HS_TT_PERIODIC_CONTEXTS to 4. The result is a significant gate count savings to the core given the limitations implied.

NOTE

Limiting the number of tracking pipes in the EMBedded –TT to four (4) will impose the restriction that no more than 4 periodic transactions (INTERRUPT/ISOCHRONOUS) can be scheduled through the embedded-tt per frame. the number 16 was chosen in the USB specification because it is sufficient to ensure that the high-speed to full-speed periodic pipeline can remain full. keeping the pipeline full puts no constraint on the number of periodic transactions that can be scheduled in a frame and the only limit becomes the flight time of the packets on the bus.

- Complete-split transaction searching.

NOTE

There is no data schedule mechanism for these transactions other than the micro-frame pipeline. The embedded TT assumes the number of packets scheduled in a frame does not exceed the frame duration (1 ms) or else undefined behavior may result.

32.12.1.5.5 Multiple Transaction Translators

The maximum number of embedded Transaction Translators that is currently supported is one as indicated by the N_TT field in the HCSPARAMS—Host Control Structural Parameters register.

32.12.2 Device Operation

The co-existence of a device operational controller within the host controller has little effect on EHCI compatibility for host operation except as noted in this section.

32.12.2.1 USBMODE Register

Given that the dual-role controller is initialized in neither host nor device mode, the USBMODE register must be programmed for host operation before the EHCI host controller driver can begin EHCI host operations.

32.12.2.2 Non-Zero Fields in the Register File

Some of the reserved fields and reserved addresses in the capability registers and operational register have use in device mode, the following must be adhered to:

- Write operations to all EHCI reserved fields (some of which are device fields) with the operation registers should always be written to zero. This is an EHCI requirement of the device controller driver that must be adhered to.
- Read operations by the host controller must properly mask EHCI reserved fields (some of which are device fields) because fields that are used exclusive for device are undefined in host mode .

32.12.2.3 SOF Interrupt

This SOF Interrupt used for device mode is shared as a free running 125us interrupt for host mode. EHCI does not specify this interrupt but it has been added for convenience and as a potential software time base. See USBSTS and USBINTR registers.

32.12.3 Embedded Design Interface

This is an Embedded USB Host Controller as defined by the EHCI specification and thus does not implement the PCI configuration registers.

32.12.3.1 Frame Adjust Register

Given that the optional PCI configuration registers are not included in this implementation, there is no corresponding bit level timing adjustments like is provided by the Frame Adjust register in the PCI configuration registers. Starts of micro-frames are timed precisely to 125 us using the transceiver clock as a reference clock. for example, 60 Mhz transceiver clock for 8-bit physical interfaces & full-speed serial interfaces or 30 Mhz transceiver clock for 16-bit physical interfaces.

32.12.4 Miscellaneous variations from EHCI

32.12.4.1 Programmable Physical Interface Behavior

This design supports multiple Physical interfaces which can operate in differing modes when the core is configured with software programmable Physical Interface Modes. Software programmability allows the selection of the Physical interface part during the board design phase instead of during the chip design phase. The control bits for selecting the Physical Interface operating mode have been added to the PORTSCx register providing a capability that is not defined by EHCI.

32.12.4.2 Discovery

32.12.4.2.1 Port Reset

The port connect methods specified by EHCI require setting the port reset bit in the PORTSCx register for a duration of 10 ms. Due to the complexity required to support the attachment of devices that are not high speed there are counter already present in the design that can count the 10ms reset pulse to alleviate the requirement of the software to measure this duration. Therefore, the basic connection is then summarized as the following:

- [Port Change Interrupt] Port connect change occurs to notify the host controller driver that a device has attached.
- Software shall write a ‘1’ to the reset the device.
- Software shall write a ‘0’ to the reset the device after 10 ms.
  - This step, which is necessary in a standard EHCI design, may be omitted with this implementation. Should the EHCI host controller driver attempt to write a ‘0’ to the reset bit

while a reset is in progress the write will simple be ignored and the reset will continue until completion.

- [Port Change Interrupt] Port enable change occurs to notify the host controller that the device in now operational and at this point the port speed has been determined.

32.12.4.2.2 Port Speed Detection

After the port change interrupt indicates that a port is enabled, the EHCI stack should determine the port speed. Unlike the EHCI implementation which will re-assign the port owner for any device that does not connect at High-Speed, this host controller supports direct attach of non High-Speed devices. Therefore, the following differences are important regarding port speed detection:

- Port Owner is read-only and always reads 0.
- A 2-bit Port Speed indicator has been added to PORTSC to provide the current operating speed of the port to the host controller driver.
- A 1-bit High Speed indicator has been added to PORTSC to signify that the port is in High-Speed vs. Full/Low Speed.

32.12.4.3 Port Test Mode

Port Test Control mode behaves fully as described in EHCI since the release of revision 3.2.1. In earlier product revisions, the test packet mode was not EHCI compatible. An alternate host controller driver procedure is no longer necessary or supported.

32.13 Device Data Structures

This section defines the interface data structures used to communicate control, status, and data between Device Controller Driver (DCD) Software and the Device Controller. The data structure definitions in this chapter support a 32-bit memory buffer address space. The interface consists of device Queue Heads and Transfer Descriptors.

NOTE

Software must ensure that no interface data structure reachable by the Device Controller spans a 4K-page boundary.

The data structures defined in the chapter are (from the device controller's perspective) a mix of read-only and read/ writeable fields. The device controller must preserve the read-only fields on all data structure writes.

The ARC USB-HS OTG High-Speed USB On-The-Go core includes DCD Software called the USB 2.0 Device API. The Device API provides an easy to use Application Program Interface for developing device (peripheral) applications using the ARC USB-HS OTG High-Speed USB On-The-Go core. The Device API incorporates and abstracts for the application developer all of the elements of the program interface.

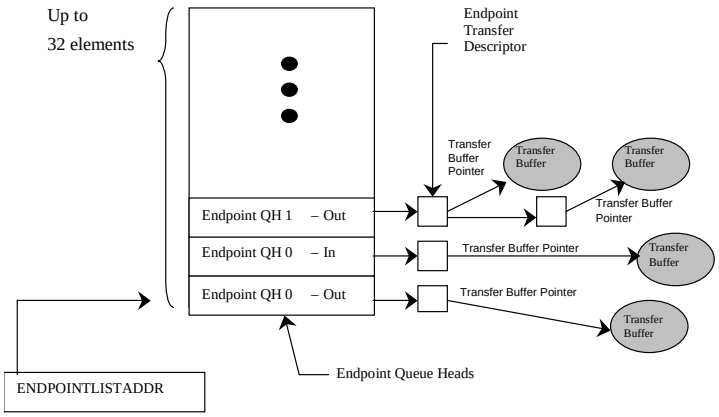


Figure 32-78. End Point Queue Head Organization

Device queue heads are arranged in an array in a continuous area of memory pointed to by the *ENDPOINTLISTADDR* pointer. The even –numbered device queue heads in the list support receive endpoints (OUT/SETUP) and the odd-numbered queue heads in the list are used for transmit endpoints (IN/INTERRUPT). The device controller will index into this array based upon the endpoint number received from the USB bus. All information necessary to respond to transactions for all primed transfers is contained in this list so the Device Controller can readily respond to incoming requests without having to traverse a linked list.

NOTE

The Endpoint Queue Head List must be aligned to a 2k boundary.

32.13.1 Endpoint Queue Head (dQH)

The device Endpoint Queue Head (dQH) is where all transfers are managed. The dQH is a 48-byte data structure, but must be aligned on 64-byte boundaries. During priming of an endpoint, the dTD (device transfer descriptor) is copied into the overlay area of the dQH, which starts at the nextTD pointer DWord and continues through the end of the buffer pointers DWords. After a transfer is complete, the dTD status DWord is updated in the dTD pointed to by the currentTD pointer. While a packet is in progress, the overlay area of the dQH is used as a staging area for the dTD so that the Device Controller can access needed information with little minimal latency.

Table 32-90. dQH Register Summary

31 30	29	28 27	26 25 24 23 22 21 20 19 18 17 16	15	14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Mult	zl	0	Maximum Packet Length	io	0
Current dTD Pointer					0
Next dTD Pointer					0 T
0	Total Bytes			io	0 MultO 0 Status
Buffer Pointer (Page 0)					Current Offset

Table 32-90. dQH Register Summary (continued)

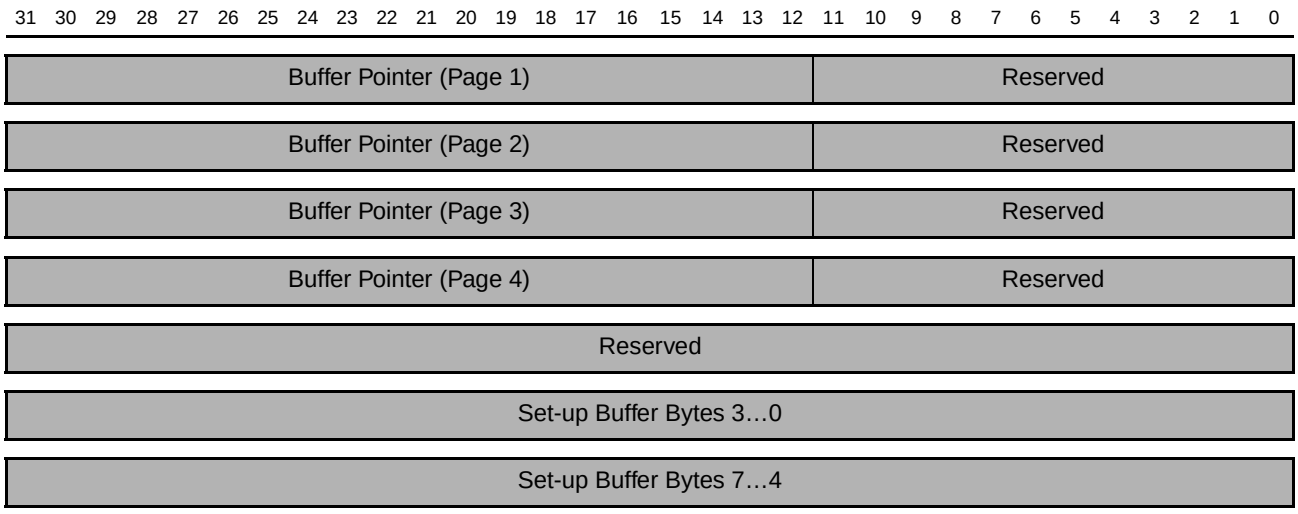


Figure 32-79. Endpoint Queue Head (dQH)

	Device Controller Read/Write		Device Controller Read Only.

32.13.1.1 Endpoint Capabilities/Characteristics

This DWord specifies static information about the endpoint, in other words, this information does not change over the lifetime of the endpoint. Device Controller software should not attempt to modify this information while the corresponding endpoint is enabled.

Table 32-91. Endpoint Capabilities/Characteristics

Field	Description
31:30	Mult. This field is used to indicate the number of packets executed per transaction description as given by the following: 00 Execute N Transactions as demonstrated by the USB variable length packet protocol where N is computed using the Maximum Packet Length (dQH) and the Total Bytes field (dTD) 01 Execute 1 Transaction. 10 Execute 2 Transactions. 11 Execute 3 Transactions. **Note:** Non-ISO endpoints must set Mult=”00”. ISO endpoints must set Mult=”01”, “10”, or “11” as needed.
29	Zero Length Termination Select. This bit is used to indicate when a zero length packet is used to terminate transfers where to total transfer length is a multiple . This bit is not relevant for Isochronous 0 Enable zero length packet to terminate transfers equal to a multiple of the Maximum Packet Length. (default). 1 Disable the zero length packet on transfers that are equal in length to a multiple Maximum Packet Length.
28:27	Reserved. These bit reserved for future use and should be set to zero.

Table 32-91. Endpoint Capabilities/Characteristics (continued)

Field	Description
26:16	Maximum Packet Length. This directly corresponds to the maximum packet size of the associated endpoint (wMaxPacketSize). The maximum value this field may contain is 0x400 (1024).
15	Interrupt On Setup (IOS). This bit is used on control type endpoints to indicate if USBINT is set in response to a setup being received.
14:0	Reserved. Bits reserved for future use and should be set to zero.

32.13.1.2 Transfer Overlay

The seven DWords in the overlay area represent a transaction working space for the device controller. The general operational model is that the device controller can detect whether the overlay area contains a description of an active transfer. If it does not contain an active transfer, then it will not read the associated endpoint.

After an endpoint is readied, the dTD will be copied into this queue head overlay area by the device controller. Until a transfer is expired, software must not write the queue head overlay area or the associated transfer descriptor. When the transfer is complete, the device controller will write the results back to the original transfer descriptor and advance the queue.

See dTD for a description of the overlay fields.

32.13.1.3 Current dTD Pointer

The current dTD pointer is used by the device controller to locate the transfer in progress. This word is for Device Controller (hardware) use only and should not be modified by DCD software.

Table 32-92. Next dTD Pointer

Field	Description
31–5	Current dTD. This field is a pointer to the dTD that is represented in the transfer overlay area. This field will be modified by the Device Controller to next dTD pointer during endpoint priming or queue advance.
4–0	Reserved. Bit reserved for future use and should be set to zero.

32.13.1.4 Set-up Buffer

The set-up buffer is dedicated storage for the 8-byte data that follows a set-up PID.

NOTE

Each endpoint has a TX and an RX dQH associated with it, and only the RX queue head is used for receiving setup data packets.

Table 32-93. Multiple Mode Control (HCCPARAMS)

DWord	Bits	Description
1	31–0	Setup Buffer 0. This buffer contains bytes 3 to 0 of an incoming setup buffer packet and is written by the device controller to be read by software.
2	31–0	Setup Buffer 1. This buffer contains bytes 7 to 4 of an incoming setup buffer packet and is written by the device controller to be read by software.

32.13.2 Endpoint Transfer Descriptor (dTD)

The dTD describes to the device controller the location and quantity of data to be sent/received for given transfer. The DCD should not attempt to modify any field in an active dTD except the Next Like Pointer, which should only be modified as described in Section 32.14.5, "Managing Transfers with Transfer Descriptors."

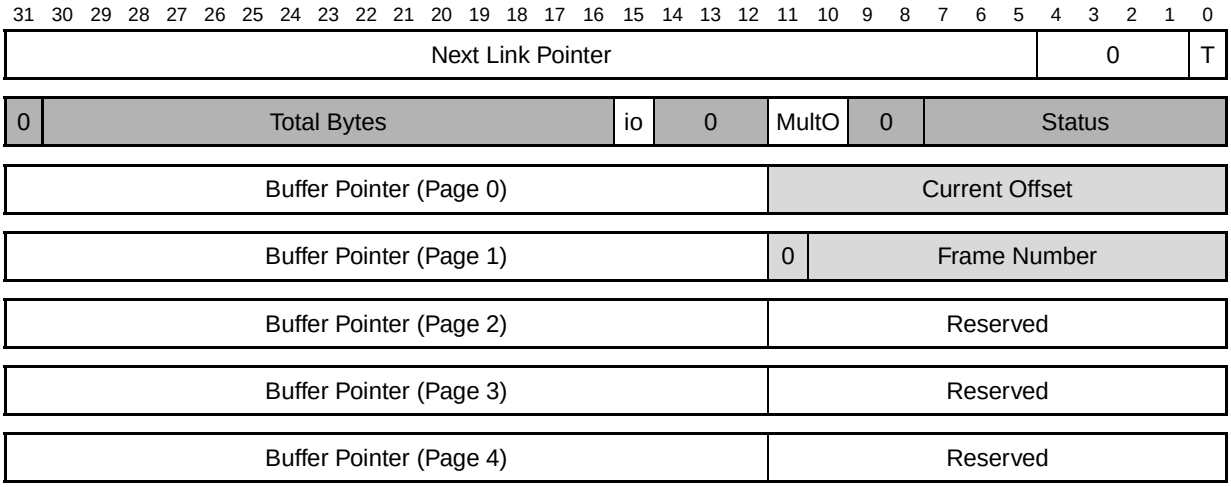


Figure 32-80. dTD Register Examples

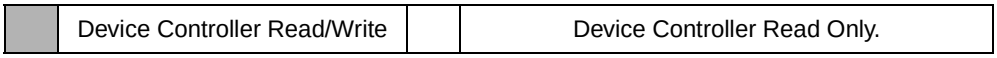


Figure 32-81. Endpoint Transfer Descriptor (dTD)

Table 32-94. Next dTD Pointer

Field	Description
31–5	Next Transfer Element Pointer. This field contains the physical memory address of the next dTD to be processed. The field corresponds to memory address signals [31:5], respectively.
4–1	Reserved. Bits reserved for future use and should be set to zero.
0	Terminate (T). 1=pointer is invalid. 0=Pointer is valid (points to a valid Transfer Element Descriptor). This bit indicates to the Device Controller that there are no more valid entries in the queue.

Table 32-95. dTD Token

Field	Description
31	Reserved. Bit reserved for future use and should be set to zero.
30–16	Total Bytes. This field specifies the total number of bytes to be moved with this transfer descriptor. This field is decremented by the number of bytes actually moved during the transaction and only on the successful completion of the transaction. The maximum value software may store in the field is 5*4K(5000H). This is the maximum number of bytes 5 page pointers can access. Although it is possible to create a transfer up to 20K this assumes the 1st offset into the first page is 0. When the offset cannot be predetermined, crossing past the 5th page can be guaranteed by limiting the total bytes to 16K**. Therefore, the maximum recommended transfer is 16K(4000H). If the value of the field is zero when the host controller fetches this transfer descriptor (and the active bit is set), the device controller executes a zero-length transaction and retires the transfer descriptor. It is not a requirement for IN transfers that Total Bytes To Transfer be an even multiple of *Maximum Packet Length*. If software builds such a transfer descriptor for an IN transfer, the last transaction will always be less that *Maximum Packet Length*.
15	Interrupt On Complete (IOC). This bit is used to indicate if USBINT is to be set in response to device controller being finished with this dTD.
14–12	Reserved. Bits reserved for future use and should be set to zero.
11–10	Multiplier Override (MultO). This field can be used for transmit ISO's (ie. ISO-IN) to override the multiplier in the QH. This field must be zero for all packet types that are not transmit-ISO. Example: if QH.multiplier = 3; Maximum packet size = 8; Total Bytes = 15; MultiO = 0 [default] Three packets are sent: {Data2(8); Data1(7); Data0(0)} if QH.multiplier = 3; Maximum packet size = 8; Total Bytes = 15; MultiO = 2 Two packets are sent: {Data1(8); Data0(7)} For maximal efficiency, software should compute MultO = greatest integer of (Total Bytes / Max. Packet Size) except for the case when Total Bytes = 0; then MultO should be 1. Note: Non-ISO and Non-TX endpoints must set MultO="00".
9–8	Reserved. Bits reserved for future use and should be set to zero.
7–0	Status. This field is used by the Device Controller to communicate individual command execution states back to the Device Controller software. This field contains the status of the last transaction performed on this qTD. The bit encodings are: 7 Active 6 Halted 5 Data Buffer Error 3 Transaction Error 4,2,0 Reserved

Table 32-96. Table 49 dTD Buffer Page Pointer List

Field	Description
31–12	Buffer Pointer. Selects the page offset in memory for the packet buffer. Non virtual memory systems will typically set the buffer pointers to a series of incrementing integers.
0;11–0	Current Offset. Offset into the 4kb buffer where the packet is to begin.
1;10–0	Frame Number. Written by the device controller to indicate the frame number in which a packet finishes. This is typically be used to correlate relative completion times of packets on an ISO endpoint.

32.14 Device Operational Model

The function of the device operation is to transfer a request in the memory image to and from the Universal Serial Bus. Using a set of linked list transfer descriptors, pointed to by a queue head, the device controller will perform the data transfers. The following sections explain the use of the device controller from the device controller driver (DCD) point-of-view and further describe how specific USB bus events relate to status changes in the device controller programmer's interface.

The ARC USB-HS OTG High-Speed USB On-The-Go is shipped with a DCD called the ARC USB-HS OTG High-Speed USB On-The-Go Device API. The ARC USB-HS OTG High-Speed USB On-The-Go Device API provides an easy to use application interface for developing USB device (peripheral) applications. The ARC USB-HS OTG High-Speed USB On-The-Go Device API incorporates and abstracts for the application developer all of the information contained in the device operational model. For more information on the ARC USB-HS OTG High-Speed USB On-The-Go Device API, refer to the "Software Design document for the Precise USB 2.0 Device API."

32.14.1 Device Controller Initialization

After hardware reset, the device is disabled until the Run/Stop bit is set to a '1'. In the disabled state, the pull-up on the USB D+ is not active which prevents an attach event from occurring. At a minimum, it is necessary to have the queue heads setup for endpoint zero before the device attach occurs. Shortly after the device is enabled, a USB reset will occur followed by setup packet arriving at endpoint 0. A Queue head must be prepared so that the device controller can store the incoming setup packet.

To initialize a device, the software should perform the following steps:

1. Set Controller Mode in the USBMODE register to device mode.
2. Allocate and Initialize device queue heads in system memory.

NOTE

Transitioning from host mode to device mode requires a device controller reset before modifying USBMODE.

— Minimum: Initialize device queue heads 0 Tx & 0 Rx.

— For information on device queue heads, refer to section—Device Data Structures

NOTE

All device queue heads must be initialized for control endpoints must be initialized before the endpoint is enabled. Non-Control device queue heads before the endpoint can be used.

3. Configure ENDPOINTLISTADDR Pointer.
   — For additional information on ENDPOINTLISTADDR, refer to the register table.
4. Enable the microprocessor interrupt associated with the ARC USB High-Speed On-The-Go core.
   — Recommended: enable all device interrupts including: USBINT, USBERRINT, Port Change Detect, USB Reset Received, DCSuspend.
   — For a list of available interrupts refer to the USBINTR and the USBSTS register tables.
5. Set Run/Stop bit to Run Mode.
   — After the Run bit is set, a device reset will occur. The DCD must monitor the reset event and adjust the software state as described in the Bus Reset section of the following Port State and Control section below.

NOTE

Endpoint 0 is designed as a control endpoint only and does not need to be configured using <XRF>ENDPTCTRL0 register.

It is also not necessary to initially prime Endpoint 0 because the first packet received will always be a setup packet. The contents of the first setup packet will require a response in accordance with USB device framework command set.

32.14.2 Port State and Control

From a chip or system reset, the device controller enters the *powered* state. A transition from the *powered* state to the *attach* state occurs when the Run/Stop bit is set to a ‘1’. After receiving a reset on the bus, the port will enter the *defaultFS or defaultHS* state in accordance with the reset protocol described in Appendix C.2 of the USB Specification Rev. 2.0. The following state diagram depicts the state of a USB 2.0 device.

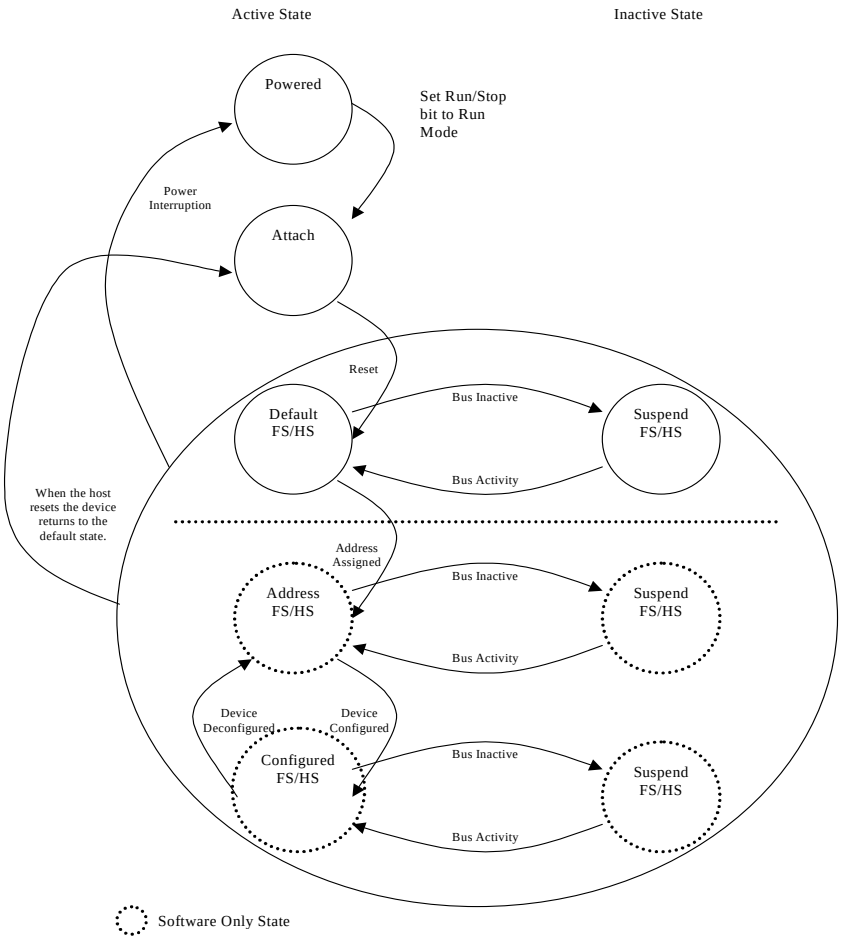


Figure 32-82. Device State Diagram

States *powered*, *attach*, *defaultFS/HS*, *suspendFS/HS* are implemented in the device controller and are communicated to the DCD using the following status bits:

Table 32-97. Device Controller State Information Bits

Bit	Register
DCSuspend	USBSTS
USB Reset Received	USBSTS
Port Change Detect	USBSTS
High-Speed Port	PORTSC

It is the responsibility of the DCD to maintain a state variable to differentiate between the *DefaultFS/HS* state and the *Address/Configured* states. Change of state from *Default* to *Address* and the *Configured* states is part of the enumeration process described in the device framework section of the USB 2.0 Specification.

As a result of entering the *Address* state, the device address register (DEVICEADDR) must be programmed by the DCD.

Entry into the *Configured* indicates that all endpoints to be used in the operation of the device have been properly initialized by programming the ENDPTCTRLx registers and initializing the associated queue heads.

32.14.2.1 Bus Reset

A bus reset is used by the host to initialize downstream devices. When a bus reset is detected, the device controller will renegotiate its attachment speed, reset the device address to 0, and notify the DCD by interrupt (assuming the USB Reset Interrupt Enable is set). After a reset is received, all endpoints (except endpoint 0) are disabled and any primed transactions will be cancelled by the device controller. The concept of priming will be clarified below, but the DCD must perform the following tasks when a reset is received:

Clear all setup token semaphores by reading the ENDPTSETUPSTAT register and writing the same value back to the ENDPTSETUPSTAT register.

Clear all the endpoint complete status bits by reading the ENDPTCOMPLETE register and writing the same value back to the ENDPTCOMPLETE register.

Cancel all primed status by waiting until all bits in the ENDPTPRIME are 0 and then writing 0xFFFFFFFF to ENDPTFLUSH.

Read the reset bit in the PORTSCx register and make sure that it is still active. A USB reset will occur for a minimum of 3 ms and the DCD must reach this point in the reset cleanup before end of the reset occurs, otherwise a hardware reset of the device controller is recommended (rare.)

- A hardware reset can be performed by writing a one to the device controller reset bit in the USBCMD reset. Note: a hardware reset will cause the device to detach from the bus by clearing the Run/Stop bit. Thus, the DCD must completely re-initialize the device controller after a hardware reset.

Free all allocated dTDs because they will no longer be executed by the device controller. If this is the first time the DCD is processing a USB reset event, then it is likely that no dTDs have been allocated.

At this time, the DCD may release control back to the OS because no further changes to the device controller are permitted until a Port Change Detect is indicated.

After a Port Change Detect, the device has reached the default state and the DCD can read the PORTSCx to determine if the device is operating in FS or HS mode. At this time, the device controller has reached normal operating mode and DCD can begin enumeration.

NOTE

The device DCD may use the FS/HS mode information to determine the bandwidth mode of the device.

In some applications, it may not be possible to enable one or more pipes while in FS mode. Beyond the data rate issue, there is no difference in DCD operation between FS and HS modes.

32.14.2.2 Suspend/Resume

32.14.2.2.1 Suspend

To conserve power, USB devices automatically enter the suspended state when the device has observed no bus traffic for a specified period. When suspended, the USB device maintains any internal status, including its address and configuration. Attached devices must be prepared to suspend at any time they are powered, regardless of if they have been assigned a non-default address, are configured, or neither. Bus activity may cease due to the host entering a suspend mode of its own. In addition, a USB device shall also enter the suspended state when the hub port it is attached to is disabled.

A USB device exits suspend mode when there is bus activity. A USB device may also request the host to exit suspend mode or selective suspend by using electrical signaling to indicate remote wakeup. The ability of a device to signal remote wakeup is optional. If the USB device is capable of remote wakeup signaling, the device must support the ability of the host to enable and disable this capability. When the device is reset, remote wakeup signaling must be disabled.

Suspend Operational Model

The device controller moves into the suspend state when suspend signaling is detected or activity is missing on the upstream port for more than a specific period. After the device controller enters the suspend state, the DCD is notified by an interrupt (assuming *DC Suspend Interrupt* is enabled). When the *DCSuspend* bit in the PORTSCx is set to a ‘1’, the device controller is suspended.

DCD response when the device controller is suspended is application specific and may involve switching to low power operation.

Information on the bus power limits in suspend state can be found in USB 2.0 specification.

NOTE

Review system level clocking issues defined in section (Ref: Signals-Clocking) for the clocking requirements of a suspended device controller.

32.14.2.2.2 Resume

If the device controller is suspended, its operation is resumed when any non-idle signaling is received on its upstream facing port. In addition, the device can signal the system to resume operation by forcing resume signaling to the upstream port. Resume signaling is sent upstream by writing a ‘1’ to the Resume bit in the in the PORTSCx while the device is in suspend state. Sending resume signal to an upstream port should cause the host to issue resume signaling and bring the suspended bus segment (one more devices) back to the active condition.

NOTE

Before resume signaling can be used, the host must enable it by using the Set Feature command defined in device framework (chapter 9) of the USB 2.0 Specification.

32.14.2.2.3 Port Test Modes

Contact ARC International for port test mode capabilities.

32.14.2.3 Managing Endpoints

The USB 2.0 specification defines an endpoint, also called a device endpoint or an address endpoint as a uniquely addressable portion of a USB device that can source or sink data in a communications channel between the host and the device. The endpoint address is specified by the combination of the endpoint number and the endpoint direction.

The channel between the host and an endpoint at a specific device represents a data pipe. Endpoint 0 for a device is always a *control* type data channel used for device discovery and enumeration. Other types of endpoints support by USB include *bulk, interrupt*, and *isochronous*. Each endpoint type has specific behavior related to packet response and error handling. More detail on endpoint operation can be found in the USB 2.0 specification.

The ARC USB-HS OTG High-Speed USB On-The-Go device controller hardware supports up to the USB 2.0 maximum of 32 endpoint specified numbers. Each additional endpoint beyond the required endpoint position adds additional hardware logic. The maximum number of endpoint numbers available to the DCD is configured at hardware synthesis timer. After synthesis, the DCD can enable, disable and configure endpoint type up to the maximum selected during synthesis.

Each endpoint direction is essentially independent and can be configured with differing behavior in each direction. For example, the DCD can configure endpoint 1-IN to be a bulk endpoint and endpoint 1-OUT to be an isochronous endpoint. This helps to conserve the total number of endpoints required for device operation. The only exception is that control endpoints must use both directions on a single endpoint number to function as a control endpoint. Endpoint 0 is, for example, is always a control endpoint and uses the pair of directions.

Each endpoint direction requires a *queue head* allocated in memory. If the maximum of 16 endpoint numbers, one for each endpoint direction are being used by the device controller, then 32 *queue heads* are required. The operation of an endpoint and use of *queue heads* are described later in this document.

32.14.2.4 Endpoint Initialization

After hardware reset, all endpoints except endpoint zero are uninitialized and disabled. The DCD must configure and enable each endpoint by writing to configuration bit in the ENDPTCTRLx register. Each 32-bit ENDPTCTRLx is split into an upper and lower half. The lower half of ENDPTCTRLx is used to configure the receive or OUT endpoint and the upper half is likewise used to configure the corresponding transmit or IN endpoint. Control endpoints must be configured the same in both the upper and lower half of the ENDPTCTRLx register otherwise the behavior is undefined. The following table shows how to construct a configuration word for endpoint initialization.

Table 32-98. Device Controller Endpoint Initialization

Field	Value
Data Toggle Reset	‘1’
Data Toggle Inhibit	‘0’
Endpoint Type	00 Control 01 Isochronous 10 Bulk 11 Interrupt
Endpoint Stall	‘0’

32.14.2.5 Stalling

There are two occasions where the device controller may need to return to the host a STALL.

The first occasion is the functional stall, which is a condition set by the DCD as described in the USB 2.0 device framework (chapter 9). A functional stall is only used on non-control endpoints and can be enabled in the device controller by setting the endpoint stall bit in the ENDPTCTRLx register associated with the given endpoint and the given direction. In a functional stall condition, the device controller will continue to return STALL responses to all transactions occurring on the respective endpoint and direction until the endpoint stall bit is cleared by the DCD.

A protocol stall, unlike a function stall, is used on control endpoints is automatically cleared by the device controller at the start of a new control transaction (setup phase). When enabling a protocol stall, the DCD should enable the stall bits (both directions) as a pair. A single write to the ENDPTCTRLx register can ensure that both stall bits are set at the same instant.

NOTE

Any write to the ENDPTCTRLx register during operational mode must preserve the endpoint type field (for example, perform a read-modify-write).

Table 32-99. Device Controller Stall Response Matrix

USB Packet	Endpoint Stall Bit.	Effect on STALL bit.	USB Response
SETUP packet received by a non-control endpoint.	N/A	None.	STALL
IN/OUT/PING packet received by a non-control endpoint.	‘1’	None.	STALL
IN/OUT/PING packet received by a non-control endpoint.	‘0’	None.	ACK/ NAK/ NYET
SETUP packet received by a control endpoint.	N/A	Cleared	ACK

Table 32-99. Device Controller Stall Response Matrix (continued)

USB Packet	Endpoint Stall Bit.	Effect on STALL bit.	USB Response
IN/OUT/PING packet received by a control endpoint	'1'	None	STALL
IN/OUT/PING packet received by a control endpoint.	'0'	None.	ACK/ NAK/ NYET

32.14.2.6 Data Toggle

Data toggle is a mechanism to maintain data coherency between host and device for any given data pipe. For more information on data toggle, refer to the USB 2.0 specification.

32.14.2.6.1 Data Toggle Reset

The DCD may reset the data toggle state bit and cause the data toggle sequence to reset in the device controller by writing a '1' to the data toggle reset bit in the ENDPTCTRLx register. This should only be necessary when configuring/initializing an endpoint or returning from a STALL condition.

32.14.2.6.2 Data Toggle Inhibit

NOTE

This feature is for test purposes only and should never be used during normal device controller operation.

Setting the *data toggle Inhibit bit* active ('1') causes the device controller to ignore the data toggle pattern that is normally sent and accept all incoming data packets regardless of the data toggle state.

In normal operation, the device controller checks the DATA0/DATA1 bit against the data toggle to determine if the packet is valid. If Data PID does not match the data toggle state bit maintained by the device controller for that endpoint, the Data toggle is considered not valid. If the data toggle is not valid, the device controller assumes the packet was already received and discards the packet (not reporting it to the DCD). To prevent the host controller from re-sending the same packet, the device controller will respond to the error packet by acknowledging it with either an ACK or NYET response.

32.14.3 Operational Model For Packet Transfers

All transactions on the USB bus are initiated by the host and in turn, the device must respond to any request from the host within the turnaround time stated in the USB 2.0 Specification. At USB 1.1 Full or Low Speed rates, this turnaround time was significant and the USB 1.1 device controllers were architected so that the device controller could access main memory or interrupt a host protocol processor to respond to the USB 1.1 transaction. The architecture of the USB 2.0 device controller must be different because same methods will not meet USB 2.0 High-speed turnaround time requirements by simply increasing clock rate.

A USB host will send requests to the device controller in an order that can not be precisely predicted as a single pipeline, so it is not possible to prepare a single packet for the device controller to execute. However,

the order of packet requests is predictable when the endpoint number and direction is considered. For example, if endpoint 3 (transmit direction) is configured as a bulk pipe, then we can expect the host will send IN requests to that endpoint. This device controller is architected in such a way that it can prepare packets for each endpoint/direction in anticipation of the host request. The process of preparing the device controller to send or receive data in response to host initiated transaction on the bus is referred to as “priming” the endpoint. This term will be used throughout the following documentation to describe the device controller operation so the DCD can be architected properly use priming. Further, note that the term “flushing” is used to describe the action of clearing a packet that was queued for execution.

32.14.3.0.1 Priming Transmit Endpoints

Priming a transmit endpoint will cause the device controller to fetch the device transfer descriptor (dTD) for the transaction pointed to by the device queue head (dQH). After the dTD is fetched, it will be stored in the dQH until the device controller completes the transfer described by the dTD. Storing the dTD in the dQH allows the device controller to fetch the operating context needed to handle a request from the host without the need to follow the linked list, starting at the dQH when the host request is received.

After the device has loaded the dTD, the leading data in the packet is stored in a FIFO in the device controller. This FIFO is split into virtual channels so that the leading data can be stored for any endpoint up to the maximum number of endpoints configured at device synthesis time.

After a priming request is complete, an endpoint state of primed is indicated in the ENDPTSTATUS register. For a primed transmit endpoint, the device controller can respond to an IN request from the host and meet the stringent bus turnaround time of High Speed USB.

Since only the leading data is stored in the device controller FIFO, it is necessary for the device controller to begin filling in behind leading data after the transaction starts. The FIFO must be sized to account for the maximum latency that can be incurred by the system memory bus. More information about FIFO sizing is presented in section .

32.14.3.0.2 Priming Receive Endpoints

Priming receive endpoints is identical to priming of transmit endpoints from the point of view of the DCD. At the device controller the major difference in the operational model is that there is no data movement of the leading packet data simply because the data is to be received from the host.

Note as part of the architecture, the FIFO for the receive endpoints is not partitioned into multiple channels like the transmit FIFO. Thus, the size of the RX FIFO does not scale with the number of endpoints.

32.14.3.1 Interrupt/Bulk Endpoint Operational Model

The behaviors of the device controller for interrupt and bulk endpoints are identical. All valid IN and OUT transactions to bulk pipes will handshake with a NAK unless the endpoint had been primed. Once the endpoint has been primed, data delivery will commence.

A dTD will be retired by the device controller when the packets described in the transfer descriptor have been completed. Each dTD describes N packets to be transferred according to the USB Variable Length transfer protocol. The formula and table on the following page describe how the device controller

computes the number and length of the packets to be sent/received by the USB vary according to the total number of bytes and maximum packet length.

With Zero Length Termination (ZLT) = 0

N = INT(Number Of Bytes/Max. Packet Length) + 1

With Zero Length Termination (ZLT) = 1

N = MAXINT(Number Of Bytes/Max. Packet Length)

Table 32-100. Variable Length Transfer Protocol Example (ZLT = 0)

Bytes (dTD)	Max. Packet Length (dQH)	N	P1	P2	P3
511	256	2	256	255	
512	256	3	256	256	0
512	512	2	512	0	

Table 32-101. Variable Length Transfer Protocol Example (ZLT = 1)

Bytes (dTD)	Max. Packet Length (dQH)	N	P1	P2	P3
511	256	2	256	255	
512	256	2	256	256	
512	512	1	512		

NOTE

The MULT field in the dQH must be set to “00” for bulk, interrupt, and control endpoints.

TX-dTD is complete when:

- All packets described dTD were successfully transmitted. *** Total bytes in dTD will equal zero when this occurs.

RX-dTD is complete when:

- All packets described in dTD were successfully received. *** Total bytes in dTD will equal zero when this occurs.
- A short packet (number of bytes < maximum packet length) was received. *** This is a successful transfer completion; DCD must check Total Bytes in dTD to determine the number of bytes that are remaining. From the total bytes remaining in the dTD, the DCD can compute the actual bytes received.
- A long packet was received (number of bytes > maximum packet size) OR (total bytes received > total bytes specified). *** This is an error condition. The device controller will discard the remaining packet, and set the Buffer Error bit in the dTD. In addition, the endpoint will be flushed and the USBERR interrupt will become active.

On the successful completion of the packet(s) described by the dTD, the active bit in the dTD will be cleared and the next pointer will be followed when the Terminate bit is clear. When the Terminate bit is set, the device controller will flush the endpoint/direction and cease operations for that endpoint/direction.

On the unsuccessful completion of a packet (see long packet above), the dQH will be left pointing to the dTD that was in error. To recover from this error condition, the DCD must properly reinitialize the dQH by clearing the active bit and update the nextTD pointer before attempting to re-prime the endpoint.

NOTE

All packet level errors such as a missing handshake or CRC error will be retried automatically by the device controller.

There is no required interaction with the DCD for handling such errors.

32.14.3.1.1 Interrupt/Bulk Endpoint Bus Response Matrix

Table 32-102. Interrupt/Bulk Endpoint Bus Response Matrix

	Stall	Not Primed	Primed	Underflow	Overflow
Setup	Ignore	Ignore	Ignore	N/A	N/A
In	STALL	NAK	Transmit	BS Error	N/A
Out	STALL	NAK	Receive + NYET/ACK	N/A	NAK
Ping	STALL	NAK	ACK	N/A	N/A
Invalid	Ignore	Ignore	Ignore	Ignore	Ignore

BS Error = Force Bit Stuff Error

NYET/ACK – NYET unless the Transfer Descriptor has packets remaining according to the USB variable length protocol then ACK.

SYSERR – System error should never occur when the latency FIFOs are correctly sized and the DCD is responsive.

32.14.3.2 Control Endpoint Operation Model

32.14.3.2.1 Setup Phase

All requests to a control endpoint begin with a setup phase followed by an optional data phase and a required status phase. The device controller will always accept the setup phase unless the setup lockout is engaged.

The setup lockout will engage so that future setup packets are ignored. Lockout of setup packets ensures that while software is reading the setup packet stored in the queue head, that data is not written as it is being read potentially causing an invalid setup packet.

In hardware versions 2.3 and later, the setup lockout mechanism can be disabled and a new tripwire type semaphore will ensure that the setup packet payload is extracted from the queue head without being corrupted be an incoming setup packet. This is the preferred behavior because ignoring repeated setup packets due to long software interrupt latency would be a compliance issue.

Setup Packet Handling (Pre-2.3 hardware)

- After receiving an interrupt and inspecting USBMODE to determine that a setup packet was received on a particular pipe:
  - 1. Duplicate contents of dQH.SsetupBuffer into local software byte array.
  - 2. Write '1' to clear corresponding ENDPTSETUPSTAT bit and thereby disabling Setup Lockout. (for example, the Setup Lockout activates as soon as a setup arrives. By writing to the ENDPTSETUPSTAT, the device controller will accept new setup packets.)
  - 3. Process setup packet using local software byte array copy and execute status/handshake phases.

NOTE

After receiving a new setup packet the status and/or handshake phases may still be pending from a previous control sequence. These should be flushed & deallocated before linking a new status and/or handshake dTD for the most recent setup packet.

To limit the exposure of setup packets to the setup lockout mechanism (if used), the DCD should designate the priority of responding to setup packets above responding to other packet completions.

Setup Packet Handling (2.3 hardware and later)

- Disable Setup Lockout by writing ‘1’ to Setup Lockout Mode (SLOM) in USBMODE. (once at initialization). Setup lockout is not necessary when using the tripwire as described below.

NOTE

Leaving the Setup Lockout Mode As ‘0’ will result in pre-2.3 hardware behavior.

- After receiving an interrupt and inspecting ENDPTSETUPSTAT to determine that a setup packet was received on a particular pipe:
  - 1. Write '1' to clear corresponding bit ENDPTSETUPSTAT.
  - 2. Write ‘1’ to Setup Tripwire (SUTW) in USBCMD register.
  - 3. Duplicate contents of dQH.SetupBuffer into local software byte array.
  - 4. Read Setup TripWire (SUTW) in USBCMD register. (if set - continue; if cleared - goto 2)
  - 5. Write '0' to clear Setup Tripwire (SUTW) in USBCMD register.
  - 6. Process setup packet using local software byte array copy and execute status/handshake phases.

NOTE

After receiving a new setup packet the status and/or handshake phases may still be pending from a previous control sequence. These should be flushed & deallocated before linking a new status and/or handshake dTD for the most recent setup packet.

32.14.3.2.2 Data Phase

Following the setup phase, the DCD must create a device transfer descriptor for the data phase and prime the transfer.

After priming the packet, the DCD must verify a new setup packet has not been received by reading the ENDPTSETUPSTAT register immediately verifying that the prime had completed. A prime will complete when the associated bit in the ENDPTPRIME register is zero and the associated bit in the ENDPTSTATUS register is a one. If a prime fails, ie. The ENDPTPRIME bit goes to zero and the ENDPTSTATUS bit is not set, then the prime has failed. This can only be due to improper setup of the dQH, dTD or a setup arriving during the prime operation. If a new setup packet is indicated after the ENDPTPRIME bit is cleared, then the transfer descriptor can be freed and the DCD must reinterpret the setup packet.

Should a setup arrive after the data stage is primed, the device controller will automatically clear the prime status (ENDPTSTATUS) to enforce data coherency with the setup packet.

32.14.3.2.3 Status Phase

Similar to the data phase, the DCD must create a transfer descriptor (with byte length equal zero) and prime the endpoint for the status phase. The DCD must also perform the same checks of the ENDPTSETUPSTAT as described above in the data phase.

NOTE

The MULT field in the dQH must be set to “00” for bulk, interrupt, and control endpoints.

Error handling of data phase packets is the same as bulk packets described previously.

32.14.3.2.4 Control Endpoint Bus Response Matrix

Shown in the following table is the device controller response to packets on a control endpoint according to the device controller state.

Table 32-103. Control Endpoint Bus Response Matrix

Token Type	Endpoint State					Setup Lockout
	Stall	Not Primed	Primed	Underflow	Overflow	
Setup	ACK	ACK	ACK	N/A	SYSERR	
In	STALL	NAK	Transmit	BS Error	N/A	N/A
Out	STALL	NAK	Receive + NYET/ACK	N/A	NAK	N/A
Ping	STALL	NAK	ACK	N/A	N/A	N/A
Invalid	Ignore	Ignore	Ignore	Ignore	Ignore	Ignore

BS Error = Force Bit Stuff Error

NYET/ACK – NYET unless the Transfer Descriptor has packets remaining according to the USB variable length protocol then ACK.

SYSERR – System error should never occur when the latency FIFOs are correctly sized and the DCD is responsive.

32.14.3.3 Isochronous Endpoint Operational Model

Isochronous endpoints are used for real-time scheduled delivery of data and their operational model is significantly different than the host throttled Bulk, Interrupt, and Control data pipes. Real time delivery by the device controller will is accomplished by the following:

- Exactly MULT Packets per (micro)Frame are transmitted/received. Note: MULT is a two-bit field in the device Queue Head. The variable length packet protocol is not used on isochronous endpoints.
- NAK responses are not used. Instead, zero length packets and sent in response to an IN request to an unprimed endpoints. For unprimed RX endpoints, the response to an OUT transaction is to ignore the packet within the device controller.
- Prime requests always schedule the transfer described in the dTD for the next (micro)frame. If the ISO-dTD is still active after that frame, then the ISO-dTD will be held ready until executed or canceled by the DCD.

An EHCI compatible host controller uses the periodic frame list to schedule data exchanges to Isochronous endpoints. The operational model for device mode does not use such a data structure. Instead, the same dTD used for Control/Bulk/Interrupt endpoints is also used for isochronous endpoints. The difference is in the handling of the dTD.

The first difference between bulk and ISO-endpoints is that priming an ISO-endpoint is a delayed operation such that an endpoint will become primed only after a SOF is received. After the DCD writes the prime bit, the prime bit will be cleared as usual to indicate to software that the device controller completed a priming the dTD for transfer. Internal to the design, the device controller hardware masks that prime start until the next frame boundary. This behavior is hidden from the DCD but occurs so that the device controller can match the dTD to a specific (micro)frame.

Another difference with isochronous endpoints is that the transaction must wholly complete in a (micro)frame. Once an ISO transaction is started in a (micro)frame it will retire the corresponding dTD when MULT transactions occur or the device controller finds a fulfillment condition.

The transaction error bit set in the status field indicates a fulfillment error condition. When a fulfillment error occurs, the frame after the transfer failed to complete wholly, the device controller will force retire the ISO-dTD and move to the next ISO-dTD.

It is important to note that fulfillment errors are only caused due to partially completed packets. If no activity occurs to a primed ISO-dTD, the transaction will stay primed indefinitely. This means it is up to software discard transmit ISO-dTDs that pile up from a failure of the host to move the data.

Finally, the last difference with ISO packets is in the data level error handling. When a CRC error occurs on a received packet, the packet is not retried similar to bulk and control endpoints. Instead, the CRC is

noted by setting the *Transaction Error* bit and the data is stored as usual for the application software to sort out.

- TX Packet Retired
  - — MULT counter reaches zero.
  - — Fulfillment Error [*Transaction Error* bit is set]
    - – # Packets Occurred > 0 AND # Packets Occurred < MULT

NOTE

For TX-ISO, MULT Counter can be loaded with a lesser value in the dTD Multiplier Override field in hardware versions 2.3 and later. If the Multiplier Override is zero, the MULT Counter is initialized to the Multiplier in the QH.

- RX Packet Retired:
  - — MULT counter reaches zero.
  - — Non-MDATA Data PID is received**
    - – ** Exit criteria only valid in hardware version 2.3 or later. Previous to hardware version 2.3, any PID sequence that did not match the MULT field exactly would be flagged as a transaction error due to PID mismatch or fulfillment error.
  - — Overflow Error:
    - – Packet received is > maximum packet length. [*Buffer Error* bit is set]
    - – Packet received exceeds total bytes allocated in dTD. [*Buffer Error* bit is set]
  - — Fulfillment Error [*Transaction Error* bit is set]
    - – # Packets Occurred > 0 AND # Packets Occurred < MULT
  - — CRC Error [*Transaction Error* bit is set]

NOTE

For ISO, when a dTD is retired, the next dTD is primed for the next frame. For continuous (micro)frame to (micro)frame operation the DCD should ensure that the dTD linked-list is out ahead of the device controller by at least two (micro)frames.

32.14.3.3.1 Isochronous Pipe Synchronization

When it is necessary to synchronize an isochroous data pipe to the host, the (micro)frame number (FRINDEX register) can be used as a marker. To cause a packet transfer to occur at a specific (micro)frame number [N], the DCD should interrupt on SOF during frame N-1. When the FRINDEX=N–1, the DCD must write the prime bit. The device controller will prime the isochronous endpoint in (micro)frame N–1 so that the device controller will execute delivery during (micro)frame N.

CAUTION

Priming an endpoint towards the end of (micro)frame N-1 will not guarantee delivery in (micro)frame N. The delivery may actually occur in (micro)frame N+1 if device controller does not have enough time to complete the prime before the SOF for packet N is received.

32.14.3.3.2 Isochronous Endpoint Bus Response Matrix

Table 32-104. Isochronous Endpoint Bus Response Matrix

	Stall	Not Primed	Primed	Underflow	Overflow
Setup	STALL	STALL	STALL	N/A	N/A
In	NULL Packet	NULL Packet	Transmit	BS Error	N/A
Out	Ignore	Ignore	Receive	N/A	Drop Packet
Ping	Ignore	Ignore	Ignore	Ignore	Ignore
Invalid	Ignore	Ignore	Ignore	Ignore	Ignore

BS Error = Force Bit Stuff Error

NULL Packet = Zero Length Packet

32.14.4 Managing Queue Heads

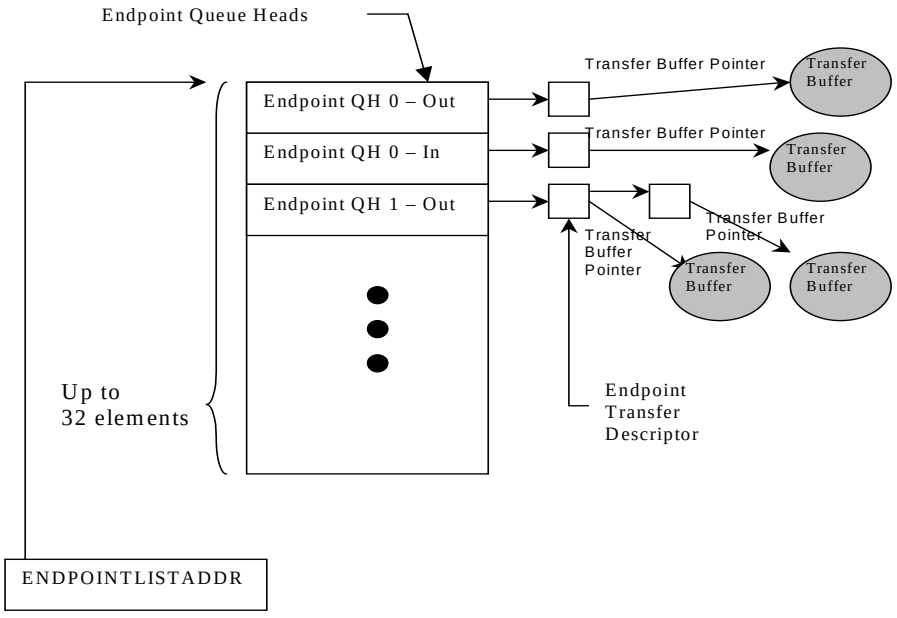


Figure 32-83. End Point Queue Head Diagram

The device queue head (dQH) points to the linked list of transfer tasks, each depicted by the device Transfer Descriptor (dTD). An area of memory pointed to by ENDPOINTLISTADDR contains a group of all dQH's in a sequential list as shown in Figure 79 End Point Queue Head Diagram79. The even elements in the list of dQH's are used for receive endpoints (OUT/SETUP) and the odd elements are used for transmit endpoints (IN/INTERRUPT). Device transfer descriptors are linked head to tail starting at the queue head and ending at a terminate bit. Once the dTD has been retired, it will no longer be part of the linked list from the queue head. Therefore, software is required to track all transfer descriptors since pointers will no longer exist within the queue head once the dTD is retired (see section Section 32.14.5.1, "Software Link Pointers").

In addition to the current and next pointers and the dTD overlay examined in section Section 32.14.3, "Operational Model For Packet Transfers," the dQH also contains the following parameters for the associated endpoint: Multipler, Maximum Packet Length, Interrupt On Setup. The complete initialization of the dQH including these fields is demonstrated in the next section.

32.14.4.1 Queue Head Initialization

One pair of device queue heads must be initialized for each active endpoint. To initialize a device queue head:

- Write the wMaxPacketSize field as required by the USB Chapter 9 or application specific protocol.
- Write the multiplier field to 0 for control, bulk, and interrupt endpoints. For ISO endpoints, set the multiplier to 1,2, or 3 as required bandwidth an in conjuction with the USB Chapter 9 protocol. *Note: In FS mode, the multiplier field can only be 1 for ISO endpoints.*
- Write the next dTD Terminate bit field to "1".
- Write the Active bit in the status field to "0".
- Write the Halt bit in the status field to "0".

NOTE

The DCD must only modify dQH if the associated endpoint is not primed and there are no outstanding dTDs.

32.14.4.2 Operational Model For Setup Transfers

As discussed in section Control Endpoint Operation Model, setup transfer requires special treatment by the DCD. A setup transfer does not use a dTD but instead stores the incoming data from a setup packet in an 8-byte buffer within the dQH.

Upon receiving notification of the setup packet, the DCD should handle the setup transfer as demonstrated here:

1. Copy setup buffer contents from dQH - RX to software buffer.
2. Acknowledge setup backup by writing a "1" to the corresponding bit in ENDPTSETUPSTAT.

NOTE

The acknowledge must occur before continuing to process the setup packet.

After the acknowledge has occurred, the DCD must not attempt to access the setup buffer in the dQH – RX. Only the local software copy should be examined.

3. Check for pending data or status dTD's from previous control transfers and flush if any exist as discussed in section Flushing/De-priming an Endpoint.
4. Decode setup packet and prepare data phase [optional] and status phase transfer as required by the USB Chapter 9 or application specific protocol.

NOTE

It is possible for the device controller to receive setup packets before previous control transfers complete. Existing control packets in progress must be flushed and the new control packet completed.

32.14.5 Managing Transfers with Transfer Descriptors

32.14.5.1 Software Link Pointers

It is necessary for the DCD software to maintain head and tail pointers to the for the linked list of dTDs for each respective queue head. This is necessary because the dQH only maintains pointers to the current working dTD and the next dTD to be executed. The operations described in next section for managing dTD will assume the DCD can use reference the head and tail of the dTD linked list.

NOTE

To conserve memory, the reserved fields at the end of the dQH can be used to store the Head & Tail pointers but it still remains the responsibility of the DCD to maintain the pointers.

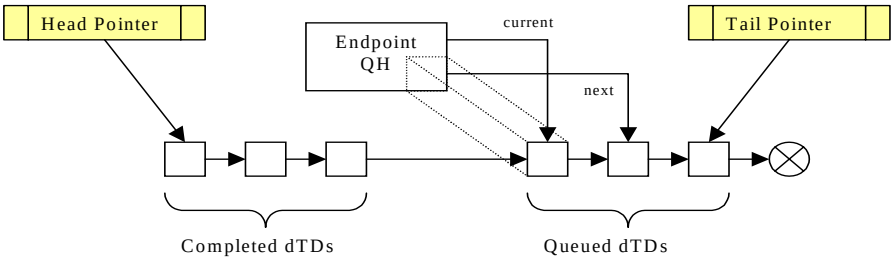


Figure 32-84. Software Link Pointers

32.14.5.2 Building a Transfer Descriptor

Before a transfer can be executed from the linked list, a dTD must be built to describe the transfer. Use the following procedure for building dTDs.

Allocate 8-DWord dTD block of memory aligned to 8-DWord boundaries. Example: bit address 4:0 would be equal to "00000"

Write the following fields:

1. Initialize first 7 DWords to 0.
2. Set the terminate bit to “1”.
3. Fill in total bytes with transfer size.
4. Set the interrupt on complete if desired.
5. Initialize the status field with the active bit set to “1” and all remaining status bits set to “0”.
6. Fill in buffer pointer page 0 and the current offset to point to the start of the data buffer.
7. Initialize buffer pointer page 1 through page 4 to be one greater than each of the previous buffer pointer.

32.14.5.3 Executing A Transfer Descriptor

To safely add a dTD, the DCD must be follow this procedure which will handle the event where the device controller reaches the end of the dTD list at the same time a new dTD is being added to the end of the list.

Determine whether the link list is empty:

— Check DCD driver to see if pipe is empty (internal representation of linked-list should indicate if any packets are outstanding)

- Case 1: Link list is empty
  - — 1. Write dQH next pointer AND dQH terminate bit to 0 as a single DWord operation.
  - — 2. Clear active & halt bit in dQH (in case set from a previous error).
  - — 3. Prime endpoint by writing ‘1’ to correct bit position in ENDPTPRIME.
- Case 2: Link list is not empty
  - — 1. Add dTD to end of linked list.
  - — 2. Read correct prime bit in ENDPTPRIME – if ‘1’ DONE.
  - — 3. Set ATDTW bit in USBCMD register to ‘1’.
  - — 4. Read correct tatus bit in ENDPTSTAT. (store in tmp. variable for later)
  - — 5. Read ATDTW bit in USBCMD register.
  - — If ‘0’ goto 3.
  - — If ‘1’ continue to 6.
  - — 6. Write ATDTW bit in USBCMD register to ‘0’.
  - — 7. If status bit read in (3) is ‘1’ DONE.
  - — 8. If status bit read in (3) is ‘0’ then Goto Case 1: Step 1.

32.14.5.4 Transfer Completion

After a dTD has been initialized and the associated endpoint primed the device controller will execute the transfer upon the host-initiated request. The DCD will be notified with a USB interrupt if the Interrupt On Complete bit was set or alternately, the DCD can poll the endpoint complete register to find when the dTD had been executed. After a dTD has been executed, DCD can check the status bits to determine success or failure.

CAUTION

Multiple dTD can be completed in a single endpoint complete notification. After clearing the notification, DCD must search the dTD linked list and retire all dTDs that have finished (Active bit cleared).

By reading the status fields of the completed dTDs, the DCD can determine if the transfers completed successfully. Success is determined with the following combination of status bits:

- Active = 0
- Halted = 0
- Transaction Error = 0
- Data Buffer Error = 0

Should any combination other than the one shown above exist, the DCD must take proper action. Transfer failure mechanisms are indicated in the Device Error Matrix.

In addition to checking the status bit the DCD must read the Transfer Bytes field to determine the actual bytes transferred. When a transfer is complete, the Total Bytes transferred is by decremented by the actual bytes transferred. For Transmit packets, a packet is only complete after the actual bytes reaches zero, but for receive packets, the host may send fewer bytes in the transfer according the USB variable length packet protocol.

32.14.5.5 Flushing/De-priming an Endpoint

It is necessary for the DCD to flush to de-prime one more endpoints on a USB device reset or during a broken control transfer. There may also be application specific requirements to stop transfers in progress. The following procedure can be used by the DCD to stop a transfer in progress:

1. Write a ‘1’ to the corresponding bit(s) in ENDPTFLUSH.
2. Wait until all bits in ENDPTFLUSH are ‘0’.
   — Software note: this operation may take a large amount of time depending on the USB bus activity. It is not desirable to have this wait loop within an interrupt service routine.
3. Read ENDPTSTAT to ensure that for all endpoints commanded to be flushed, that the corresponding bits are now ‘0’. If the corresponding bits are ‘1’ after step #2 has finished, then the flush failed as described in the following:
   — Explanation: In very rare cases, a packet is in progress to the particular endpoint when commanded flush using ENDPTFLUSH. A safeguard is in place to refuse the flush to ensure that the packet in progress completes successfully. The DCD may need to repeatedly flush any endpoints that fail to flush be repeating steps 1-3 until each endpoint is successfully flushed.

32.14.5.6 Device Error Matrix

The following table summarizes packet errors that are not automatically handled by the Device Controller.

Table 32-105. Device Error Matrix

Error	Direction	Packet Type	Data Buffer Error Bit	Transaction Error Bit
Overflow **	RX	Any	1	0
ISO Packet Error	RX	ISO	0	1
ISO Fulfillment Error	Both	ISO	0	1

Notice that the device controller handles all errors on Bulk/Control/Interrupt Endpoints except for a data buffer overflow. However, for ISO endpoints, errors packets are not retried and errors are tagged as indicated.

Table 32-106. Error Descriptions

Error	Description
Overflow	Number of bytes received exceeded max. packet size or total buffer length.
	** This error will also set the Halt bit in the dQH and if there are dTDs remaining in the linked list for the endpoint, then those will not be executed.
ISO Packet Error	CRC Error on received ISO packet. Contents not guaranteed to be correct.
ISO Fulfillment Error	Hst failed to complete the number of packets defined in the dQH mult field within the given (micro)frame. For scheduled data delivery the DCD may need to readjust the data queue because a fulfillment error will cause Device Controller to cease data transfers on the pipe for one (micro)frame. During the “dead” (micro)frame, the Device Controller reports error on the pipe and primes for the following frame.

32.14.6 Servicing Interrupts

The interrupt service routine must consider that there are high-frequency, low-frequency operations, and error operations and order accordingly.

32.14.6.1 High-Frequency Interrupts

High frequency interrupts in particular should be handed in the order below. The most important of these is listed first because the DCD must acknowledge a setup buffer in the timeliest manner possible.

Table 32-107. High Frequency Interrupt Events

Execution Order	Interrupt	Action
1a	USB Interrupt ** - ENDPTSETUPSTATUS	Copy contents of setup buffer and acknowledge setup packet (as indicated in section Managing Queue Heads). Process setup packet according to USB 2.0 Chapter 9 or application specific protocol.

Table 32-107. High Frequency Interrupt Events

1b	USB Interrupt ** - ENDPTCOMPLETE	Handle completion of dTD as indicated in section Managing Queue Heads.
2	SOF Interrupt	Action as deemed necessary by application. This interrupt may not have a use in all applications.

** It is likely that multiple interrupts to stack up on any call to the Interrupt Service Routine AND during the Interrupt Service Routine.

32.14.6.2 Low-Frequency Interrupts

The low frequency events include the following interrupts. These interrupt can be handled in any order since they don't occur often in comparison to the high-frequency interrupts.

Table 32-108. Low Frequency Interrupt Events

Interrupt	Action
Port Change	Change software state information.
Sleep Enable (Suspend)	Change software state information. Low power handling as necessary.
Reset Received	Change software state information. Abort pending transfers.

32.14.6.3 Error Interrupts

Error interrupts will be least frequent and should be placed last in the interrupt service routine.

Table 32-109. Error Interrupt Events

Interrupt	Action
USB Error Interrupt	This error is redundant because it combines USB Interrupt and an error status in the dTD. The DCD will more aptly handle packet-level errors by checking dTD status field upon receipt of USB Interrupt (w/ ENDPTCOMPLETE).
System Error	Unrecoverable error. Immediate Reset of core; free transfers buffers in progress and restart the DCD.

Book II, Part 6: Timer Peripherals

Introduction

The i.MX31 and i.MX31L provide a variety of timers for timing and scheduling of software applications, as well as peripherals and audio waveform generation. The timers covered in this part are as follows:

Enhanced Periodic Interrupt Timer (EPIT)

The Enhanced Periodic Interrupt Timer (EPIT) is a 32-bit “set and forget” timer that starts counting after the EPIT is enabled by software. The i.MX31 and i.MX31L contain two identical EPIT timers. It is capable of providing precise interrupts at regular intervals with minimal processor intervention.The EPIT follows the IP Bus protocol for interfacing with the ARM11. The i.MX31 and i.MX31L have two identical EPIT timers. It does not have any interface signals with any other module inside the i.MX31 and i.MX31L except for clock and reset inputs from the Clock Controller Module (CCM) and interrupt signals to the processor interrupt handler.

General Purpose Timer (GPT)

The General Purpose Timer (GPT) has a 32 bit up-counter. The timer counter value can be captured in a register using an event on an external pin. The capture trigger can be programmed to be a rising or/and falling edge. The GPT can also generate an event on ipp_do_cmpout pins and an interrupt when the timer reaches a programmed value. It has a 12-bit prescaler providing a programmable clock frequency derived from multiple clock sources. The GPT has one 32 bit up-counter with clock source selection, including external clock, two input capture channels with programmable trigger edge, and three output compare channels with programmable output mode. The GPT can perform a forced compare and can configured to be programmed to be active in low power and debug modes Interrupt generation can be programmed for capture, compare, rollover events and the timers offers both restart or free-run modes of operation.

Pulse-Width Modulation Module (PWM)

While the Pulse-Width Modulation Module (PWM) is a timer/counter, its primary use is for sound and melody generation. The PWM of the i.MX31 and i.MX31L uses a 16-bit counter which is optimized to generate sound from stored sample audio images and it can also generate tones. It uses the 16-bit resolution and a 4 × 16 data FIFO to generate sound. The PWM follows IP Bus protocol for interfacing with the ARM11 processor core. It does not have any interface signals with any other module inside the chip except for clock and reset inputs from the Clock and Reset Controller module and interrupt signals to the processor interrupt handler. There is a single output signal going to a pin in the i.MX31 and i.MX31L.

The PWM module of the i.MX31 and i.MX31L offers the following features:

- Pulse-width modulation (PWM) module has the following features:
- 4 x 16 FIFO to minimize interrupt overhead
- 16-bit resolution
- Sound and melody generation
- Secondary Display contrast control

Real Time Clock (RTC)

The Real Time Clock (RTC) module provides a current stamp of seconds, minutes, hours and days. Alarm and Timer functions are also available for programming. RTC support dates from the year 1980 to 2050. The RTC module includes the following features:

- Full clock—days, hours, minutes, seconds
- Minute countdown timer with interrupt
- Programmable daily alarm with interrupt
- Sampling timer with interrupt
- Once-per-day, once-per-hour, once-per-minute, and once-per-second interrupts
- Operation at 32.768 kHz or 32 kHz, or 38.4 kHz (determined by reference clock crystal)

Watchdog Timer (WDOG)

The Watchdog (WDOG) timer module protects the i.MX31 and i.MX31L against system failures by providing a method of escaping from unexpected events or programming errors.

Once the WDOG module is activated, it must be serviced by software on a periodic basis. If servicing does not take place, the timer times out. Upon a time-out, the WDOG Timer module either asserts the $\overline{\text{wdog}}$ signal or a system reset signal $\overline{\text{wdog\_rst}}$ depending on software configuration. The WDOG Timer module also generates a system reset via a software write to the Watchdog Control Register (WCR) a detection of a clock monitor event, an external reset, an external JTAG reset signal, or if a power-on-reset has occurred. The $\overline{\text{wdog}}$ signal is asserted via a software write to the WCR, a detection of a clock monitor event, or upon a watchdog time-out.

In case of a Time-out Even, the following actions can be programmed:

- Interrupt to the ARM11

- Internal Reset (can follow the interrupt after a predefined time-out)
- External pin toggle for external devices reset (issued together with the Internal Reset)

The WDOG module can continue/suspend the timer operation in the low power modes (WAIT, DOZE and STOP). For ARM (DOZE and STOP), it emulates these modes.

Chapter 33
Enhanced Periodic Interrupt Timer (EPIT 1, 2)

The Enhanced Periodic Interrupt Timer (EPIT) is a 32-bit timer that can be used to provide precise interrupts at regular intervals with minimal processor intervention. Figure 33-1 illustrates the block diagram of the EPIT.

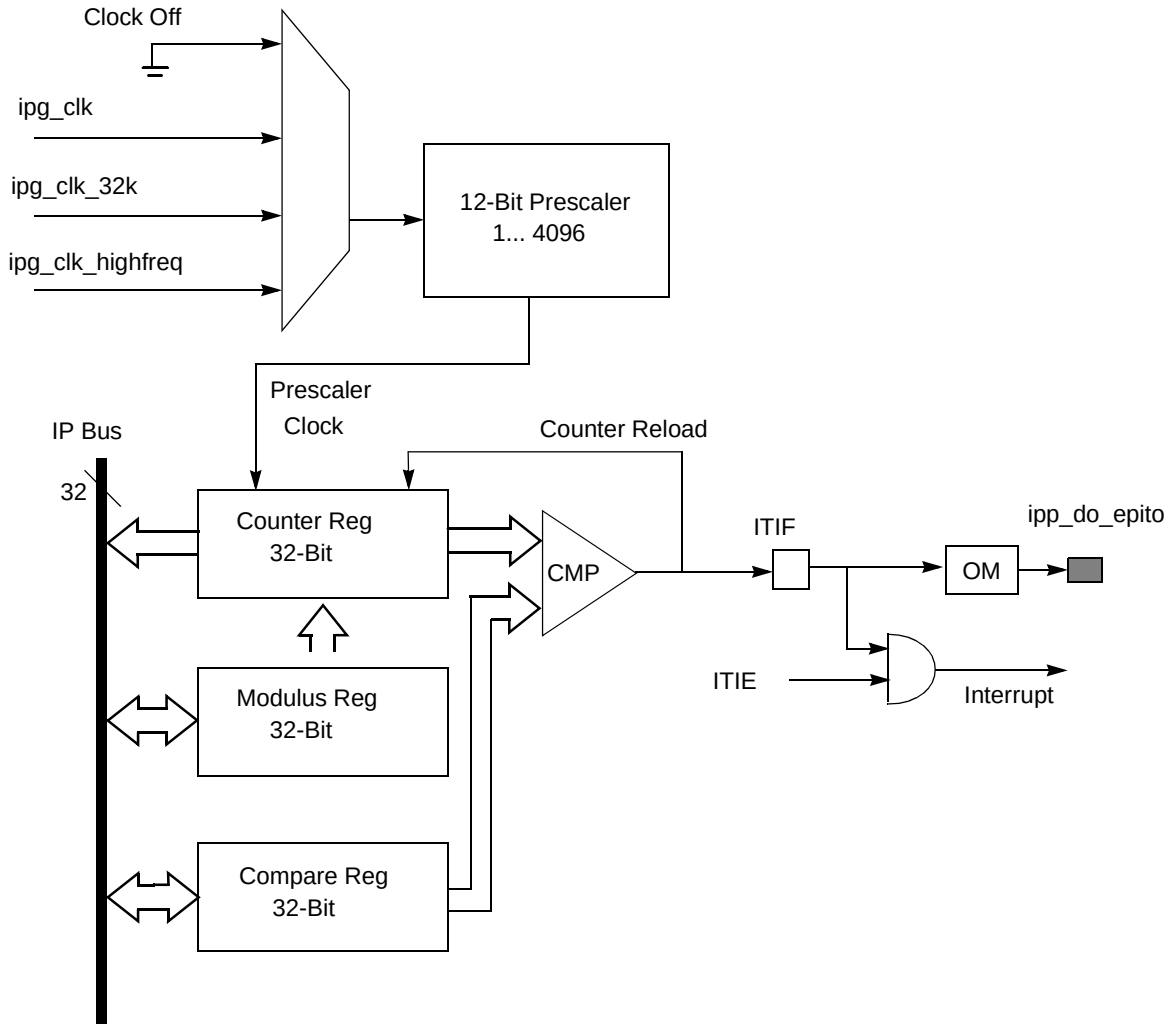


Figure 33-1. Enhanced Periodic Interrupt Timer (EPIT) Block Diagram

33.1 Overview

The EPIT is a 32-bit “set and forget” timer which starts counting after the EPIT is enabled by software. It is capable of providing precise interrupts at regular intervals with minimal processor intervention. The IC provides two identical EPIT timers. Only one timer is described but the memory map for both EPIT timers is provided in this chapter.

33.1.1 Features

The following features characterize the EPIT:

- 32-bit down counter with clock source selection
- 12-bit prescaler for division of input clock frequency
- Counter value can be programmed on the fly
- Can be programmed to be active in low-power and debug modes
- Interrupt generation when counter reaches the Compare value

33.1.2 Modes of Operation

The EPIT can be programmed to function in “set-and-forget” or “free-running” modes, as follows:

- Set-and-Forget mode—This mode of operation is selected when the RLD bit in control register (EPITCR) is set to a value of one. The counter cannot be written from the module data bus. Instead, it gets its data from the load register (EPITLR). Whenever the counter reaches a count of zero, the value in EPITLR is loaded into the counter to be decremented toward zero. The counter may by directly initialized, without having to wait for the count to reach zero, when the EPITLR is written with the IOVW bit in EPITCR set.
- Free-Running mode—This mode of operation is selected when the RLD bit is cleared to a value of zero. In this mode, the counter rolls over from 0x00000000 to 0xFFFFFFFF without reloading from the modulus register, and continues to count down. In this mode the counter can also be directly initialized by writing to EPITLR with the required initialization value after having set the IOVW bit.

33.2 Signal Description

33.2.1 Overview

The EPIT follows IP Bus protocol for interfacing with the ARM11. It does not have any interface signals with any other module inside the IC except for clock and reset inputs from the Clock Controller Module (CCM) and interrupt signals to the processor interrupt handler. There is a single output signal going outside the chip boundary.

Figure 33-2 shows the EPIT signals at the module boundary.

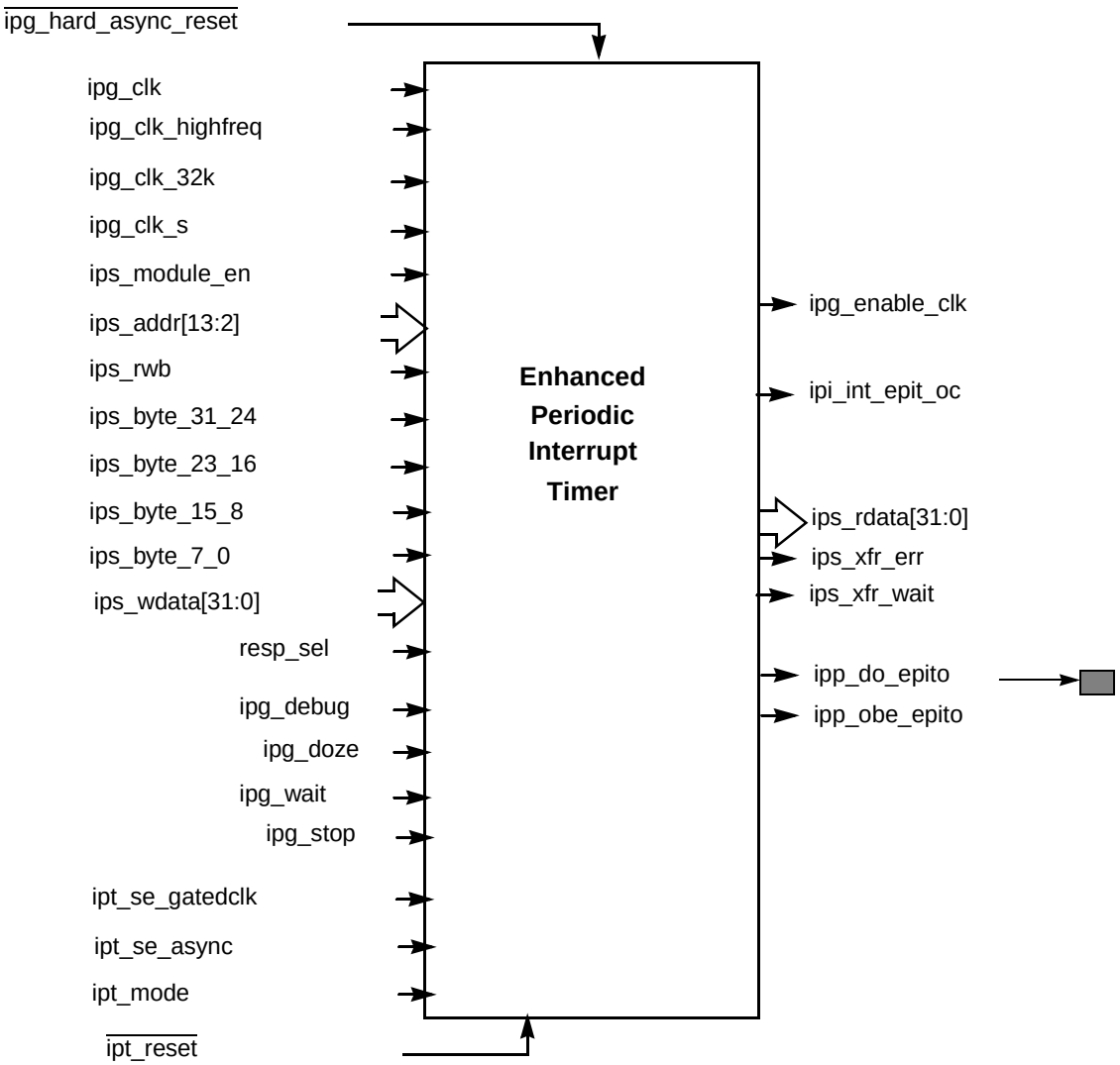


Figure 33-2. EPIT Module Signals

33.3 External Signals

The EPIT module has one output signal to the chip boundary called ipp_do_epito. Table 33-1 describes the signals.

Table 33-1. External EPIT Signals

Name	Direction	Function	Reset State
ipp_do_epito	Output	Output pin for indication of occurrence of output compare event through a specified transition.	0

33.4 EPIT Module Signals

There are four clock inputs, one interrupt line, one hardware reset line, four low power and debug mode signals and four scan signals. The remaining signals are standard IP slave bus signals.

Table 33-2. Module Signal Description

Name	Function	Direction	Reset State
ipg_clk_highfreq	High frequency clock. The Clock Controller provides this clock is synchronized to ahb_clk in normal mode and switches to a raw non-synchronous clock in low power mode when ahb_clk is off. This clock continues to run in low-power mode.	Input	—
ipg_clk_32k	Low frequency clock. The Clock Controller provide this clock which is synchronized to ahb_clk in normal mode and switches to the raw non-synchronous clock in low-power mode when ahb_clk is off. This clock continues to run in low power mode.	Input	—
ipg_clk	IP global functional clock. This clock is always on except in low-power mode.	Input	—
ipg_clk_s	IP slave bus clock. This clock is synchronized to ipg_clk and is used only for register read/writes.	Input	—
ips_module_en	IP slave bus module enable signal. This signal indicates when a module bus transaction is occurring.	Input	—
ips_addr[13:2]	IP slave bus addr signal. Indicates which register is being accessed during a bus transaction	Input	—
ips_rwb	IP slave bus read/write signal. Shows whether a bus transaction is read or write. (Active low)	Input	—
ips_byte_31_24	IP slave bus byte enable signal for bits 31 to 24	Input	—
ips_byte_23_16	IP slave bus byte enable signal for bits 23 to 16	Input	—
ips_byte_15_8	IP slave bus byte enable signal for bits 15 to 8	Input	—
ips_byte_7_0	IP slave bus byte enable signal for bits 7 to 0	Input	—
ips_wdata[31:0]	IP slave bus write date line	Input	—
resp_sel	Signal to determine ips_xfr_err generation conditions. If set, error will not be generated if unimplemented register space is accessed. If not set error will be generated	Input	—
ipg_debug	IP global signal to indicate that EPIT should enter debug mode operation	Input	—
ipg_doze	IP global signal to indicate that EPIT should enter low-power doze mode operation	Input	—
ipg_wait	IP global signal to indicate that EPIT should enter low-power wait mode operation	Input	—
ipg_stop	IP global signal indicating that EPIT should enter low-power stop mode operation	Input	—
ipt_se_gatedclk	IP scan signal for bypassing of clock gating in scan mode	Input	—

Table 33-2. Module Signal Description (continued)

Name	Function	Direction	Reset State
ipt_se_async	IP scan signal for bypassing generated resets and making latches transparent in scan mode	Input	—
ipt_mode	IP scan reset signal used to put module in scan mode	Input	—
$\overline{\text{ipt\_reset}}$	IP scan reset signal used to work in place of generated resets in scan mode (Active low)	Input	—
$\overline{\text{ipg\_hard\_async\_reset}}$	IP global hardware reset (Active low)	Input	—
ipg_enable_clk	IP global clock gating signal. It can be used by the Clock Controller to gate off the clock to the EPIT for power saving. This is essentially the EPIT enable bit value	Output	0
ips_rdata[31:0]	IP slave bus read date line	Output	0
ips_xfr_err	IP slave bus transfer error indicator	Output	0
ips_xfr_wait	IP slave bus transfer wait indicator	Output	0
ipi_int_epit_oc	EPIT output compare interrupt	Output	0
ipp_obe_epito	Output enable for ipp_do_epito pad signal	Output	0

33.5 Memory Map and Register Definition

The EPIT module includes five user-accessible 32-bit registers.

33.5.1 Memory Map

Table 33-3 shows the EPIT memory map.

Table 33-3. EPIT Memory Map

Address	Register	Access	Reset Value	Section/Page
General Registers				
0x53F9_4000 (EPITCR1) 0x53F9_8000 (EPITCR2)	EPIT Control Register	R/W	0x0000_0000	33.5.2.1/33-8
0x53F9_4004 (EPITSR1) 0x53F9_8004 (EPITSR2)	EPIT Status Register	R/W	0x0000_0000	33.5.2.2/33-10
0x53F9_4008 (EPITLR1) 0x53F9_8008 (EPITLR2)	EPIT Load Register	R/W	0xFFFF_FFFF	33.5.2.3/33-11
0x53F9_400C (EPITCMPR1) 0x53F9_800C (EPITCMPR2)	EPIT Compare Register	R/W	0x0000_0000	33.5.2.4/33-11
0x53F9_4010 (EPITCNT1) 0x53F9_8010 (EPITCNT2)	EPIT Counter Register	R	0x0000_0000	33.5.2.5/33-12

33.5.2 Register Summary

Figure 33-3 shows the key to the register fields, and Table 33-4 shows the register figure conventions.

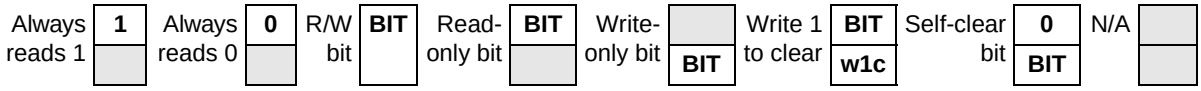


Figure 33-3. Key to Register Fields

Table 33-4. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Self-clearing bit. Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 33-5 shows the EPIT register summary.

Table 33-5. EPIT Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x53F9_4000 (EPITCR1) 0x53F9_8000 (EPITCR2)	R	0	0	0	0	0	0	CLKSRC		OM		STOPEN	DOZEN	WAITEN	DBGEN	IOVW	SWR
	W																
	R	PRESCALER												RLD	OCIEN	ENMOD	EN
	W																

Table 33-5. EPIT Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x53F9_4004 (EPITSR1) 0x53F9_8004 (EPITSR2)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	OCIF
	W																w1c
0x53F9_4008 (EPITLR1) 0x53F9_8008 (EPITLR2)	R	LOAD															
	W																
	R	LOAD															
	W																
0x53F9_400C (EPITCMPR1) 0x53F9_800C (EPITCMPR2)	R	COMPARE															
	W																
	R	COMPARE															
	W																
0x53F9_4010 (EPITCNT1) 0x53F9_8010 (EPITCNT2)	R	COUNT															
	W																
	R	COUNT															
	W																

33.5.2.1 EPIT Control Register (EPITCR)

The EPIT control register (EPITCR) is used to configure the operating settings of the EPIT. It contains the clock division prescaler value and also the interrupt enable bit. Figure 33-4 shows the register, and Table 33-6 provides its field descriptions.

0x53F9_4000 (EPITCR1)
0x53F9_8000 (EPITCR2)

Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	CLKSRC		OM		STOP EN	DOZE N	WAIT EN	DBG EN	IOVW	SWR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PRESCALER												RLD	OCIE N	ENM OD	EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 33-4. EPIT Control Register

Table 33-6. EPITCR Register Field Descriptions

Field	Description
31–26 Reserved	Reserved. These reserved bits are always read as zero.
25–24 CLKSRC	Select Clock Source. These bits select the clock input used to run the counter. After reset, the system functional clock is selected. The input clock can also be turned off if these bits are set to 00. This field value should only be changed when the EPIT is disabled. 00 Clock is off 01 ipg_clk 10 ipg_clk_highfreq 11 ipg_clk_32k
23–22 OM	EPIT Output Configuration. This bit field determines the output mode of EPIT output pin 00 EPIT output is disconnected from pad 01 Toggle output pin 10 Clear output pin 11 Set output pin
21 STOPEN	EPIT Stop Mode Enable. This read/write control bit enables the operation of the EPIT during stop mode. This bit is reset by a hardware reset. A software reset does not affect this bit. 0 EPIT is disabled in stop mode 1 EPIT is enabled in stop mode
20 DOZEN	EPIT Doze Mode Enable. This read/write control bit enables the operation of the EPIT during doze mode. This bit is reset by a hardware reset. A software reset does not affect this bit. 0 EPIT is disabled in stop mode 1 EPIT is enabled in stop mode

Table 33-6. EPITCR Register Field Descriptions (continued)

Field	Description
19 EPIT	EPITWait Mode Enable. This read/write control bit enables the operation of the EPIT during wait mode. This bit is reset by a hardware reset. A software reset does not affect this bit. 0 EPIT is disabled in wait mode 1 EPIT is enabled in wait mode
18 DBGEN	Debug Mode enable. This bit is used to keep the EPIT functional in debug mode. When this bit is cleared, the input clock is gated off while in debug mode. This bit is reset by a hardware reset. A software reset does not affect this bit. 0 Inactive in debug mode 1 Active in debug mode
17 IOVW	EPIT Counter Overwrite Enable. This bit controls the counter data when the modulus register is written. When this bit is set, all writes to the load register will overwrite the counter contents and the counter will subsequently start counting down from the programmed value. 0 Write to load register does not result in counter value being overwritten. 1 Write to load register immediately overwriting current counter value.
16 SWR	Software Reset. The EPIT module is reset when this bit is set to 1. It is a self-clearing bit. This bit is set when the module is in reset state and is cleared when the reset procedure is over. Writing 1 to this bit produces a single wait state write cycle. Setting this bit resets all the registers to their reset values, except for the EN, ENMOD, STOPEN, DOZEN, WAITEN, and DBGEN bits in this control register. 0 EPIT is out of reset 1 EPIT is undergoing reset
15–4 PRESCALER	Counter Clock Prescaler Value. This bit field determines the prescaler value by which the clock is divided before it goes to the counter. 0x000 Divide by 1 0x001 Divide by 2 . . . 0xfff Divide by 4096
3 RLD	Counter Reload Control. This bit is cleared by hardware reset. It controls whether the counter runs in *free running* mode OR *set and forget* mode. 0 When the counter reaches zero, it rolls over to 0xFFFFFFFF. 1 When the counter reaches zero, it reloads from the modulus register.
2 OCIEN	Output Compare Interrupt Enable. This bit enables the generation of interrupt when a compare event occurs. 0 Compare interrupt disabled 1 Compare interrupt enabled

Table 33-6. EPITCR Register Field Descriptions (continued)

Field	Description
1 ENMOD	EPIT Enable Mode. When EPIT is disabled (EN=0), then both Main Counter and Prescaler Counter freeze their count at current count values. ENMOD bit is a R/W bit which decides the counter value when the EPIT is enabled again by setting EN bit. If ENMOD bit is set, then Main Counter is loaded with the load value (If RLD=1)/0xffffffff(If RLD=0) and Prescaler Counter is reset, when EPIT is enabled(EN=1). If ENMOD is programmed to 0 then both Main Counter and Prescaler Counter restarts counting from their frozen values, when EPIT is enabled(EN=1). 0 Counter will start counting from the value it had when it was disabled. 1 Counter will start count from load value (If RLD=1) or 0xFFFFFFFF(If RLD=0).
0 EN	EPIT Enable. This bit enables the EPIT. The EPIT Counter and Prescaler value when EPIT is enabled (EN=1) is dependent upon the ENMOD and RLD bit s, as described for the ENMOD bit. It is recommended that all registers be properly programmed before setting this bit. This bit is reset by a hardware reset. A software reset does not affect this bit. 0 EPIT is disabled. 1 EPIT is enabled.

33.5.2.2 EPIT Status Register (EPITSR)

The EPIT Status Register (EPITSR) has a single status bit for the output compare event. It is a write 1 to clear bit.

Figure 33-5 shows the register, and Table 33-7 provides its field descriptions.

0x53F9_4004 (EPITSR1)
0x53F9_8004 (EPITSR2)

Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	OCIF
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 33-5. EPIT Status Register

Table 33-7. EPTISR Register Field Descriptions

Field	Description
31–1 Reserved	Reserved. These reserved bits are always read as zero.
0 OCIF	Output Compare Interrupt Flag. This bit is the interrupt flag that is set when the content of counter equals the content of the compare register (EPITCMPR). This is write one to clear bit. 0 Compare event has not occurred. 1 Compare event has occurred.

33.5.2.3 EPIT Load Register (EPITLR)

The EPIT load register contains the value that is loaded into the counter at the start of each count cycle if the RLD bit in EPITCR is set. If the IOVW bit in the EPITCR is set then writing to this register will overwrite the value of the EPIT counter register in addition to updating this register's value. This overwrite feature is active even if the RLD bit is not set.

Figure 33-6 shows the register, and Table 33-8 provides its field descriptions.

0x53F9_4008 (EPITLR1)
0x53F9_8008 (EPITLR2)

Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	LOAD															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	LOAD															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 33-6. EPITLR Load Register

Table 33-8. EPITLR Register Field Descriptions

Field	Description
31–0 LOAD	Load Value. Value that is loaded into the counter at the start of each count cycle.

33.5.2.4 EPIT Compare Register (EPITCMPR)

The EPIT Compare Register holds the value that determines when a compare event is generated.

Figure 33-7 shows the register, and Table 33-9 provides its field descriptions.

0x53F9_400C (EPITCMPR1)
0x53F9_800C (EPITCMPR2)

Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	COMPARE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	COMPARE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 33-7. EPITC Compare Register (EPTICCMPR)

Table 33-9. EPITCMPR Register Field Descriptions

Field	Description
31–0 COMPARE	Compare Value. When the counter value equals this COMPARE field value a compare event is generated.

33.5.2.5 EPIT Counter Register (EPITCNT)

The EPIT counter register (EPITCNT) contains the current count value and can be read at any time without disturbing the counter. This is a read-only register, and any attempt to write into it generates a transfer error unless the IOVW bit in EPITCR is set, in which case the value of this register can be overwritten with a write to EPITLR. This change is reflected when this register is subsequently read.

Figure 33-8 shows the register, and Table 33-10 provides its field descriptions.

0x53F9_4010 (EPITCNT1)
0x53F9_8010 (EPITCNT2)

Access: User read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	COUNT															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COUNT															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 33-8. EPIT Counter Register

Table 33-10. EPITCNT Register Field Descriptions

Field	Description
31–0 COUNT	Counter Value. The counter register contains the current count.

33.6 Functional Description

33.6.1 Operation

The EPIT has a single 32-bit down counter which starts counting when the module is enabled by software. The start value of the counter is loaded from the EPIT Load Register, which can be programed with a new value at any time by the processor. The value in the compare register determines the time of occurrence of the interrupt.

When the EPIT is disabled the counter may be frozen at its current state or reset to 0x0000 0000, depending on the ENMOD bit in control register. If ENMOD is set, the counter value is reset, and on subsequent enabling, it starts its count from the Load register value or 0xFFFF FFFF (depending on the state of the RLD bit). If ENMOD is zero, the current value of the counter is frozen and it restarts from the frozen value when enabled.

A hardware reset resets all EPIT registers to their respective reset values. There is a software reset that has the same effect on all registers except for the EN, ENMOD, STOPEN, DOZEN, and WAITEN bits in the control register. The state of these bits are not affected by a software reset. A software reset can be asserted even when the EPIT is disabled.

33.6.1.1 Clocks

The clock source that feeds the prescaler can be selected from one of the following:

- High frequency Clock (ipg_clk_highfreq)

  This is a high frequency clock provided by the Clock Controller module (CCM). This clock remains on in low-power mode when ipg_clk is turned off allowing the EPIT to be run using this clock while in low-power mode. This clock is synchronized to ahb_clk in normal mode and is not synchronized when in low-power mode.
- Low Frequency Reference Clock (ipg_clk_32k)

  This is the 32-KHz low-reference clock provided by the CCM. This clock is on during low-power mode when ipg_clk is turned off allowing EPIT to be operate using this clock during low-power mode. This clock is synchronized to ahb_clk in normal mode and is not synchronized when in low-power mode.
- Global Functional Clock (ipg_clk)

  This clock is on during normal operation. If ipg_clk s selected(CLKSRC=01) as Clock Source. In low power modes, if EPIT is programmed to be disabled (STOPEN or WAITEN or DOZEN=0), then ipg_clk can be switched off.

The clock input to the prescaler can also be disabled by programming the control register. The clock input source is determined by the clock source (CLKSRC) field in the control register. This field value should be changed only after disabling the EPIT by clearing the EN bit in the EPITCR. The PRESCALER field is used to select the divide ratio of the input clock that drives the main counter as shown in Figure 33-9. The prescaler can divide the input clock by a value between 1 and 4096. A change in the value of the PRESCALER field is immediately reflected on its output clock frequency.

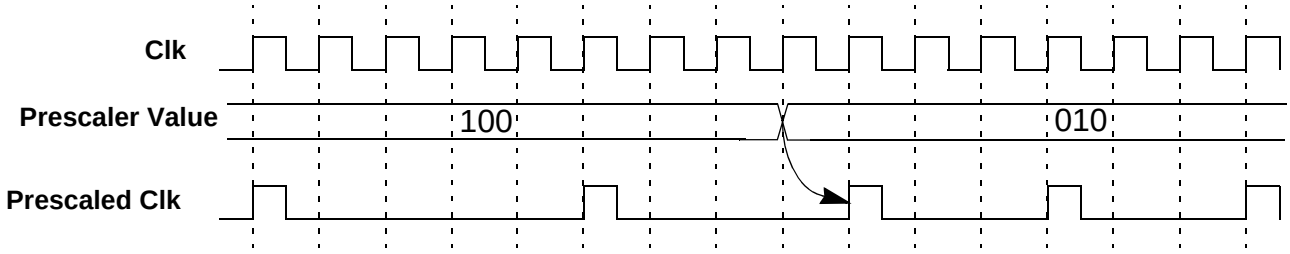


Figure 33-9. Prescaler Value Change Timing Diagram

When the programmed content of EPITCMPR matches the value in EPITCNR, a compare status flag is set and an interrupt is generated if the corresponding bit is set in the control register. Consequently, the compare output pin is set, cleared, toggled, or not affected at all depending how the mode bits are programmed (Figure 33-10). If an interrupt is required at rollover (when counter value reaches 0x0000 0000 and new value is loaded), a compare register value should be made equal to the load register value in *Set-and-Forget* mode or equal to 0xFFFF FFFF in *Free-Running* mode.

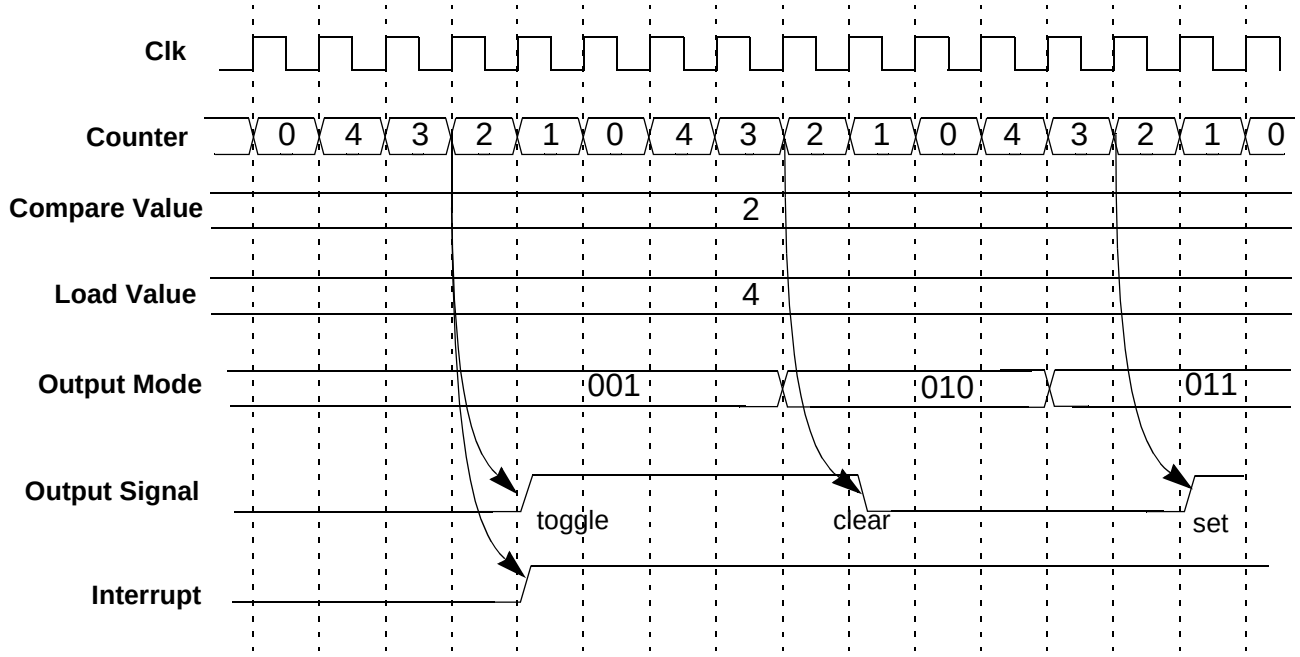


Figure 33-10. Compare Event and Interrupt Timing Diagram

33.6.1.2 Overwriting the Counter Value

The EPIT counter value can be overwritten with a new value at any point in time. The procedure for this is to set the IOVW bit in the control register and then write the desired value into the load register. This results in the load register acquiring that value and also the counter being overwritten with it. If the EPIT is running, the counter resumes counting from the overwritten value.

33.6.1.3 Low-Power Mode Behavior

The EPIT timer's behavior in low-power modes depends on the clock source being used. If the selected clock source is available and the corresponding low-power enable bit is set, then the EPIT continues to function in the low-power mode.

Care should be taken when operating in low-power mode to ensure that any register read/write operations are only done if the selected clock source running the counter is synchronized to ipg_clk_s. If the clock selected is not available or the enable bit is not set then the counter value freezes at its current value and resumes counting once low-power mode is exited.

33.6.1.4 Debug Mode Behavior

In debug mode, the EPIT timers have the option of continuing to run or be halted. If the DBGEN bit is set in the EPIT Control Register, the timer is halted. When exiting debug mode, the timer operation reverts to the state it was in before entering debug mode.

Chapter 34
General Purpose Timer (GPT)

The General Purpose Timer (GPT) is a 32-bit up-counter whose counter value can be captured in a register when a programmed compare value is reached or when an event trigger is applied to an external pin. Figure 34-1 shows the block diagram of the GPT outlining its basic functionality.

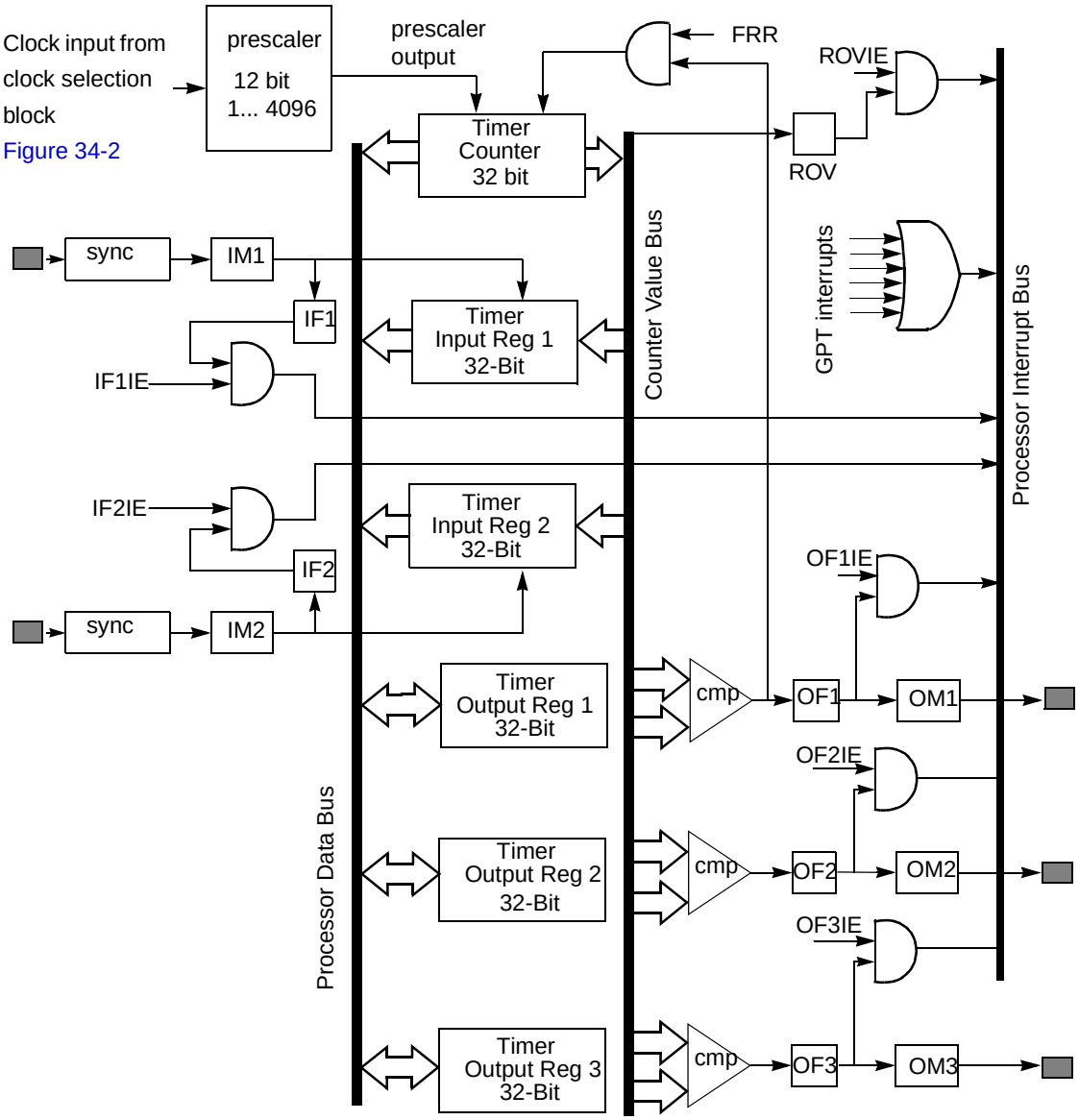


Figure 34-1. General Purpose Timer (GPT) Block Diagram

Figure 34-2 shows the functional clocking scheme for the GPT module.

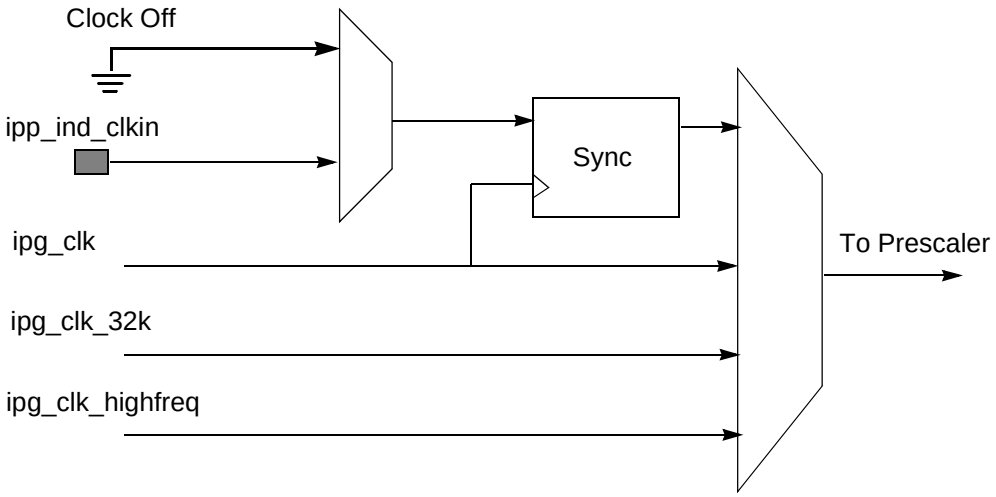


Figure 34-2. GPT Counter Clocks Diagram

34.1 Overview

The GPT consists of a 32-bit up-counter. The timer counter value can be captured in a register when a programmed compare value is reached or by using either an event trigger on an external pin. The capture trigger can be programmed to be a rising or/and falling edge. The GPT can also generate an event on a chip boundary signal and an interrupt when the timer reaches a programmed value. It has a 12-bit prescaler providing a programmable clock frequency derived from multiple clock sources.

34.1.1 Features

The following are the GPT features:

- One 32-bit up-counter with clock source selection, including external clock
- Two input capture channels with programmable trigger edge
- Three output compare channels with programmable output mode. A forced compare feature is available
- Can be programmed to be active in low-power and debug modes
- Interrupt generation at capture, compare, rollover events
- Restart or free-run modes for counter operation

34.1.2 Modes of Operation

The GPT can be programmed to run in either free-run or restart modes, as follows:

- Restart mode—When operating in restart mode, (selected through the control register), every time the counter reaches the compare value it resets and begins counting from 0x00000000. This feature is only available on compare channel 1. Any write access to the channel 1 compare register resets the GPT counter to avoid missing a compare event when compare value changes from a higher to

a lower value while counting. Restart mode is not available for the counter in Compare channels 2 and 3.

- Free-run mode—In free-run mode, the counter value is not reset when a compare event occurs. The counter continues to count till 0xFFFFFFFF and rolls over to 0x00000000.

34.2 Signal Description

34.2.1 Overview

The GPT uses standard IP Bus protocol for interfacing with the ARM11 and does not communicate with any other module inside the IC except for clock and reset inputs from the Clock and Controller module and interrupt signals to the processor interrupt handler. There are functional and clock inputs and functional output signals going outside the chip boundary.

Figure 34-3 shows the GPT signals at the module boundary.

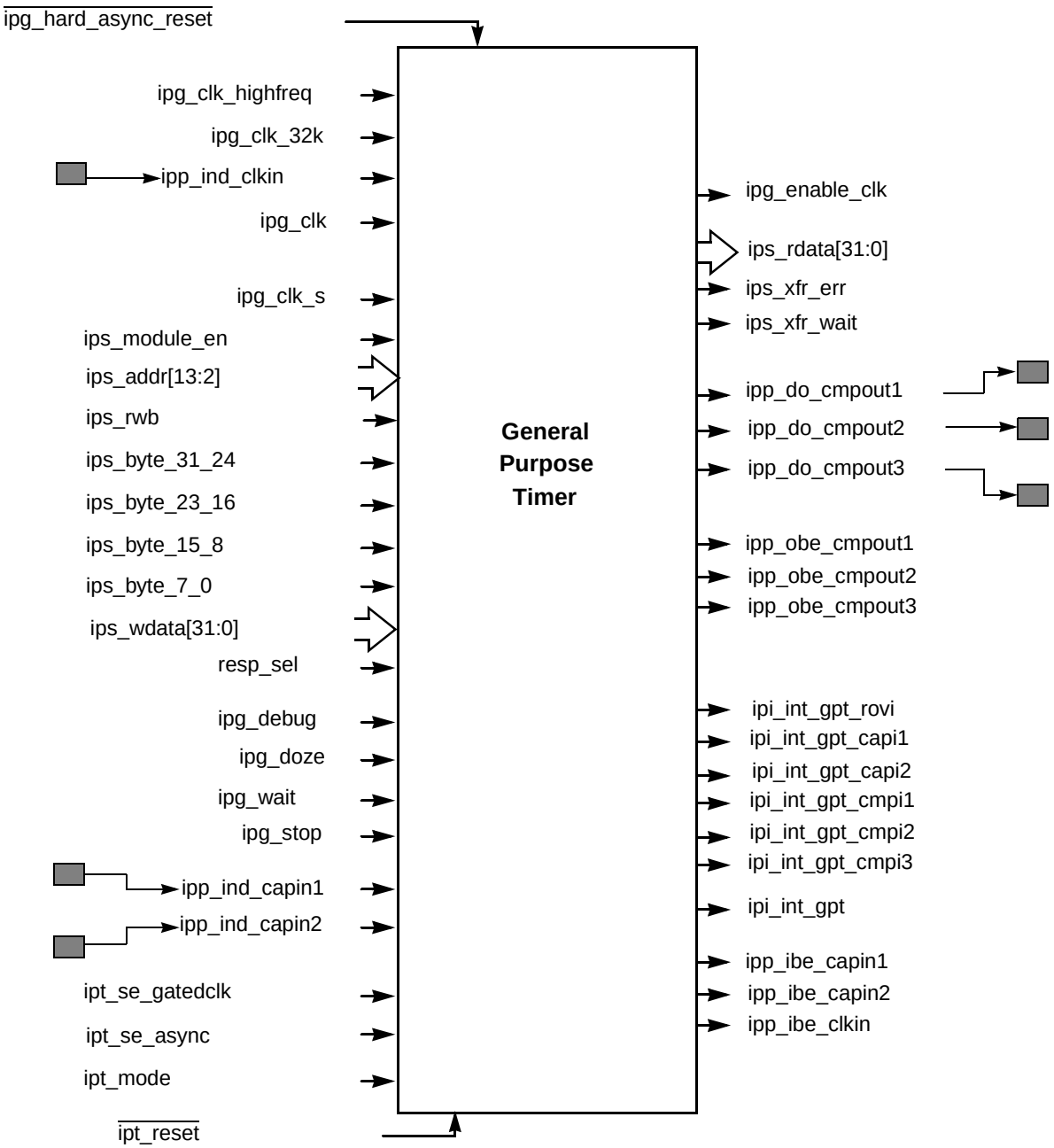


Figure 34-3. GPT Module Signals

34.2.2 External Signals

There are six external signals (three input and three output) in the GPT module. Table 34-1 describes these signals.

Table 34-1. External Signal Description

Name	Direction	Function	Reset State	Pull Up
ipp_ind_clkin	Input	Input pin for external clock on which can be used to run the counter.	—	Passive Hysteresis
ipp_ind_capin1	Input	Input pin for capture event–capture channel 1	—	Passive
ipp_ind_capin2	Input	Input pin for capture event–capture channel 2	—	Passive
ipp_do_cmpout1	Output	Output pin which indicates a compare event in compare channel 1	0	Passive
ipp_do_cmpout2	Output	Output pin which indicates a compare event in compare channel 2	0	Passive
ipp_do_cmpout3	Output	Output pin which indicates a compare event in compare channel 3	0	Passive

34.2.2.1 External Clock Input (ipp_ind_clkin)

The GPT counter can be run using an external clock from outside the IC connected to this pin. This clock is treated as asynchronous to ipg_clk. Its frequency should be less than 1/4 of frequency of ipg_clk for proper functioning of GPT. Hysteresis characteristics on this pad will be required as this is a clock input.

34.2.2.2 Input Capture Trigger Signals (ipp_ind_capin1, ipp_ind_capin2)

The GPT counter value can be stored into a register with an event from outside the IC. A positive or/and negative edge on these signals can trigger this capture event. These signals are treated as asynchronous to ipg_clk. Only those transitions which occur at least a single clock cycle (clock selected to run the counter) after the previous recorded transition will be guaranteed to trigger a capture event.

34.2.2.3 Output Compare Signals (ipp_do_cmpout1, ipp_do_cmpout2, ipp_do_cmpout3)

These signals indicate an output compare event has occurred through a specified transition.

34.2.3 GPT Module Internal Signals

There are five clock inputs, seven interrupt lines, one hardware reset line, and four scan signals. The remaining signals are typical IP slave bus signals.

Table 34-2 provides the GPT module signal descriptions.

Table 34-2. Module Signal Description

Name	Direction	Function	Reset State
ipg_clk_highfreq	Input	High frequency clock which continues to run in low-power mode. The Clock Controller Module (CCM) provides this clock synchronized to ahb_clk in the normal mode and switches to normal mode and switches to a non-synchronized signal when the ahb_clk is off in low-power mode.	—
ipg_clk_32k	Input	Low frequency clock which continues to run in low-power mode. CCM provides this clock synchronous to ahb_clk in the normal mode and switches to a non-synchronized signal when the ahb_clk is off in low-power mode.	—
ipg_clk	Input	IP global functional clock. This clock should be always on except in low power mode.	—
ipg_clk_s	Input	IP slave bus clock. This clock is synchronized to ipg_clk and is used only for register read/write operations.	—
ips_module_en	Input	IP slave bus module enable signal. This signal indicates when a module bus transaction is occurring.	—
ips_addr[13:2]	Input	IP slave bus address signal. Denotes which register is being accessed during a bus transaction	—
ips_rwb	Input	IP slave bus read/write signal. Shows whether a bus transaction is read or write. (Active low)	—
ips_byte_31_24	Input	IP slave bus byte enable signal for bits 31 to 24	—
ips_byte_23_16	Input	IP slave bus byte enable signal for bits 23 to 16	—
ips_byte_15_8	Input	IP slave bus byte enable signal for bits 15 to 8	—
ips_byte_7_0	Input	IP slave bus byte enable signal for bits 7 to 0	—
ips_wdata[31:0]	Input	IP slave bus write date line	—
resp_sel	Input	Signal to determine ips_xfr_err generation conditions If set, error will not be generated if unimplemented register space is accessed. If not set error will be generated	—
ipg_debug	Input	IP global signal to indicate that GPT should enter debug mode operation	—
ipg_doze	Input	IP global signal to indicate that GPT should enter low-power doze mode operation	—
ipg_wait	Input	IP global signal to indicate that GPT should enter low-power wait mode operation	—
ipg_stop	Input	IP global signal to indicate that GPT should enter low-power stop mode operation	—
ipt_se_gatedclk	Input	IP scan signal for bypassing of clock gating in scan mode	—
ipt_se_async	Input	IP scan signal for bypassing generated resets and making latches transparent in scan mode	—
ipt_mode	Input	IP scan reset signal used to put module in scan mode	—

Table 34-2. Module Signal Description (continued)

Name	Direction	Function	Reset State
$\overline{\text{ipt\_reset}}$	Input	IP scan reset signal used to work in place of generated resets in scan mode (Active low)	—
$\overline{\text{ipg\_hard\_async\_reset}}$	Input	IP global hardware reset (Active low)	—
ipg_enable_clk	Output	IP global clock gating signal. It can be used by the CCM to gate off ipg_clk to the GPT for power saving	0
ips_rdata[31:0]	Output	IP slave bus read date line	0
ips_xfr_err	Output	IP slave bus transfer error indicator	0
ips_xfr_wait	Output	IP slave bus transfer wait indicator	0
ipi_int_gpt_rovi	Output	GPT rollover interrupt line	0
ipi_int_gpt_capi1	Output	GPT Input capture channel 1 interrupt line	0
ipi_int_gpt_capi2	Output	GPT Input capture channel 2 interrupt line	0
ipi_int_gpt_cmpi1	Output	GPT Output Compare channel 1 interrupt line	0
ipi_int_gpt_cmpi2	Output	GPT Output Compare channel 2 interrupt line	0
ipi_int_gpt_cmpi3	Output	GPT Output Compare channel 3 interrupt line	0
ipi_int_gpt	Output	Cumulative interrupt line. Any interrupt on any of the 6 interrupt lines will be reflected on this line also	0
ipp_ibe_capin1	Output	Input enable for ipp_ind_capin1 pad signal	0
ipp_ibe_capin2	Output	Input enable for ipp_ind_capin2 pad signal	0
ipp_ibe_clkin	Output	Input enable for ipp_ind_clkin pad signal	0
ipp_obe_cmpout1	Output	Output enable for ipp_do_cmpout1 pad signal	0
ipp_obe_cmpout2	Output	Output enable for ipp_do_cmpout2 pad signal	0
ipp_obe_cmpout3	Output	Output enable for ipp_do_cmpout3 pad signal	0

34.3 Register Definition and Memory Map

The GPT has 10 user accessible 32-bit registers used to configure, operate and monitor the state of the GPT. Section 34.3.3, “Register Descriptions” provides the detailed descriptions for all of the GPT registers.

34.3.1 Memory Map

Table 34-3 shows the GPT memory map.

Table 34-3. GPT Memory Map

Address	Register	Access	Reset Value	Section/Page
0x53F9_0000 (GPTCR)	GPT Control Register (GPTCR)	R/W	0x0000_0000	34.3.3.1/34-10
0x53F9_0004 (GPTPR)	GPT Prescaler Register (GPTPR)	R/W	0x0000_0000	34.3.3.2/34-14
0x53F9_0008 (GPTSR)	GPT Status Register (GPTSR)	R/W	0x0000_0000	34.3.3.3/34-14
0x53F9_000C (GPTIR)	GPT Interrupt Register (GPTIR)	R/W	0x0000_0000	34.3.3.4/34-15
0x53F9_0010 (GPTOCR1)	GPT Output Compare Register 1 (GPTOCR1)	R/W	0xFFFF_FFFF	34.3.3.5/34-16
0x53F9_0014 (GPTOCR2)	GPT Output Compare Register 2 (GPTOCR2)	R/W	0xFFFF_FFFF	34.3.3.6/34-17
0x53F9_0018 (GPTOCR3)	GPT Output Compare Register 3 (GPTOCR3)	R/W	0xFFFF_FFFF	34.3.3.7/34-18
0x53F9_001C (GPTICR1)	GPT Input Capture Register 1 (GPTICR1)	R	0x0000_0000	34.3.3.8/34-18
0x53F9_20 (GPTICR2)	GPT Input Capture Register 2 (GPTICR2)	R	0x0000_0000	34.3.3.9/34-19
0x53F9_0024 (GPTCNT)	GPT Counter Register (GPTCNT)	R	0x0000_0000	34.3.3.10/34-20

34.3.2 Register Summary

Figure 34-4 shows the key to the register fields, and Table 34-4 shows the register figure conventions.

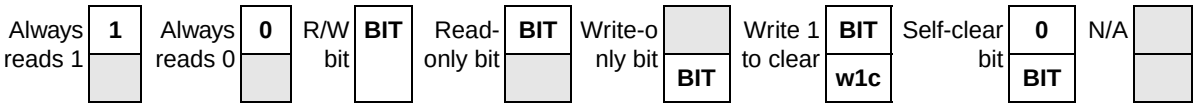


Figure 34-4. Key to Register Fields

Table 34-4. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.

Table 34-4. Register Figure Conventions (continued)

Convention	Description
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 34-5 shows the GPT register summary.

Table 34-5. GPT Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x53F9_0000 (GPTCR)	R	0	0	0	OM3			OM2			OM1			IM2		IM1	
	W	FO3	FO2	FO1													
	R	SWR	0	0	0	0	0	FRR	CLKSRC			STOPEN	DOZEN	WAITEN	DBGEN	ENMOD	EN
	W																
0x53F9_0004 (GPTPR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	PRESCALER											
	W																
0x53F9_0008 (GPTSR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	ROV	IF2	IF1	OF3	OF2	OF1
	W											w1c	w1c	w1c	w1c	w1c	w1c
0x53F9_000C (GPTIR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	ROVIE	IF2IE	IF1IE	OF3IE	OF2IE	OF1IE
	W																

Table 34-5. GPT Register Summary (continued)

Name		31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0x53F9_0010 (GPTOCR1)	R	COMP[31:16]
	W	
	R	COMP[15:0]
	W	
0x53F9_0014 (GPTOCR2)	R	COMP[31:16]
	W	
	R	COMP[15:0]
	W	
0x53F9_0018 (GPTOCR3)	R	COMP[31:16]
	W	
	R	COMP[15:0]
	W	
0x53F9_001C (GPTICR1)	R	CAPT[31:16]
	W	
	R	CAPT[15:0]
	W	
0x53F9_20 (GPTICR2)	R	CAPT[31:16]
	W	
	R	CAPT[15:0]
	W	
0x53F9_0024 (GPTCNT)	R	COUNT[31:16]
	W	
	R	COUNT[15:0]
	W	

34.3.3 Register Descriptions

This section contains the detailed register descriptions for the GPT registers.

34.3.3.1 GPT Control Register (GPTCR)

The GPT Control Register (GPTCR) is used to program and configure the GPT for appropriate operation and control of its features. Figure 34-5 shows the register; Table 34-6 provides its field descriptions.

0x53F9_0000 (GPTCR) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	OM3			OM2			OM1			IM2		IM1	
W	FO3	FO2	FO1													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SWR	0	0	0	0	0	FRR	CLKSRC			STOP EN	DOZE N	WAIT EN	DBG EN	ENM OD	EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 34-5. GPT Control Register (GPTCR)

Table 34-6. GPT Control Register Field Descriptions

Field	Description
31 FO3	Force Output Compare Channel 3. The bit determines the action of the timer output compare 3 pin (according to the bits OM3 in this register). The OF3 flag in the status register is not affected. This bit is self negating and always read as zero. 0 Writing a 0 has no effect 1 Causes pin action on timer output compare 3 pin, OF3 flag is not set
30 F02	Force Output Compare Channel 2. The bit causes the pin action programmed for the timer output compare 2 pin (according to the bits OM3 in this register). The OF2 flag in the status register is not affected. This bit is self negating and always read as zero. 0 Writing a 0 has no effect. 1 Causes pin action on timer output compare 2 pin, OF2 flag is not set
29 FO1	Force Output Compare Channel 1. The bit causes the pin action programmed for the timer output compare 1 pin (according to the bits OM3 in this register). The OF1 flag in the status register is not affected. This bit is self negating and always read as zero. 0 Writing a 0 has no effect. 1 Causes pin action on timer output compare 1pin, OF1 flag is not set
28–26 OM3	Output Compare Channel 3 Operating Mode. This bit field specifies the response that a compare event will generate on the output pin of Output Compare channel 3. The toggle, clear and set options cause a change in the output pin only on occurrence of a compare event. When OM3 is programmed as 1xx (active low pulse), the output pin is set to one, immediately on the next input clock, and the low pulse occurs when there is a compare event. This low pulse width is input clock wide. Here input clock means clock selected by CLKSRC bits of GPT Control Register. 000 Output disconnected. No response on pin 001 Toggle output pin 010 Clear output pin 011 Set output pin 100 Generate a single count duration pulse on output 101 Generate a single count duration pulse on output 110 Generate a single count duration pulse on output 111 Generate a single count duration pulse on output

Table 34-6. GPT Control Register Field Descriptions (continued)

Field	Description
25–23 OM2	Output Compare Channel 2 Operating Mode. This bit field specifies the response that a compare event will generate on the output pin of Output Compare channel 2. The toggle, clear and set options cause a change in the output pin only on occurrence of a compare event. When OM2 is programmed as 1xx (active low pulse), the output pin is set to one, immediately on the next input clock, and the low pulse occurs when there is a compare event. This low pulse width is input clock wide. Here input clock means clock selected by CLKSRC bits of GPT Control Register. 000 Output disconnected. No response on pin 001 Toggle output pin 010 Clear output pin 011 Set output pin 100 Generate a single count duration pulse on output 101 Generate a single count duration pulse on output 110 Generate a single count duration pulse on output 111 Generate a single count duration pulse on output
22–20 OM1	Output Compare Channel 1 Operating Mode. This bit field specifies the response that a compare event will generate on the output pin of Output Compare channel 1. The toggle, clear and set options cause a change in the output pin only on occurrence of a compare event. When OM1 is programmed as 1xx (active low pulse), the output pin is set to one, immediately on the next input clock, and the low pulse occurs when there is a compare event. This low pulse width is input clock wide. Here input clock means clock selected by CLKSRC bits of GPT Control Register. 000 Output disconnected. No response on pin 001 Toggle output pin 010 Clear output pin 011 Set output pin 100 Generate a single count duration pulse on output 101 Generate a single count duration pulse on output 110 Generate a single count duration pulse on output 111 Generate a single count duration pulse on output
19–18 IM2	Input Capture Channel 2 Operating Mode. This bit field determines the transition on the input pin for Input capture channel 2 which will trigger a capture event. 00 Capture disabled 01 Capture only on rising edge 10 Capture only on falling edge 11 Capture on both edges
17–16 IM1	Input Capture Channel 1 Operating Mode. This bit field determines the transition on the input pin for Input capture channel 1 which will trigger a capture event. 00 Capture disabled 01 Capture only on rising edge 10 Capture only on falling edge 11 Capture on both edges
15 SWR	Software Reset. This is the software reset of the GPT module. It is a self-clearing bit. This bit is set when the module is in reset state and is cleared when the reset procedure is over. Writing a 1 to this bit produces a single wait state write cycle. Setting this bit resets all the registers to their default reset values except for the EN, ENMOD, STOPEN, DOZEN, WAITEN and DBGEN bits in this control register. 0 GPT is not in reset state 1 GPT is in reset state
14–10	Reserved bits. These are reserved bits and writing a value will not affect the functionality of GPT. These reserved bits are always read as zero.

Table 34-6. GPT Control Register Field Descriptions (continued)

Field	Description
9 FRR	Freerun or Restart mode. This bit determines the behavior of the GPT when a compare event on channel 1. In restart mode the counter resets to 0x00000000 and resumes counting after the occurrence of a compare event. In freerun mode, after a compare event the counter continues counting till 0xFFFFFFFF and then rolls over to 0. 0 Restart mode 1 Freerun mode
8–6 CLKSRC	Clock Source Select. These bits selects the clock source for the prescaler and subsequently be used to run the GPT counter. **Note:** This bit field value should only be changed after disabling the GPT. If GPT is enabled during this period, then internal state of GPT may become unpredictable. 000 No clock source selected 001 ipg_clk 010 ipg_clk_highfreq 011 not defined 100 ipg_clk_32k 101 not defined 110 not defined 111 not defined
5 STOPEN	GPT Stop Mode Enable. This read/write control bit enables the operation of the GPT during stop mode. This bit is reset by a hardware reset. A software reset does not affect this bit. 0 GPT is disabled in stop mode 1 GPT is enabled in stop mode
4 DOZEN	GPT Doze Mode Enable. This read/write control bit enables the operation of the GPT during doze mode. This bit is reset by a hardware reset. A software reset does not affect this bit. 0 GPT is disabled in doze mode 1 GPT is enabled in doze mode
3 WAITEN	GPT Wait Mode Enable. This read/write control bit enables the operation of the GPT during wait mode. This bit is reset by a hardware reset. A software reset does not affect this bit. 0 GPT is disabled in wait mode 1 GPT is enabled in wait mode
2 DBGEN	GPT Debug Mode Enable. This read/write control bit enables the operation of the GPT during debug mode. This bit is reset by a hardware reset. A software reset does not affect this bit 0 GPT is disabled in debug mode 1 GPT is enabled in debug mode
1 ENMODE	GPT Enable Mode. When GPT is disabled(EN=0), then GPT Main Counter freezes its count at current count value and Prescaler Counter is reset. ENMOD bit determines value of GPT counter when EN bit is set and Counter is enabled again. If ENMOD bit is set then the Main Counter is reset to 0, when GPT is enabled(EN=1). If ENMOD bit is programmed to 0, then the Main Counter restarts counting from frozen value, when GPT is enabled again(EN=1). Setting the SWR bit will clear the counter value regardless of the value of EN or ENMOD bits. This bit is reset by a hardware reset. A software reset does not affect this bit 0 GPT counter retains its value when it is disabled. 1 GPT counter value is reset to 0 when it is disabled.
0 EN	GPT Enable. This bit is the module enable bit. It is recommended that all registers be properly programmed before setting this bit. This bit is reset by a hardware reset. A software reset does not affect this bit. 0 GPT is disabled. 1 GPT is enabled.

34.3.3.2 GPT Prescaler Register (GPTPR)

The GPT Prescaler Register (GPTPR) contains bits that determine the divide value of the clock that runs the counter. Figure 34-6 shows the register; Table 34-7 provides its field descriptions.

0x53F9_0004 (GPTPR) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	PRESCALER											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 34-6. GPT Prescaler Register

Table 34-7. GPT Prescaler Register Field Descriptions

Field	Description
31–12	Reserved. These are reserved bits and writing a value will not affect the functionality of GPT. These reserved bits are always read as zero.
11–0 PRESCALER	This field contains the divisor for the clock source selected by the CLKSRC field. The clock signal is divided by [PRESCALER + 1] and used to run the counter. 0x000 Divide by 1 0x001 Divide by 2 —— 0xFFF Divide by 4096

34.3.3.3 GPT Status Register (GPTSR)

The GPT Status Register (GPTSR) contains bits that indicate the occurrence of rollover of the counter and any event on input capture and output compare channels. The bits are write one to clear.

Figure 34-7 shows the register; Table 34-8 provides its field descriptions.

0x53F9_0008 (GPTSR) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	ROV	IF2	IF1	OF3	OF2	OF1
W											w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 34-7. GPT Status Register

Table 34-8. GPT Status Register Field Descriptions

Field	Description
31–6	Reserved. These are reserved bits and writing a value will not affect the functionality of GPT. These reserved bits are always read as zero.
5 ROV	Rollover Flag. This bit indicates that the counter has reached its maximum possible value and rolled over to 0 from which it continues counting. This bit is only set if the counter has reached 0xFFFFFFFF in both restart and freerun modes. 0 Rollover has not occurred. 1 Rollover has occurred.
4 IF2	Input Capture 2 Flag. This bit indicates that a capture event has occurred on Input Capture channel 2. 0 Capture event has not occurred. 1 Capture event has occurred.
3 IF1	Input Capture 1 Flag. This bit indicates that a capture event has occurred on Input Capture channel 1. 0 Capture event has not occurred. 1 Capture event has occurred.
2 OF3	Output Compare 3 Flag. This bit indicates that a compare event has occurred on Output Compare channel 3. 0 Compare event has not occurred. 1 Compare event has occurred.
1 OF2	Output Compare 2 Flag. This bit indicates that a compare event has occurred on Output Compare channel 3. 0 Compare event has not occurred. 1 Compare event has occurred.
0 OF1	Output Compare 1Flag. This bit indicates that a compare event has occurred on Output Compare channel 3. 0 Compare event has not occurred. 1 Compare event has occurred.

34.3.3.4 GPT Interrupt Register (GPTIR)

The GPT Interrupt Register (GPTIR) contains bits that control the generation of interrupt on rollover, input capture and output compare events.

Figure 34-8 shows the register; Table 34-9 provides its field descriptions.

0x53F9_000C (GPTIR) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	ROVIE	IF2IE	IF1IE	OF3IE	OF2IE	OF1IE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 34-8. GPT Interrupt Register

Table 34-9. GPT Interrupt Register Field Descriptions

Field	Description
31–6	Reserved. These are reserved bits and writing a value will not affect the functionality of GPT. These reserved bits are always read as zero.
5 ROV	Rollover Interrupt Enable. This bit controls the occurrence of rollover interrupt. 0 Interrupt not enabled 1 Interrupt enabled
4 IF2IE	Input Capture 2 Interrupt Enable. This bit controls the occurrence of input capture channel 2 interrupt. 0 Interrupt disabled 1 Interrupt enabled
3 IF1IE	Input Capture 1 Interrupt Enable. This bit controls the occurrence of input capture channel 1interrupt. 0 Interrupt disabled 1 Interrupt enabled
2 OF3IE	Output Compare 3 Interrupt Enable. This bit controls the occurrence of output compare channel 3 interrupt. 0 Interrupt disabled 1 Interrupt enabled
1 OF2IE	Output Compare 2Interrupt Enable. This bit controls the occurrence of output compare channel 2 interrupt. 0 Interrupt disabled 1 Interrupt enabled
0 OF1IE	Output Compare 1Interrupt Enable. This bit controls the occurrence of output compare channel 1 interrupt. 0 Interrupt disabled 1 Interrupt enabled

34.3.3.5 GPT Output Compare Register 1 (GPTOCR1)

The GPT Compare Register 1 (GPTOCR1) holds the value that determines when a compare event will be generated on output compare channel 1. Any write access to the compare register of channel 1 while in restart mode(FRR=0) will result in the GPT counter being reset.

IP Bus Write access to GPT Output Compare Register1 (GPTOCR1) results in one cycle of wait state (ips_xfr_wait high for 1 cycle), while IP Bus Read access is with 0 wait state.

Figure 34-9 shows the register; Table 34-10 provides its field descriptions.

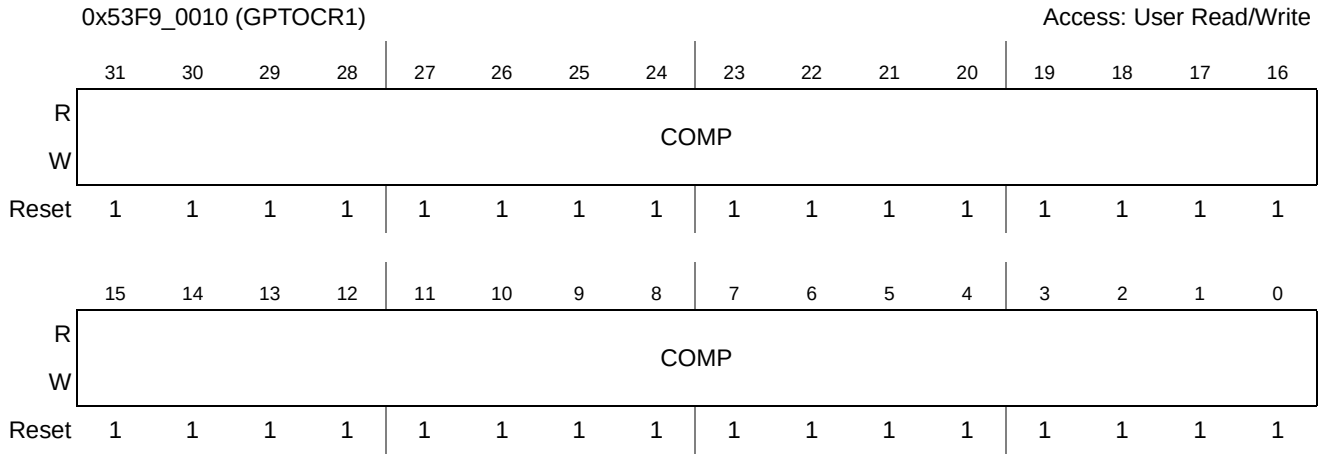


Figure 34-9. GPT Output Compare Register 1

Table 34-10. GPT Output Compare Register 1 Field Descriptions

Field	Description
31–0 COMP	Compare Value. When the counter value equals this bit field value, a compare event is generated on output compare channel 1.

34.3.3.6 GPT Output Compare Register 2 (GPTOCR2)

The GPT Compare Register 2 (GPTOCR2) holds the value that determines when a compare event will be generated on output compare channel 2.

Figure 34-10 shows the register; Table 34-11 provides its field descriptions.

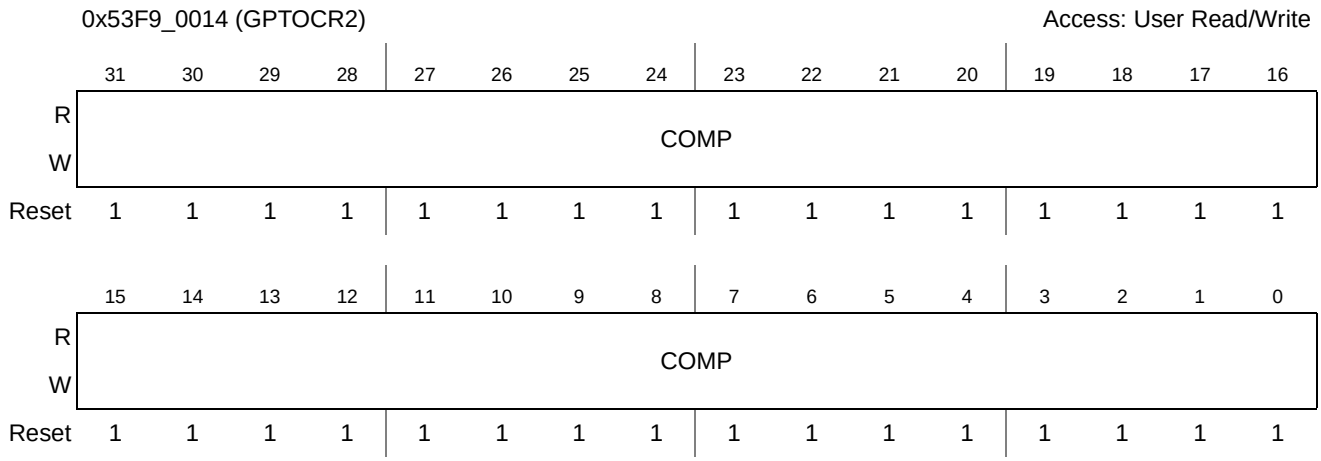


Figure 34-10. GPT Output Compare Register 2

Table 34-11. GPT Output Compare Register 2 Field Descriptions

Field	Description
31–0 COMP	Compare Value. When the counter value equals this bit field value, a compare event is generated on output compare channel 2.

34.3.3.7 GPT Output Compare Register 3 (GPTOCR3)

The GPT Compare Register 3 (GPTOCR3) holds the value that determines when a compare event will be generated on output compare channel 3.

Figure 34-11 shows the register; Table 34-12 provides its field descriptions.

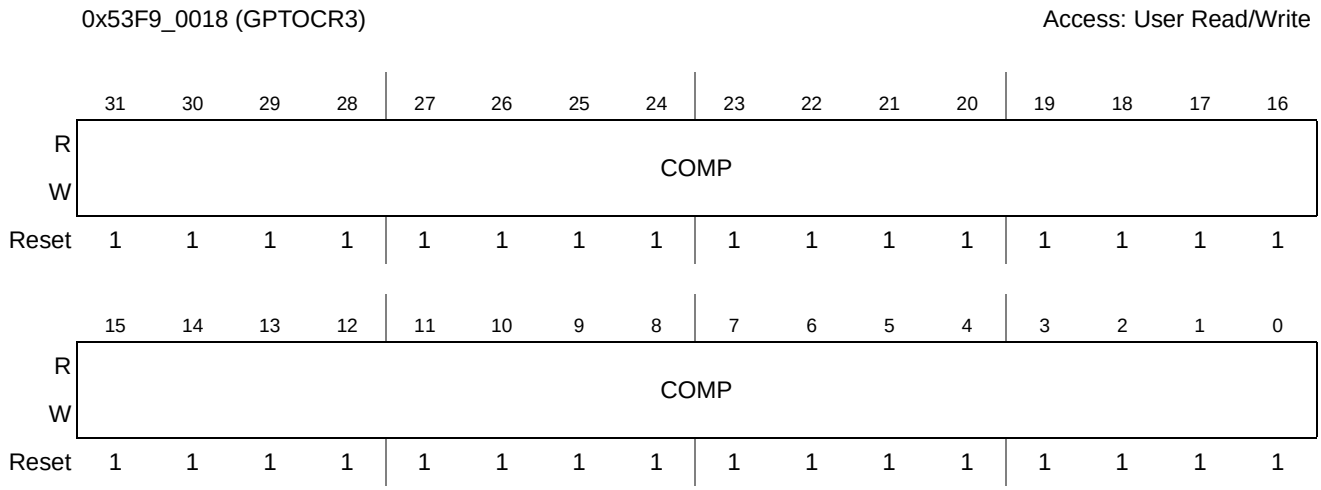


Figure 34-11. GPT Output Compare Register 3

Table 34-12. GPT Output Compare Register 3 Field Descriptions

Field	Description
31–0 COMP	Compare Value. When the counter value equals this bit field value, a compare event is generated on output compare channel 1.

34.3.3.8 GPT Input Capture Register 1 (GPTICR1)

The GPT Input capture Register 1 (GPTICR1) is a read-only register which holds the value that was in the counter during the last capture event on input capture channel 1.

Figure 34-12 shows the register; Table 34-13 provides its field descriptions.

0x53F9_001C (GPTICR1) Access: User read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CAPT[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CAPT[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 34-12. GPT Input Capture Register 1

Table 34-13. GPT Input Capture Register 1 Field Descriptions

Field	Description
31–0 CAPT	Capture Value. When a capture event occurs on Input capture channel 1, the current value of the counter is loaded into this register.

34.3.3.9 GPT Input Capture Register 2 (GPTICR2)

The GPT Input capture Register 2 (GPTICR2) is a read-only register that holds the value that was in the counter during the last capture event on input capture channel 2.

Figure 34-13 shows the register; Table 34-14 provides its field descriptions.

0x53F9_20 (GPTICR2) Access: User read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CAPT[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CAPT[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 34-13. GPT Input Capture Register 2

Table 34-14. GPT Input Capture Register 2 Field Descriptions

Field	Description
31–0 CAPT	Capture Value. When a capture event occurs on Input capture channel 2, the current value of the counter is loaded into this register.

34.3.3.10 GPT Counter Register (GPTCNT)

The GPT Counter Register (GPTCNT) is the main counter register. It is a read only register and can be read without affecting the counting process of the GPT.

Figure 34-14 shows the register; Table 34-15 provides its field descriptions.

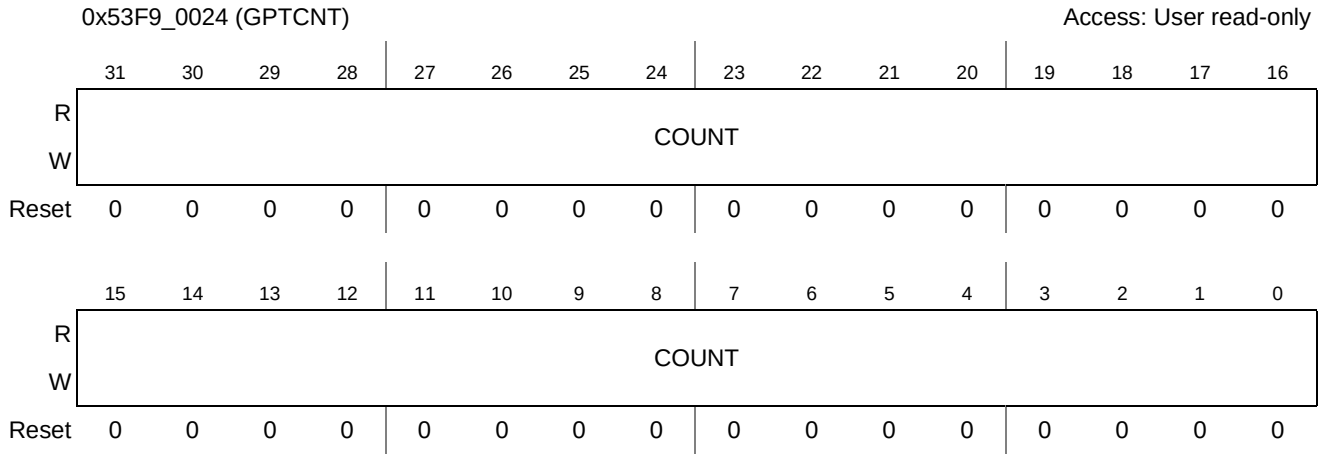


Figure 34-14. GPT Counter Register

Table 34-15. GPT Counter Register Field Descriptions

Field	Description
31–0 COUNT	Counter Value. These bits show the current count value in the counter register.

34.4 Functional Description

This section provides the functional description for the GPT module.

34.4.1 Operation

The GPT has a single counter (GPTCNT). It is a 32-bit free-running up counter which starts counting after it is enabled by software (EN=1). The clock source for the counter is the prescaler output labeled “Prescaler output” in Figure 34-1. If the GPT is disabled (EN=0), the counter freezes at its current value or is reset to 0 according to the setting of the ENMOD bit in the control register. On re-enabling the timers, GPTCNT resumes counting up. The value of the counter (GPTCNT) can be read at any time by the processor. Both input capture channels read the same counter (GPTCNT).

A hardware reset resets all the GPT registers to their respective reset values. All registers except the output compare registers (GPTOCR1,GPTOCR2, and GPTOCR3) reset to a value of 0x00000000. The compare registers are reset to 0xFFFFFFFF. There is a software reset available (SWR bit in control register) which resets all the register bits except the EN, ENMOD, STOPEN, DOZEN, WAITEN and DBGEN bits. The state of these bits are not affected by a software reset. Software reset can be applied even when the GPT is disabled.

34.4.1.1 Clocks

The clock that feeds the prescaler can be selected from the following clock inputs:

High Frequency Clock (ipg_clk_highfreq)

This is a high frequency clock source from the Clock Controller Module (CCM). This clock is available in low-power mode when the ipg_clk is turned off allowing the GPT to use this clock in low-power mode. The CCM synchronizes this clock to ahb_clk in normal mode and it becomes a non-synchronized clock signal in the low-power mode.

Low Frequency Reference Clock (ipg_clk_32k)

This is the 32 KHz low frequency reference clock from the CCM. This clock is available in low-power mode when the ipg_clk is turned off allowing the GPT to use this clock in low-power mode. The CCM synchronizes this clock to ahb_clk in normal mode and it becomes a non-synchronized clock signal in the low-power mode.

External Clock (ipp_ind_clkin)

This is the external clock from outside the IC which can be used to run the counter. This clock is treated as asynchronous to ipg_clk and thus its frequency is limited to < 1/4 frequency of ipg_clk for proper functioning of the GPT.

Global Functional Clock (ipg_clk)

This clock is on in normal operation and in low-power mode it can be switched off.

The clock input to the prescaler can also be disabled by programming the control register. The clock input source is determined by the clock source (CLKSOURCE) field in the control register (Figure 34-15). This field value should be changed only after disabling the GPT.

The prescaler field selects the divide ratio of the input clock that drives the main counter. The prescaler can divide the input clock by a value from 1 to 4096. A change in the value of the prescaler field is immediately reflected in its output clock frequency.

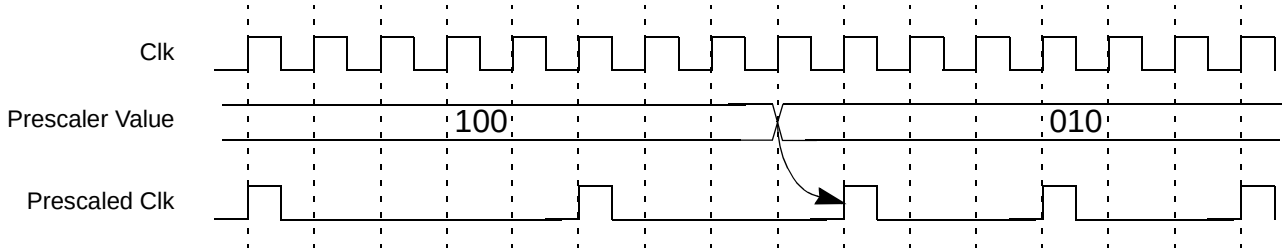
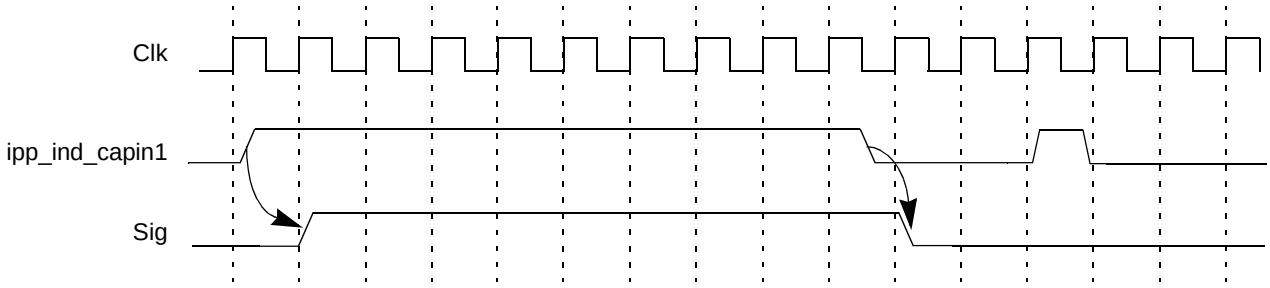


Figure 34-15. Prescaler Value Change Timing Diagram

34.4.1.2 Input Capture

There are two input capture channels and each input capture channel has a dedicated capture pin, capture register and input edge detection/selection logic. Each input capture function has an associated status flag, and can cause the processor to generate an interrupt service request. When a selected edge transition occurs on an input capture pin, the current counter value (GPTCNT) is captured on the corresponding capture register and the associated interrupt status flag is set. An interrupt request can be generated when the transition is detected if its corresponding enable bit is set in the Interrupt Register. The capture can be programmed to occur on the rising edge, falling edge, or on both edges of the input pin or it can be disabled completely. The timer events are synchronized with the clock source selected to run the counter (Figure 34-16). Only those transitions which occur at least a single clock cycle (clock selected to run the counter) after the previous recorded transition will be guaranteed to trigger a capture event. There can be up to one clock cycle of uncertainty in latching of the input transition. The input capture registers can be read at any time without affecting their values.



Clk—The clock selected from CLKSRC bit field setting
ipp_ind_capin1—Pad signal for capture channel 1
Sig—Capture signal as sensed by the module

Figure 34-16. Input Capture Event Timing Diagram

34.4.1.3 Output Compare

The three output compare channels read the same counter (GPTCNT) as the input capture channels. When the programmed content of an output compare register matches the value in GPTCNT, an output compare status flag is set and an interrupt is generated if the corresponding bit is set in the interrupt register. Depending on the Mode bit setting, the output compare timer pin is set, cleared, toggled, not affected at all or it provides an active-low pulse for one count-period (this is subject to the restriction on the maximum

frequency allowed on the pad). There also exists a forced-compare feature allowing the software to generate compare event when required without the condition of the counter value being equal to the compare value. The action taken as a result of a forced compare is the same as when an output compare match occurs, except that the status flags are not set and no interrupt can be generated (Figure 34-17). Forced channels take programmed action immediately after writing to the associated force-compare bit. These bits are self-negating and always read as zeros.

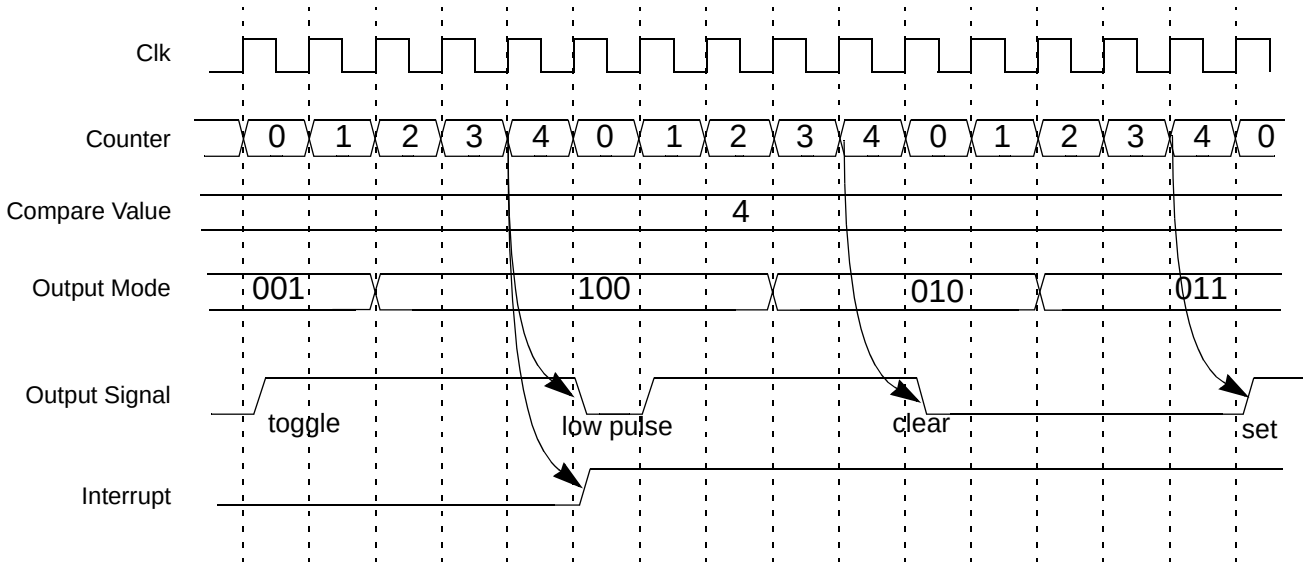


Figure 34-17. Output Compare and Interrupt Timing Diagram

34.4.1.4 Interrupts

There are six different interrupts that are generated by the GPT. All interrupts can be generated in low-power and debug modes if the selected clock for running the counter is available.

- Rollover Interrupt

  This interrupt is generated when the GPT counter reaches 0xFFFFFFFF and resets to 0x00000000 and continues counting. The interrupt is enabled by the ROVIE bit in GPTIR register and its associated status bit is ROV bit in GPTSR register.

- Input Capture Interrupts 1 and 2

  On occurrence of a capture event, interrupts can be generated by each input capture channel. These interrupts are enabled by IF2IE and IF1IE bits in GPTIR register and their associated status bits are IF2 and IF1 in GPTSR register. The capture of the counter value due to a capture event is not affected by a pending capture interrupt. Capture register is updated with new counter value on occurrence of capture event irrespective of that capture channels interrupt has been serviced or not.

- Output Compare Interrupts 1, 2, and 3

  On occurrence of a compare event, interrupts can be generated by each output compare channel. These interrupts are enabled by OF3IE,OF2IE and OF1IE bits in GPTIR register and their associated status bits are OF3,OF2 and OF1 in GPTSR register. No interrupt is generated due to a forced compare.

An cumulative interrupt line is also present which is asserted whenever any of the above interrupts are posted. This has no associated enables or status bits in the GPT module.

34.4.1.5 Low-Power Mode Behavior

In low power modes if the clock from the selected clock source is available (except ipp_ind_clkin which can be used only if ipg_clk is available), the counter continues to run depending on whether the control bit for that mode is set. In absence of the clock itself or the corresponding low power bit in control register being 0, the counter freezes at its current value and resumes counting when the low power mode is exited.

34.4.1.6 Debug Mode Behavior

In debug mode, the modules have the option of continuing to run or be halted. If the DBGEN bit is not set in the GPTCR, the GPT timer is halted. If the DBGEN bit is set, then the GPT timer will continue to run during debug mode.

Chapter 35
Pulse-Width Modulator (PWM)

The pulse-width modulator (PWM) has a 16-bit counter, and is optimized to generate sound from stored sample audio images and it can also generate tones. It uses 16-bit resolution and a 4 x 16 data FIFO to generate sound.

35.1 Overview

This section presents an overview of the PWM. Figure 35-1 illustrates the PWM block diagram.

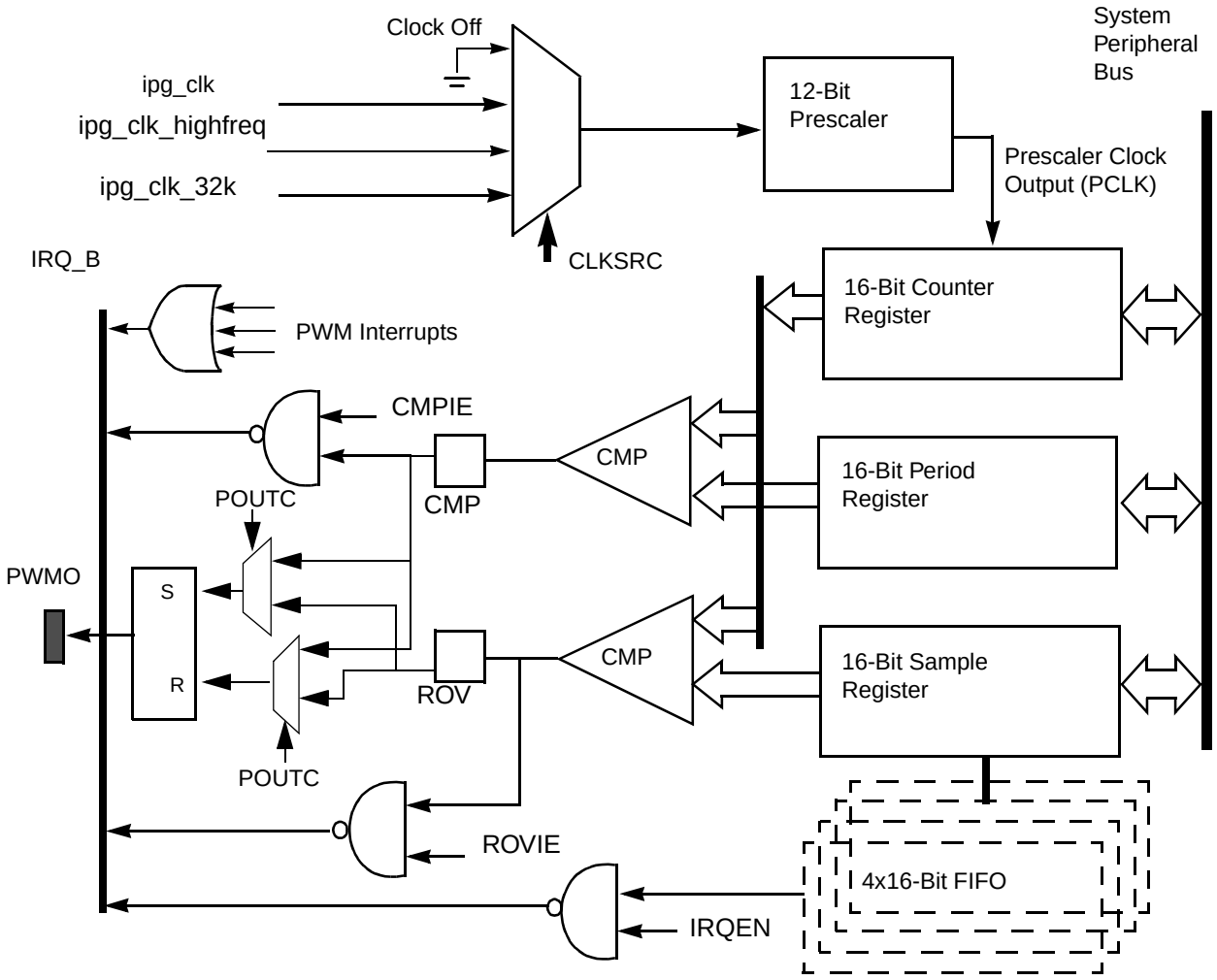


Figure 35-1. Pulse-Width Modulator Block Diagram

The following features characterize the PWM:

- 16-bit up-counter with clock source selection
- 4 x 16 FIFO to minimize interrupt overhead
- 12-bit prescaler for division of clock
- Sound and melody generation
- Active high or active low configurable output
- Can be programmed to be active in low power and debug modes
- Interrupts at compare and rollover

35.2 Signal Description

The PWM follows IP bus protocol for interfacing with the processor core. It does not have any interface signals with any other module inside the chip except for clock and reset inputs from the Clock Control module (CCM) and interrupt signals to the processor interrupt handler. There is a single output signal going outside the chip boundary.

Figure 35-2 shows the PWM signals at the module boundary.

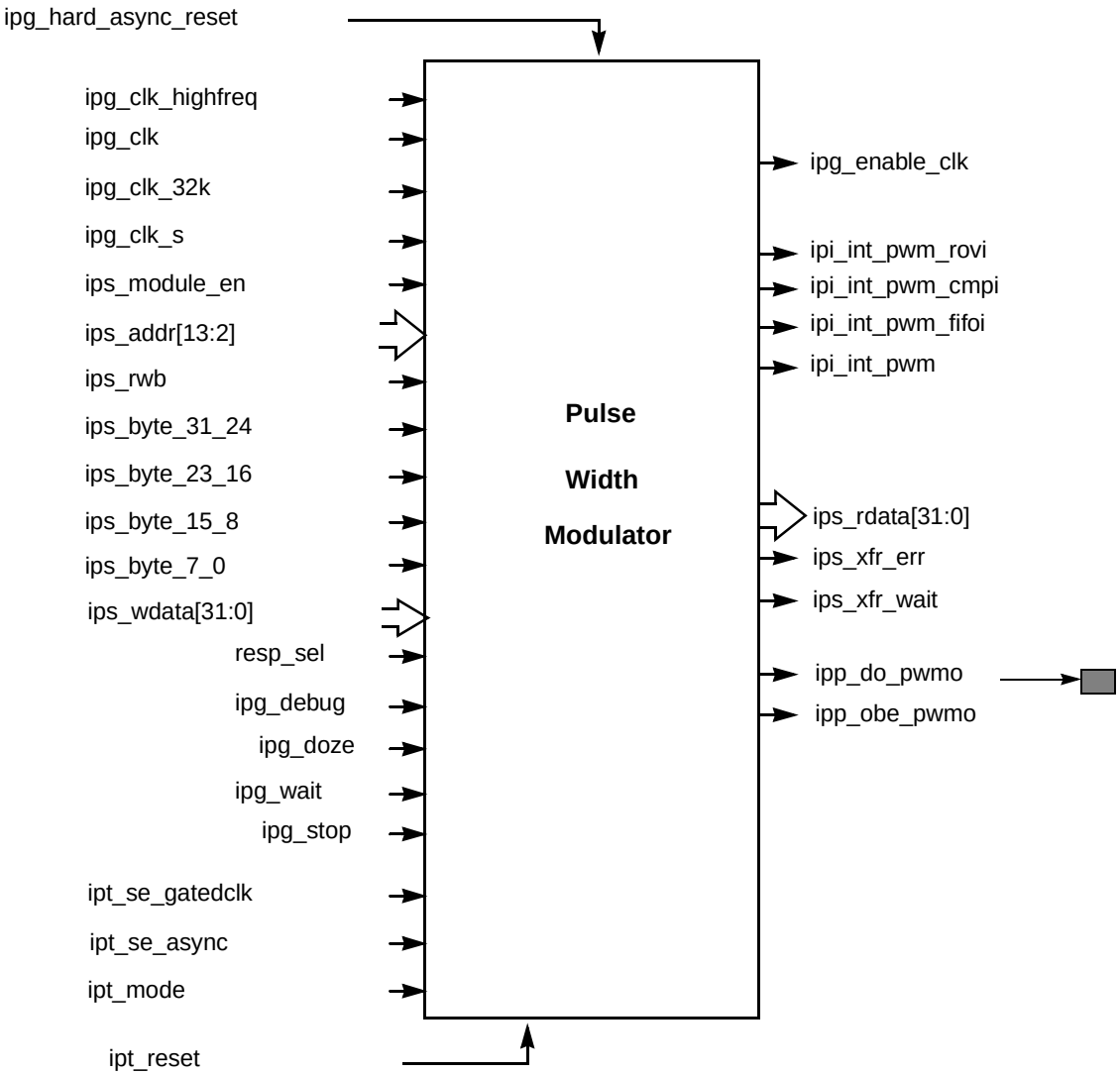


Figure 35-2. PWM Module Signals

35.2.1 External Signals

PWM has a single output signal to the chip boundary named ipp_do_pwmo.

Table 35-1 outlines the external signals.

Table 35-1. External Signals

Name	Direction	Function	Reset State	Pull-Up
ipp_do_pwmo	Output	This is the functional output of the PWM. The modulated signal of the module is observed at this pin	0	—

35.2.1.1 ipp_do_pwmo

The ipp_do_pwmo is the modulated output signal of the PWM. This signal can be viewed as a clock signal whose period and duty cycle can be varied with different settings of the PWM. The smallest period can be two ipg_clk clock periods with duty cycle of 50%.

35.2.2 PWM Module Boundary Signals

There are three clock inputs, four interrupt lines, one hardware reset line, four low power and debug mode signals, and four scan signals. The remaining signals are standard IP slave bus signals. Table 35-2 lists the module boundary signals.

Table 35-2. Module Boundary Signal Description

Name	Direction	Function	Reset State
ipg_clk_highfreq	Input	High frequency clock which will continue to run in low power mode. Clock Controller will provide this clock synchronous to ahb_clk in normal mode and switch to the raw non-synchronous version in low power mode when the ahb_clk is off.	—
ipg_clk_32k	Input	Low frequency clock which will continue to run in low power mode. Clock Controller will provide this clock synchronous to ahb_clk in normal mode and switch to the raw non-synchronous version in low power mode when the ahb_clk is off.	—
ipg_clk	Input	IP global functional clock. This clock should be always on except in low power mode.	—
ipg_clk_s	Input	IP slave bus clock. This clock is synchronous to ipg_clk and is used only for register read/write operations.	—
ips_module_en	Input	IP slave bus module enable signal. This signal indicates when a module bus transaction is occurring.	—
ips_addr[13:2]	Input	IP slave bus address signal. Denotes the register which is being accessed during a bus transaction	—
ips_rwb	Input	IP slave bus read/write signal. Shows whether a bus transaction is read or write. (Active low)	—
ips_byte_31_24	Input	IP slave bus byte enable signal for bits 31 to 24	—
ips_byte_23_16	Input	IP slave bus byte enable signal for bits 23 to 16	—
ips_byte_15_8	Input	IP slave bus byte enable signal for bits 15 to 8	—
ips_byte_7_0	Input	IP slave bus byte enable signal for bits 7 to 0	—
ips_wdata[31:0]	Input	IP slave bus write date line	—
resp_sel	Input	Signal to determine ips_xfr_err generation conditions If set, the error will not be generated if unimplemented register space is accessed. If not set error will be generated	—
ipg_debug	Input	IP global signal to indicate that PWM should enter debug mode operation	—

Table 35-2. Module Boundary Signal Description (continued)

Name	Direction	Function	Reset State
ipg_doze	Input	IP global signal to indicate that PWM should enter low-power doze mode operation	—
ipg_wait	Input	IP global signal to indicate that PWM should enter low-power wait mode operation	—
ipg_stop	Input	IP global signal to indicate that PWM should enter low-power stop mode operation	—
ipt_se_gatedclk	Input	IP scan signal for bypassing of clock gating in scan mode	—
ipt_se_async	Input	IP scan signal for bypassing generated resets and making latches transparent in scan mode	—
ipt_mode	Input	IP scan reset signal used to put module in scan mode	—
ipt_reset	Input	IP scan reset signal used to work in place of generated resets in scan mode (Active low)	—
ipg_hard_async_reset	Input	IP global hardware reset (Active low)	—
ipg_enable_clk	Output	IP global clock gating signal. It can be used by the Clock Controller to gate off the clock to the PWM for power saving. This is essentially the PWM enable bit value	0
ips_rdata[31:0]	Output	IP slave bus read date line	0
ips_xfr_err	Output	IP slave bus transfer error indicator	0
ips_xfr_wait	Output	IP slave bus transfer wait indicator	0
ipi_int_pwm_rovi	Output	PWM rollover interrupt line	0
ipi_int_pwm_cmpi	Output	PWM compare interrupt line	0
ipi_int_pwm_fifoi	Output	PWM FIFO water level crossing interrupt line	0
ipi_int_pwm	Output	Cumulative interrupt line. Any interrupt on any of the 3 interrupt lines will be reflected on this line as well	0
ipp_obe_pwmo	Output	Output enable for ipp_do_pwmo pad signal	0

35.3 Memory Map and Register Definition

The PWM module includes 6 user-accessible 32-bit registers. Section 35.3.2, “Register Descriptions” provides the detailed descriptions for all of the PWM registers.

Table 35-3. PWM Memory Map

Address	Register	Access	Reset Value	Section/Page
0x53FE_0000 (PWMCR)	PWM Control Register (PWMCR)	R/W	0x0000_0000	35.3.2.1/35-8
0x53FE_0004 (PWMSR)	PWM Status Register (PWMSR)	R/W	0x0000_0008	35.3.2.2/35-9
0x53FE_0008 (PWMIR)	PWM Interrupt Register (PWMIR)	R/W	0x0000_0000	35.3.2.3/35-11
0x53FE_000C (PWMSAR)	PWM Sample Register (PWMSAR)	R/W	0x0000_0000	35.3.2.4/35-11

Table 35-3. PWM Memory Map (continued)

Address	Register	Access	Reset Value	Section/Page
0x53FE_0010 (PWMPR)	PWM Period Register (PWMPR)	R/W	0x0000_FFFE	35.3.2.5/35-12
0x53FE_0014 (PWMCNR)	PWM Counter Register (PWMCNR)	R	0x0000_0000	35.3.2.6/35-13

35.3.1 Register Summary

Figure 35-3 shows the key to the register fields, and Table 35-4 shows the register figure conventions.

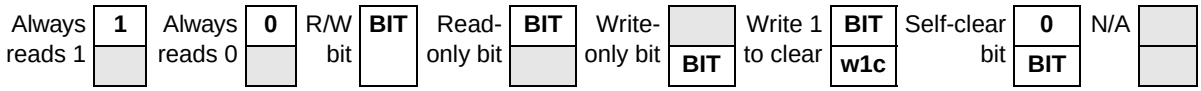


Figure 35-3. Key to Register Fields

Table 35-4. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writeable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 35-5 shows the PWM register summary.

Table 35-5. PWM Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x53FE_0000 (PWMCR)	R	0	0	0	0	FWM		STOPEN	DOZEN	WAITEN	DBGEN	BCTR	HCTR	POUTC		CLKSRC	
	W																
	R	PRESCALER												SWR	REPEAT		EN
	W																
0x53FE_0004 (PWMSR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	FWE	CMP	ROV	FE	FIFOAV		
	W										w1c	w1c	w1c	w1c			
0x53FE_0008 (PWMIR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	CIE	RIE	FIE
	W																
0x53FE_000C (PWMSAR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	SAMPLE[15:0]															
	W																
0x53FE_0010 (PWMPR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	PERIOD[15:0]															
	W																
0x53FE_0014 (PWMCNR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	COUNT[15:0]															
	W																

35.3.2 Register Descriptions

This section contains the detailed register descriptions for the PWM registers.

35.3.2.1 PWM Control Register (PWMCR)

The PWM control register (PWMCR) is used to configure the operating settings of the PWM. It contains the prescaler for the clock division. Figure 35-4 shows the register; Table 35-6 provides its field descriptions.

0x53FE_0000 (PWMCR) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	FWM		STOP EN	DOZ EN	WAIT EN	DBG EN	BCT R	HCT R	POUTC		CLKSRC	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PRESCALER												SWR	REPEAT		EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 35-4. PWM Control Register (PWMCR)

Table 35-6. PWMCR Field Descriptions

Field	Description
31–28 Reserved	Reserved. These reserved bits are always read as zero.
27–26	FIFO Water Mark. These bits are used to set the data level at which the FIFO empty flag will be set and the corresponding interrupt generated 00 FIFO empty flag is set when there are more than or equal to 1 empty slots in FIFO. 01 FIFO empty flag is set when there are more than or equal to 2 empty slots in FIFO. 10 FIFO empty flag is set when there are more than or equal to 3 empty slots in FIFO. 11 FIFO empty flag is set when there are more than or equal to 4 empty slots in FIFO.
25 STOPEN	Stop Mode Enable. This bit keeps the PWM functional while in stop mode. When this bit is cleared, the input clock is gated off in stop mode. This bit is not affected by software reset. It is cleared by hardware reset. 0 Inactive in stop mode 1 Active in stop mode
24 DOZEN	Doze Mode Enable. This bit keeps the PWM functional in doze mode. When this bit is cleared, the input clock is gated off in doze mode. This bit is not affected by software reset. It is cleared by hardware reset. 0 Inactive in doze mode 1 Active in doze mode
23 WAITEN	Wait Mode Enable. This bit keeps the PWM functional in wait mode. When this bit is cleared, the input clock is gated off in wait mode. This bit is not affected by software reset. It is cleared by hardware reset. 0 Inactive in wait mode 1 Active in wait mode
22 DBGEN	Debug Mode Enable. This bit keeps the PWM functional in debug mode. When this bit is cleared, the input clock is gated off in debug mode. This bit is not affected by software reset. It is cleared by hardware reset. 0 Inactive in debug mode 1 Active in debug mode

Table 35-6. PWMCR Field Descriptions (continued)

Field	Description
21 BCTR	Byte Data Swap Control. This bit determines the byte ordering of the 16-bit data when it goes into the FIFO from the sample register. 0 byte ordering remains the same 1 byte ordering is reversed
20 HCTR	Halfword Data Swap Control. This bit determines which halfword data from the 32-bit IP-Bus interface is written into the lower 16 bits of the sample register. 0 Halfword swapping does not take place 1 Halfwords from write data bus are swapped
19–18 POUTC	PWM Output Configuration. This bit field determines the mode of PWM output on the output pin. 00 Output pin is set at rollover and cleared at comparison 01 Output pin is cleared at rollover and set at comparison 10 PWM output is disconnected. 11 PWM output is disconnected.
17–16 CLKSRC	Select Clock Source. These bits determine which clock input will be selected for running the counter. After reset the system functional clock is selected. The input clock can also be turned off if these bits are set to 00. This field value should only be changed when the PWM is disabled 00 Clock is off 01 ipg_clk 10 ipg_clk_highfreq 11 ipg_clk_32k
15–4 PRESCALER	Counter Clock Prescaler Value. This bit field determines the value by which the clock will be divided before it goes to the counter. 0x000 Divide by 1 0x001 Divide by 2 ... 0xFFF Divide by 4096
3 SWR	Software Reset. PWM is reset when this bit is set to 1. It is a self clearing bit. A write 1 to this bit is a single wait state write cycle. When the module is in reset state this bit is set and is cleared when the reset procedure is over. Setting this bit resets all the registers to their reset values except for the STOPEN, DOZEN, WAITEN, and DBGEN bits in this control register. 0 PWM is out of reset 1 PWM is undergoing reset
2–1 REPEAT	Sample Repeat. This bit field determines the number of times each sample from the FIFO is to be used. 00 Use each sample once 01 Use each sample twice 10 Use each sample four times 11 Use each sample eight times
0 EN	PWM Enable. This bit enables the PWM. If this bit is not enabled, the clock prescaler and the counter is reset. When the PWM is enabled, it begins a new period, the output pin is set to start a new period while the prescaler and counter are released and counting begins. 0 PWM disabled 1 PWM enabled

35.3.2.2 PWM Status Register (PWMSR)

The PWM status register (PWMSR) contains seven bits which display the state of the FIFO and the occurrence of rollover and compare events. The FIFOAV bit is read-only but the other four bits can be cleared by writing 1 to them. FE, ROV, and CMP bits are associated to FIFO-Empty, Roll-over, and

Compare interrupts, respectively. Figure 35-5 shows the register; Table 35-7 provides its field descriptions.

0x53FE_0004 (PWMSR) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	FWE	CMP	ROV	FE	FIFOAV		
W										w1c	w1c	w1c	w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Figure 35-5. PWM Status Register (PWMSR)

Table 35-7. PWMSR Field Descriptions

Field	Description
31–7 Reserved	Reserved. These reserved bits are always read as zero.
6 FWE	FIFO Write Error Status. This bit shows that an attempt has been made to write FIFO when it is full. 0 FIFO write error has not occurred 1 FIFO write error has occurred.
5 CMP	Compare Status. This bit shows that a compare event has occurred. 0 Compare event has not occurred. 1 Compare event has occurred.
4 ROV	Roll-over Status. This bit shows that a roll-over event has occurred. 0 Roll-over event has not occurred. 1 Roll-over event has occurred.
3 FE	FIFO Empty Status Bit. This bit indicates the FIFO data level in comparison to the water level set by FWM field in the control register. 0 Data level is above water mark. 1 The data level falls below the mark set by the FWM field.
2–0 FIFOAV	FIFO Available. These read-only bits indicate the data level remaining in the FIFO. An attempted write to these bits will not affect their value and no transfer error is generated. 000 No data available 001 1 word of data in FIFO 010 2 words of data in FIFO 011 3 words of data in FIFO 100 4 words of data in FIFO 101 Unused 110 Unused 111 Unused

35.3.2.3 PWM Interrupt Register (PWMIR)

The PWM Interrupt register (PWMIR) contains three bits that control the generation of the compare, rollover and FIFO empty interrupts. Figure 35-6 shows the register; Table 35-8 provides its field descriptions.

0x53FE_0008 (PWMIR) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	CIE	RIE	FIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 35-6. PWM Interrupt Register (PWMIR)

Table 35-8. PWMIR Field Descriptions

Field	Description
31–3 Reserved	Reserved. These reserved bits are always read as zero.
2 CIE	Compare Interrupt Enable. This bit controls the generation of the Compare interrupt. 0 Compare Interrupt not enabled 1 Compare Interrupt enabled
1 RIE	Roll-over Interrupt Enable. This bit controls the generation of the Rollover interrupt. 0 Roll-over interrupt not enabled 1 Roll-over Interrupt enabled
0 FIE	FIFO Empty Interrupt Enable. This bit controls the generation of the FIFO Empty interrupt. 0 FIFO Empty interrupt disabled 1 FIFO Empty interrupt enabled

35.3.2.4 PWM Sample Register (PWMSAR)

The PWM sample register (PWMSAR) is the input to the FIFO. 16-bit words are loaded into the FIFO. The FIFO can be written and read when the PWM is disabled. The PWM runs at the last set duty-cycle setting if all the values of the FIFO has been utilized, until the FIFO is reloaded or the PWM is disabled. When a new value is written, the duty cycle changes after the current period is over.

A value of zero in the sample register will result in the ipp_pwm_pwmo output signal being always low/high (POUTC =00 it will be low and POUTC = 01 it will be high), and hence no output waveform will be produced. If the value in this register is higher than the PERIOD + 1, the output will never be reset/set depending on POUTC value. Figure 35-7 shows the register; Table 35-9 provides its field descriptions.

0x53FE_000C (PWMSAR) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SAMPLE															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 35-7. PWM Sample Register (PWMSAR)

Table 35-9. PWMSAR Field Descriptions

Field	Description
31–16 Reserved	These are reserved bits and writing a value will not affect the functionality of PWM and are always read as zero.
15–0 SAMPLE	Sample Value. This is the input to the 4x16 FIFO. The value in this register denotes the value of the sample being currently used.

35.3.2.5 PWM Period Register (PWMPR)

The PWM period register (PWMPR) determines the period of the PWM output signal. After the counter value matches PERIOD + 1, the counter is reset to start another period.

PWMO (Hz) = PCLK(Hz)/(period +2)

A value of zero in the PWMPR results in a period of two clock cycles for the output signal. Writing 0xFFFF to this register achieves the same result as writing 0xFFFE.

A change in the period value due to a write in PWMPR results in the counter being reset to zero and the start of a new count period. Figure 35-8 shows the register; Table 35-10 provides its field descriptions.

0x53FE_0010 (PWMPR) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PERIOD															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

Figure 35-8. PWM Period Register (PWMPR)

Table 35-10. PWMPR Field Descriptions

Field	Description
31–16 Reserved	These are reserved bits and writing a value will not affect the functionality of PWM and are always read as zero.
15–0 PERIOD	Period Value. These bits determine the Period of the count cycle. The counter counts up to [Period Value] +1 and is then reset to 0x0000.

35.3.2.6 PWM Counter Register (PWMCNR)

The read-only pulse-width modulator counter register (PWMCNR) contains the current count value and can be read at any time without disturbing the counter. Figure 35-9 shows the register; Table 35-11 provides its field descriptions.

0x53FE_0014 (PWMCNR) Access: User read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COUNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 35-9. PWM Counter Register (PWMCNR)

Table 35-11. PWMCNR Field Descriptions

Field	Description
31–16 Reserved	These are reserved bits and writing a value will not affect the functionality of PWM and are always read as zero.
15–0 COUNT	Counter Value. These bits are the counter register value and denotes the current count state the counter register is in.

35.4 Functional Description

The following sections detail the PWM operation and function.

35.4.1 Operation

The output of the PWM is a toggling signal whose frequency and duty cycle can be modulated by programming the appropriate registers. It has a 16-bit up counter which counts from 0x0000 until the counter value equals the [Value in Period register] + 1. After this match occurs the Counter is reset to 0x0000.

At the beginning of a count period cycle, the PWMO pin is set to one (default) and the counter begins counting up from 0x0000. The sample value in the sample FIFO is compared on each count of prescaler clock. When the sample and count values match, the PWMO signal is cleared to zero (default). The counter continues counting until the period match occurs and subsequently another period cycle begins.

When the PWM is enabled the counter starts running and generates an output with the reset values in the period and sample registers. It is recommended that the programming of these registers be done before PWM is enabled.

A hardware reset results in all the PWM count and sample registers begin cleared and the FIFO being flushed. The control register shows that FIFO is empty and it can be written into, and the PWM is disabled. A software reset has the same results, however the state of the STOPEN, DOZEN, WAITEN, and DBGEN bits in the control register are not affected. Software reset can be asserted even when the PWM is in disabled state.

35.4.1.1 Clocks

The clock that feeds the prescaler can be selected from the following:

- High Frequency Clock (ipg_clk_highfreq) pat_ref or ckih

  This is a high frequency clock provided by the Clock Control Module (CCM). This clock is supposed to be on in the low power mode when the ipg_clk is turned off. Thus the PWM can be run on this clock in the low power mode. The CRM is expected to provide this clock after synchronizing it to ahb_clk in the normal functional mode and switch to the unsynchronized version in the low power mode.
- Low Reference Clock (ipg_clk_32k) ckil

  This is the 32 KHz low reference clock which is provided by the CCM. This clock is supposed to be on in the low power mode when ipg_clk is turned off. Thus PWM can be run on this clock in

the low power mode. The CRM is expected to provide this clock after synchronizing it to ahb_clk in the normal functional mode and switch to the unsynchronized version in the low power mode.

- Global Functional Clock (ipg_clk)

  This clock is supposed to be on in normal operations. In low power modes it can be switched off.

The clock input source is determined by the CLKSRC field of the PWM control register. The CLKSRC value should only be changed when the PWM is disabled.

The PWM input clock can be divided from 1 to 4096 by using a prescaler by appropriately setting the PRESCALER field in the control register. A change in the value of the PRESCALER field is immediately reflected on its output clock frequency.

35.4.1.2 FIFO

Digital sample values can be loaded into the pulse-width modulator as 16-bit words. The Endianness can be changed using the BCTR and HCTR bits of the control register. A 4-word (16-bit) FIFO minimizes interrupt overhead. A maskable interrupt is generated when the number of data words fall below the water level set by the FWM field in the control register.

A write in the sample register results in the value being stored into the FIFO if it is not full. A write when the FIFO is full sets FWE (FIFO write error) bit in the status register and the FIFO contents remain unchanged. The FIFO can be written into when the PWM is disabled. The FIFOAV field shows how many data words are currently contained in the FIFO and if it can be written into.

A read on the sample register yields the current FIFO value being used or will be used by the PWM for generation on the output signal. Therefore a write and a subsequent read on the sample register may result in different values being obtained.

35.4.1.3 Rollover and Compare Event

The counter is reset to 0x0000 after its value equals the PERIOD + 1 and resumes counting thereafter. This event is referred to as a rollover. When PERIOD = 0x0000, the counter is reset after count reaches 0x0001. Therefore PERIOD = 0xFFFF or 0xFFFE results in the counter value being reset after count till 0xFFFF. During a rollover event the output is either set (default), reset or has no effect according to the programming of the POUTC field in the control register. This event can also generate an interrupt if the respective interrupt enable bit is set in the control register.

When the counter value reaches the sample value the output of the PWM is reset (default), set or has no effect according to the programming of the POUTC field of control register. This event is referred to as a compare event. This event can also generate an interrupt if the respective interrupt enable bit is set in the control register.

If the rollover event sets the PWM output signal the compare event will reset it and vice versa for a particular programming configuration of POUTC field.

35.4.1.4 Low Power Mode Behavior

In low power modes if the clock from the selected clock source is available, the PWM counter continues to run and an output is produced depending on whether the control bit for that mode is set. In the absence of the clock itself or if the corresponding low power bit in the control register is 0, the counter is reset and resumes counting when it exits the low power mode.

35.4.1.5 Debug Mode Behavior

In debug mode, PWM has the option of continuing to run or be halted. If the DBGEN bit is not set in the PWMCR, the PWM is halted. If the DBGEN bit is set, then the PWM will continue to run in the debug mode.

35.5 PWM Clocking

Figure 35-10 shows the relationship of clocks used by the PWM.

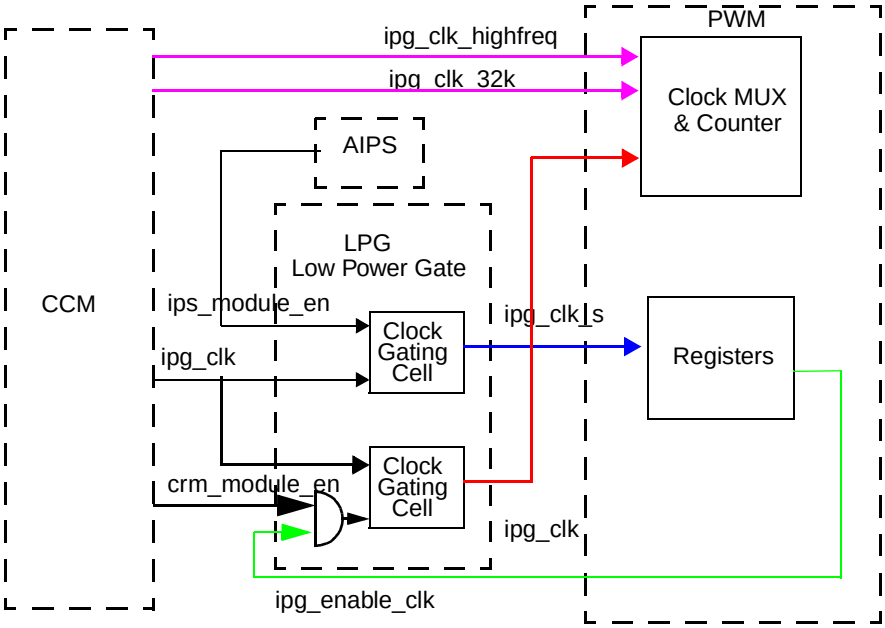


Figure 35-10. PWM Clocking

35.5.1 PWM Clock Inputs

Figure 35-11 shows the clock that feeds the prescaler (sys_clk).

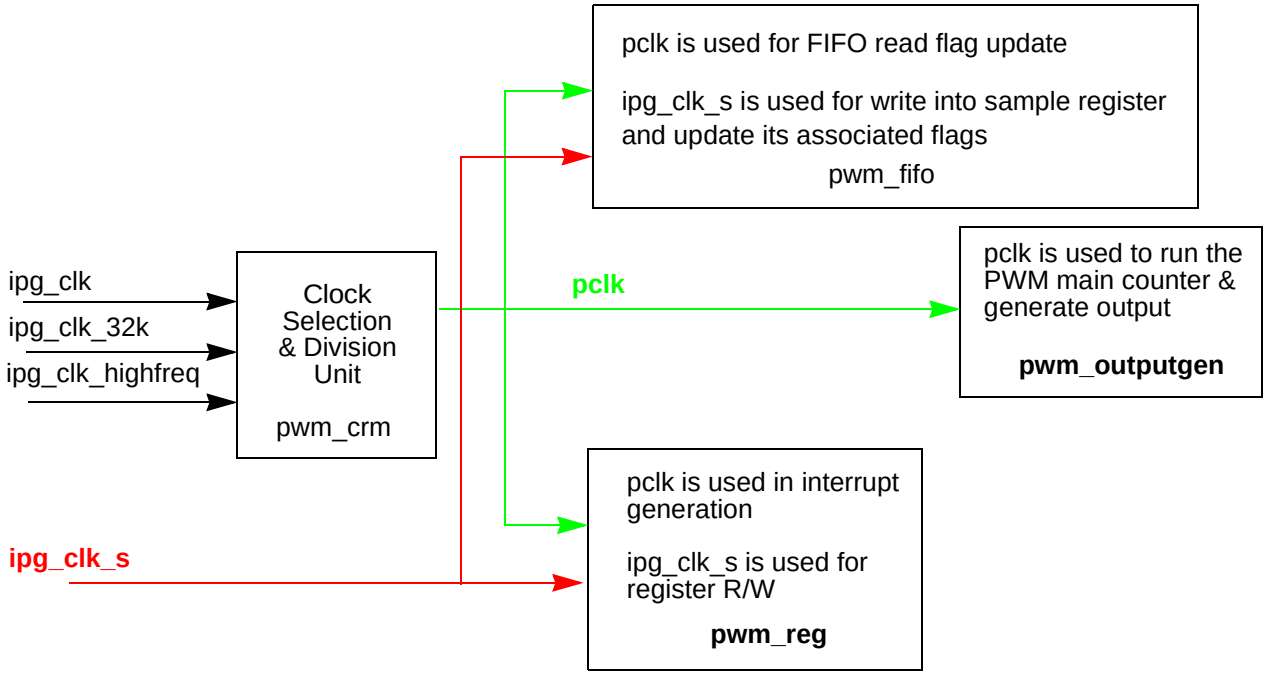


Figure 35-11. Clock Distribution Inside PWM

The clock that feeds the prescaler shown in Figure 35-11 can be selected from the following clock inputs:

- High frequency Clock (ipg_clk_highfreq) pat_ref or ckih
- Low Reference Clock (ipg_clk_32k) ckil
- Global Functional Clock (ipg_clk)

The selected clock sys_clk is prescaled with a 12-bit prescaler value. The sys_clk is gated with an enable signal that is generated from a prescaler counter and PWM enable to get the pclk. The main PWM counter runs on pclk.

pclk is used to generate the output signal of PWM and also the interrupts.

ipg_clk_s is the clock used for register read/write.

The only hand instantiated clock gating inside the module is done inside pwm_crm submodule for division of sys_clk to generate the pclk.

Figure 35-12 shows an overview of the clock selection and division unit.

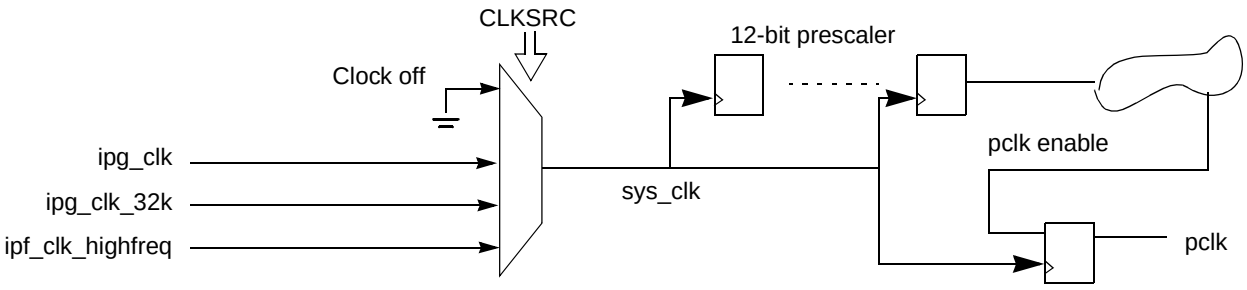


Figure 35-12. Clock Selection and Division Unit

35.5.2 ipg_enable_clk Generation

Whenever the module is enabled and ipg_clk is selected, this signal is asserted. Figure 35-13 shows the ipg_enable_clk generation logic.

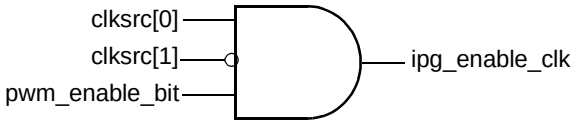


Figure 35-13. ipg_enable_clk Generation Logic

Chapter 36
Real Time Clock (RTC)

The Real Time Clock (RTC) module maintains the system clock, provides stopwatch, alarm, and interrupt functions, and supports the features described in Section 36.1.1, “Features.”

The RTC consists of the following blocks:

- Prescaler
- Time-of-day (TOD) clock counter
- Alarm
- Sampling timer
- Minute stopwatch
- Associated control and bus interface hardware

See Figure 36-1 for a block diagram of the functional organization of the RTC module.

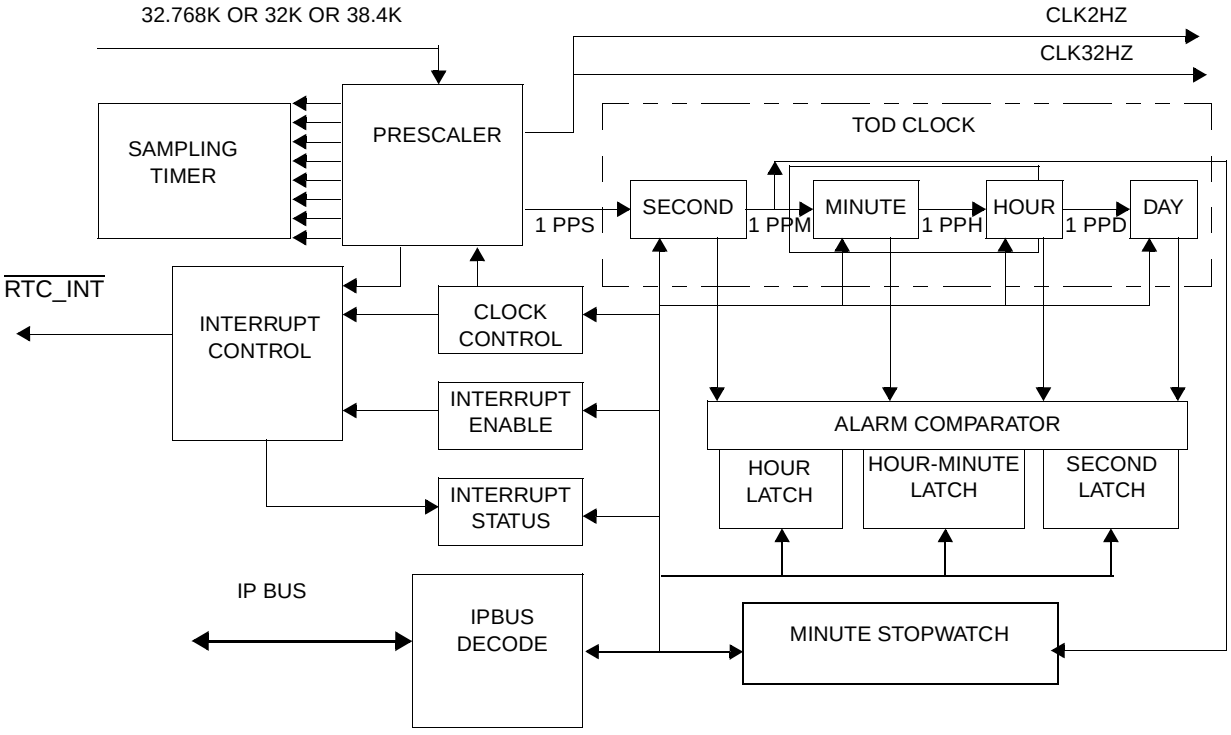


Figure 36-1. Real Time Clock Block Diagram

36.1 Overview

This section discusses how to operate and program the RTC.

36.1.1 Features

The RTC module includes the following features:

- Full clock—days, hours, minutes, seconds
- Minute countdown timer with interrupt
- Programmable daily alarm with interrupt
- Sampling timer with interrupt
- Once-per-day, once-per-hour, once-per-minute, and once-per-second interrupts
- Operation at 32.768 kHz or 32 kHz, or 38.4 kHz (determined by reference clock crystal)

36.1.2 Modes of Operation

The prescaler converts the incoming crystal reference clock to a 1 Hz signal, which is used to increment the seconds, minutes, hours, and days TOD counters. The alarm functions, when enabled, generate RTC interrupts when the TOD settings reach programmed values. The sampling timer generates fixed-frequency interrupts, and the minute stopwatch allows for efficient interrupts on minute boundaries.

- Prescaler and counter

  The prescaler divides the reference clock down to 1 Hz. The reference frequencies of 32.768 kHz, 38.4 kHz, and 32 kHz are supported.

  The counter portion of the RTC module consists of four groups of counters that are physically located in three registers:

  — The 6-bit seconds counter is located in the SECONDS register

  — The 6-bit minutes counter and the 5-bit hours counter are located in the HOURMIN register

  — The 16-bit day counter is located in the DAYR register

- Alarm

  There are three alarm registers that mirror the three counter registers. An alarm is set by accessing the real-time clock alarm registers (ALRM_HM, ALRM_SEC, and DAYALARM) and loading the exact time that the alarm should generate an interrupt. When the TOD clock value and the alarm value coincide, an interrupt occurs.

- Sampling Timer

  The sampling timer is designed to support application software. The sampling timer generates a periodic interrupt with the frequency specified by the SAMx bits of the RTCIENR register. This timer can be used for digitizer sampling, keyboard debouncing, or communication polling. The sampling timer operates only if the real-time clock is enabled. See Table 36-14 for the list of the interrupt frequencies of the sampling timer for the possible reference clocks.

- Minute Stopwatch

The minute stopwatch performs a countdown with a one minute resolution. It can be used to generate an interrupt on a minute boundary.

36.2 External Signal Description

The RTC has no external signals.

36.2.1 Overview

There are no signals connected off the chip.

36.3 Memory Map and Register Definition

The RTC module has ten 32-bit registers. Section 36.3.3, "Register Descriptions" provides the detailed descriptions for all of the RTC registers.

36.3.1 Memory Map

Table 36-1 shows the RTC memory map.

Table 36-1. RTC Register Memory Map

Address	Register	Access	Reset Value	Section/Page
0x53FD_8000 (HOURMIN)	RTC Hours and Minutes Counter Register (HOURMIN)	R/W	0x0000_– – – –	36.3.3.1/36-6
0x53FD_8004 (SECONDS)	RTC Seconds Counter Register (SECONDS)	R/W	0x0000_00 – –	36.3.3.2/36-7
0x53FD_8008 (ALRM_HM)	RTC Hours and Minutes Alarm Register (ALRM_HM)	R/W	0x0000_0000	36.3.3.3/36-7
0x53FD_800C (ALRM_SEC)	RTC Seconds Alarm Register (ALRM_SEC)	R/W	0x0000_0000	36.3.3.4/36-8
0x53FD_8010 (RTCCTL)	RTC Control Register (RCCTL)	R/W	0x0000_0080	36.3.3.5/36-9
0x53FD_8014 (RTCISR)	RTC Interrupt Status Register (RTCISR)	R/W	0x0000_0000	36.3.3.6/36-10
0x53FD_8018 (RTCIENR)	RTC Interrupt Enable Register (RTCIENR)	R/W	0x0000_0000	36.3.3.7/36-12
0x53FD_801C (STPWCH)	Stopwatch Minutes Register (STPWCH)	R/W	0x0000_0000	36.3.3.8/36-14
0x53FD_8020 (DAYR)	RTC Days Counter Register (DAYR)	R/W	0x0000_– – – –	36.3.3.9/36-15
0x53FD_8024 (DAYALARM)	RTC Days Alarm Register (DAYALARM)	R/W	0x0000_0000	36.3.3.10/36-16

36.3.2 Register Summary

Figure 36-2 shows the key to the register fields, and Table 36-2 shows the register figure conventions.

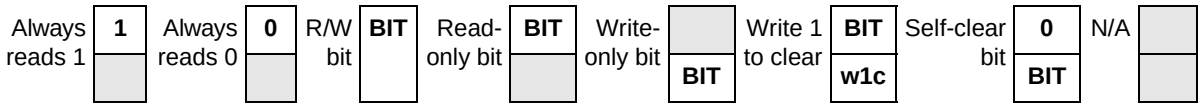


Figure 36-2. Key to Register Fields

Table 36-2. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 36-3 shows the RTC register summary.

Table 36-3. RTC Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x53FD_8000 (HOURMIN)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	HOUR					0	0	MINUTES					
	W																
0x53FD_8004 (SECONDS)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	SECONDS					
	W																

Table 36-3. RTC Register Summary (continued)

Name		31 15	30 14	29 13	28 12	27 11	26 10	25 9	24 8	23 7	22 6	21 5	20 4	19 3	18 2	17 1	16 0
0x53FD_8008 (ALRM_HM)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	HOUR					0	0	MINUTES					
	W																
0x53FD_800C (ALRM_SEC)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	SECONDS					
	W																
0x53FD_8010 (RTCCTL)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0		EN	XTL		0	0	0	GEN	SWR
	W																
0x53FD_8014 (RTCISR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	SAM7	SAM6	SAM5	SAM4	SAM3	SAM2	SAM1	SAM0	2HZ	0	HR	1HZ	DAY	ALM	MIN	SW
	W																
0x53FD_8018 (RTCIENR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	SAM7	SAM6	SAM5	SAM4	SAM3	SAM2	SAM1	SAM0	2HZ	0	HR	1HZ	DAY	ALM	MIN	SW
	W																
0x53FD_801C (STPWCH)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	CNT					
	W																
0x53FD_8020 (DAYR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	DAYS															
	W																
0x53FD_8024 (DAYALARM)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	DAYSAL															
	W																

36.3.3 Register Descriptions

This section consists of register descriptions in address order. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

36.3.3.1 RTC Hours and Minutes Counter Register (HOURMIN)

The real-time clock hours and minutes counter register (HOURMIN) is used to program the hours and minutes for the TOD clock. It can be read or written to at any time. After a write, the time changes to the new value. This register cannot be reset, since the real-time clock is always enabled at reset.

Figure 36-3 shows the HOURMIN register; Table 36-4 shows the register's field descriptions.

0x53FD_8000 (HOURMIN) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	HOUR					0	0	MINUTES					
W																
Reset	0	0	0	—	—	—	—	—	0	0	—	—	—	—	—	—

Figure 36-3. RTC Hours and Minutes Counter Register

Table 36-4. RTC Hours and Minutes Counter Register Field Descriptions

Field	Description
31–13	Reserved
12–8 HOUR	Hour setting indicates the current hour that can be set to any value between 0 and 23. 00000 Current hour is 0. 00001 Current hour is 1. .... 10111 Current hour is 23. Indicates the current hour that can be set to any value between 0 and 23.
7–5	Reserved
5–0 MINUTES	Minutes setting indicates the current minutes that can be set to any value between 0 and 59. 000000 Current minute is 0. 000001 Current minute is 1. .... 111011 Current minute is 59.

36.3.3.2 RTC Seconds Counter Register (SECONDS)

The real-time clock seconds register (SECONDS) is used to program the seconds for the TOD clock. It can be read or written to at any time. After a write, the time changes to the new value. This register cannot be reset, since the real-time clock is always enabled at reset.

Figure 36-4 shows the SECONDS register, and Figure 36-4 shows the register's field descriptions.

0x53FD_8004 (SECONDS) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	SECONDS					
W																
Reset	0	0	0	0	0	0	0	0	0	0	—	—	—	—	—	—

Figure 36-4. RTC Seconds Counter Register

Table 36-5. RTC Seconds Counter Register Field Descriptions

Field	Description
31–6	Reserved
5–0 SECONDS	Seconds setting indicates the current seconds that can be set to any value between 0 and 59. 000000 Current second is 0. 000001 Current second is 1. .... 111011 Current second is 59.

36.3.3.3 RTC Hours and Minutes Alarm Register (ALRM_HM)

The real-time clock hours and minutes alarm (ALRM_HM) register is used to configure the hours and minutes setting for the alarm. The alarm settings can be read or written to at any time.

Figure 36-5 shows the ALRM_HM register, and Table 36-6 shows the register's field descriptions.

0x53FD_8008 (ALRM_HM) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	HOUR					0	0	MINUTES					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 36-5. RTC Hours and Minutes Alarm Register

Table 36-6. RTC Hours and Minutes Alarm Register Field Descriptions

Field	Description
31–13	Reserved
12–8 HOURS	Hour setting of the alarm hours that can be set to any value between 0 and 23. 00000 Current hour is 0. 00001 Current hour is 1. .... 10111 Current hour is 23.
7–6	Reserved
5–0 MINUTES	Minutes setting of the alarm minutes that can be set to any value between 0 and 59. 000000 Current minute is 0. 000001 Current minute is 1. .... 111011 Current minute is 59.

36.3.3.4 RTC Seconds Alarm Register (ALRM_SEC)

The real-time clock seconds alarm (ALRM_SEC) register is used to configure the seconds setting for the alarm. The alarm settings can be read or written to at any time.

Figure 36-6 shows the ALRM_SEC register, and Table 36-7 shows the register's field descriptions.

0x53FD_800C (ALRM_SEC) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	SECONDS					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 36-6. RTC Seconds Alarm Register

Table 36-7. RTC Seconds Alarm Register Field Descriptions

Field	Description
31–6	Reserved
5–0 SECONDS	Seconds setting of the alarm seconds, can be set to any value between 0 and 59. 000000 Current second is 0. 000001 Current second is 1. .... 111011 Current second is 59.

36.3.3.5 RTC Control Register (RTCCTL)

The real-time clock control (RTCCTL) register is used to enable the real-time clock module and specify the reference frequency information for the prescaler.

Figure 36-7 shows the RTCCTL register, and Table 36-8 shows the register's field descriptions.

0x53FD_8010 (RTCCTL) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	EN	XTL		0	0	0	GEN	SWR
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Figure 36-7. RTC Control Register

Table 36-8. RTC Control Register Field Descriptions

Field	Description
31–8	Reserved
7 EN	Enables/disables the real-time clock. The software reset bit (SWR) has no effect on this bit. Bit description 0 Disable the real-time clock. 1 Enable the real-time clock.
6–5 XTL	Crystal selection. Selects the proper input crystal frequency. It is important to set these bits correctly or the real-time clock will be inaccurate. 00 Input crystal frequency is 32.768 kHz. 01 Input crystal frequency is 32 kHz. 10 Input crystal frequency is 38.4 kHz. 11 Input crystal frequency is 32.768 kHz.
4–2	Reserved
1 GEN	IPG_CLK gating enable. Decides whether to enable or disable the ipg_clk gating. Upon reset, the ipg_clk gating is enabled. 0 Enable ipg_clk gating. 1 Disable ipg_clk gating.
0 SWR	Software reset. Resets the module to its default state. However, a software reset will have no effect on the RTC enable (EN) bit. 0 No effect 1 Reset the module to its default state.

36.3.3.6 RTC Interrupt Status Register (RTCISR)

The real-time clock interrupt status register (RTCISR) indicates the status of the various real-time clock interrupts. When an event of the types included in this register occurs, then the bit is set in this register regardless of its corresponding interrupt enable bit.These bits are cleared by writing a value of ‘1’, which also clears the interrupt. Interrupts can occur while the system clock is idle or in sleep mode. Every interrupt status bit is independent of each other.

Figure 36-8 shows the RTCISR register, and Table 36-9 shows the register’s field descriptions.

0x53FD_8014 (RTCISR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SAM7	SAM6	SAM5	SAM4	SAM3	SAM2	SAM1	SAM0	2HZ	0	HR	1HZ	DAY	ALM	MIN	SW
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 36-8. RTC Interrupt Status Register

Table 36-9. RTC Interrupt Status Register Field Descriptions

Field	Description
31–16	Reserved
15 SAM7	Sampling timer interrupt flag at SAM7 frequency. Indicates that an interrupt has occurred. If enabled, this bit is periodically set at a rate of 512, 500, or 600 Hz, depending on different input clock. 0 No SAM7 interrupt occurred. 1 A SAM7 interrupt occurred.
14 SAM6	Sampling timer interrupt flag at SAM6 frequency. Indicates that an interrupt has occurred. If enabled, this bit is periodically set at a rate of 256, 250, or 300 Hz, depending on different input clock. 0 No SAM6 interrupt occurred. 1 A SAM6 interrupt occurred.
13 SAM5	Sampling timer interrupt flag at SAM5 frequency. Indicates that an interrupt has occurred. If enabled, this bit is periodically set at a rate of 128, 125, or 150 Hz, depending on different input clock. 0 No SAM5 interrupt occurred. 1 A SAM5 interrupt occurred.
12 SAM4	Sampling timer interrupt flag at SAM4 frequency. Indicates that an interrupt has occurred. If enabled, this bit is periodically set at a rate of 64, 62.5, or 75 Hz, depending on different input clock. 0 No SAM4 interrupt occurred. 1 A SAM4 interrupt occurred.
11 SAM3	Sampling timer interrupt flag at SAM3 frequency. Indicates that an interrupt has occurred. If enabled, this bit is periodically set at a rate of 32, 31.25, or 37.5 Hz, depending on different input clock. 0 No SAM3 interrupt occurred. 1 A SAM3 interrupt occurred.
10 SAM2	Sampling timer interrupt flag at SAM2 frequency. Indicates that an interrupt has occurred. If enabled, this bit is periodically set at a rate of 16, 15.625, or 18.75 Hz, depending on different input clock. 0 No SAM2 interrupt occurred. 1 A SAM2 interrupt occurred.
9 SAM1	Sampling timer interrupt flag at SAM1 frequency. Indicates that an interrupt has occurred. If enabled, this bit is periodically set at a rate of 8, 7.8125, or 9.375 Hz, depending on different input clock. 0 No SAM1 interrupt occurred. 1 A SAM1 interrupt occurred.
8 SAM0	Sampling timer interrupt flag at SAM0 frequency. Indicates that an interrupt has occurred. If enabled, this bit is periodically set at a rate of 4, 3.90625, or 4.6875 Hz, depending on different input clock. 0 No SAM0 interrupt occurred. 1 A SAM0 interrupt occurred.
7 2 Hz	2 Hz flag. Indicates that an interrupt has occurred. If enabled, this bit is set at every 2 Hz frequency. 0 No 2 Hz interrupt occurred. 1 A 2 Hz interrupt has occurred.
6	Reserved

Table 36-9. RTC Interrupt Status Register Field Descriptions (continued)

Field	Description
5 HR	Hour flag. Indicates that the hour counter has increment. If enabled, this bit is set on every increment of the hour counter in the time-of-day clock. 0 No 1-hour interrupt occurred. 1 A 1-hour interrupt has occurred.
4 1 Hz	1 Hz flag. Indicates that the second counter has increment. If enabled, this bit is set on every increment of the second counter of the time-of-day clock. 0 No 1 Hz interrupt occurred. 1 A 1 Hz interrupt has occurred.
3 DAY	Day flag. Indicates that the day counter has increment. If enabled, this bit is set on every increment of the day counter of the time-of-day clock. 0 No 24-hour rollover interrupt occurred. 1 A 24-hour rollover interrupt has occurred.
2 ALM	Alarm flag. Indicates that the real-time clock matches the value in the alarm registers. The alarm will reoccur every 65536 days. For a single alarm, clear the interrupt enable for this bit in the interrupt service routine. 0 No alarm interrupt occurred. 1 An alarm interrupt has occurred.
1 MIN	Minute flag. Indicates that the minute counter has increment. If enabled, this bit is set on every increment of the minute counter in the time-of-day clock. 0 No 1-minute interrupt occurred. 1 A 1-minute interrupt has occurred.
0 SW	Stopwatch flag. Indicates that the stopwatch countdown timed out. 0 The stopwatch did not time out. 1 The stopwatch timed out.

36.3.3.7 RTC Interrupt Enable Register (RTCIENR)

The real-time clock interrupt enable register (RTCIENR) is used to enable/disable the various real-time clock interrupts. Masking an interrupt bit has no effect on its corresponding status bit. Every interrupt enable bit is independent of each other.

Figure 36-9 shows the RTCIENR register, and Table 36-10 shows the register's field descriptions.

0x53FD_8018 (RTCIENR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SAM7	SAM6	SAM5	SAM4	SAM3	SAM2	SAM1	SAM0	2HZ	0	HR	1HZ	DAY	ALM	MIN	SW
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 36-9. RTC Interrupt Enable Register

Table 36-10. RTC Interrupt Enable Register Field Descriptions

Field	Description
31–16	Reserved
15 SAM7	Sampling timer interrupt flag at SAM7 interrupt. Enables/disables the real-time sampling timer interrupt 7. 0 The SAM7 interrupt is disabled. 1 The SAM7 interrupt is enabled.
14 SAM6	Sampling timer interrupt flag at SAM 6 interrupt. Enables/disables the real-time sampling timer interrupt 6. 0 The SAM6 interrupt is disabled. 1 The SAM6 interrupt is enabled.
13 SAM5	Sampling timer interrupt flag at SAM5 interrupt. Enables/disables the real-time sampling timer interrupt 5. 0 The SAM5 interrupt is disabled. 1 The SAM5 interrupt is enabled.
12 SAM4	Sampling timer interrupt flag at SAM4 interrupt. Enables/disables the real-time sampling timer interrupt 4. 0 The SAM4 interrupt is disabled. 1 The SAM4 interrupt is enabled.
11 SAM3	Sampling timer interrupt flag at SAM3 interrupt. Enables/disables the real-time sampling timer interrupt 3. 0 The SAM3 interrupt is disabled. 1 The SAM3 interrupt is enabled.
10 SAM2	Sampling timer interrupt flag at SAM2 interrupt. Enables/disables the real-time sampling timer interrupt 2. 0 The SAM2 interrupt is disabled. 1 The SAM2 interrupt is enabled.
9 SAM1	Sampling timer interrupt flag at SAM1 interrupt. Enables/disables the real-time sampling timer interrupt 1. 0 The SAM1 interrupt is disabled. 1 The SAM1 interrupt is enabled.

Table 36-10. RTC Interrupt Enable Register Field Descriptions (continued)

Field	Description
8 SAM0	Sampling timer interrupt flag at SAM0 interrupt. Enables/disables the real-time sampling timer interrupt 0. 0 The SAM0 interrupt is disabled. 1 The SAM0 interrupt is enabled.
7 2 Hz	2 Hz interrupt enable. Enables/disables an interrupt at a 2 Hz rate. 0 The 2-Hz interrupt is disabled. 1 The 2-Hz interrupt is enabled.
6	Reserved
5 HR	Hour interrupt enable. Enables/disables an interrupt whenever the hour counter of the real-time clock increments. 0 The 1-hour interrupt is disabled. 1 The 1-hour interrupt is enabled.
4 1HZ	1 Hz Interrupt Enable. Enables/disables an interrupt whenever the second counter of the real-time clock increments. 0 The 1-Hz interrupt is disabled. 1 The 1-Hz interrupt is enabled.
3 DAY	Day interrupt enable. Enables/disables an interrupt whenever the hours counter rolls over from 23 to 0. (midnight rollover) 0 The 24-hour interrupt is disabled. 1 The 24-hour interrupt is enabled.
2 ALM	Alarm interrupt enable. Enables/disables the alarm interrupt. 0 The alarm interrupt is disabled. 1 The alarm interrupt is enabled.
1 MIN	Minute interrupt enable. Enables/disables an interrupt whenever the minute counter of the real-time clock increments. 0 The 1-minute interrupt is disabled. 1 The 1-minute interrupt is enabled.
0 SW	Stopwatch interrupt enable. Enables/disables the stopwatch interrupt. **Note:** The stopwatch counts down and remains at decimal -1 until it is reprogrammed. If this bit is enabled with -1 (decimal) in the STPWCH register, an interrupt is posted on the next minute tick. 0 Stopwatch interrupt is disabled. 1 Stopwatch interrupt is enabled.

36.3.3.8 RTC Stopwatch Minutes Register (STPWCH)

The stopwatch minutes (STPWCH) register contains the current stopwatch countdown value. When the minute counter of the TOD clock increments, the value in this register decrements.

Figure 36-10 shows the STPWCH register, and Table 36-11 shows the register's field descriptions.

0x53FD_801C (STPWCH) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	CNT					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 36-10. RTC Stopwatch Minutes Register

Table 36-11. RTC Stopwatch Minutes Register Field Descriptions

Field	Description
31–6	Reserved
5–0 CNT	Stopwatch count. Contains the stopwatch countdown value. **Note:** The stopwatch counter is decremented by the minute (MIN) tick output from the real-time clock, so the average tolerance of the count is 0.5 minutes. For better accuracy, enable the stopwatch by polling the MIN bit of the RTCISR register or by polling the minute interrupt service routine. 000000 Stopwatch countdown value is 0. 000001 Stopwatch countdown value is 1. .... 111111 Stopwatch countdown value is 63.

36.3.3.9 RTC Days Counter Register (DAYR)

The real-time clock days counter register (DAYR) is used to program the day for the TOD clock. When the HOUR field of the HOURMIN register rolls over from 23 to 00, the day counter increments. It can be read or written to at any time. After a write, the time changes to the new value. This register cannot be reset, since the real-time clock is always enabled at reset.

Figure 36-11 shows the DAYR register, and Table 36-12 shows the register's field descriptions.

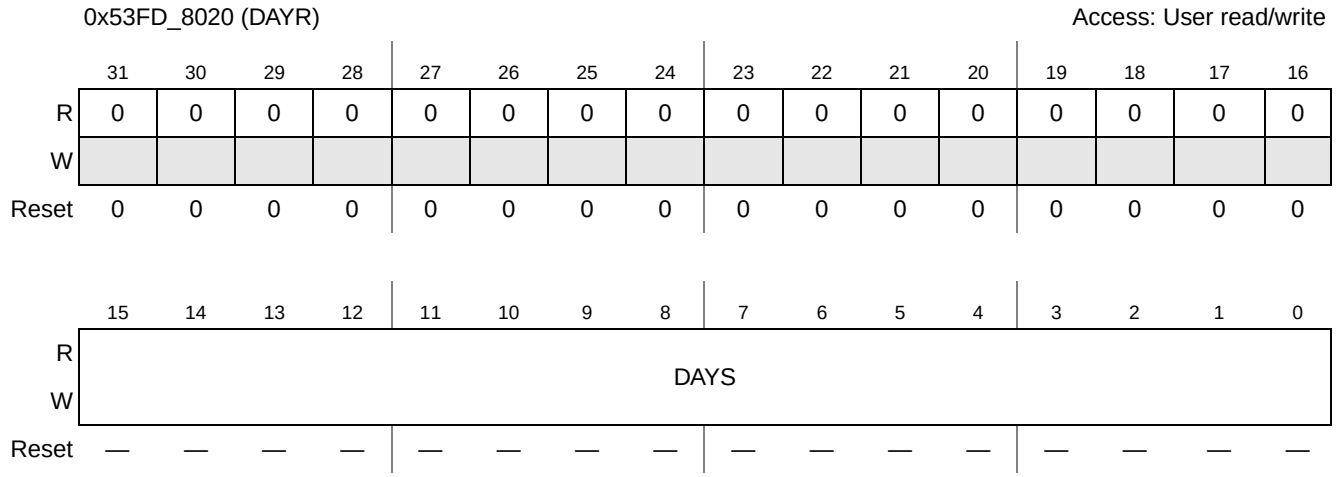


Figure 36-11. RTC Days Counter Register

Table 36-12. RTC Days Counter Register Field Descriptions

Field	Description
31–16	Reserved
15–0 DAYS	Day setting. Indicates the current day count, can be set to any values between 0 and 65535. 0x0000 Current day count is 0. 0x0001 Current day count is 1. .... 0xFFFF Current day count is 65535.

36.3.3.10 RTC Day Alarm Register (DAYALARM)

The real-time clock day alarm (DAYALARM) register is used to configure the day for the alarm. The alarm settings can be read or written to at any time.

Figure 36-12 shows the DAYALARM register, and Table 36-13 shows the register's field descriptions.

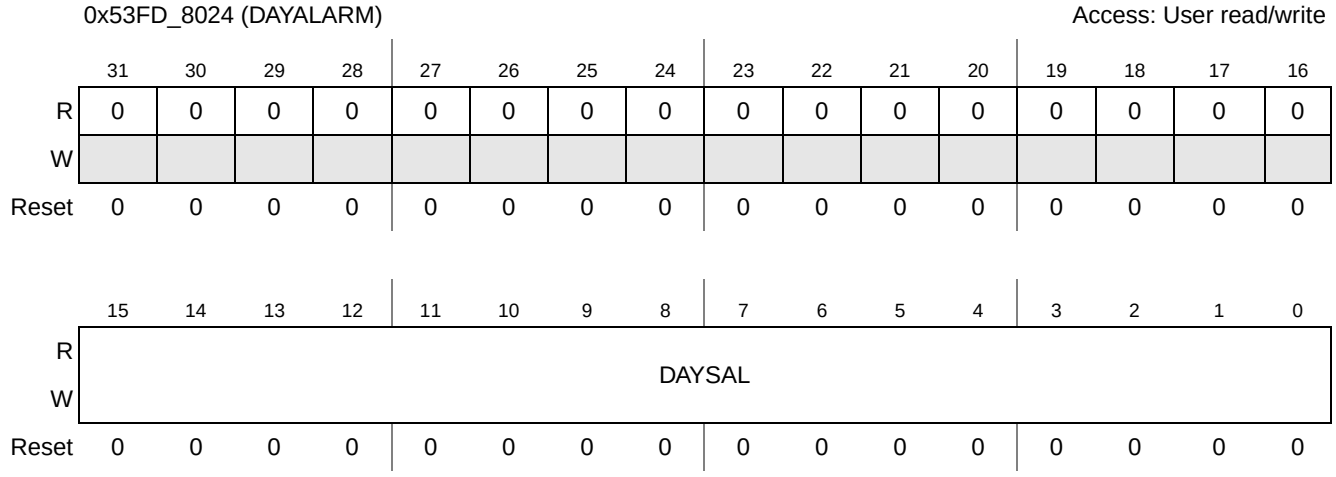


Figure 36-12. RTC Day Alarm Register

Table 36-13. RTC Day Alarm Register Field Descriptions

Field	Description
31–16	Reserved
15–0 DAYSAL	Day Setting of the Alarm. Indicates the current day setting of the alarm. It can be set to any value between 0 and 65535. 0x0000 Current day setting of alarm is 0. 0x0001 Current day setting of alarm is 1. .... 0xFFFF Current day setting of alarm is 65535

36.4 Functional Description

The prescaler converts the incoming crystal reference clock to a 1 Hz signal that is used to increment the seconds, minutes, hours, and days TOD counters. The alarm functions, when enabled, generate RTC interrupts when the TOD settings reach programmed values. The sampling timer generates fixed-frequency interrupts, and the minute stopwatch allows for efficient interrupts on minute boundaries.

36.4.1 Prescaler and Counter

The prescaler divides the reference clock down to 1 Hz. The reference frequencies of 32.768 kHz, 38.4 kHz, and 32 kHz are supported. The prescaler stages are tapped to support the sampling timer.

The counter portion of the RTC module consists of four groups of counters that are physically located in three registers:

- The 6-bit seconds counter is located in the SECONDS register.
- The 6-bit minutes counter and the 5-bit hours counter are located in the HOURMIN register.
- The 16-bit day counter is located in the DAYR register.

These counters cover a 24-hour clock over 65536 days. All three registers can be read or written to at any time.

Interrupts signal when each of the four counters increments, and can be used to indicate when a counter rolls over. For example, each tick of the seconds counter causes the 1 Hz interrupt flag to be set. When the seconds counter rolls from 59 to 00, the minute counter increments and the MIN interrupt flag is set. The same is true for the minute counter with the HR signal, and the hour counter with the DAY signal.

36.4.2 Alarm

There are three alarm registers that mirror the three counter registers:

- ALRM_HM
- ALRM_SEC
- DAYALARM

An alarm is set by accessing the three real-time clock alarm registers and loading the exact time that the alarm should generate an interrupt. When the TOD clock value and the alarm value coincide, if the ALM bit in the real-time clock interrupt enable register (RTCIENR) is set, an interrupt occurs.

NOTE

If the alarm is not disabled, it will reoccur every 65536 days. If a single alarm is desired, the alarm function must be disabled through the RTC Interrupt Enable Register (RTCIENR).

36.4.3 Sampling Timer

The sampling timer is designed to support application software. The sampling timer generates a periodic interrupt with the frequency specified by the SAMx bits of the RTCIENR register. This timer can be used for digitizer sampling, keyboard debouncing, or communication polling. The sampling timer operates only if the real-time clock is enabled. See Table 36-14 for the list of the interrupt frequencies of the sampling timer for the possible reference clocks.

Multiple SAMx bits may be set in the RTC Interrupt Enable Register (RTCIENR). The corresponding bits in the RTC Interrupt Status Register (RTCISR) are set at the noted frequencies.

Table 36-14. Sampling Timer Frequencies

Sampling Frequency	32.768 kHz Reference Clock	32 kHz Reference Clock	38.4 kHz Reference Clock
SAM7	512 Hz	500 Hz	600 Hz
SAM6	256 Hz	250 Hz	300 Hz
SAM5	128 Hz	125 Hz	150 Hz
SAM4	64 Hz	62.5 Hz	75 Hz
SAM3	32Hz	31.25 Hz	37.5 Hz
SAM2	16 Hz	15.625 Hz	18.75 Hz
SAM1	8 Hz	7.8125 Hz	9.375 Hz
SAM0	4 Hz	3.90625 Hz	4.6875 Hz

36.4.4 Minute Stopwatch

The minute stopwatch performs a countdown with a one minute resolution. It can be used to generate an interrupt on a minute boundary. For example, to turn off the LCD controller after five minutes of inactivity, program a value of 0x04 into the Stopwatch Count (CNT) field of the Stopwatch Minutes (STPWCH) register. At each minute, the value in the stopwatch is decremented. When the stopwatch value reaches -1, the interrupt occurs. The value of the register does not change until it is reprogrammed.

NOTE

The actual delay includes the seconds from setting the stopwatch to the next minute tick.

36.5 Initialization/Application Information

36.5.1 Flowchart of RTC Operation

See Figure 36-13 for the illustration of the flowchart of a typical RTC operation. Refer to Example 36-1 for the code example of ARM instruction for configuring RTC.

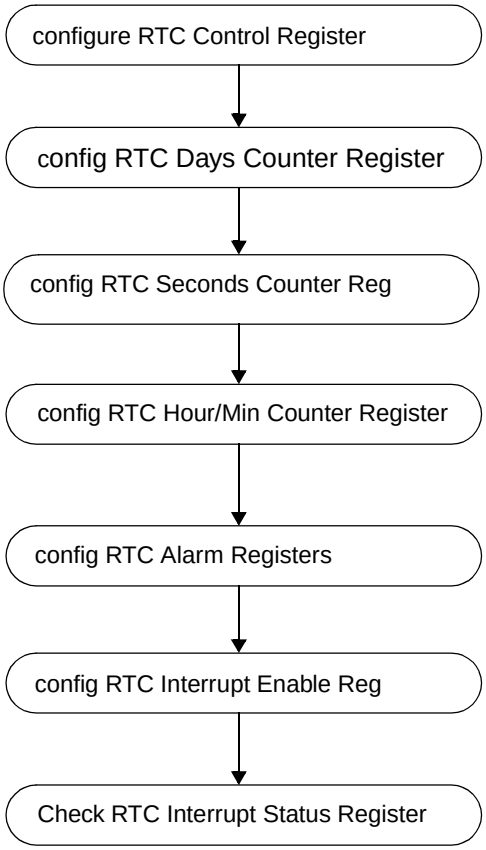


Figure 36-13. Flowchart of RTC Operation

36.5.2 Code Example of ARM Instruction

Example 36-1. Code Example of ARM Instruction

```
LDR     r1,=RTC_BASE_ADDR

LDR     r2,[r1,#0x10]
ORR     r2,r2,#0x21
STR     r2,[r1,#0x10]          ; Software reset and 32k crystal


LDR     r3,=0x0000
STR     r3,[r1,#0x20]         ;DAY
```

```
        LDR     r3,=0x00038
        STR     r3,[r1,#0x04]         ;SECOND
        LDR     r3,=0x173B
        STR     r3,[r1]               ;HR, MIN

        LDR     r3,=0x0001
        STR     r3,[r1,#0x24]          ;Alarm Day
        LDR     r3,=0x0000
        STR     r3,[r1,#0x08]         ;Alarm hour, minute
        LDR     r3,=0x01
        STR     r3,[r1,#0x0C]         ;Alarm seconds

        LDR     r2,[r1,#0x18]         ;set ALARM interrupt
        ORR     r2,r2,#0x4
        STR     r2,[r1,#0x18]

ALARM_STATUS_3
        LDR     r2,[r1,#0x18]          ;check ALARM STATUS FLAG
        TST     r2,#0x04
        BNE     ALARM_STATUS_3
```

Chapter 37
Watchdog Timer (WDOG)

The Watchdog (WDOG) timer module protects against system failures by providing a method of escaping from unexpected events or programming errors. Figure 37-1 shows the WDOG block diagram.

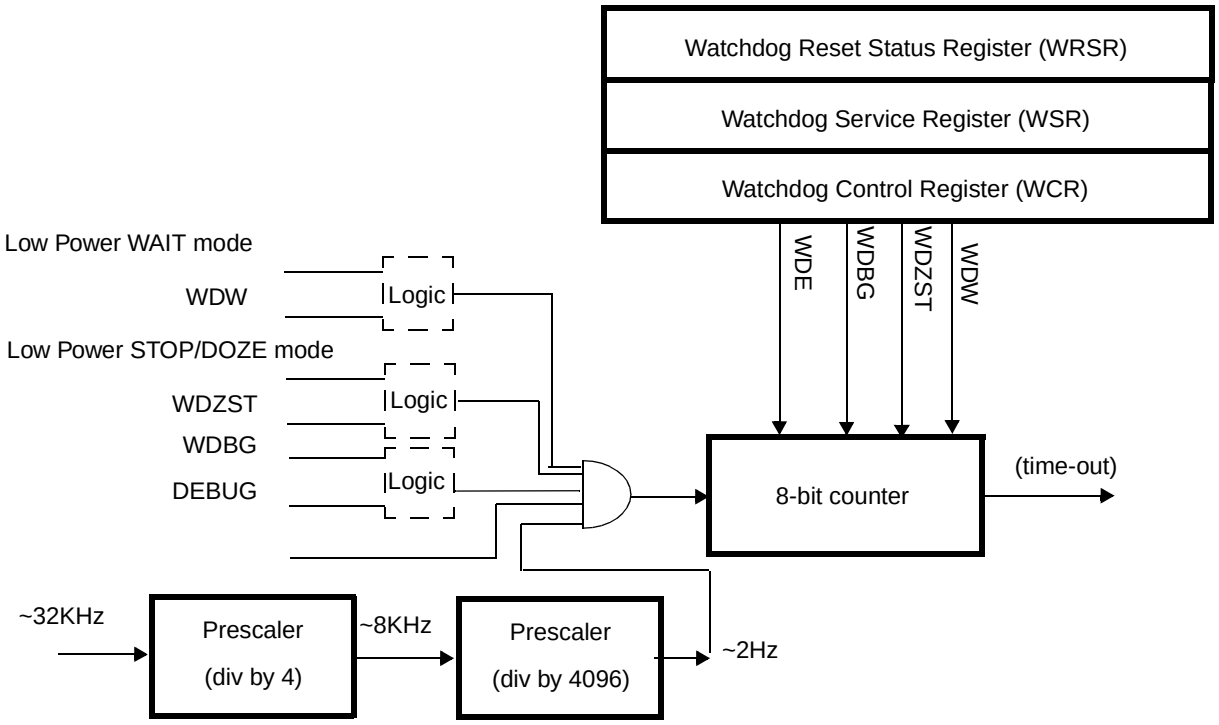


Figure 37-1. WDOG Block Diagram

37.1 Overview

Once the WDOG module is activated, it must be serviced by the software on a periodic basis. If servicing does not take place, the timer times out. Upon a time-out, the WDOG Timer module either asserts the $\overline{\text{WDOG}}$ signal or a system reset signal, wdog_rst, depending on the software configuration. The WDOG Timer module also generates a system reset via a software write to the Watchdog Control Register (WCR), a detection of a clock monitor event, an external reset, an external JTAG reset signal, or if a power-on-reset has occurred. The $\overline{\text{WDOG}}$ signal is asserted via a software write to the WCR, a detection of a clock monitor event, or upon a watchdog time-out. A state machine of the counter operation is shown in Figure 37-7, which demonstrates the time-out operation.

The WDOG module cannot be deactivated again after activation.

The input clocks to the WDOG module (ipg_clk_32k, ipg_clk, and ipg_clk_s) must be synchronized with each other.

NOTE

ipg_clk_32k is the synchronized version of the input ~32k clock (CKIL) on the IP global functional clock. Because the synchronized version is not available (the IP global functional clock is off) in low-power mode, it will receive the raw CKIL (ipg_clk_32k) clock from the CCM (Clock Controller Module). These synchronizers will be bypassed while going into low power modes in CRM.

The WDOG module can continue or suspend the timer operation in the low power modes (WAIT and STOP). For ARM (DOZE and STOP), it emulates these modes.

37.1.1 Features

The WDOG features are as follows:

- Time-out periods from 0.5 seconds up to 128 seconds
- Time resolution of 0.5 seconds
- Configurable counters that can be programed to run or stop during low-power modes
- Configurable counters that can be programed to run or stop during DEBUG mode

37.2 External Signal Description

The WDOG module port signals going to pins are listed in Table 37-2.

Table 37-2. Signal Properties

Name	Port	Function	Reset State	Pull-Up
$\overline{\text{IPP\_WDOG}}$	—	Asserted by software, timeout, or clock monitor event	1	—
IPP_WDOG_OE	—	$\overline{\text{WDOG}}$ output enable at pin	0	—

37.2.1 Detailed External Signal Descriptions

37.2.1.1 $\overline{\text{IPP\_WDOG}}$, IPP_WDOG_OE

The $\overline{\text{IPP\_WDOG}}$ signal can be provided externally to the IC. It is asserted by a software request (setting the WCR bits), timeout, or a clock monitor event. The IPP_WDOG_OE signal is the output enable for the pin.

37.2.2 Internal Port Signals

The WDOG internal port signals are listed in Table 37-3.

Table 37-3. WDOG Module Port List

Port Name	Direction	Description
ipg_enable_clk	Output (O)	IP global clock gating signal. It is available to the Clock Controller to gate off the clock to the WDOG module for power saving.
ipg_clk	Input (I)	IP Global functional clock. All functionality inside the WDOG module is synchronized to this clock.
ipg_clk_s	I	IP slave bus clock. This clock is synchronized to ipg_clk and is only used for register read/write operations.
ipg_clk_32k	I	Low frequency (32.768 kHz) clock that continues to run in low-power mode. It is assumed that the Clock Controller will provide this clock signal synchronized to ipg_clk in the normal mode, and switch to a non-synchronized signal in low-power mode when the ipg_clk is off.
$\overline{\text{ipg\_hard\_async\_reset}}$	I	IP global hardware reset (active low)
$\overline{\text{ipg\_por}}$	I	Power-on-reset signal (active low)
$\overline{\text{ipg\_jtag\_reset}}$	I	JTAG Reset signal (active low)
ipg_debug	I	IP global signal indicating that WDOG should enter debug mode operation
ipg_stop	I	IP global signal indicating that WDOG should enter low-power stop mode operation
ipg_doze	I	IP global signal indicating that WDOG should enter low-power doze mode operation
ipg_wait	I	IP global signal indicating that WDOG should enter low-power wait mode operation
ips_rdata[15:0]	O	IP slave bus read data line
ips_xfr_wait	O	IP slave bus transfer wait indicator
ips_xfr_err	O	IP slave bus transfer error indicator
ips_module_en	I	IP slave bus module enable signal. This signal indicates when a module bus transaction is occurring.
ips_rwb	I	IP slave bus read/write signal. Shows whether a bus transaction is read or write.
ips_addr[13:1]	I	IP slave bus address signal. Denotes the register being accessed during a bus transaction.
ips_wdata[15:0]	I	IP slave bus write date line
ips_byte_15_8	I	IP slave bus byte enable signal for bits 15 to 8
ips_byte_7_0	I	IP slave bus byte enable signal for bits 7 to 0
ipt_reset	I	IP scan reset signal used to work in place of generated resets in scan mode (active low)
ipt_se_async	I	IP scan signal for bypassing generated resets and making latches transparent in scan mode
ipt_scan_mode	I	IP scan mode signal
ipt_se_gatedclk	I	IP scan mode clock gating signal
$\overline{\text{wdog\_rst}}$	O	Watchdog Timer reset (active low)

Table 37-3. WDOG Module Port List (continued)

Port Name	Direction	Description
$\overline{\text{ccm\_ext\_rst}}$	I	External Reset (Active low)
cmon_noclk	I	Clock Monitor event
resp_sel	I	Indicates if the error response needs to be generated on access on unimplemented registers or not. Refer to Section 37.5.4, “Generation of Transfer Error on the IP Bus” for details.

37.3 Memory Map and Register Definitions

The WDOG module has three, user-accessible, 16-bit registers used to configure, operate, and monitor the state of the Watchdog Timer. Section 37.4, “Register Descriptions” provides the detailed descriptions for all of the WDOG registers.

37.3.1 Watchdog Timer Memory Map

Table 37-4 shows the WDOG memory map.

Table 37-4. WDOG Memory Map

Address	Register	Access	Reset Value	Section/Page
0x53FD_C000 (WCR)	Watchdog Control Register	R/W	0x0030	37.4.1/37-5
0x53FD_C002 (WSR)	Watchdog Status Register	R/W	0x0010	37.4.2/37-6
0x53FD_C004 (WRSR)	Watchdog Reset Status Register	R	0x00– –	37.5.6/37-10

37.3.2 Register Summary

Figure 37-2 shows the key to the register fields, and Table 37-5 shows the register figure conventions.

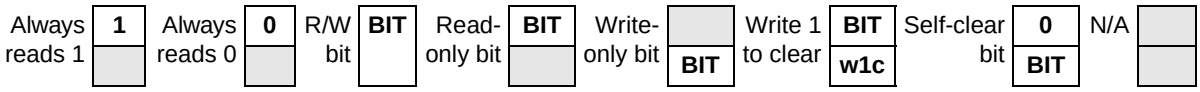


Figure 37-2. Key to Register Fields

Table 37-5. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writeable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit’s value (other than a hardware reset).

Table 37-5. Register Figure Conventions (continued)

Convention	Description
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 37-6 shows the WDOG register summary.

Table 37-6. WDOG Register Summary

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x53FD_C000 (WCR)	R	WT								0	WOE	WDA	SRS	WRE	WDE	WDBG	WDZST
	W																
0x53FD_C002 (WSR)	R	WSR															
	W																
0x53FD_C004 (WRSR)	R	0	0	0	0	0	0	0	0	0	0	JRST	PWR	EXT	CMON	TOUT	SFTW
	W																

37.4 Register Descriptions

37.4.1 Watchdog Control Register (WCR)

The Watchdog Control Register (WCR) is a 16-bit read/write register. It controls the WDOG operation. Figure 37-3 shows the register; Table 37-7 provides its field descriptions.

0x53FD_C000 (WCR) Access: User Read/Write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	WT								0	WOE	WDA	SRS	WRE	WDE	WDBG	WDZST
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0

Figure 37-3. Watchdog Control Register

Table 37-7. WCR Register Descriptions

Field	Description
15–8 WT	Watchdog Timeout Field. This 8-bit field contains the time-out value that is loaded into the Watchdog counter after the service routine has been performed. After reset, WT[7:0] must have a value written to it before enabling the Watchdog.
7	Reserved
6 WOE	$\overline{\text{WDOG}}$ Output Enable. Determines if the WDOG pin is configured as an output or as an input if the GPIO gives the WDOG the ability to control this pin's direction after reset. This bit is not initialized by the hard reset generated by the CCM because of a software reset (SRS bit reset) assertion by the WDOG. Refer to the SRS bit in this register. **Note:** Regarding functionality of the WOE bit with a software reset: When the source for a hard reset is due to a software reset, then the WOE bit retains its value prior to assertion of the software reset. For all other resets, the WOE bit is cleared (for example, the output will be tri-stated). 0 Pin is tri-stated 1 Pin configured as output
5 WDA	$\overline{\text{WDOG}}$ Assertion. Controls the software assertion of the $\overline{\text{WDOG}}$ signal. 0 Assert $\overline{\text{WDOG}}$ output. 1 No effect on system.
4 SRS	Software Reset Signal. Controls the software assertion of the WDOG-generated reset signal. This bit automatically resets to "1" after it has been asserted to "0". **Note:** This bit does not generate the software reset to the module. The ipg_clk must be on to write to this bit. 0 Assert system reset signal 1 No effect on the system
3 WRE	$\overline{\text{wdog}}$/$\overline{\text{wdog\_rst}}$ Enable. Determines if the Watchdog generates a reset signal or a $\overline{\text{WDOG}}$ signal upon a Watchdog timeout or clock monitor event. This is a write once-only bit. 0 Generate a $\overline{\text{reset signal}}$ 1 Generate a $\overline{\text{WDOG signal}}$
2 WDE	Watchdog Enable. Enables or disables the WDOG module. Software can only write "1" in this bit. It is not possible to reset this bit by a software write, once the bit is set. **Note:** This bit can be set/reset as per the IP writes in debug mode (exception). 0 Disable the Watchdog 1 Enable the Watchdog
1 WDBG	Watchdog DEBUG Enable. Determines the operation of the WDOG module during DEBUG mode. This bit is write once-only. 0 Continue WDOG timer operation 1 Suspend the watchdog timer
0 WDZST	Watchdog Low Power. Determines the operation of the WDOG module during low-power modes. This bit is write once-only. **Note:** The WDOG module can continue/suspend the timer operation in the low-power modes (WAIT, DOZE and STOP). In the ARM modes DOZE and STOP, it emulates these modes. 0 Continue timer operation 1 Suspend the watchdog timer

37.4.2 Watchdog Service Register (WSR)

When enabled, the WDOG requires that a service sequence be written to the Watchdog Service Register (WSR). Figure 37-4 shows the register; Table 37-8 provides its field descriptions.

0x53FD_C002 (WSR) Access: User Read/Write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	WSR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 37-4. Watchdog Service Register (WSR)

Table 37-8. Watchdog Service Register Description

Field	Description
15–0 WSR	Watchdog Service Register. This 15-bit field contains the Watchdog service sequence. Both writes must occur in the order listed prior to the time-out, but any number of instructions can be executed between the two writes. The service sequence must be performed as follows: Write 0x 5555 to the Watchdog Service Register (WSR) Write 0x AAAA to the Watchdog Service Register (WSR)

37.4.2.1 Watchdog Reset Status Register (WRSR)

The WRSR is a read-only register that records the source of the output reset assertion. It is not cleared by a hard reset. It records the source of the output reset assertion. Therefore, only one bit in the WRSR will always be asserted high. The register will always indicate the source of the last reset.

A reset can be generated by the following sources, as listed in priority from highest to lowest:

- Power-On Reset
- External Reset
- Clock Monitor Event
- Watchdog Timeout
- Software Reset

Figure 37-5 shows the register; Table 37-8 provides its field descriptions.

0x53FD_C004 (WRSR) Access: User read-only

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	JRST	PWR	EXT	CMON	TOUT	SFTW
W																
Reset	0	0	0	0	0	0	0	0	0	0	—	—	—	—	—	—

Figure 37-5. Watchdog Reset Status Register (WRSR)

Table 37-9. WRSR Descriptions

Field	Description
15–6	Reserved
5 JRST	JTAG Reset. Indicates whether the reset is the result of a JTAG reset. 0 Reset is not the result of a JTAG reset. 1 Reset is the result of a JTAG reset.
4 PWR	Power-On Reset. Indicates whether the reset is the result of a power-on reset. 0 Reset is not the result of a power-on reset. 1 Reset is the result of a power-on reset.
3 EXT	External Reset. Indicates whether the reset is the result of an external reset. 0 Reset is not the result of an external reset. 1 Reset is the result of an external reset.
2 CMON	Clock Monitor. Indicates whether the reset is the result of a clock monitor event. 0 Reset is not the result of clock monitor event. 1 Reset is the result of clock monitor event.
1 TOUT	Time-out. Indicates whether the reset is the result of a WDOG time-out. 0 Reset is not the result of a WDOG time-out. 1 Reset is the result of a WDOG time-out.
0 SFTW	Software Reset. Indicates whether the reset is the result of a software reset. 0 Reset is not the result of a software reset. 1 Reset is the result of a software reset. **Note:** Writing to the WRSR register will generate an ips_xfr_err.

37.5 Functional Description

This section describes the timing information for the WDOG module.

37.5.1 Timing Specifications

The WDOG provides time-out periods from 0.5 seconds up to 128 seconds with a time resolution of 0.5 seconds. It uses the ipg_clk_32k clock (32.768 kHz frequency clock) as an input to prescalers. The prescalers divide the clock by a fixed value of 16384 (divide by 4 and divide by 4096) to achieve a resolution of 0.5 seconds at a frequency of 2 Hz. The output of the prescaler circuitry is connected to the input of an 8-bit counter to obtain a range of 0.5 to 128 seconds. The user can determine the time-out period by writing a time-out value to the WDOG Time-out field (WT[7:0]) in the WDOG Control Register (WCR).

37.5.2 Watchdog During Reset

A system reset resets all registers (except the WRSR) to their initial default values, and places the counter in the idle state until the WDOG is enabled. The Watchdog Reset Status Register (WRSR) contains the source of the reset event and is not reset by a system reset.

37.5.3 Watchdog After Reset

The following subsections define the WDOG Timer state after reset.

37.5.3.1 Initial Load

The Watchdog Control Register (WCR) field WT[7:0] must have a time-out value written to it before the Watchdog can be enabled. The WDOG is enabled by setting the Watchdog Enable (WDE) bit in the WCR. The time-out value is loaded into the counter after the service sequence is written to the Watchdog Service Register (WSR), or after the WDOG has been enabled. The service sequence is described in Section 37.5.3.3, "Reloading the Counter." (The counter state machine is shown in Figure 37-7.)

37.5.3.2 Timer Countdown

The counter is activated and begins to count down from its initial programmed value after the WDOG is enabled. If any system errors occur that prevent the software from servicing the Watchdog Service Register (WSR), the timer will time-out when the counter reaches zero. If the WSR is serviced prior to the counter reaching zero, the WDOG reloads its counter to the time-out value indicated by bits WT[7:0] of the WCR, and it re-starts the countdown. A system reset will reset the counter and place it in the idle state at any time during the countdown. (The counter state machine is shown in Figure 37-7.)

37.5.3.3 Reloading the Counter

The proper service sequence to write a time-out value to the counter begins by writing 0x 5555 followed by 0x AAAA to the WSR. To reload the counter, the writes must take place within the time-out value indicated by bits WT[7:0] of the WCR. Any number of instructions can be executed between the two writes. This service sequence is also used to activate the counter during the initial load. See Section 37.5.3.1, "Initial Load."

If the WSR is not loaded with 0x 5555 prior to writing 0x AAAA to the WSR, the counter is not reloaded. If any value other than 0x AAAA is written to the WSR after 0x 5555, the counter is not reloaded.

37.5.3.4 Time-Out

If the counter reaches zero, the WDOG outputs either a system reset or the $\overline{\text{WDOG}}$ signal, depending on the state of the WRE bit in the WCR. A "0" written to the WRE bit configures the WDOG to generate a system reset. A "1" causes the WDOG to generate the $\overline{\text{WDOG}}$ signal. (The counter state machine is shown in Figure 37-7.)

37.5.4 Generation of Transfer Error on the IP Bus

The WDOG module asserts a transfer error signal (IPS_XFR_ERROR) on the IP bus in the following cases:

- Receiving an IP access to an address that is not implemented
- A write access to the WRSR register that is a read-only register

An error on an unimplemented address is generated only if the input pin resp_sel is low. Otherwise, an error is only asserted on a write access to the WRSR register (a read-only register).

37.5.5 Low-Power and DEBUG Modes

The WDOG module can either continue or suspend the timer operation during low-power modes (WAIT, DOZE, and STOP) and DEBUG mode.

37.5.5.1 Low-Power Mode (WAIT, DOZE, STOP)

While in low-power mode, the WDOG Timer can be configured for continual operation or its operation can be suspended. If the WDOG low-power Enable (WDZST) bit in the WCR is set to “0”, the WDOG continues to operate using the ipg_clk_32k clock (32.768 kHz frequency clock) as its source. If the low-power enable (WDZST) bit is set to “1”, then the WDOG operation is suspended during low-power mode. Upon exiting low-power mode, the WDOG returns to the operational mode it was in prior to entering low-power mode.

37.5.5.2 DEBUG Mode

The WDOG Timer can be configured for continual operation, or the operation can be suspended during debug mode. If the WDOG debug enable (WDBG) bit is set to “1” in the Watchdog Control Register (WCR), the WDOG module operation is suspended in debug mode. At this point, the counter is stopped, but register read and write accesses continue to function normally. Also, while in DEBUG mode, the WDE bit can be enabled-disabled directly.

NOTE

If the WDE bit is cleared while in DEBUG mode, it will remain cleared upon exiting DEBUG mode. If the WDE bit is not cleared while in DEBUG mode, the WDOG count will continue from its value before DEBUG mode was entered.

37.5.6 Watchdog Reset Control

The WDOG generated reset signal $\overline{\text{wdog\_rst}}$ can be asserted by a software write to the Software Reset Signal (SRS) bit of the WCR. It can also be generated by the following events:

- WDOG time-out
- High frequency clock monitor event
- External JTAG reset signal
- Power-on-reset

The $\overline{\text{wdog\_rst}}$ is generated for 0.5 seconds for a time-out, but is deasserted early if a system reset is detected. In case of a software reset, the $\overline{\text{wdog\_rst}}$ is asserted after three clocks of resetting the SRS bit and remains asserted for three IPG clocks (IP global functional clock). If a system reset is asserted in between, it deasserts before three IPG clocks have elapsed.

If a clock monitor event occurs, it remains asserted until a system reset is asserted with ipg_clk running. For the power-on reset and the external reset, the signal is asserted as long as they remain active.

The watchdog-generated reset signal wdog_rst is an output to the Clock Control Module (CCM) for system reset generation.

NOTE

The CCM of the IC generates the system reset signal on assertion of wdog_rst.

37.5.7 WDOG Operation

WDOG can be asserted through software writes to the WDOG Assertion (WDA) bit of the WCR. It can also be generated as a result of a WDOG time-out or a high frequency clock monitor event.

If asserted by a software write to the WDA bit, it remains asserted as long as the WDA bit is “0”. The counter timeout asserts it for 0.5 seconds. Both a system reset and a power-on reset can deassert it in between.

In case of a clock monitor event the WDOG signal remains asserted until a system reset (with ipg_clk running) or a power-on reset is asserted.

WDOG is applied to the WDOG pin. After reset, the WOE bit in the WCR register controls the direction of this pin.

37.5.8 Clock Monitor

A clock monitor event results in the WDOG generated signals wdog_rst or WDOG being asserted, depending on the setting of the WRE bit in the WCR. Figure 37-6 shows the clock monitor signals.

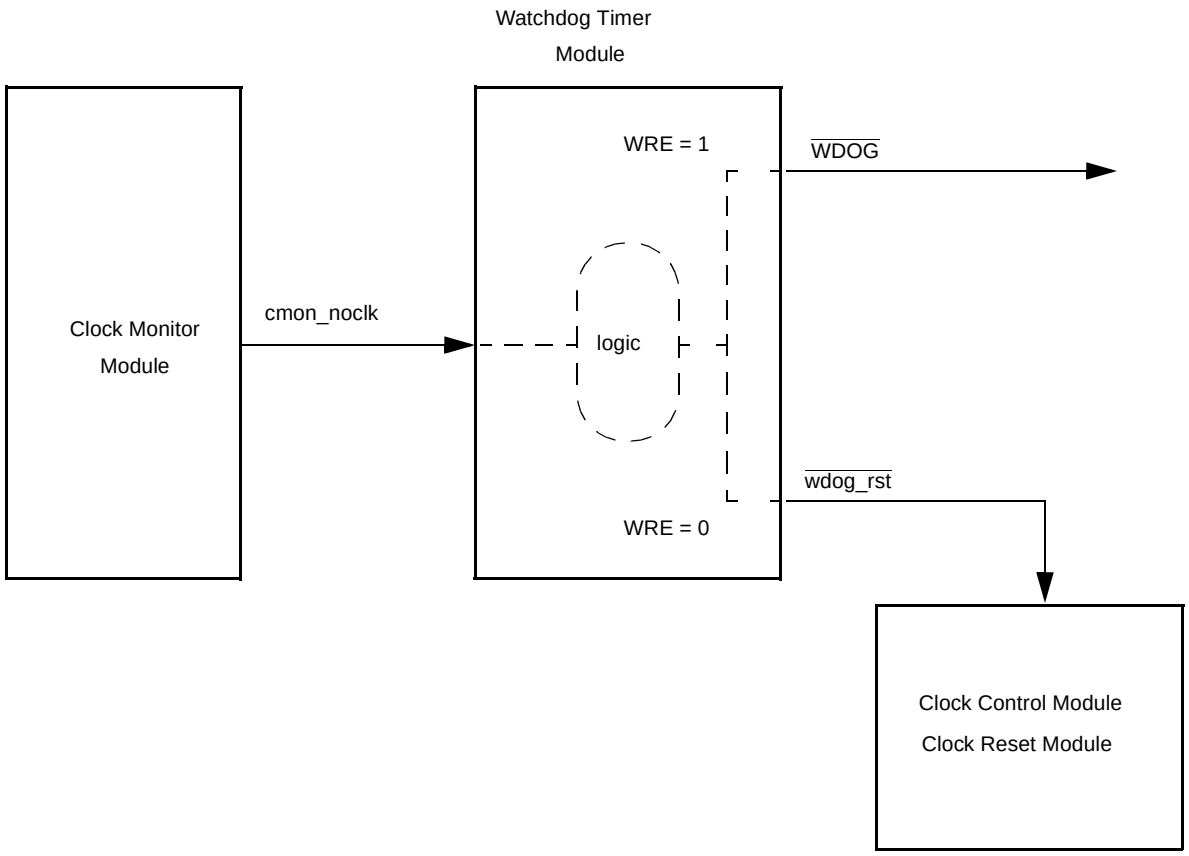
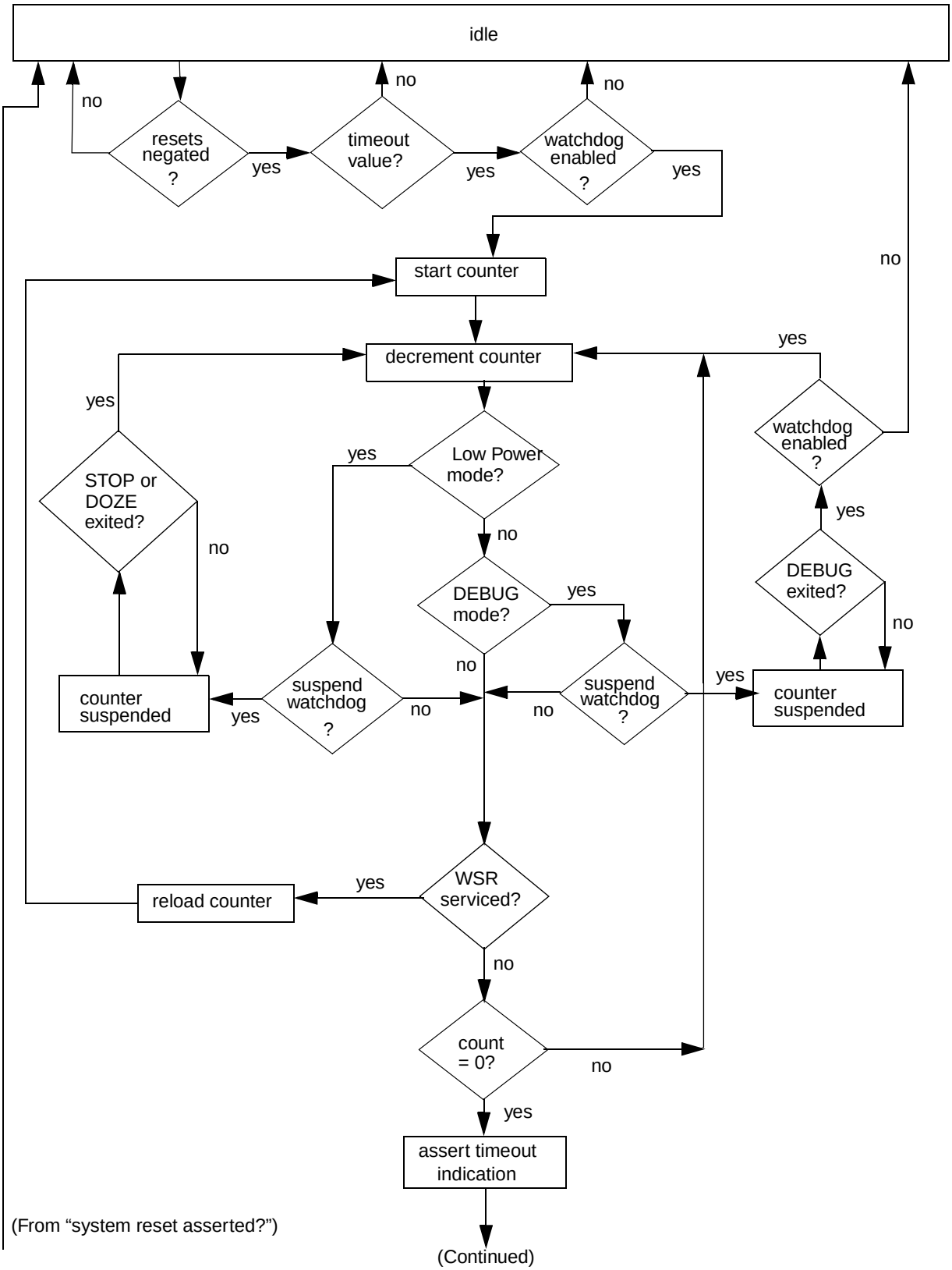


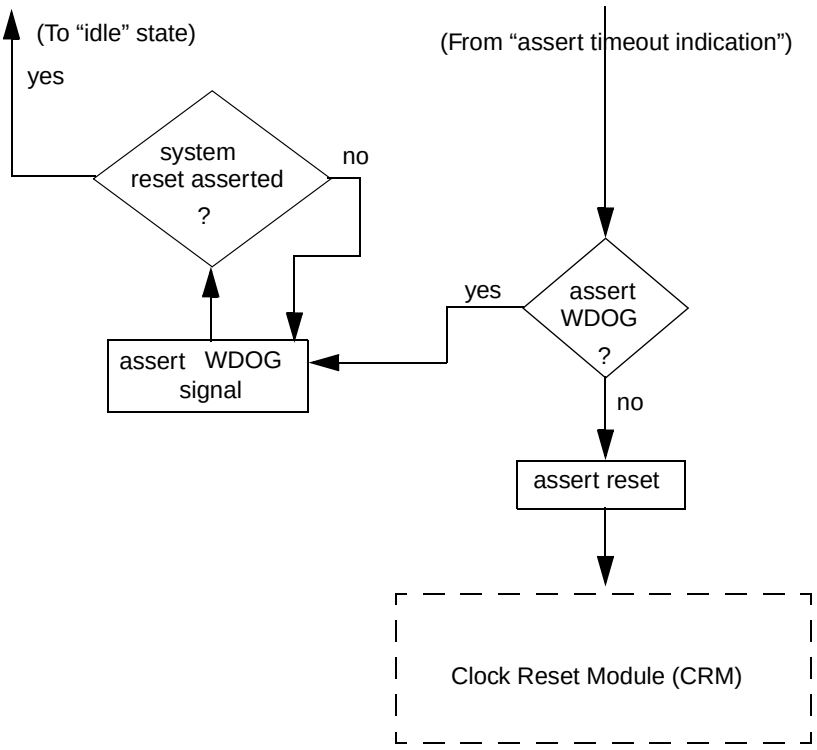
Figure 37-6. Clock Monitor Signals

37.6 Initialization/Application Information

37.6.1 State Machine

The watchdog state machine is shown in Figure 37-7.


idle
no
resets negated ?
yes
no
timeout value?
yes
no
watchdog enabled ?
yes
start counter
no
decrement counter
yes
yes
STOP or DOZE exited?
no
yes
Low Power mode?
no
watchdog enabled ?
yes
DEBUG mode?
yes
DEBUG exited?
no
no
counter suspended
yes
suspend watchdog ?
no
no
suspend watchdog ?
yes
counter suspended
WSR serviced?
yes
reload counter
no
count = 0?
no
yes
assert timeout indication
(From “system reset asserted?”)
(Continued)



NOTE: A system reset will force the state machine to “idle” at any time during countdown.

Figure 37-7. Counter State Machine

Book II, Part 7: System Control Peripherals

Introduction

The transfer of data between modules are controlled by the following system control peripherals:

AHB-Lite to IPS (AIPS)

The AHB-Lite to IPS (AIPS) interfaces—AIPS A and AIPS B—facilitate proper communication between the ARM-11 platform and devices that use AHB-Lite with peripherals that are IPS-compliant. Each AIPS in this device handles a separate MCU peripheral bus: AIPS A interfaces to the MCU Peripheral Bus 1, which supports 16 on-platform peripherals; AIPS B interfaces to the MCU Peripheral Bus 2, which also supports 32 on-platform peripherals). The AIPS provides all of the necessary bus structure and handshaking signals to allow high-speed communication between these peripheral devices and the internal bus structure of the ARM-11.

Multi-Layer AHB Crossbar Switch (MAX)

The purpose of the MAX is to concurrently support up to five simultaneous connections between master ports and slave ports. The MAX supports a 32-bit address bus width, and a 32-bit data bus width at all master and slave ports.

Smart Direct Memory Access (SDMA)

The Smart Direct Memory Access (SDMA) architecture offers highly-competitive DMA Controller features combined with software-based virtual-DMA flexibility. It enables data transfers between peripheral I/O devices and internal/external memories.

The Direct Memory Access (DMA) controller is a critical piece of hardware in a highly integrated IC, such as a 3G baseband or multimedia chip. It helps maximize system performance by off-loading the CPU in dynamic data routing.

Shared Peripheral Bus Arbiter (SPBA)

The Shared Peripheral Bus Arbiter (SPBA) is a three-to-one IP SkyBlue line interface (IP-Bus) arbiter, with a resource locking mechanism. The masters can access up to thirty-one shared peripherals through the SPBA.

Chapter 38
AHB-Lite 2.v6 to IP Bus Interface (AIPS)

This chapter provides an overview of the AMBA high-performance bus (AHB and AHB-Lite) to intellectual property (IP) bus interfaces (AIPS A and AIPS B) for 32-bit and 64-bit platforms.

38.1 Overview

The AHB-Lite to IPS (AIPS) interfaces—AIPS A and AIPS B—facilitate proper communication between the ARM-11 platform and devices that use AHB-Lite with peripherals that are IPS-compliant. Each AIPS in this device handles a separate MCU peripheral bus: AIPS A interfaces to the MCU Peripheral Bus 1, which supports 16 on-platform peripherals; AIPS B interfaces to the MCU Peripheral Bus 2, which also supports 32 on-platform peripherals. The AIPS provides all of the necessary bus structure and handshaking signals to enable high-speed communication between these peripheral devices and the internal bus structure of the ARM-11.

38.1.1 Features

The following list summarizes the key features of the AIPS:

- AIPS supports the IPS slave bus (SkyBlue) signals as shown in Figure 38-1. This interface is only meant for slave peripherals.
- AIPS supports 8-, 16-, and 32-bit IPS peripherals. (Accesses larger than the size of a peripheral are not supported, except to 32-bit memory.)
- AIPS supports a pair of IPS accesses for 64-bit and certain misaligned AHB transfers to 32-bit memory in 64-bit platforms.
- AIPS directly supports up to 32 on-platform IPS peripherals, 32 16-Kbyte external IPS peripherals, and two global external IPS peripheral spaces. AIPS occupies 64 Mbytes of total address space.
- AIPS provides configurable per-module write buffering support.
- AIPS provides configurable per-module and per-master access protections.
- Peripheral read transactions require a minimum of two hclk clocks, and unbuffered write transactions require a minimum of three hclk clocks.
- The AIPS uses one single asynchronous reset and one global clock.
- The AIPS is implemented using MUX-D scan methodology for testability.

The following IPS signals/operations are not supported:

- Slave bus (SkyBlue) signal ips_test_access is not directly supported.

NOTE

An external MUX is required since only one 32-bit read data bus, one ips_xfr_err, and one ips_xfr_wait input is provided.

38.1.2 General Operation

The AIPS is the interface between the AHB-Lite 2.v6 interface and on-chip IPS v3.0 peripherals, as shown in Figure 38-1.

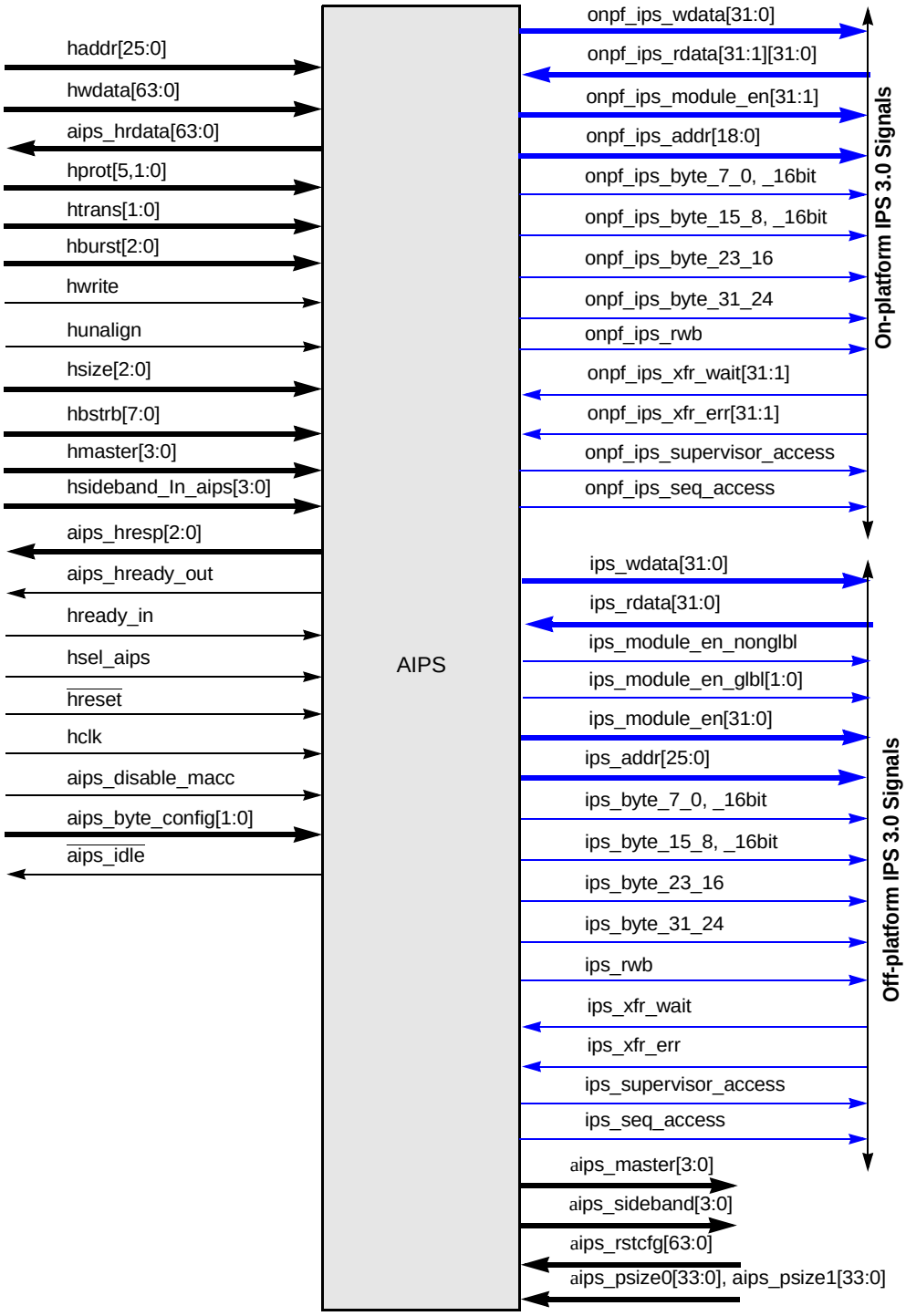


Figure 38-1. AIPS Interface

IPS peripherals are modules that contain readable/writable control and status registers. The AHB master reads and writes these registers through the AIPS. The AIPS generates module enables, the module address, transfer attributes, byte enables, and write data as inputs to the IPS peripherals. The AIPS captures

read data from the IPS interface and drives it on the AHB. The AIPS terminates the transfer by asserting aips_hready_out.

Separate interface ports are provided for on-platform and off-platform peripherals. The distinction is made between on-platform and off-platform to enable platform-based designs that incorporate the AIPS to separate the interface ports, allowing for ease of timing closure. In addition, module selects and control register storage for on-platform peripherals are allocated at synthesis time, allowing only needed resources to be implemented. Off-platform module selects and control register storage do not have the same degree of configurability.

The AIPS occupies a 64-Mbyte portion of the address space. A 0.5-Mbyte portion of this space is allocated to on-platform peripherals. The remaining 63.5 Mbytes are available for off-platform devices. The register maps of the IPS peripherals are located on 16 kilobyte boundaries. Each IPS peripheral is allocated one 16-Kbyte block of the memory map, and is activated by one of the module enables from the AIPS. Up to thirty-two 16-Kbyte external IPS peripherals may be implemented, occupying contiguous blocks of 16 Kbytes. Two global external IPS module enables are available for the remaining 63 Mbytes of address space to allow for customization and expansion of addressed peripheral devices. In addition, a single “non-global” module enable is also asserted whenever any of the thirty-two non-global module enables is asserted.

The following three synthesis configuration options are available:

- To select whether the system bus is 32 bits or 64 bits.
- To select whether 8 or 16 masters are supported.
- To select whether 16 or 32 fixed-size non-global off-platform module enables are supported. This synthesis-time parameter may be used to reduce AIPS overhead when 16 or fewer off-platform modules need support.

Two memory map configurations are supported by the AIPS and the configuration is selected at synthesis time. The AIPS memory map options are shown in Figure 38-2 and Figure 38-3. Two maps are supported to allow for flexible placement of off-platform peripherals at either the low or high end of the memory map.

The connection of a particular module enable to a peripheral, and hence the exact address assignment for an IPS peripheral is system dependent, and is defined in the system specification. Each IPS peripheral selects its internal registers based on the address driven on the ips_addr signals.

The size of an off-platform peripheral is determined by statically driven configuration inputs aips_psize0[33:0] and aips_psize1[33:0]. Synthesis parameters are used to determine the sizes of all on-platform peripherals. The AIPS uses the size of the addressed peripheral to perform proper data byte lane routing only; no bus decomposition (bus sizing) is performed.

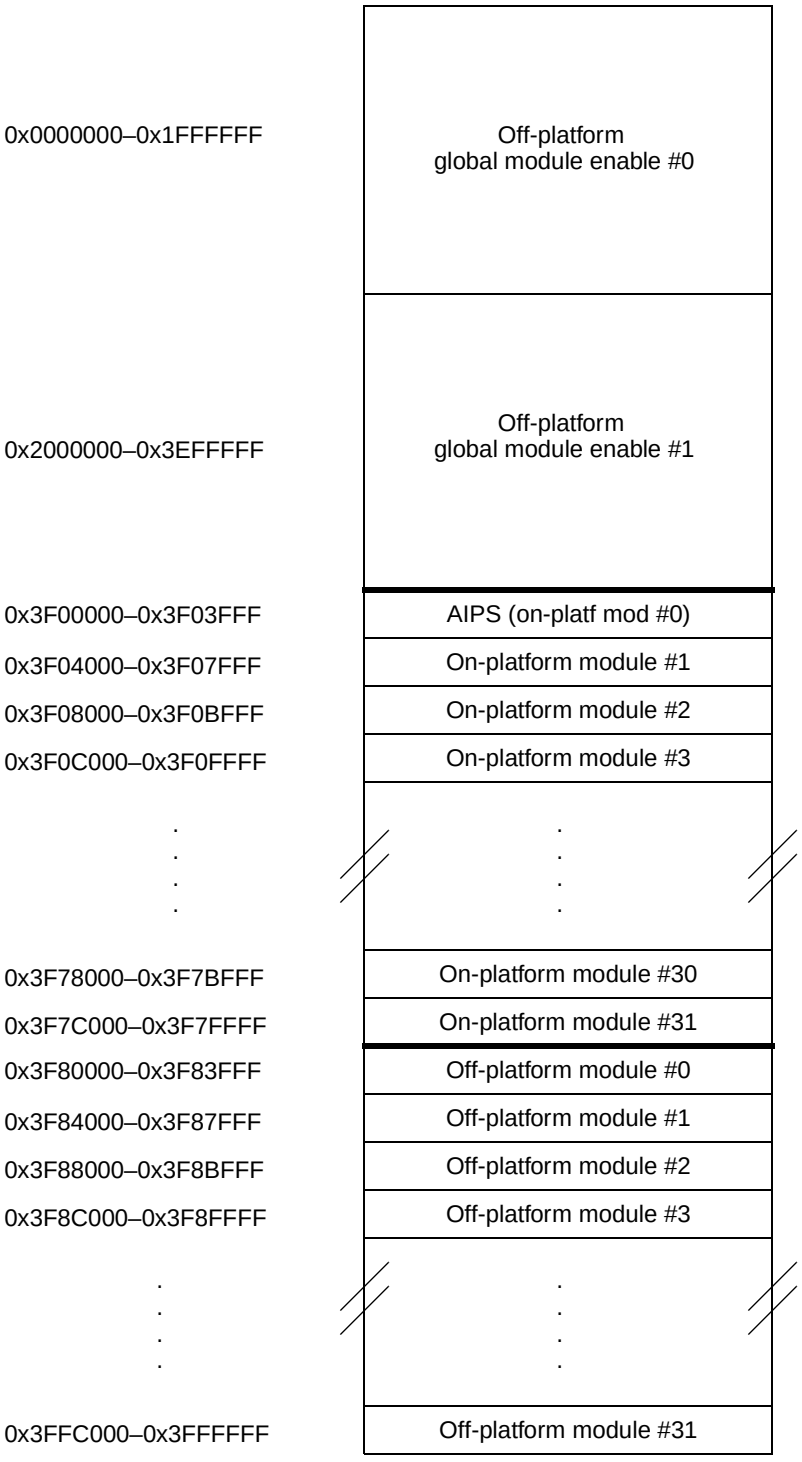


Figure 38-2. AIPS Memory Map #1

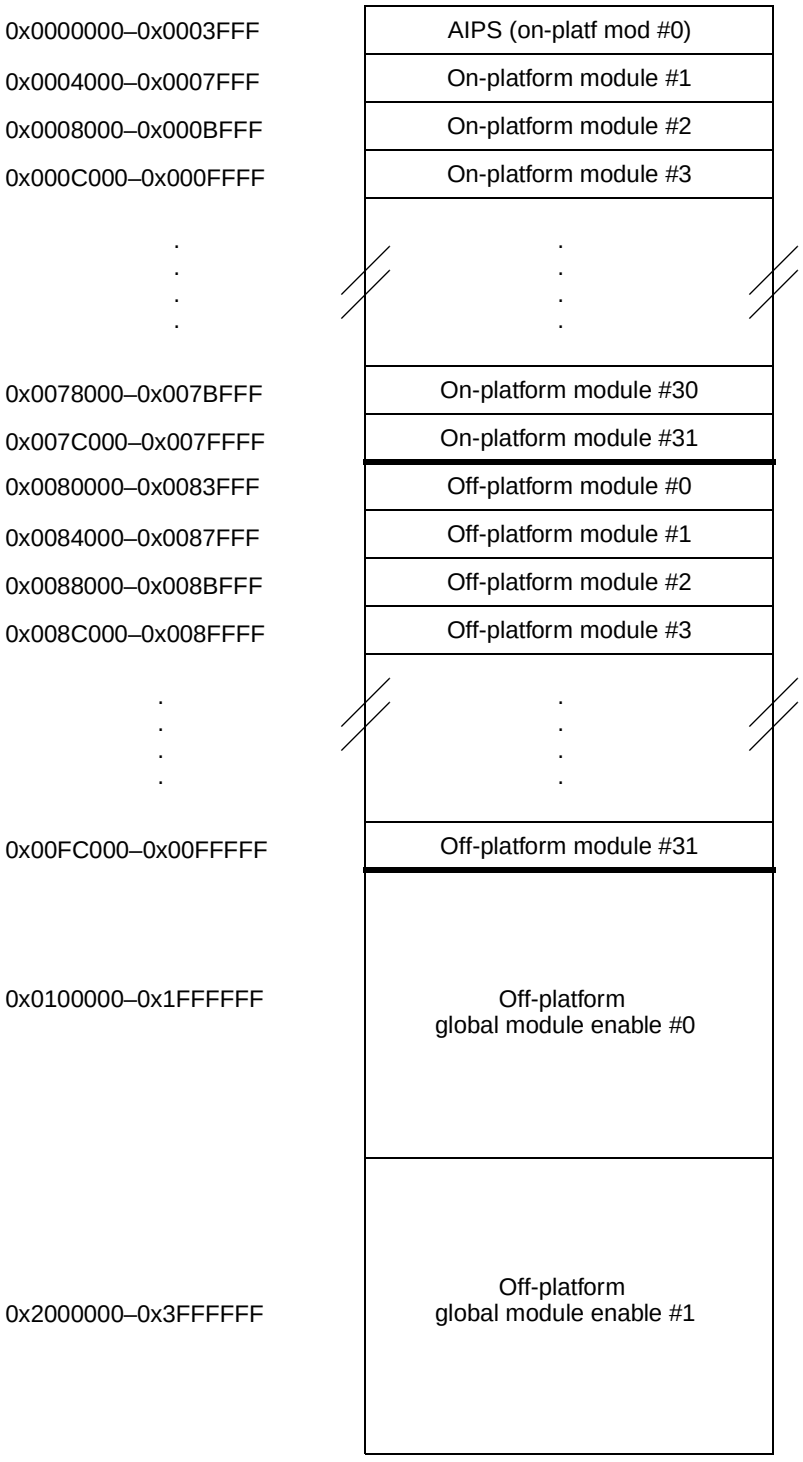


Figure 38-3. AIPS Memory Map #2

The AIPS is responsible for indicating to IPS peripherals if an access is in supervisor or user mode. The AIPS may block user mode accesses to certain IPS peripherals, or it may allow the individual IPS

peripherals to determine if user mode accesses are allowed. In addition, peripherals may be designated as write-protected. The AIPS supports the notion of “trusted” masters for security purposes. Masters may be individually designated as trusted for reads, trusted for writes, or trusted for both reads and writes, as well as being forced to look as though all accesses from a master are in user-mode privilege level. Refer to Section 38.3.2, “Register Summary” for more information.

All peripheral devices are expected to only require aligned accesses equal to or smaller in size than the peripheral size. An exception to this rule is supported for 32-bit peripherals to allow memory to be placed on the IPS.

On 64-bit platforms, the AIPS will initiate a pair of accesses (to 32-bit IPS peripherals only) when the AHB master requests data transfers that are 64 bits, since the IPS data width is limited to 32 bits. These accesses will be performed starting with word offset 0, then 1. A pair of accesses will also be initiated (to 32-bit IPS peripherals only) to handle misaligned system bus accesses which cross a 32-bit boundary. No other bus sizing (sometimes referred to as “bus-decomposition”) is supported. Bus sizing for data accesses may be optionally disabled via an input pin (aips_disable_macc), and an error termination generated instead.

The AIPS also supports buffered writes, allowing write accesses to be terminated on the system bus in a single clock cycle, and then subsequently performed on the IPS interface. Write buffering is controllable on a per-peripheral basis. The AIPS implements a two-entry 32-bit write buffer. Refer to Section 38.3.2, “Register Summary” for more information.

To support higher-speed platforms, the AIPS also supports the delaying of read access and buffered writes by an additional cycle to allow for more decode time following a master-side AHB address phase. This option is configurable via a synthesis parameter, and extends the minimum read cycle to 3 clocks, and buffered writes to 2 clocks.

Synthesis parameters for controlling configurations of the AIPS are described in Section 38.4.1, “AIPS Scalability.”

38.2 AIPS Interface Signals

This section provides information on AIPS interface signals, including the AHB 2.v6 master interface and IPS interface signals.

38.2.1 AIPS Signal Overview

Table 38-1 provides of overview of AIPS module interface signals.

NOTE

Table 38-1 assumes a 64-bit AHB data width. If the AHB data width is 32-bits, the aips_hrdata and hwdata bus indices are 31:0 instead of 63:0, and the HBSTRB indices are 3:0 instead of 7:0.

Table 38-1. AIPS Signals

Signal	Type	Description
AHB Interface		
hclk	Input	AHB reference clock
$\overline{\text{hreset}}$	Input	AHB reset signal
hsel_aips	Input	AIPS module select signal
hmaster[3:0]	Input	Master ID
hsideband_in_aips[3:0]	Input	Sideband inputs from AHB
haddr[25:0]	Input	Address bus
hwdata[63:0]	Input	Write data bus
aips_hrdata[63:0]	Output	Read data bus
htrans[1:0]	Input	Transfer type
hburst[2:0]	Input	Burst type
hwrite	Input	Transfer direction
hprot[5,1:0]	Input	Protection control
hsize[1:0]	Input	Transfer size
hunalign	Input	Unaligned transfer indicator
hbstrb[7:0]	Input	Byte strobes
aips_hready_out	Output	AIPS transfer done
hready_in	Input	Transfer done in
aips_hresp[2:0]	Output	Transfer response
aips_byte_config[1:0]	Input	Byte ordering configuration
aips_disable_macc	Input	Disable Data access bus sizing
$\overline{\text{aips\_idle}}$	Output	AIPS Idle
IPS v3.0 Off-Platform Interface Signals		
ips_addr[25:0]	Output	IPS address bus
ips_rdata[31:0]	Input	IPS read data bus
ips_wdata[31:0]	Output	IPS write data bus
ips_module_en[31:0]	Output	Module enable bus
ips_module_en_glbl[1:0]	Output	Global module enables
ips_module_en_nonglbl	Output	Non-global module enable
ips_rwb	Output	Read/write access signal

Table 38-1. AIPS Signals (continued)

Signal	Type	Description
ips_byte_31_24, ips_byte_23_16, ips_byte_15_8, ips_byte_7_0	Output	IPS byte enables (for 32-bit peripherals)
ips_byte_15_8_16bit ips_byte_7_0_16bit	Output	IPS byte enables (for 16-bit peripherals)
ips_xfr_wait	Input	IPS peripheral wait signal
ips_xfr_err	Input	IPS peripheral error signal
ips_supervisor_access	Output	Supervisor mode access signal
ips_seq_access	Output	Sequential access signal
IPS Off-Platform Sideband Signals		
aips_master[3:0]	Output	IPS Master ID (from hmaster[3:0])
aips_sideband[3:0]	Output	IPS sideband signals (from hsideband_in_aips[3:0])
IPS v3.0 On-Platform Interface Signals		
onpf_ips_addr[18:0]	Output	IPS address bus
onpf_ips_rdata[31:1][31:0]	Input	IPS read data buses
onpf_ips_wdata[31:0]	Output	IPS write data bus
onpf_ips_module_en[31:1]	Output	Module enable bus
onpf_ips_rwb	Output	Read/write access signal
onpf_ips_byte_31_24, onpf_ips_byte_23_16, onpf_ips_byte_15_8, onpf_ips_byte_7_0	Output	IPS byte enables (for 32-bit peripherals)
onpf_ips_byte_15_8_16bit onpf_ips_byte_7_0_16bit	Output	IPS byte enables (for 16-bit peripherals)
onpf_ips_xfr_wait[31:1]	Input	IPS peripheral wait bus
onpf_ips_xfr_err[31:1]	Input	IPS peripheral error bus
onpf_ips_supervisor_access	Output	Supervisor mode access signal
onpf_ips_seq_access	Output	Sequential access signal
AIPS Configuration Signals		
aips_rstcfg[63:0]	Input	AIPS master protection configuration

38.2.2 AIPS Signal Descriptions

The following paragraphs provide descriptions of AIPS signals.

38.2.2.1 AHB Interface Signals

38.2.2.1.1 System Clock—hclk

This signal is the AHB reference clock.

38.2.2.1.2 System Reset—$\overline{\text{hreset}}$

This signal is the AHB reset signal.

38.2.2.1.3 AIPS Select Signal—hsel_aips

The AIPS select signal comes from the AHB slave address decoder (hsel_aips). If the AIPS is the only module on the AHB port, this signal should be tied asserted.

38.2.2.1.4 AHB Master—hmaster[3:0]

These input signals identify the current AHB bus master. They are used in the protection scheme to control master access to peripherals.

38.2.2.1.5 AIPS Sideband Inputs—hsideband_in_aips[3:0]

These input signals are provided for system specific use. They are driven as inputs from the AHB during the AHB address phase, and are provided on the IPS interface for the duration of an IPS access.

38.2.2.1.6 Address Bus—haddr[25:0]

These input signals are the address bus from the AHB. The AIPS decodes haddr[25:14] to determine which of the on-platform or off-platform IPS peripherals is being accessed (see Figure 38-2). The address bus is also passed on to the IPS peripherals to allow decoding of their registers.

38.2.2.1.7 Write Data Bus—hwdata[63:0] (hwdata[31:0])

These input signals are the write data bus from the AHB master. This bus is used to transfer data from the AHB master to the bus slaves during write operations. When the AIPS is configured for a 32-bit system, **hwdata[63:32]** are not present.

38.2.2.1.8 Read Data Bus—aips_hrdata[63:0] (aips_hrdata[31:0])

These output signals are the read data bus going to the read data MUX to be sent on to the AHB master. This bus is used to transfer data from the AIPS to the AHB master during read operations. When the AIPS is configured for a 32-bit system, aips_hrdata[63:32] are not present.

38.2.2.1.9 Transfer Type—htrans[1:0]

These inputs indicate the type of the current transfer, which can be sequential/non-sequential or idle/busy. Table 38-2 shows htrans[1:0] encoding.

Table 38-2. Transfer Type Encoding

htrans[1]	htrans[0]	Access Type
0	0	IDLE. No data transfer is required
0	1	BUSY. Master is busy, burst transfer continues.
1	0	NONSEQ. Indicates the first transfer of a burst, or a single transfer. Address and control signals are unrelated to the previous transfer
1	1	SEQ. Indicates the continuation of a burst. Address and control signals are related to the previous transfer. Control signals are the same, Address has been incremented by the size of the data transferred (optionally wrapped)

38.2.2.1.10 Burst Type—hburst[2:0]

The hburst[2:0] signals indicate the burst type for a bus transfer. Table 38-3 shows the definitions of the hburst[2:0] encodings.

Table 38-3. hburst[2:0] Burst Type Encoding

hburst[2:0]	Burst Type
000	SINGLE—No burst, single beat only
001	INCR—Incrementing burst of unspecified length
010	WRAP4—4-beat wrapping burst
011	INCR4—4-beat incrementing burst
100	WRAP8—8-beat wrapping burst
101	INCR8—8-beat incrementing burst
110	WRAP16—16-beat wrapping burst
111	INCR16- 16-beat incrementing burst

38.2.2.1.11 Transfer Direction—hwrite

This input indicates a write transfer when high and a read transfer when low.

38.2.2.1.12 Protection Control—hprot[5,1:0]

The hprot5 input indicates an exclusive transfer when high and a non-exclusive transfer when low. Any exclusive write access will be terminated by the AIPS with error termination, and no IPS access will be initiated. The exclusive attribute is ignored for read accesses.

The hprot1 input indicates a privileged transfer when high and a user transfer when low.

The hprot0 input indicates an instruction transfer when low and a data transfer when high. It is used in conjunction with the aips_disable_macc signal to qualify instruction vs. data transfers.

38.2.2.1.13 Transfer Size—hsize[1:0]

These input signals indicate the size of the transfer, which is byte (8-bit), halfword (16-bit), word (32-bit), or doubleword (64-bit). Table 38-4 shows the encoding of this bit field.

Table 38-4. Transfer Size Encoding

hsize[1:0]	Transfer Size
0 0	Byte
0 1	Halfword (2 bytes)
1 0	Word (4 bytes)
1 1	Doubleword (8 bytes)

38.2.2.1.14 Transfer Unaligned—hunalign

This input signal indicates that the transfer is unaligned, and the active byte lanes cannot be determined directly from the low order address and size information.

38.2.2.1.15 Transfer Byte Strobes—hbstrb[7:0] (hbstrb[3:0])

These input signals indicate the active byte lanes for the transfer. Table 38-5 shows the mapping of AHB 2.v6 bytes strobes to data bus lanes for a 64-bit AHB data bus.

Table 38-5. hbstrb[7:0] to Byte Address Mappings, 64-bit AHB

Memory Byte Address haddr[2:0]	Wired To h{r,w} Data bits	Corresponding Byte Strobe Signal
000	7:0	hbstrb[0]
001	15:8	hbstrb[1]
010	23:16	hbstrb[2]
011	31:24	hbstrb[3]
100	39:32	hbstrb[4]
101	47:40	hbstrb[5]
110	55:48	hbstrb[6]
111	63:56	hbstrb[7]

Table 38-6 shows the mapping of AHB 2.v6 bytes strobes to data bus lanes for a 32-bit AHB data bus.

Table 38-6. hbstrb[3:0] to Byte Address Mappings, 32-bit AHB

Memory Byte Address haddr[1:0]	Wired To h{r,w} Data Bits	Corresponding Byte Strobe Signal
00	7:0	hbstrb[0]
01	15:8	hbstrb[1]

Table 38-6. hbstrb[3:0] to Byte Address Mappings, 32-bit AHB (continued)

Memory Byte Address haddr[1:0]	Wired To h{r,w} Data Bits	Corresponding Byte Strobe Signal
10	23:16	hbstrb[2]
11	31:24	hbstrb[3]

38.2.2.1.16 AIPS Transfer Done—aips_hready_out

This output signal indicates to the AHB master when an AIPS transfer has finished on the bus. This signal may be driven low to extend a transfer.

38.2.2.1.17 Transfer Done In—hready_in

This input signal indicates to the AIPS when any of the slaves has finished their transfer on the AHB bus. It may be tied to aips_hready_out if the AIPS is the only slave on the AHB port.

38.2.2.1.18 Transfer Response—aips_hresp[2:0]

These output signals provide the AHB master with additional information on the status of the transfer. If aips_hresp[0] is high, then an error has occurred. If aips_hresp[0] is low, then the transfer was completed successfully. aips_hresp[2:1] are tied low since no SPLIT or RETRY support is provided, and no XFAIL response is possible from the IPS interface.

38.2.2.1.19 Byte Ordering Configuration—aips_byte_config[1:0]

These input signals configure the byte ordering of connections between the IPS and the AHB data buses. Refer to Section 38.4.4, “Data Byte Lane and Byte Strobe Mapping” for a detailed description of operation.

38.2.2.1.20 Disable Multi-access for Data—aips_disable_macc

This input signal provides the capability of disabling bus sizing for data accesses (hprot0 is high). Data accesses which cross a 32-bit boundary and 64-bit data accesses will be terminated with an error, and no module select will be asserted. This signal is ignored for instruction accesses (hprot0 is low). This signal should be tied to a static value.

38.2.2.1.21 AIPS Idle—$\overline{\text{aips\_idle}}$

This active-low output signal indicates whether the AIPS has a current or pending transfer to perform on the IPS bus, or whether it is currently idle. This signal negates when an access initially becomes pending, and will assert once all transfers to the IPS (buffered and non-buffered) have completed with no further access pending.

38.2.2.2 Off-Platform IPS V3.0 Interface Signals

38.2.2.2.1 IP Address Bus—ips_addr[25:0]

These outputs from the AIPS provide the address to the various off-platform IPS peripherals to allow them to decode and select from the 16 kilobyte space allocated to each IPS peripheral. They also are provided to allow external module selects to be generated using the ips_module_en_glbl signal. This address is registered from the AHB. IPS peripherals do not have to decode all of the address bits (the registers may be multiply-mapped).

38.2.2.2.2 IP Read Data Bus—ips_rdata[31:0]

This input bus is the read data path between the AIPS and the off-platform IPS peripherals. The AIPS captures the data from the IPS peripherals on reads and drives it to the AHB on aips_hrdata[63:0]. For 32-bit peripherals, read data is provided on ips_rdata[31:0]. For 16-bit peripherals, read data is provided on ips_rdata[15:0]. For 8-bit peripherals, read data is provided on ips_rdata[7:0].

Since there are multiple read data buses coming from the off-platform IPS peripherals but only one read data bus input to the AIPS, the integrator must externally MUX all the off-platform peripheral read data buses based on the ips_module_en outputs of the AIPS.

38.2.2.2.3 IP Write Data Bus

This output bus is the write data path between the AIPS and the off-platform IPS peripherals. The AIPS drives data on writes from hwdata[31:0] (hwdata[63:0] for 64-bit systems) to ips_wdata[31:0] This write data bus is common to all off-platform IPS peripherals. For 32-bit peripherals, write data is provided on ips_wdata[31:0]. For 16-bit peripherals, write data is provided on ips_wdata[15:0]. For 8-bit peripherals, write data is provided on ips_wdata[7:0].

38.2.2.2.4 ips_module_en[31:0], ips_module_en_glbl[1:0], ips_module_en_nonglbl

The AIPS asserts one (and only one) module enable at a time from ips_module_en[31:0] and ips_module_en_glbl[1:0]. The module enable signal is driven off the rising edge of hclk and remains valid until the rising edge of hclk when the ips_xfr_wait signal is negated. The ips_module_en_nonglbl is asserted whenever one of ips_module_en[31:0] is asserted, allowing it to cover the entire address space of the ips_module_en[31:0] outputs.

The haddr[25:14] bits of the AHB address are decoded to determine which module enable is to be generated. Table 38-7 and Table 38-8 show the address decoding performed to determine which off-platform module enable is asserted, based upon the memory map configuration (Map #1 vs. Map #2).

Table 38-7. Off-Platform IPS Peripheral Location Decoder for AIPS MAP #1[1]

haddr[25:14]	ips_module_en	haddr[25:14]	ips_module_en	haddr[25:14]	ips_module_en
111111100000	0[2]	111111101011	11	111111110110	22
111111100001	1	111111101100	12	111111110111	23
111111100010	2	111111101101	13	111111111000	24
111111100011	3	111111101110	14	111111111001	25

Table 38-7. Off-Platform IPS Peripheral Location Decoder for AIPS MAP #1[1] (continued)

haddr[25:14]	ips_module_en	haddr[25:14]	ips_module_en	haddr[25:14]	ips_module_en
111111100100	4	111111101111	15	111111111010	26
111111100101	5	111111110000	16	111111111011	27
111111100110	6	111111110001	17	111111111100	28
111111100111	7	111111110010	18	111111111101	29
111111101000	8	111111110011	19	111111111110	30
111111101001	9	111111110100	20	111111111111	31
111111101010	10	111111110101	21	000000000000–011111111111	ips_module_en_glbl[0]
				100000000000–111110111111	ips_module_en_glbl[1]

1 Assumes hsel_aips is asserted.

2 ips_module_en_nonglbl also asserted

Table 38-8. Off-Platform IPS Peripheral Location Decoder for AIPS MAP #2[1]

haddr[25:14]	ips_module_en	haddr[25:14]	ips_module_en	haddr[25:14]	ips_module_en
000000100000	0[2]	000000101011	11	000000110110	22
000000100001	1	000000101100	12	000000110111	23
000000100010	2	000000101101	13	000000111000	24
000000100011	3	000000101110	14	000000111001	25
000000100100	4	000000101111	15	000000111010	26
000000100101	5	000000110000	16	000000111011	27
000000100110	6	000000110001	17	000000111100	28
000000100111	7	000000110010	18	000000111101	29
000000101000	8	000000110011	19	000000111110	30
000000101001	9	000000110100	20	000000111111	31
000000101010	10	000000110101	21	000001000000–011111111111	ips_module_en_glbl[0]
				100000000000–111111111111	ips_module_en_glbl[1]

1 Assumes hsel_aips is asserted.

2 ips_module_en_nonglbl also asserted

38.2.2.2.5 ips_rwb

This signal is used to indicate to the off-platform peripheral that the current access is either a read or write access. A logic high indicates a read and a logic low indicates a write. It is asserted after the rising edge of hclk along with the ips_module_en. This signal remains asserted for the duration of the peripheral access.

38.2.2.2.6 ips_byte_31_24, ips_byte_23_16, ips_byte_15_8, ips_byte_7_0

Byte enables are generated based on the hbstrb[7:0] signals. These byte enables are used to select the corresponding bytes on the IPS for 32-bit peripherals. Mapping of the byte strobes is dependent on the aips_byte_config[1:0] inputs and is detailed in Section 38.4.4, "Data Byte Lane and Byte Strobe Mapping."

38.2.2.2.7 ips_byte_15_8_16bit, ips_byte_7_0_16bit

Byte enables are generated based on the hbstrb[7:0] signals. These byte enables are used to select the corresponding bytes on the IPS for 16-bit peripherals. Mapping of the byte strobes is dependent on the aips_byte_config[1:0] inputs and is detailed in Section 38.4.4, "Data Byte Lane and Byte Strobe Mapping."

38.2.2.2.8 ips_xfr_wait

Each connected off-platform IPS peripheral has a separate ips_xfr_wait signal. The ips_xfr_wait signal is sampled by the AIPS on the rising edge of hclk. It should be asserted by the selected peripheral to indicate that the current access cannot be completed in the given cycle and a wait state is needed.

Since there are multiple ips_xfr_wait signals coming from the off-platform peripherals and only one ips_xfr_wait input to the AIPS, the integrator must externally MUX all off-platform peripheral ips_xfr_wait signals based on the ips_module_en outputs of the AIPS. The integrator should ensure that if a vacant off-platform IPS peripheral location is selected, the ips_xfr_wait input to the AIPS is driven to a logic low.

38.2.2.2.9 ips_xfr_err

Each off-platform IPS peripheral has a separate ips_xfr_err signal. This error signal is asserted by the peripheral to indicate that the current access is not valid and needs to be terminated with an error. This signal is sampled by the AIPS at the positive edge of hclk when the ips_xfr_wait signal is negated.

Since there are multiple ips_xfr_err signals coming from the off-platform peripherals and only one ips_xfr_err input to the AIPS, the integrator must externally MUX all off-platform peripheral ips_xfr_err signals based on the ips_module_en outputs of the AIPS. The integrator should ensure that if a vacant off-platform IPS peripheral location is selected, the ips_xfr_err input to the AIPS is driven to a logic high (forcing an error for any access to an unoccupied off-platform IPS peripheral location).

38.2.2.2.10 ips_supervisor_access

This signal is asserted at the positive edge of hclk to indicate to the off-platform peripheral that the current access is being performed in supervisor mode.

38.2.2.2.11 ips_seq_access

This signal is asserted at the positive edge of hclk to indicate to the off-platform peripheral that the next access will be sequential to the current access. This signal is based on the assertion of a qualified hburst[2:0] burst type. If ips_seq_access is asserted for the current transfer, and the next transfer on the AHB is not of type SEQ, then a dead cycle in which ips_module_en is negated may need to be inserted to

correctly implement the IPS protocol. The qualified burst types are INCR bursts, and certain WRAP bursts which are effectively equivalent to INCR bursts (for example, WRAP bursts that begin on a burst size boundary).

38.2.2.3 Off-Platform IPS Sideband Signals

38.2.2.3.1 aips_master[3:0]

These signals are driven along with ips_addr to indicate which master on the AHB bus is performing the current off-platform IPS access. Most slave devices will not need this information, but it is provided to allow a slave to condition its response based on the accessing master.

38.2.2.3.2 aips_sideband[3:0]

These signals are driven during an IPS access with the values provided on hsideband_in_aips[3:0] during the address phase of the AHB bus access which targets the AIPS memory map. These signals are provided for system-dependent functions.

38.2.2.4 On-Platform IPS V3.0 Interface Signals

38.2.2.4.1 IP Address Bus—onpf_ips_addr[18:0]

These outputs from the AIPS provide the address to the various on-platform IPS peripherals to allow them to decode and select from the 16-kilobyte space allocated to each IPS peripheral. This address is registered from the AHB. IPS peripherals do not have to decode all of the address bits (the registers may be multiply-mapped).

38.2.2.4.2 IP Read Data Buses—onpf_ips_rdata[31:1][31:0]

These input buses are the read data paths between the AIPS and the on-platform IPS peripherals. The AIPS captures the data from the on-platform IPS peripherals on reads and drives it to the AHB on aips_hrdata[63:0]. For 32-bit peripherals, read data is provided on ips_rdata[31:0]. For 16-bit peripherals, read data is provided on ips_rdata[15:0]. For 8-bit peripherals, read data is provided on ips_rdata[7:0]. Separate read data buses are provided for each on-platform peripheral which is actually present (see Section 38.4.1, “AIPS Scalability”). If a peripheral is not present, the bus is not implemented, and muxing is simplified internally.

Since there are multiple read data buses coming from the on-platform IPS peripherals, the AIPS muxes all on-platform peripheral read data buses based on the onpf_ips_module_en outputs of the AIPS.

38.2.2.4.3 IP Write Data Bus—onpf_ips_wdata[31:0]

This output bus is the write data path between the AIPS and the on-platform IPS peripherals. The AIPS drives data on writes from hwdata[31:0] or hwdata[63:32] (for 64-bit systems) to ips_wdata[31:0] depending on the value of haddr[2]. This write data bus is common to all on-platform IPS peripherals. For 32-bit peripherals, write data is provided on ips_wdata[31:0]. For 16-bit peripherals, write data is provided on ips_wdata[15:0]. For 8-bit peripherals, write data is provided on ips_wdata[7:0].

38.2.2.4.4 onpf_ips_module_en[31:1]

The AIPS asserts one (and only one) module enable at a time from onpf_ips_module_en[31:1]. The module enable signal is driven off the rising edge of hclk and remains valid until the rising edge of hclk when the onpf_ips_xfr_wait signal is negated.

The haddr[25:14**]** bits of the AHB address are decoded to determine which on-platform module enable is to be generated. Table 38-9 and Table 38-10 show the address decoding performed to determine which on-platform module enable is asserted, based upon the memory map configuration (Map #1 vs. Map #2).

Table 38-9. On-Platform IPS Peripheral Location Decoder for AIPS MAP #1[1]

haddr[25:14]	onpf_ips_module_en	haddr[25:14]	onpf_ips_module_en	haddr[25:14]	onpf_ips_module_en
111111000000	0[2]	111111001011	11	111111010110	22
111111000001	1	111111001100	12	111111010111	23
111111000010	2	111111001101	13	111111011000	24
111111000011	3	111111001110	14	111111011001	25
111111000100	4	111111001111	15	111111011010	26
111111000101	5	111111010000	16	111111011011	27
111111000110	6	111111010001	17	111111011100	28
111111000111	7	111111010010	18	111111011101	29
111111001000	8	111111010011	19	111111011110	30
111111001001	9	111111010100	20	111111011111	31
111111001010	10	111111010101	21		

1 Assumes hsel_aips is asserted.

2 This is an internal signal

Table 38-10. On-Platform IPS Peripheral Location Decoder for AIPS MAP #2[1]

haddr[25:14]	onpf_ips_module_en	haddr[25:14]	onpf_ips_module_en	haddr[25:14]	onpf_ips_module_en
000000000000	0	000000001011	11	000000010110	22
000000000001	1	000000001100	12	000000010111	23
000000000010	2	000000001101	13	000000011000	24
000000000011	3	000000001110	14	000000011001	25
000000000100	4	000000001111	15	000000011010	26
000000000101	5	000000010000	16	000000011011	27
000000000110	6	000000010001	17	000000011100	28
000000000111	7	000000010010	18	000000011101	29
000000001000	8	000000010011	19	000000011110	30

Table 38-10. On-Platform IPS Peripheral Location Decoder for AIPS MAP #2[1] (continued)

haddr[25:14]	onpf_ips_module_en	haddr[25:14]	onpf_ips_module_en	haddr[25:14]	onpf_ips_module_en
000000001001	9	000000010100	20	000000011111	31
000000001010	10	000000010101	21		

1 Assumes hsel_aips is asserted.

38.2.2.4.5 onpf_ips_rwb

This signal is used to indicate to the on-platform peripheral that the current access is either a read or write access. A logic high indicates a read and a logic low indicates a write. It is asserted after the rising edge of hclk along with the onpf_ips_module_en. This signal remains asserted for the duration of the peripheral access.

38.2.2.4.6 onpf_ips_byte_31_24, onpf_ips_byte_23_16, onpf_ips_byte_15_8, onpf_ips_byte_7_0

Byte enables are generated based on the hbstrb[7:0] signals. These byte enables are used to select the corresponding bytes on the IPS for 32-bit peripherals. Mapping of the byte strobes is dependent on the aips_byte_config[1:0] inputs and is detailed in Section 38.4.4, “Data Byte Lane and Byte Strobe Mapping.”

38.2.2.4.7 onpf_ips_byte_15_8_16bit, onpf_ips_byte_7_0_16bit

Byte enables are generated based on the hbstrb[7:0] signals. These byte enables are used to select the corresponding bytes on the IPS for 16-bit peripherals. Mapping of the byte strobes is dependent on the aips_byte_config[1:0] inputs and is detailed in Section 38.4.4, “Data Byte Lane and Byte Strobe Mapping.”

38.2.2.4.8 onpf_ips_xfr_wait[31:1]

Each on-platform IPS peripheral has a separate ips_xfr_wait signal. The onpf_ips_xfr_wait signal is sampled by the AIPS on the rising edge of hclk. It should be asserted by the selected peripheral to indicate that the current access cannot be completed in the given cycle and a wait state is needed.

Separate ips_xfr_wait signals are provided for each on-platform peripheral which is actually present (see Section 38.4.1, “AIPS Scalability”).

Since there are multiple ips_xfr_wait signals coming from the on-platform IPS peripherals, the AIPS muxes all on-platform peripheral ips_xfr_wait signals based on the onpf_ips_module_en outputs of the AIPS. If a peripheral is not present, the signal is not implemented, and muxing is simplified internally.

38.2.2.4.9 onpf_ips_xfr_err[31:1]

Each on-platform IPS peripheral has a separate ips_xfr_err signal. This error signal is asserted by the peripheral to indicate that the current access is not valid and needs to be terminated with an error. The

onpf_ips_xfr_err signal is sampled by the AIPS at the positive edge of hclk when the onpf_ips_xfr_wait signal is negated.

Separate ips_xfr_err signals are provided for each on-platform peripheral which is actually present (see Section 38.4.1, “AIPS Scalability”).

Since there are multiple ips_xfr_err signals coming from the on-platform IPS peripherals, the AIPS muxes all on-platform peripheral ips_xfr_err signals based on the onpf_ips_module_en outputs of the AIPS. If a peripheral is not present, the signal is not implemented, and muxing is simplified internally. The AIPS will generate internal error responses for on-platform peripherals which are not present.

38.2.2.4.10 onpf_ips_supervisor_access

This signal is asserted at the positive edge of hclk to indicate to the on-platform peripheral that the current access is being performed in supervisor mode.

38.2.2.4.11 onpf_ips_seq_access

This signal is asserted at the positive edge of hclk to indicate to the on-platform peripheral that the current access is sequential to the previous access. This signal is based on the assertion of a SEQ htrans[1:0] type qualified with the hburst[2:0] burst type.

38.2.2.5 AIPS Configuration Signals

38.2.2.5.1 aips_rstcfg[63:0]

These signals are loaded into the Master Privilege registers (MPRs) during reset. They should be driven according to the desired default master privilege values. aips_rstcfg[63:32] maps to MPROT[8:15] with aips_rstcfg[63:60] corresponding to MPROT[8], and aips_rstcfg[31:0] maps to MPROT[0:7] with aips_rstcfg[31:28] corresponding to MPROT[0].

38.2.2.5.2 aips_psize1[33:0], aips_psize0[33:0]

These signals are used to indicate the size of each off-platform peripheral. They should be driven in a static manner (tied-off). Based on the size, the AIPS will properly drive/receive data on the appropriate byte lanes of the IPS interface. Bit 0 of these buses corresponds to off-platform peripheral 0, bit 1 to off-platform peripheral 1, ——, bit 31 to off-platform peripheral 31, bit 32 to off-platform glbl0, and bit 33 to off-platform glbl1.

Table 38-11 shows the size definitions used.

Table 38-11. Off-Platform Peripheral Size Control

aips_psize1[n]	aips_psize0[n]	Corresponding Peripheral n Size
0	0	8-bit
0	1	16-bit

Table 38-11. Off-Platform Peripheral Size Control (continued)

aips_psize1[n]	aips_psize0[n]	Corresponding Peripheral n Size
1	0	32-bit
1	1	reserved (64-bit)

38.3 Memory Map and Register Definition

This section provides information on AIPS registers. Section 38.3.3, "Register Descriptions" provides the detailed descriptions for all of the AIPS registers.

There are eleven registers that control the AIPS. All registers are 32-bit registers and can only be accessed in supervisor mode by trusted bus masters. Additionally, these registers must only be read from or written to by a 32-bit aligned access. AIPS registers are mapped into the PACR0 address space.

Two system clocks are required for read accesses and three system clocks are required for write accesses to the AIPS registers.

38.3.1 AIPS A and AIPS B Memory Map

The memory map for the AIPS program-visible registers is shown in Table 38-12.

Table 38-12. AIPS A and AIPS B Memory Map

Address	Register	Access	Reset Value	Section/Page
General Registers				
0x43F0_0000 (MPR_1) 0x53F0_0000 (MPR_1)	Master Privilege Register 1	R/W	Undefined	38.3.3.1/38-24
0x43F0_0004 (MPR_2) 0x53F0_0004 (MPR_2)	Master Privilege Register 2	R/W	Undefined	38.3.3.1/38-24
0x43F0_0020 (PACR_1) 0x53F0_0020 (PACR_1)	Peripheral Access Control Register 1	R/W	0x5444_4444	38.3.3.2/38-26
0x43F0_0024 (PACR_2) 0x53F0_0024 (PACR_2)	Peripheral Access Control Register 2	R/W	0x4444_4444	38.3.3.2/38-26
0x43F0_0028 (PACR_3) 0x53F0_0028 (PACR_3)	Peripheral Access Control Register 3	R/W	0x4444_4444	38.3.3.2/38-26
0x43F0_002C (PACR_4) 0x53F0_002C (PACR_4)	Peripheral Access Control Register 4	R/W	0x4444_4444	38.3.3.2/38-26
0x43F0_0040 (OPACR_1) 0x53F0_0040 (OPACR_1)	Off-Platform Peripheral Access Control Register 1	R/W	0x5444_4444	38.3.3.3/38-29
0x43F0_0044 (OPACR_2) 0x53F0_0044 (OPACR_2)	Off-Platform Peripheral Access Control Register 2	R/W	0x4444_4444	38.3.3.3/38-29
0x43F0_0048 (OPACR_3) 0x53F0_0048 (OPACR_3)	Off-Platform Peripheral Access Control Register 3	R/W	0x4444_4444	38.3.3.3/38-29

Table 38-12. AIPS A and AIPS B Memory Map (continued)

Address	Register	Access	Reset Value	Section/Page
0x43F0_004C (OPACR_4) 0x53F0_004C (OPACR_4)	Off-Platform Peripheral Access Control Register 4	R/W	0x4444_4444	38.3.3.3/38-29
0x43F0_0050 (OPACR_5) 0x53F0_0050 (OPACR_5)	Off-Platform Peripheral Access Control Register 5	R/W	0x4444_4444	38.3.3.3/38-29

38.3.2 Register Summary

Figure 38-4 shows the key to the register fields, and Table 38-14 shows the register figure conventions.

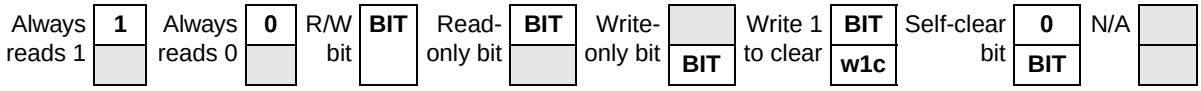


Figure 38-4. Key to Register Fields

Table 38-14. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writeable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Figure 38-15 shows the AIPS register summary.

Table 38-15. AIPS Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x43F0_0000 (MPR_1) 0x53F0_0000 (MPR_1)	R	MBW0	MTR0	MTW0	MPL0	MBW1	MTR1	MTW1	MPL1	MBW2	MTR2	MTW2	MPL2	MBW3	MTR3	MTW3	MPL3
	W																
	R	MBW4	MTR4	MTW4	MPL4	MBW5	MTR5	MTW5	MPL5	MBW6	MTR6	MTW6	MPL6	MBW7	MTR7	MTW7	MPL7
	W																
0x43F0_0004 (MPR_2) 0x53F0_0004 (MPR_2)	R	MBW8	MTR8	MTW8	MPL8	MBW9	MTR9	MTW9	MPL9	MBW10	MTR10	MTW10	MPL10	MBW11	MTR11	MTW11	MPL11
	W																
	R	MBW12	MTR12	MTW12	MPL12	MBW13	MTR13	MTW13	MPL13	MBW14	MTR14	MTW14	MPL14	MBW15	MTR15	MTW15	MPL15
	W																
0x43F0_0020 (PACR_1) 0x53F0_0020 (PACR_1)	R	BW0	SP0	WP0	TP0	BW1	SP1	WP1	TP1	BW2	SP2	WP2	TP2	BW3	SP3	WP3	TP3
	W																
	R	BW4	SP4	WP4	TP4	BW5	SP5	WP5	TP5	BW6	SP6	WP6	TP6	BW7	SP7	WP7	TP7
	W																
0x43F0_0024 (PACR_2) 0x53F0_0024 (PACR_2)	R	BW8	SP8	WP8	TP8	BW9	SP9	WP9	TP9	BW10	SP10	WP10	TP10	BW11	SP11	WP11	TP11
	W																
	R	BW12	SP12	WP12	TP12	BW13	SP13	WP13	TP13	BW14	SP14	WP14	TP14	BW15	SP15	WP15	TP15
	W																
0x43F0_0028 (PACR_3) 0x53F0_0028 (PACR_3)	R	BW16	SP16	WP16	TP16	BW17	SP17	WP17	TP17	BW18	SP18	WP18	TP18	BW19	SP19	WP19	TP19
	W																
	R	BW20	SP20	WP20	TP20	BW21	SP21	WP21	TP21	BW22	SP22	WP22	TP22	BW23	SP23	WP23	TP23
	W																
0x43F0_002C (PACR_4) 0x53F0_002C (PACR_4)	R	BW24	SP24	WP24	TP24	BW25	SP25	WP25	TP25	BW26	SP26	WP26	TP26	BW27	SP27	WP27	TP27
	W																
	R	BW28	SP28	WP28	TP28	BW29	SP29	WP29	TP29	BW30	SP30	WP30	TP30	BW31	SP31	WP31	TP31
	W																
0x43F0_0040 (OPACR_1) 0x53F0_0040 (OPACR_1)	R	BW0	SP0	WP0	TP0	BW1	SP1	WP1	TP1	BW2	SP2	WP2	TP2	BW3	SP3	WP3	TP3
	W																
	R	BW4	SP4	WP4	TP4	BW5	SP5	WP5	TP5	BW6	SP6	WP6	TP6	BW7	SP7	WP7	TP7
	W																

Table 38-15. AIPS Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x43F0_0044 (OPACR_2) 0x53F0_0044 (OPACR_2)	R	BW8	SP8	WP8	TP8	BW9	SP9	WP9	TP9	BW10	SP10	WP10	TP10	BW11	SP11	WP11	TP11
	W																
	R	BW12	SP12	WP12	TP12	BW13	SP13	WP13	TP13	BW14	SP14	WP14	TP14	BW15	SP15	WP15	TP15
	W																
0x43F0_0048 (OPACR_3) 0x53F0_0048 (OPACR_3)	R	BW16	SP16	WP16	TP16	BW17	SP17	WP17	TP17	BW18	SP18	WP18	TP18	BW19	SP19	WP19	TP19
	W																
	R	BW20	SP20	WP20	TP20	BW21	SP21	WP21	TP21	BW22	SP22	WP22	TP22	BW23	SP23	WP23	TP23
	W																
0x43F0_004C (OPACR_4) 0x53F0_004C (OPACR_4)	R	BW24	SP24	WP24	TP24	BW25	SP25	WP25	TP25	BW26	SP26	WP26	TP26	BW27	SP27	WP27	TP27
	W																
	R	BW28	SP28	WP28	TP28	BW29	SP29	WP29	TP29	BW30	SP30	WP30	TP30	BW31	SP31	WP31	TP31
	W																
0x43F0_0050 (OPACR_5) 0x53F0_0050 (OPACR_5)	R	BW32	SP32	WP32	TP32	BW33	SP33	WP33	TP33	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																

38.3.3 Register Descriptions

This section contains the detailed register descriptions for the ATA registers.

38.3.3.1 Master Privilege Registers (MPR_1 and MPR_2)

Each MPR specifies eight 4-bit fields defining the access privilege level associated with a bus master in the platform, as well as specifying whether write accesses from this master are bufferable. The registers provide one field per bus master, where field 15 corresponds to master 15, field 14 to master 14, and so on, field 0 to master 0 (typically the processor core).

0x43F0_0000 (MPR_1)
0x53F0_0000 (MPR_1)

Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	MBW 0	MTR0	MTW 0	MPL0	MBW 1	MTR1	MTW 1	MPL1	MBW 2	MTR2	MTW 2	MPL2	MBW 3	MTR3	MTW 3	MPL3
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MBW 4	MTR4	MTW 4	MPL4	MBW 5	MTR5	MTW 5	MPL5	MBW 6	MTR6	MTW 6	MPL6	MBW 7	MTR7	MTW 7	MPL7
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Figure 38-5. Master Privilege Register 1

0x43F0_0004 (MPR_2)
0x53F0_0004 (MPR_2)

Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	MBW 8	MTR8	MTW 8	MPL8	MBW 9	MTR9	MTW 9	MPL9	MBW 10	MTR1 0	MTW 10	MPL1 0	MBW 11	MTR1 1	MTW 11	MPL1 1
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MBW 12	MTR1 2	MTW 12	MPL1 2	MBW 13	MTR1 3	MTW 13	MPL1 3	MBW 14	MTR1 4	MTW 14	MPL1 4	MBW 15	MTR1 5	MTW 15	MPL1 5
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Figure 38-6. Master Privilege Register 2

NOTE

Table 38-16 describes each nibble in the MPROT registers.

Table 38-16. Master Protection Field Descriptions

Field	Description
31, 27, 23, 20, 19, 15, 11, 7, 3 MBWn[1]	Master Buffer Writes. This bit determines whether the AIPS is enabled to buffer writes from this master. 0 Write accesses from this master are not bufferable 1 Write accesses from this master are allowed to be buffered
30, 26, 22, 18, 14, 10, 6, 2 MTRn	Master Trusted for Reads. This bit determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.

Table 38-16. Master Protection Field Descriptions (continued)

Field	Description
29, 25, 21, 17, 13, 9, 5, 1 MTWn	Master Trusted for Writes. This bit determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
28, 24, 20, 16, 12, 8, 4, 0 MPLn	Master Privilege Level. This bit determines how the privilege level of the master is determined. 0 Accesses from this master are forced to user-mode (ips_supervisor_access is forced to zero) regardless of the hprot[1] access attribute. 1 Accesses from this master are not forced to user-mode. The hprot[1] access attribute is used directly to determine ips_supervisor_access.

[1] n = the field bit name number listed in the register figure.

NOTE

At reset, the default value loaded into the MPROT15–0 fields is dependent on the state of a 64-bit platform configuration input vector (aips_rstcfg[63:0]). This input vector allows any of the bus masters to be identified as a trusted bus master. The processor core is always treated as a trusted bus master out of reset. Accordingly, the MPROTx field for the processor should be forced to 0111 at reset. Typically, the processor is bus master 0, thus MPROT0 should be driven to 0111 at reset.

If a given bit in this input vector is set, the reset state of the corresponding MPR bit is also set. In addition, the state of the input configuration vector at reset time is saved by the AIPS.

Accesses to registers or register fields which correspond to master or peripheral locations which are not implemented will return zeros on reads, and will be ignored on writes.

aips_rstcfg[63:32] maps to MPROT[8:15] with aips_rstcfg[63:60] corresponding to MPROT[8], and aips_rstcfg[31:0] maps to MPROT[0:7] with aips_rstcfg[31:28] corresponding to MPROT[0].

38.3.3.2 Peripheral Access Control Registers (PACR_1, PACR_2, PACR_3, and PACR_4)

Each of the on-platform peripherals have a Peripheral Access Control Register that defines the access levels supported by the given module.

0x43F0_0020 (PACR_1)
0x53F0_0020 (PACR_1)

Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BW0 (note)	SP0 (note)	WP0 (note)	TP0 (note)	BW1	SP1	WP1	TP1	BW2	SP2	WP2	TP2	BW3	SP3	WP3	TP3
Reset	0	1	0	1	0	1	0	0	0	1	0	0	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BW4	SP4	WP4	TP4	BW5	SP5	WP5	TP5	BW6	SP6	WP6	TP6	BW7	SP7	WP7	TP7
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

The reset state of [TP0] is 1, not 0. Also [BW0] and [WP0] are hardwired to “0.” SP0 is hardwired to “1.”

Figure 38-7. Peripheral Access Control Register 1

0x43F0_0024 (PACR_2)
0x53F0_0024 (PACR_2)

Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BW8	SP8	WP8	TP8	BW9	SP9	WP9	TP9	BW10	SP10	WP10	TP10	BW11	SP11	WP11	TP11
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BW12	SP12	WP12	TP12	BW13	SP13	WP13	TP13	BW14	SP14	WP14	TP14	BW15	SP15	WP15	TP15
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

Figure 38-8. Peripheral Access Control Register 2

0x43F0_0028 (PACR_3)
0x53F0_0028 (PACR_3)

Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BW16	SP16	WP16	TP16	BW17	SP17	WP17	TP17	BW18	SP18	WP18	TP18	BW19	SP19	WP19	TP19
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BW20	SP20	WP20	TP20	BW21	SP21	WP21	TP21	BW22	SP22	WP22	TP22	BW23	SP23	WP23	TP23
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

Figure 38-9. Peripheral Access Control Register 3

0x43F0_002C (PACR_4)
0x53F0_002C (PACR_4)

Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BW24	SP24	WP24	TP24	BW25	SP25	WP25	TP25	BW26	SP26	WP26	TP26	BW27	SP27	WP27	TP27
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BW28	SP28	WP28	TP28	BW29	SP29	WP29	TP29	BW30	SP30	WP30	TP30	BW31	SP31	WP31	TP31
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

Figure 38-10. Peripheral Access Control Register 4

NOTE

Table 38-17 describes each nibble in the PACR registers.

Table 38-17. Peripheral Access Control Register Field Descriptions

Field	Description
31, 27, 23, 19, 15, 11, 7, 3 BWn[1]	Buffer Writes. This bit determines whether write accesses to this peripheral are allowed to be buffered.[2] 0 Write accesses to this peripheral are not bufferable by the AIPS. 1 Write accesses to this peripheral are allowed to be buffered by the AIPS.
30, 26, 22, 18, 14, 10, 6, 2 SPn	Supervisor Protect. This bit determines whether the peripheral requires supervisor privilege level for access. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus.
29, 25, 21, 17, 13, 9, 5, 1 WPn	Write Protect. This bit determines whether the peripheral allows write accesses. 0 This peripheral allows write accesses. 1 This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus.
28, 24, 20, 16, 12, 8, 4, 0 TPn	Trusted Protect. This bit determines whether the peripheral allows accesses from an untrusted master. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus.

1 n = The field bit name number listed in the register figure.

2 Buffered writes are not available for PACR0. In PACR0, this bit is hardwired to ‘0’.

Presence or absence of a particular PACR is based on whether the associated peripheral is present in the platform, as denoted by the synthesis configuration parameters defined in Section 38.4.1, “AIPS Scalability.” When absent, the corresponding PACR is not implemented and will read as 0s. Writes will be ignored.

38.3.3.3 Off-Platform Peripheral Access Control Registers (OPACR_1, OPACR_2, OPACR_3, OPACR_4, and OPACR_5)

Each of the off-platform peripherals have an Off-Platform Peripheral Access Control Register (OPACR) which defines the access levels supported by the given module. Each OPACR has a format identical to the PACR described in Section 38.3.3.2, “Peripheral Access Control Registers (PACR_1, PACR_2, PACR_3, and PACR_4).”

Each OPACR corresponds to the equivalent ips_module_en; OPACR0 corresponds to ips_module_en[0], and so on, with OPACR32 corresponding to ips_module_en_glbl[0], and OPACR33 corresponding to ips_module_en_glbl[1].

Presence or absence of a particular OPACR is based on the synthesis configuration parameters defined in Section 38.4.1, “AIPS Scalability.” When absent, the corresponding OPACR is not implemented and will read as 0. Writes will be ignored.

0x43F0_0040 (OPACR_1)
0x53F0_0040 (OPACR_1)

Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BW0 (note)	SP0 (note)	WP0 (note)	TP0 (note)	BW1	SP1	WP1	TP1	BW2	SP2	WP2	TP2	BW3	SP3	WP3	TP3
Reset	0	1	0	1	0	1	0	0	0	1	0	0	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BW4	SP4	WP4	TP4	BW5	SP5	WP5	TP5	BW6	SP6	WP6	TP6	BW7	SP7	WP7	TP7
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

Note: The reset state of [TP0] is 1, not 0. Also [BW0] and [WP0] are hardwired to “0.” SP0 is hardwired to “1.”

Figure 38-11. Off-Platform Peripheral Access Control Register 1

0x43F0_0044 (OPACR_2)
0x53F0_0044 (OPACR_2)

Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BW8	SP8	WP8	TP8	BW9	SP9	WP9	TP9	BW10	SP10	WP10	TP10	BW11	SP11	WP11	TP11
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BW12	SP12	WP12	TP12	BW13	SP13	WP13	TP13	BW14	SP14	WP14	TP14	BW15	SP15	WP15	TP15
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

Figure 38-12. Off-Platform Peripheral Access Control Register 2

0x43F0_0048 (OPACR_3)
0x53F0_0048 (OPACR_3)

Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BW16	SP16	WP16	TP16	BW17	SP17	WP17	TP17	BW18	SP18	WP18	TP18	BW19	SP19	WP19	TP19
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BW20	SP20	WP20	TP20	BW21	SP21	WP21	TP21	BW22	SP22	WP22	TP22	BW23	SP23	WP23	TP23
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

Figure 38-13. Off-Platform Peripheral Access Control Register 3

0x43F0_004C (OPACR_4)
0x53F0_004C (OPACR_4)

Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BW24	SP24	WP24	TP24	BW25	SP25	WP25	TP25	BW26	SP26	WP26	TP26	BW27	SP27	WP27	TP27
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BW28	SP28	WP28	TP28	BW29	SP29	WP29	TP29	BW30	SP30	WP30	TP30	BW31	SP31	WP31	TP31
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

Figure 38-14. Off-Platform Peripheral Access Control Register 4

0x43F0_0050 (OPACR_5)
0x53F0_0050 (OPACR_5)

Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BW32	SP32	WP32	TP32	BW33	SP33	WP33	TP33	0	0	0	0	0	0	0	0
W																
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

Figure 38-15. Off-Platform Peripheral Access Control Register 5

NOTE

Table 38-18 describes each nibble in the OPACR registers.

Table 38-18. Off-Platform Peripheral Access Control Register Field Descriptions

Field	Description
31, 27, 23, 19, 15, 11, 7, 3 BWn[1]	Buffer Writes. This bit determines whether write accesses to this peripheral are allowed to be buffered.[2] 0 Write accesses to this peripheral are not buffer-able by the AIPS. 1 Write accesses to this peripheral are allowed to be buffered by the AIPS.
30, 26, 22, 18, 14, 10, 6, 2 SPn	Supervisor Protect. This bit determines whether the peripheral requires supervisor privilege level for access. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus.
29, 25, 21, 17, 13, 9, 5, 1 WPn	Write Protect. This bit determines whether the peripheral allows write accesses 0 This peripheral allows write accesses. 1 This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus.
28, 24, 20, 16, 12, 8, 4, 0 TPn	Trusted Protect. This bit determines whether the peripheral allows accesses from an untrusted master. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus.

1 n = the field bit name number listed in the register figure.

2 Buffered writes are not available for OPACR0. In OPACR0, this bit is hardwired to ‘0’.

NOTE

Bits 23:0 are reserved for OPACR_5.

38.4 Functional Description

The AIPS serves as an interface between an AHB 2.v6 system bus and the IP SkyBlue peripheral bus. It functions as a protocol translator. Support is provided for generating a pair of 32-bit IP SkyBlue accesses when targeted by a 64-bit system bus access, or a misaligned access which crosses a 32-bit boundary. No other bus-sizing access support is provided.

Accesses which fall within the address space of the AIPS are decoded to provide individual module selects for peripheral devices on the IP SkyBlue interface, as described in Section 38.2.2.2.4, "ips_module_en[31:0], ips_module_en_glbl[1:0], ips_module_en_nonglbl" and Section 38.2.2.4.4, "onpf_ips_module_en[31:1]."

38.4.1 AIPS Scalability

The AIPS is configurable to support either eight or sixteen masters, two different memory maps, from one to thirty-two on-platform peripherals, and either sixteen or thirty-two fixed-size off-platform peripherals depending on the requirements of the system. The minimum read access time can be parameterized to be either two or three cycles. In addition, the AIPS can be configured for both 32-bit and 64-bit system data buses.

The scalability of the AIPS relies on the use of synthesis configuration parameters. The parameters in the AIPS and their default values in the order of declaration can be found in Table 38-19.

Table 38-19. AIPS Parameters and Defaults

Parameter Name	Supported Values	Default Value
DATA_PORT_WIDTH	32, 64	64
AIPS_MEMMAP	1 (Map #1), 2(Map #2)	1
AIPS_DLY_CYCLE	0, 1	1
MSTR_8_15_PRESENT	0, 1	1
OPACR16_31_PRESENT	0, 1	1
PACR1_PRESENT	0, 1	1
PACR2_PRESENT	0, 1	1
PACR3_PRESENT	0, 1	1
PACR4_PRESENT	0, 1	1
PACR5_PRESENT	0, 1	1
PACR6_PRESENT	0, 1	1
PACR7_PRESENT	0, 1	1
PACR8_PRESENT	0, 1	1
PACR9_PRESENT	0, 1	1
PACR10_PRESENT	0, 1	1
PACR11_PRESENT	0, 1	1

Table 38-19. AIPS Parameters and Defaults (continued)

Parameter Name	Supported Values	Default Value
PACR12_PRESENT	0, 1	1
PACR13_PRESENT	0, 1	1
PACR14_PRESENT	0, 1	1
PACR15_PRESENT	0, 1	1
PACR16_PRESENT	0, 1	1
PACR17_PRESENT	0, 1	1
PACR18_PRESENT	0, 1	1
PACR19_PRESENT	0, 1	1
PACR20_PRESENT	0, 1	1
PACR21_PRESENT	0, 1	1
PACR22_PRESENT	0, 1	1
PACR23_PRESENT	0, 1	1
PACR24_PRESENT	0, 1	1
PACR25_PRESENT	0, 1	1
PACR26_PRESENT	0, 1	1
PACR27_PRESENT	0, 1	1
PACR28_PRESENT	0, 1	1
PACR29_PRESENT	0, 1	1
PACR30_PRESENT	0, 1	1
PACR31_PRESENT	0, 1	1
PACR1_SIZE	8,16,32	32
PACR2_SIZE	8,16,32	32
PACR3_SIZE	8,16,32	32
PACR4_SIZE	8,16,32	32
PACR5_SIZE	8,16,32	32
PACR6_SIZE	8,16,32	32
PACR7_SIZE	8,16,32	32
PACR8_SIZE	8,16,32	32
PACR9_SIZE	8,16,32	32
PACR10_SIZE	8,16,32	32
PACR11_SIZE	8,16,32	32
PACR12_SIZE	8,16,32	32
PACR13_SIZE	8,16,32	32

Table 38-19. AIPS Parameters and Defaults (continued)

Parameter Name	Supported Values	Default Value
PACR14_SIZE	8,16,32	32
PACR15_SIZE	8,16,32	32
PACR16_SIZE	8,16,32	32
PACR17_SIZE	8,16,32	32
PACR18_SIZE	8,16,32	32
PACR19_SIZE	8,16,32	32
PACR20_SIZE	8,16,32	32
PACR21_SIZE	8,16,32	32
PACR22_SIZE	8,16,32	32
PACR23_SIZE	8,16,32	32
PACR24_SIZE	8,16,32	32
PACR25_SIZE	8,16,32	32
PACR26_SIZE	8,16,32	32
PACR27_SIZE	8,16,32	32
PACR28_SIZE	8,16,32	32
PACR29_SIZE	8,16,32	32
PACR30_SIZE	8,16,32	32
PACR31_SIZE	8,16,32	32

To scale the AIPS, the integrator must specify via parameters which resources need support in the design.

38.4.1.1 Master Presence

The parameter MSTR_8_15_PRESENT should be set to ‘1’ if hmaster ID’s 8-15 are to be supported and should be set to ‘0’ otherwise. When hmaster ID’s 8-15 are supported, hmaster[3:0] are used to identify the current master for an access to an AIPS and select an MPROT control field. When unsupported, hmaster[3] is ignored, and only hmaster[2:0] are used to select an MPROT control field.

38.4.1.2 Peripheral Presence

The parameter PACRx_PRESENT should be set to ‘1’ if the specific on-platform peripheral is going to be present in the final design and should be set to ‘0’ otherwise. If the peripheral is present, the corresponding PACRx_SIZE parameter will be used as well.

NOTE

PACR0 is present in all designs using the AIPS since it corresponds to AIPS registers.

The parameter OPACR16_31_PRESENT should be set to ‘1’ if more than sixteen fixed-size off-platform peripherals are going to be present in the final design and should be set to ‘0’ otherwise.

When a particular peripheral or set of peripherals are not present in a design, those peripheral memory spaces which are not present will not be accessible by software and will be terminated with an AHB ERROR response by the AIPS. In addition, the corresponding module enable outputs, on-platform read data buses, and on-platform transfer wait and error signals will not be present in the design.

38.4.1.3 AHB Data Port Width

The AIPS is designed to support either a 32-bit or 64-bit AHB data width. The parameter DATA_PORT_WIDTH is used to determine the actual width. This affects the width of the AHB data bus (hwdata, aips_hrdata) as well as the byte strobe bus (hbstrb). For 32-bit designs, 64-bit bus transfers are not applicable. Refer to Section 38.4.4, “Data Byte Lane and Byte Strobe Mapping.”

38.4.1.4 Memory Map

The memory map for the AIPS can be configured at synthesis time to correspond to Map #1 as shown in Figure 38-2 (and further defined in Table 38-7 and Table 38-9) or to Map #2 as shown in Figure 38-3 (and further defined in Table 38-8 and Table 38-10) by means of the AIPS_MEMMAP configuration parameter.

If any of the on- or off-platform peripheral ports are not included in the final instantiation of the AIPS, the address space allocated to that peripheral will no longer be available. If a peripheral is not present in the final design and a master tries to make an access to that address space through the AIPS, the AIPS will terminate the access with an ERROR response.

38.4.1.5 Registers

The parameters outlined in Table 38-19 will be used to configure which registers as well as which bits in the registers end up in the final instantiation of the AIPS. Since the registers are peripheral and master oriented, the presence or absence of a group of masters, a group of off-platform peripherals, or individual on-platform peripherals will determine whether or not a particular register is present.

Any attempt to access registers or register bit fields that are not present in the final instantiation will result in a read returning zeroed data and writes being ignored.

38.4.1.6 Read Cycle and Buffered Write Delay Control

The AIPS_DLY_CYCLE parameter may be used to decide in higher-frequency designs that read cycle initiation to a peripheral and write buffering should be delayed for one cycle to allow for additional control decoding within the AIPS.

38.4.2 Access Protections

The AIPS provides programmable access protections for both masters and peripherals. It allows the privilege level of a master to be overridden, forcing it to user-mode privilege, and allows masters to be designated as trusted or untrusted. Peripherals may require supervisor privilege level for access, may restrict access to a trusted master only, and may be write-protected.

This functionality is described in Section 38.3.3.1, “Master Privilege Registers (MPR_1 and MPR_2),” Section 38.3.3.2, “Peripheral Access Control Registers (PACR_1, PACR_2, PACR_3, and PACR_4),” and Section 38.3.3.3, “Off-Platform Peripheral Access Control Registers (OPACR_1, OPACR_2, OPACR_3, OPACR_4, and OPACR_5).”

38.4.3 Peripheral Write Buffering

The AIPS provides programmable write buffering capability to allow certain write accesses to be buffered in the AIPS for later completion, while terminating the AHB access early. This provides improved performance in systems where frequent writes to a slow peripheral are performed. Write buffering must only be enabled for masters and peripherals for which an error termination from the IPS bus will either not occur, or is safe to ignore, since the AIPS controller ignores the ips_xfr_err signal on termination of buffered writes. In addition, write buffering must not be enabled for peripherals which have been marked write protected.

When write buffering is enabled, all accesses through the AIPS will still occur in-order; no bypassing of buffered writes is supported.

Write buffering is controllable on a per-master and per-peripheral basis. Enabling this functionality is described in Section 38.3.3.1, “Master Privilege Registers (MPR_1 and MPR_2),” Section 38.3.3.2, “Peripheral Access Control Registers (PACR_1, PACR_2, PACR_3, and PACR_4),” and Section 38.3.3.3, “Off-Platform Peripheral Access Control Registers (OPACR_1, OPACR_2, OPACR_3, OPACR_4, and OPACR_5).”

Write buffering is not available for the AIPS control registers (PACR0).

38.4.4 Data Byte Lane and Byte Strobe Mapping

For maximum flexibility, the AIPS provides input control signals aips_byte_config[1:0], which allows the byte ordering of the IPS to be configured to support both byte-invariant (BE8) and word invariant (BE32) Big-Endian operation, as well as Little Endian operation.

Data byte lanes on the master bus are wired according to the convention used by the system bus masters for assigning byte lanes to byte addresses. For the AHB 2.v6, byte address 0 corresponds to h[r,w]data[7:0]. In this convention (often associated with a Little-Endian ordering, even though AHB 2.v6 is Endian-neutral), the aips_byte_config[1:0] signals are normally negated for byte-invariant Big-Endian mode, assuming that the IPS peripheral wiring to the IPS bus is Big-Endian, but may be asserted to support alternate Little-Endian systems where the peripheral connections to the IPS are Little-Endian. For Little Endian systems with Little-Endian IPS peripheral wiring connections, the aips_byte_config[1:0] signals are normally set to 01. For word-invariant Big-Endian (BE32) mode systems with Big-Endian peripheral wiring, the aips_byte_config[1:0] signals are normally set to 11. For byte-invariant Big-Endian (BE8) mode systems with Big-Endian peripheral wiring, the aips_byte_config[1:0] signals are normally set to 00. It is expected that the signals are in a valid state prior to an access to the AIPS, and are not expected to change on a cycle-by-cycle basis.

NOTE

If an implementation requires a different byte ordering for instructions vs. data, this may be accommodated by proper loading of instruction memory on the IPS such that a change in the aips_byte_config[1:0] signals is not required on a cycle-by-cycle basis. This may require a byte reversal or other rearrangement of text segments when loading into program memory to accommodate operation of the AIPS with a given setting of the aips_byte_config[1:0] signals.

38.4.4.1 32-Bit Peripherals

Data byte lanes of the AHB 2.v6 read and write data buses are mapped to data byte lanes of the IPS based on the access ips_addr[2] value, and on the aips_byte_config[1:0] control signals. Byte enables on the IPS Bus are asserted based on the value of the AHB 2.v6 byte strobes, the access ips_addr[2] value, and on the aips_byte_config[1:0] control signals. Figure 38-16 through Figure 38-19 show the mapping of data byte lanes and byte strobes based on the state of aips_byte_config[1:0].

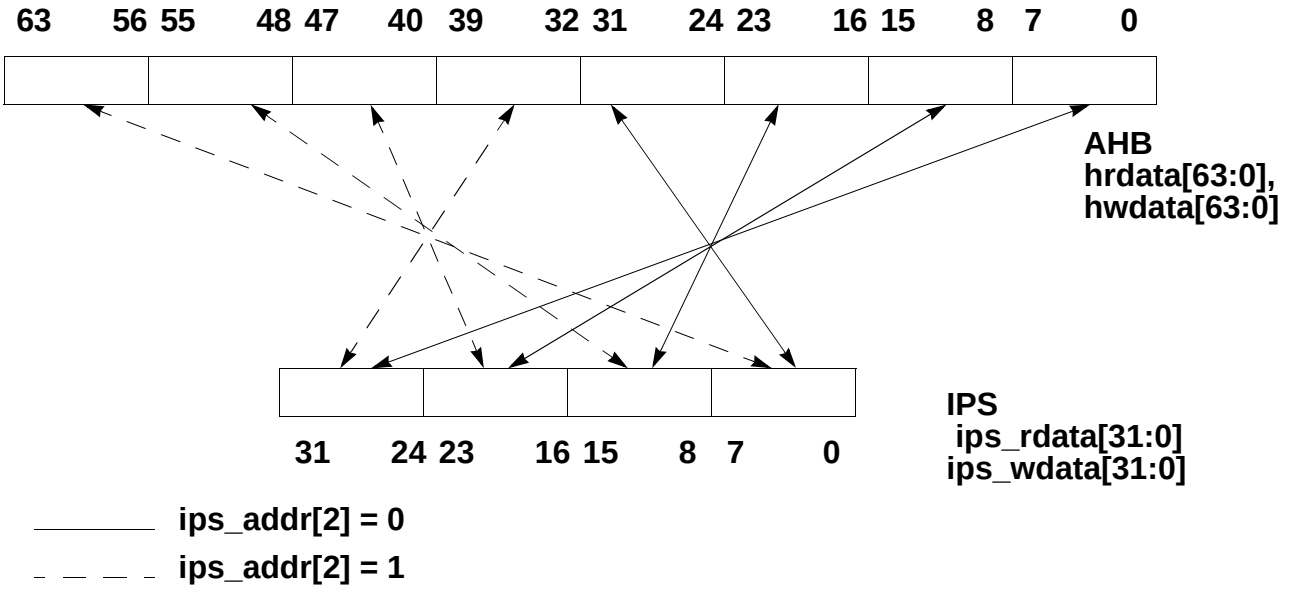


Figure 38-16. Data Bus Byte Lane Mapping for 32-Bit Peripherals, 64-Bit System, aips_byte_config[1:0] = 00

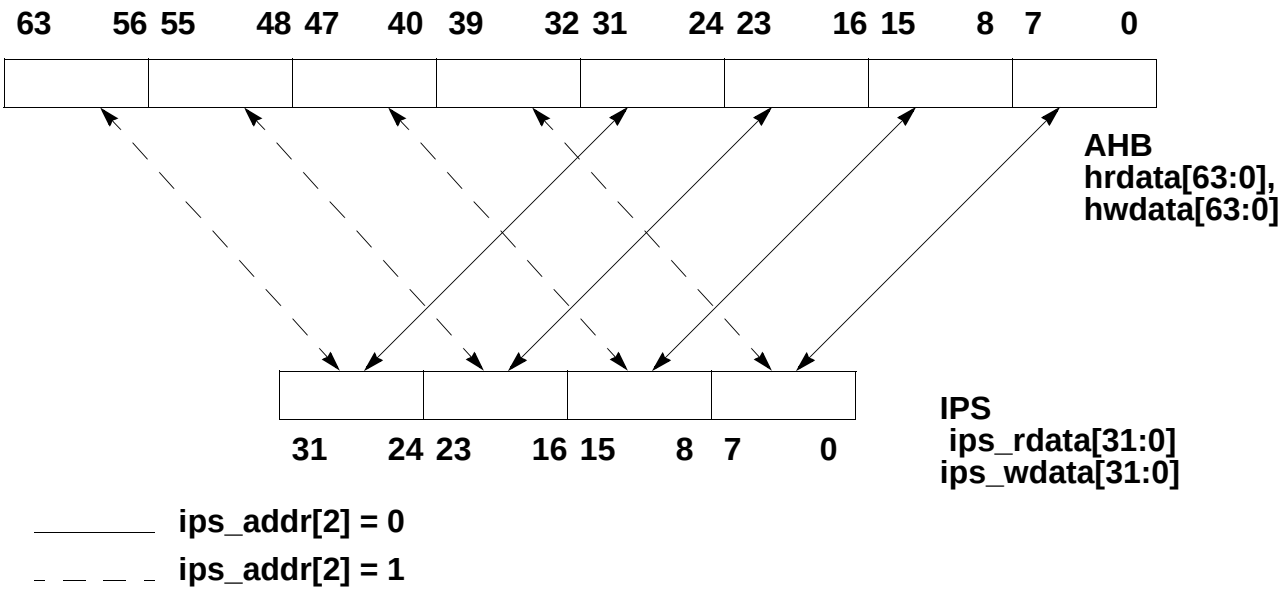


Figure 38-17. Data Bus Byte Lane Mapping for 32-Bit Peripherals, 64-Bit System, aips_byte_config[1:0] = 01, 11

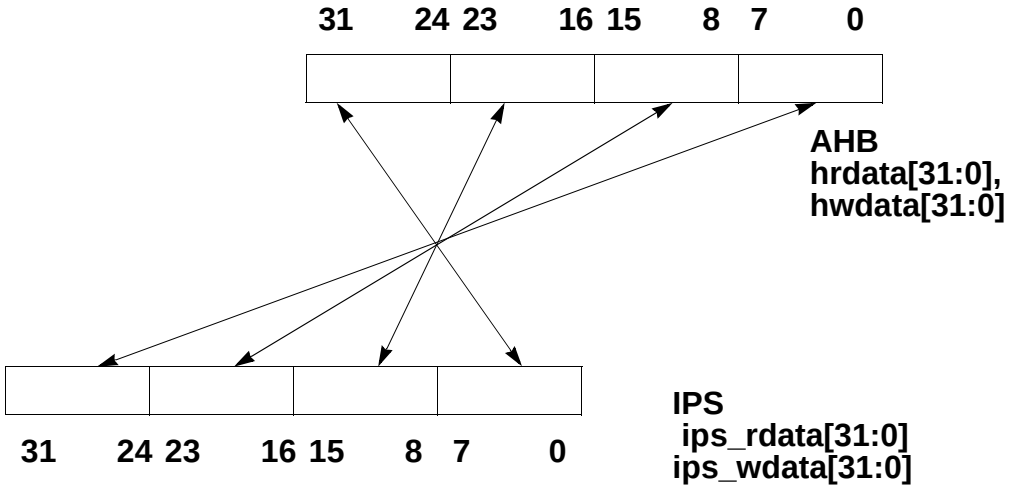


Figure 38-18. Data Bus Byte Lane Mapping for 32-Bit Peripherals, 32-Bit System, aips_byte_config[1:0] = 00

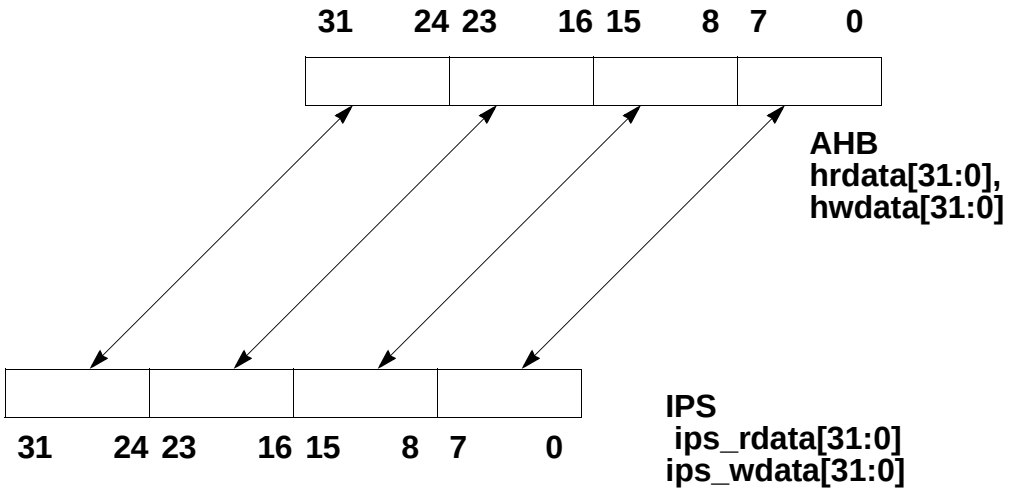


Figure 38-19. Data Bus Byte Lane Mapping for 32-Bit Peripherals, 32-Bit System, aips_byte_config[1:0] = 01, 11

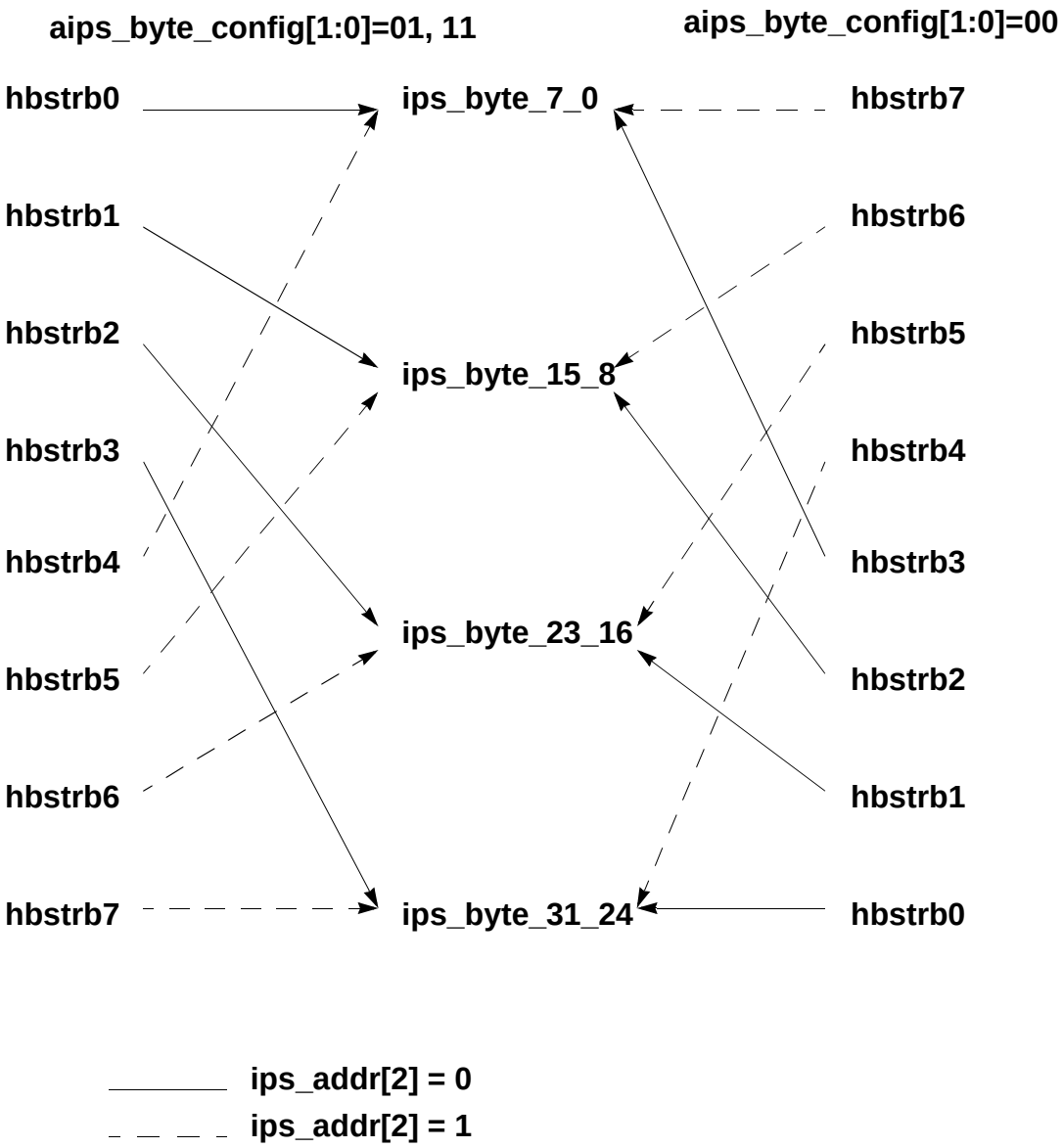


Figure 38-20. Mapping of 32-Bit Byte Strobes, 64-Bit System

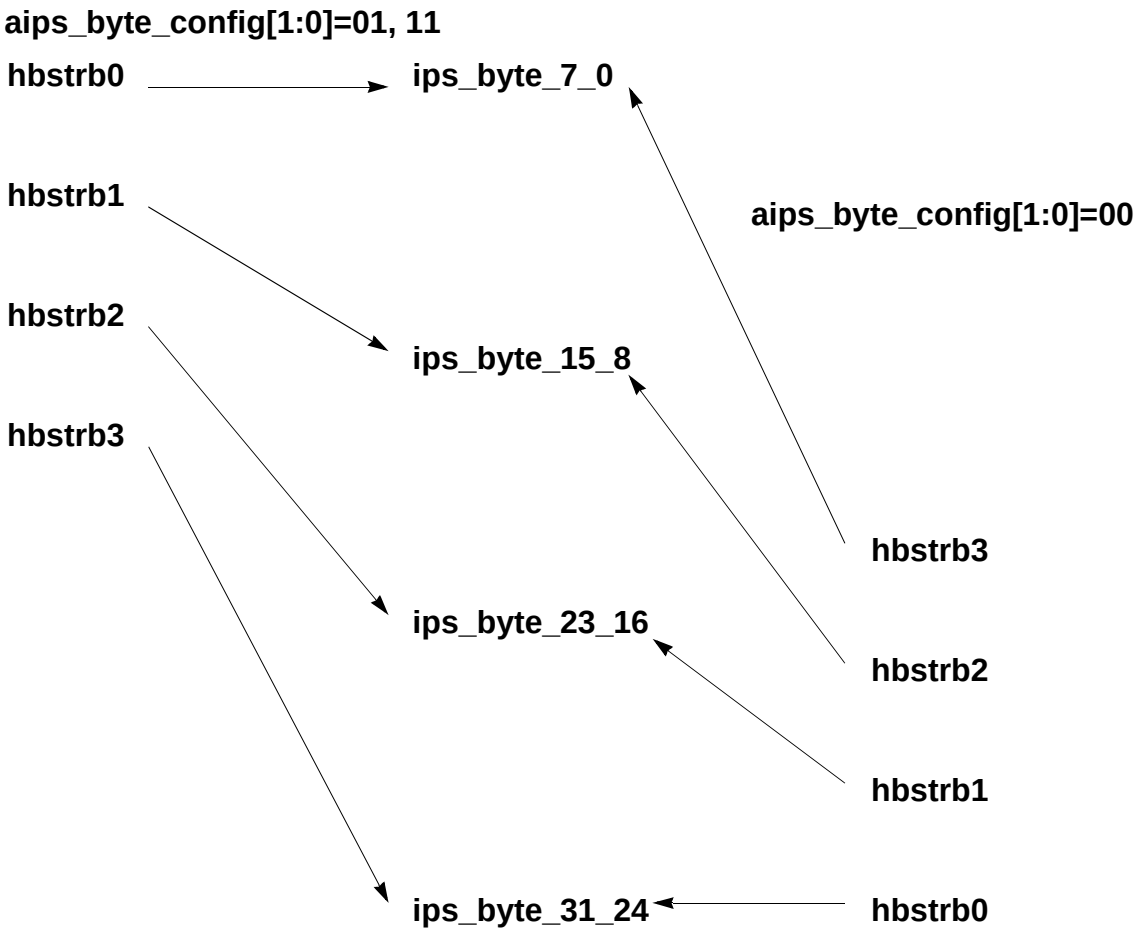


Figure 38-21. Mapping of 32-Bit Byte Strobes, 32-Bit System

38.4.4.2 16-Bit Peripherals

Data byte lanes of the AHB 2.v6 read and write data buses are mapped to data byte lanes of the IPS for 16-bit peripherals based on the access ips_addr[2:1] values, and on the aips_byte_config[1:0] control signals. Byte enables on the IPS Bus are asserted based on the value of the AHB 2.v6 byte strobes, the access ips_addr[2:1] values, and on the aips_byte_config[1:0] control signals. The ips_byte_15_8_16bit and ips_byte_7_0_16bit byte strobes are provided for 16-bit peripheral use.

For maximum flexibility, the AIPS provides control signals aips_byte_config[1:0], which allows the byte ordering of the IPS to be configured to support both byte-invariant (BE8) and word invariant (BE32) Big-Endian operation, as well as Little Endian operation. Figure 38-22 through Figure 38-33 show the mapping of data byte lanes and byte strobes for 16-bit peripherals based on the state of aips_byte_config[1:0].

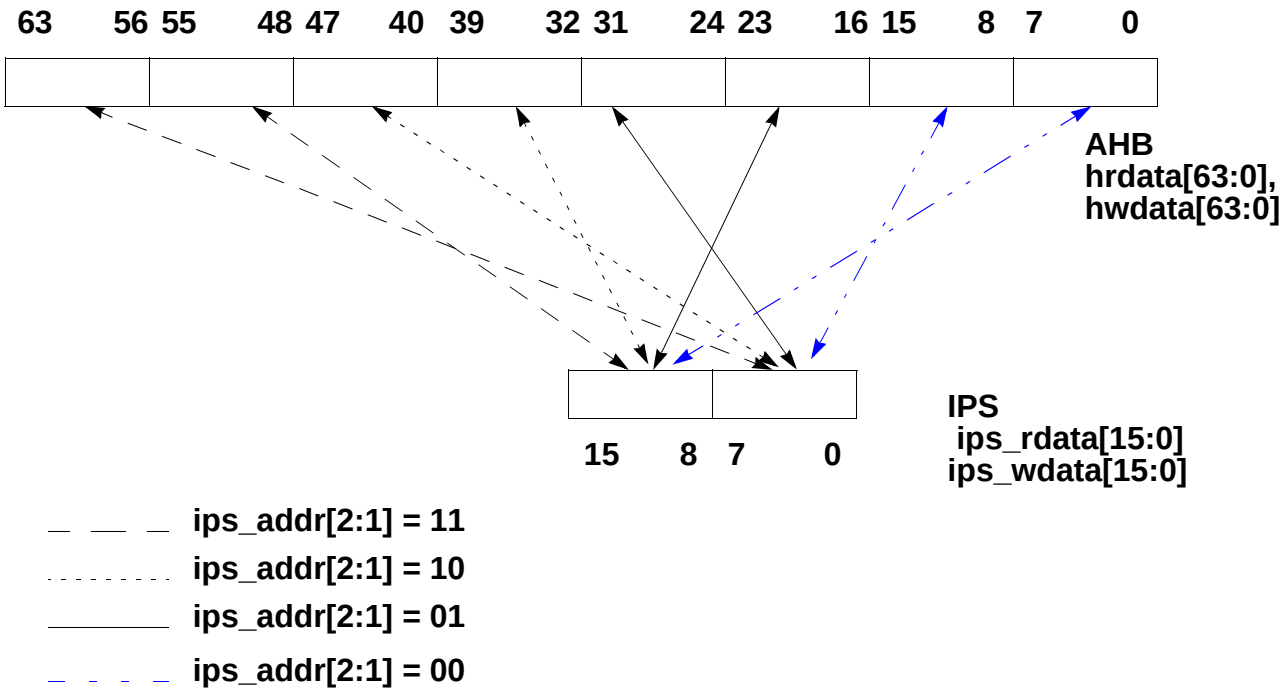


Figure 38-22. Data Bus Byte Lane Mapping for 16-Bit Peripherals, 64-Bit System, aips_byte_config[1:0] = 00

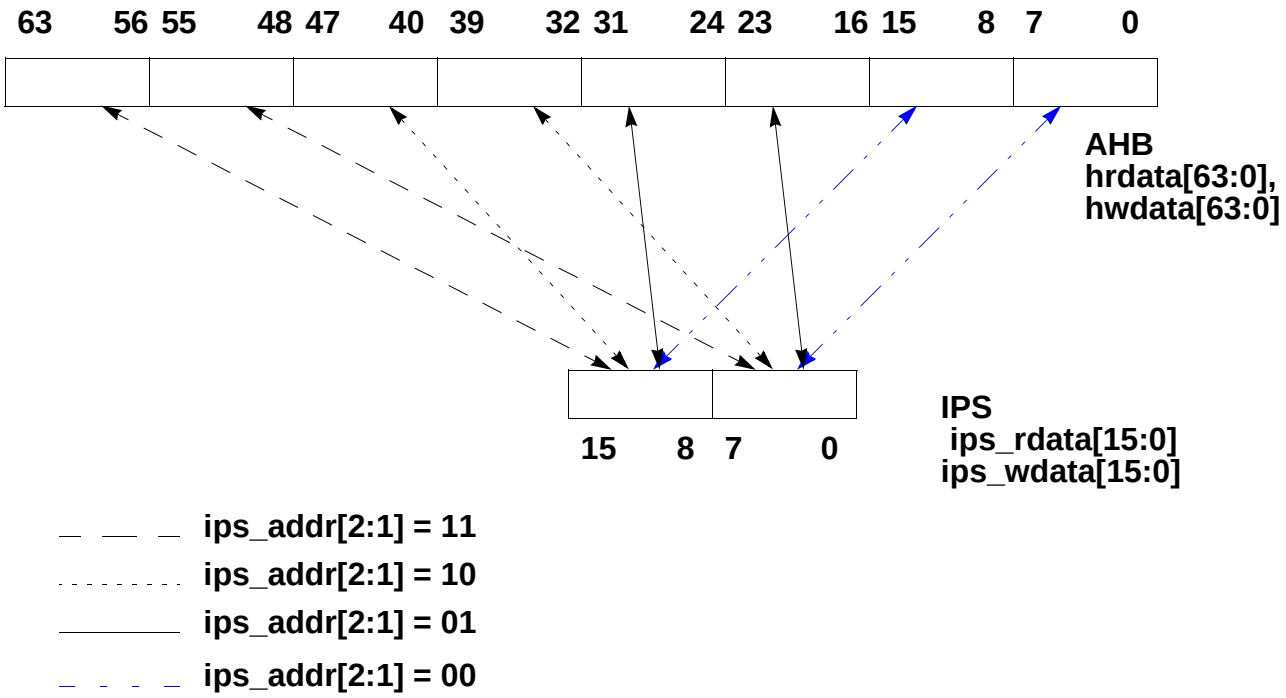


Figure 38-23. Data Bus Byte Lane Mapping for 16-Bit Peripherals, 64-Bit System, aips_byte_config[1:0] = 01

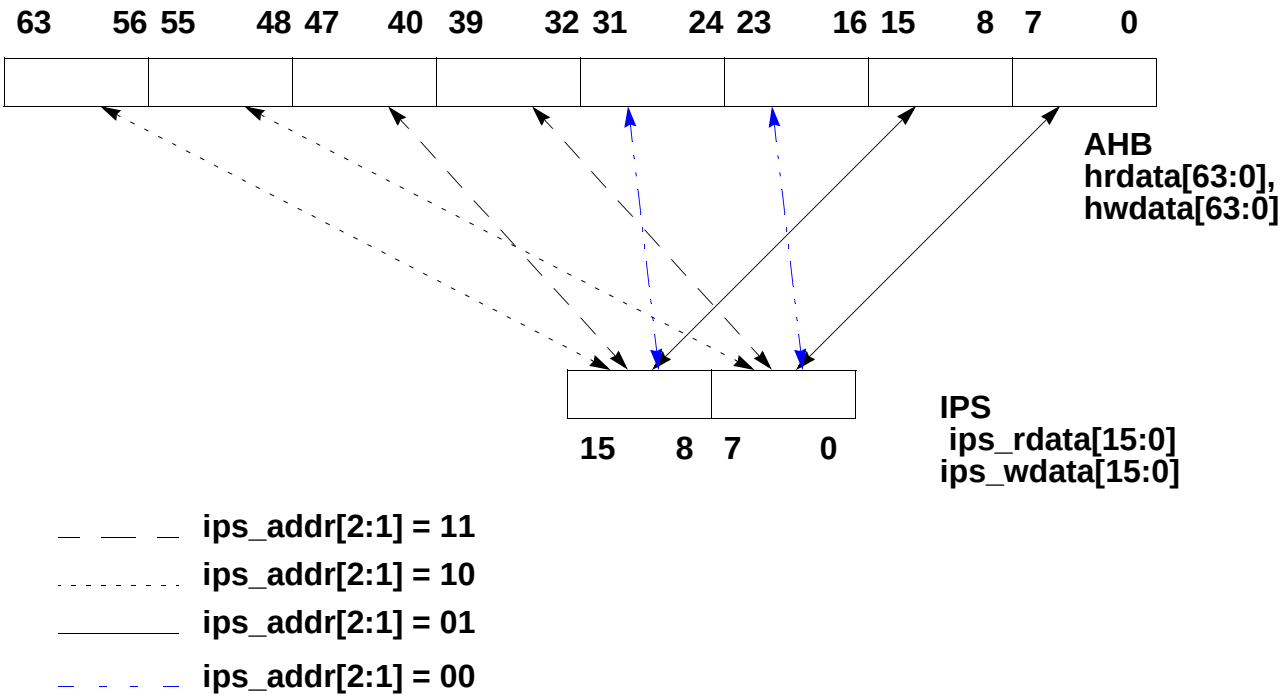


Figure 38-24. Data Bus Byte Lane Mapping for 16-Bit Peripherals, 64-Bit System, aips_byte_config[1:0] = 11

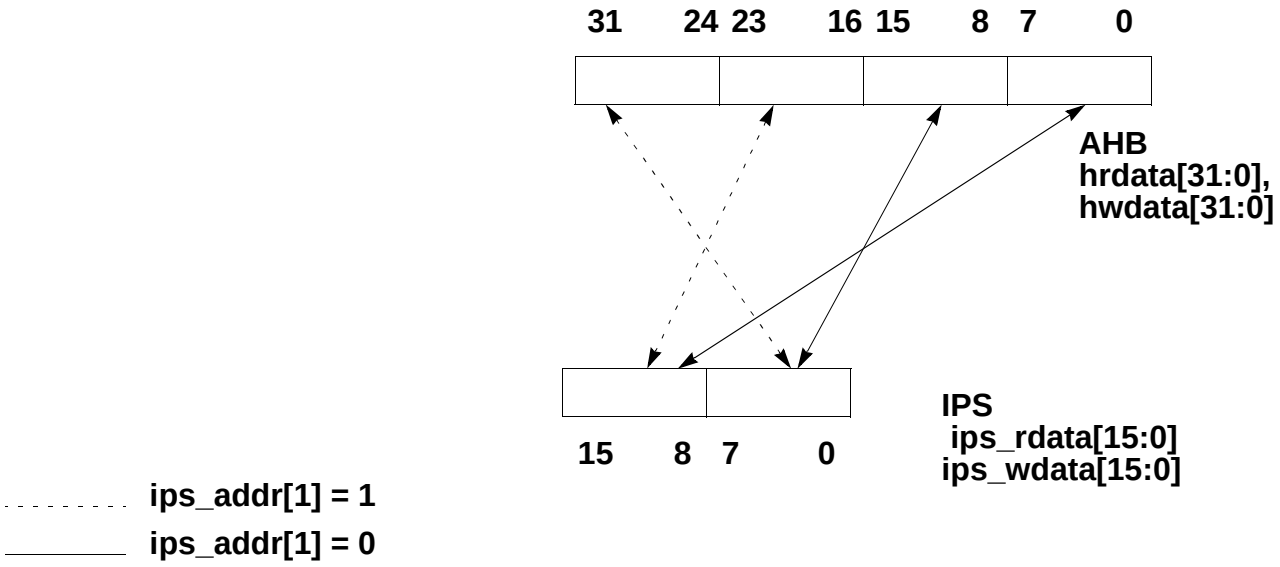


Figure 38-25. Data Bus Byte Lane Mapping for 16-Bit Peripherals, 32-Bit System, aips_byte_config[1:0] = 00

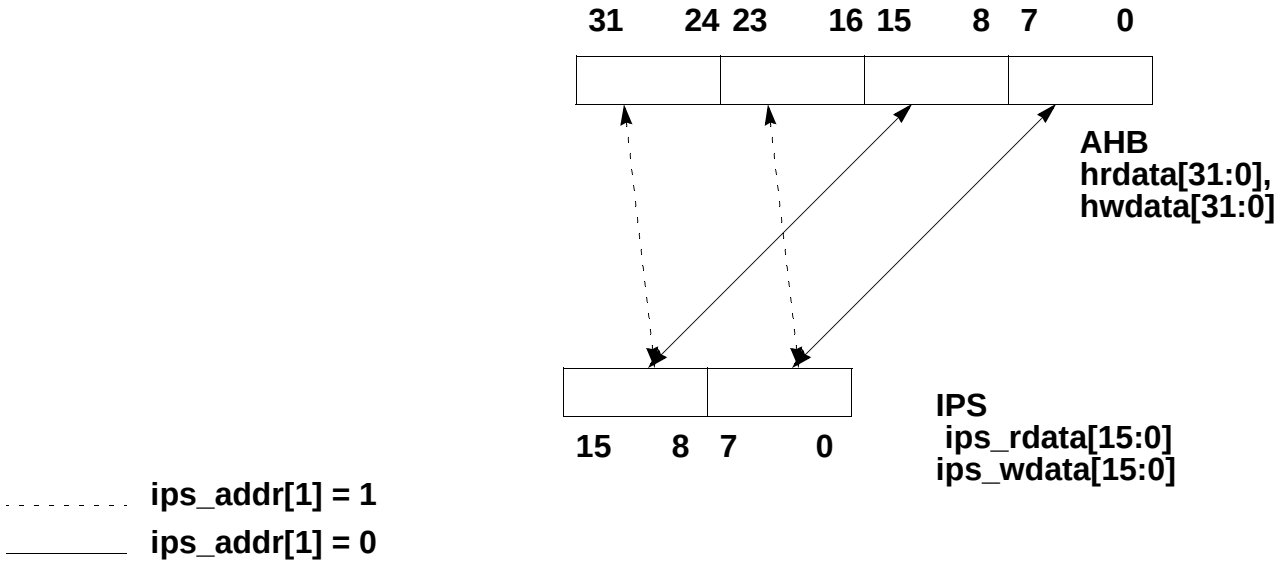


Figure 38-26. Data Bus Byte Lane Mapping for 16-Bit Peripherals, 32-Bit System, aips_byte_config[1:0] = 01

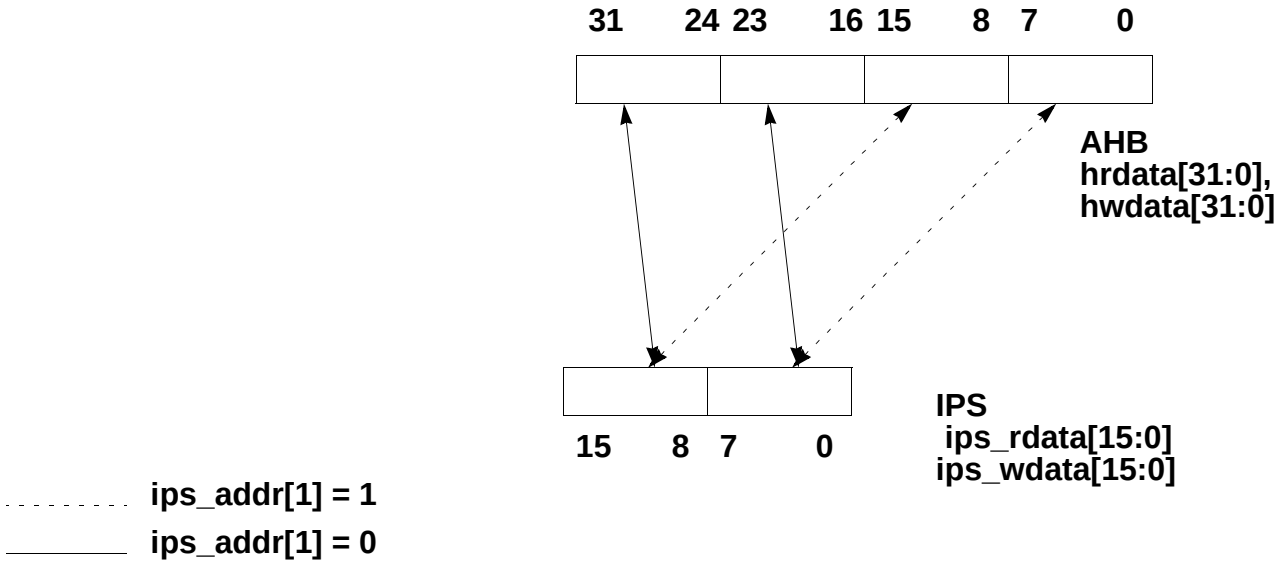


Figure 38-27. Data Bus Byte Lane Mapping for 16-Bit Peripherals, 32-Bit System, aips_byte_config[1:0] = 11

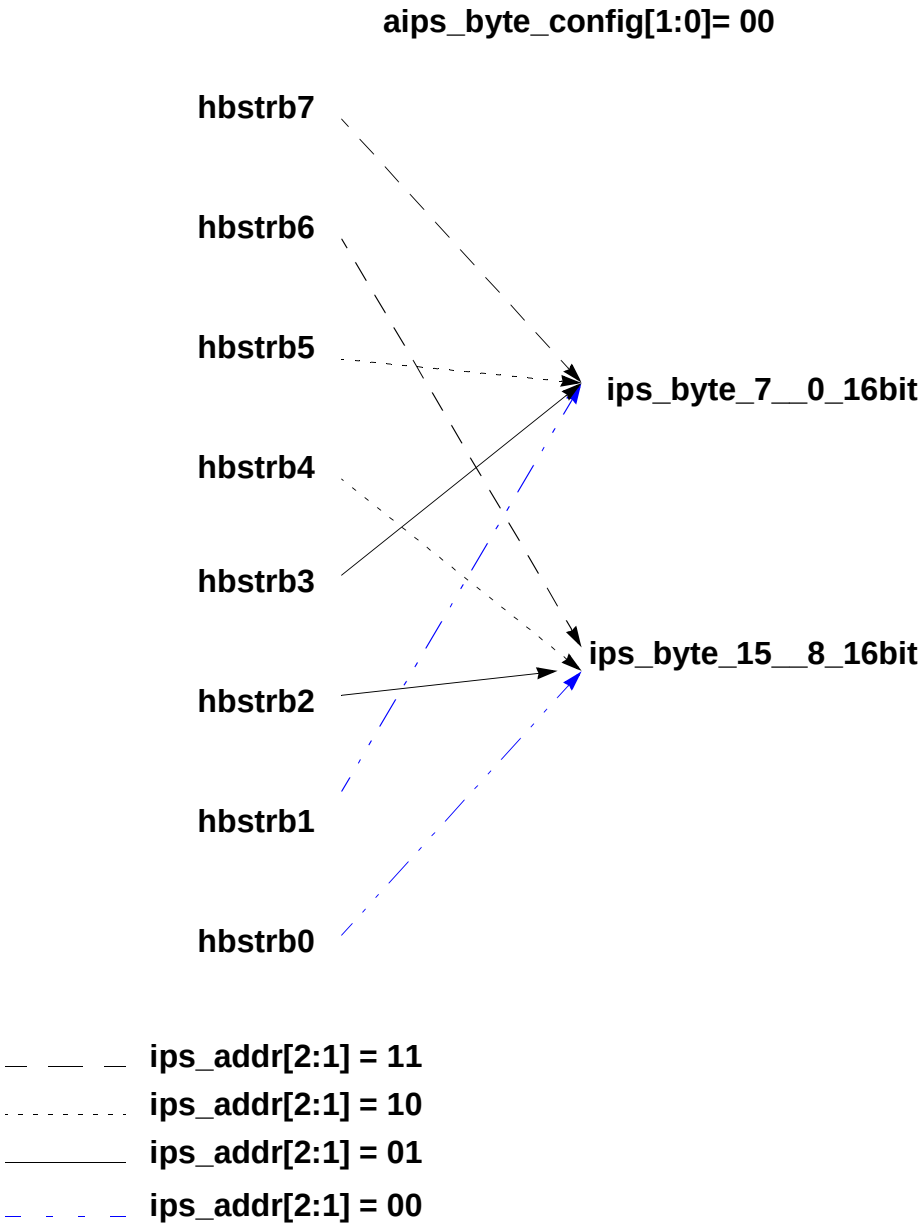


Figure 38-28. Mapping of 16-Bit Byte Strobes, 64-Bit System, aips_byte_config[1:0]= 00

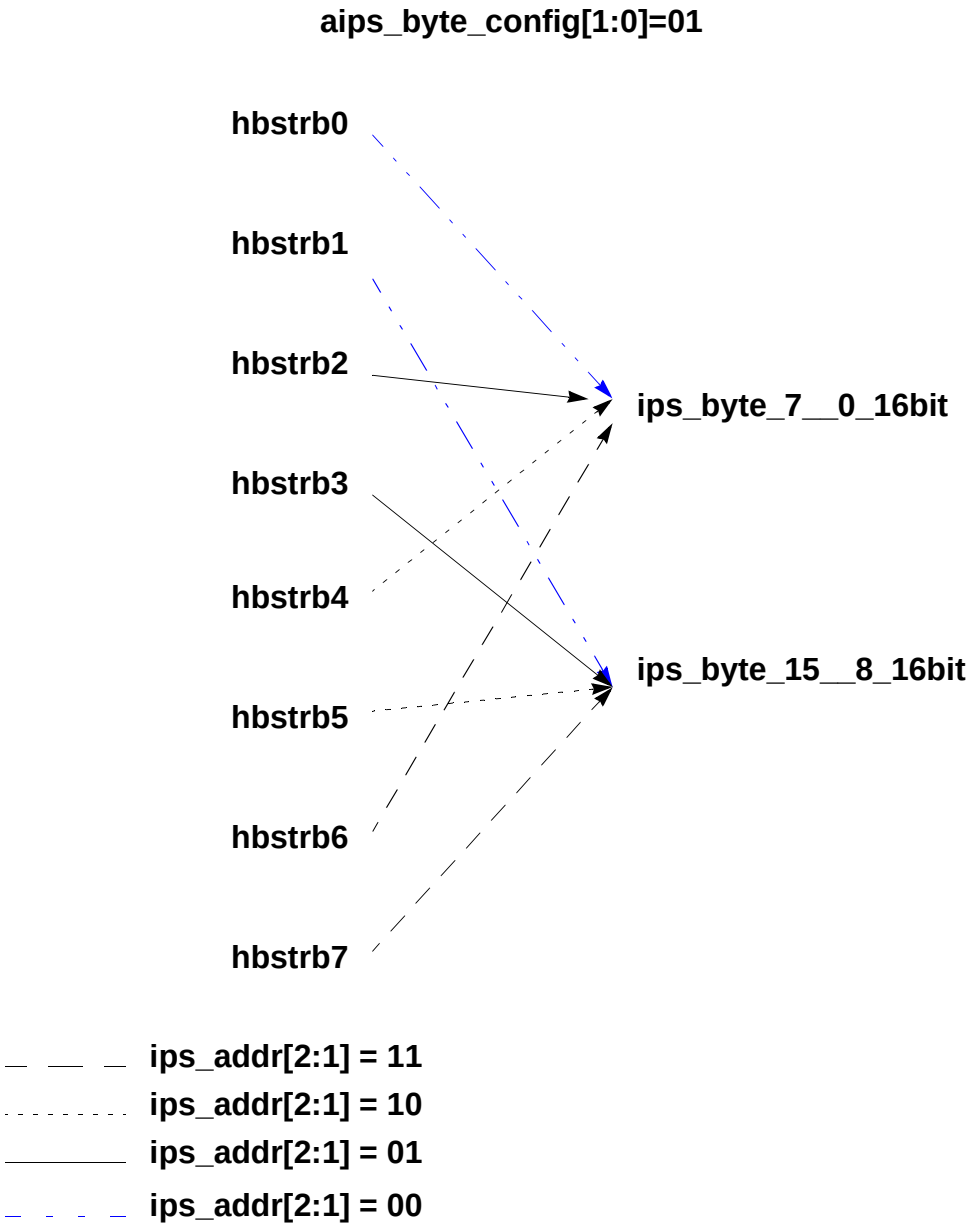


Figure 38-29. Mapping of 16-Bit Byte Strobes, 64-Bit System, aips_byte_config[1:0]=01

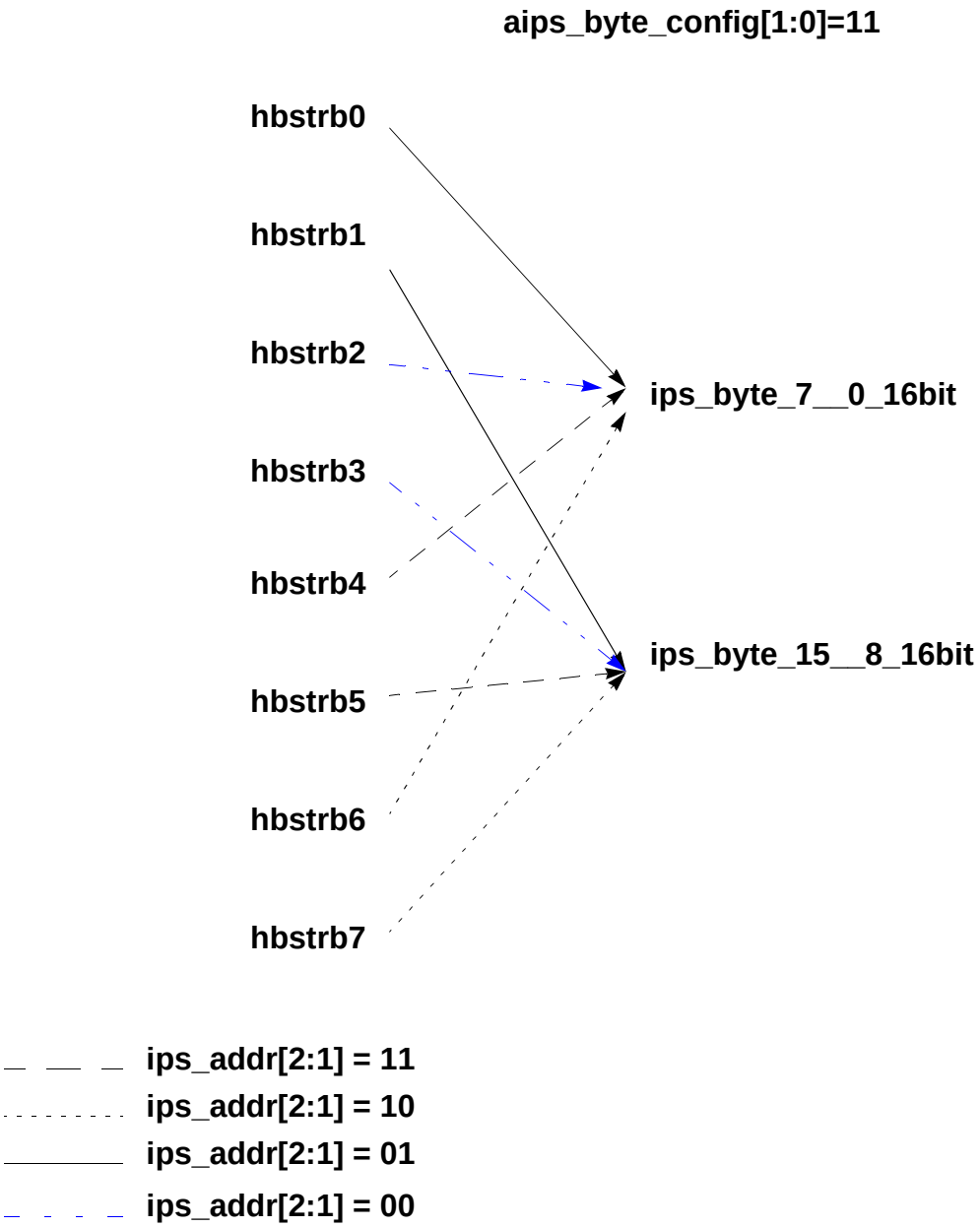


Figure 38-30. Mapping of 16-Bit Byte Strobes, 64-Bit System, aips_byte_config[1:0]= 11

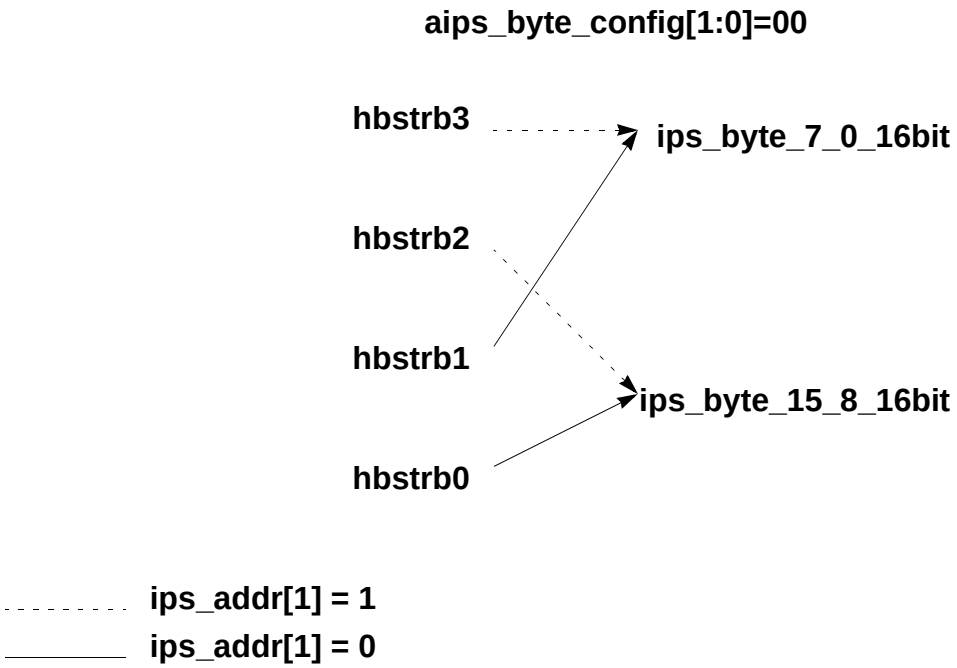


Figure 38-31. Mapping of 16-Bit Byte Strobes, 32-Bit System, aips_byte_config[1:0]= 00

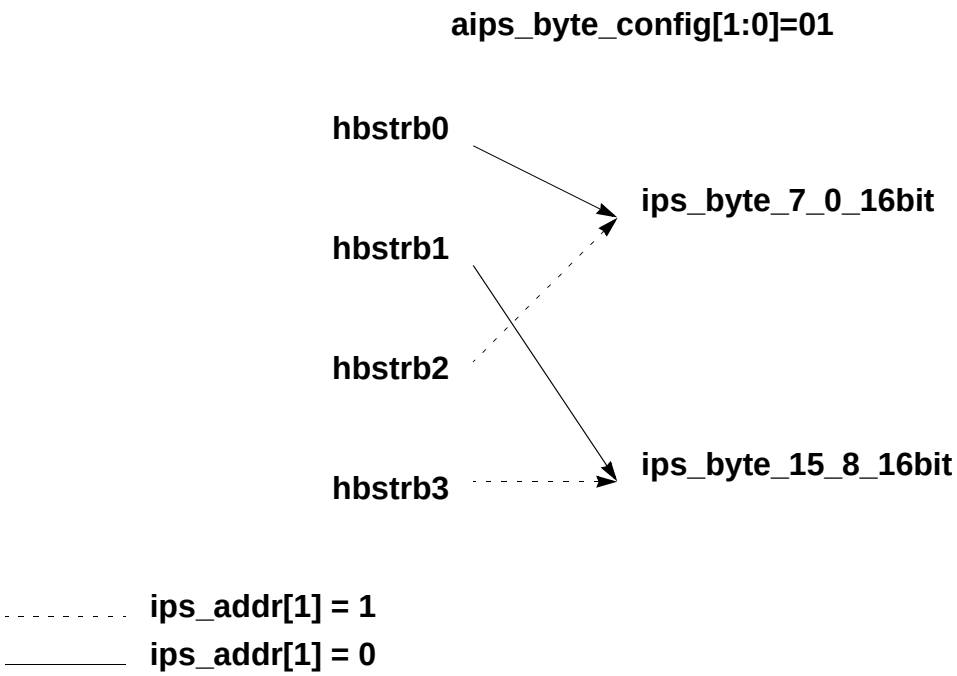


Figure 38-32. Mapping of 16-Bit Byte Strobes, 32-Bit System, aips_byte_config[1:0]= 01

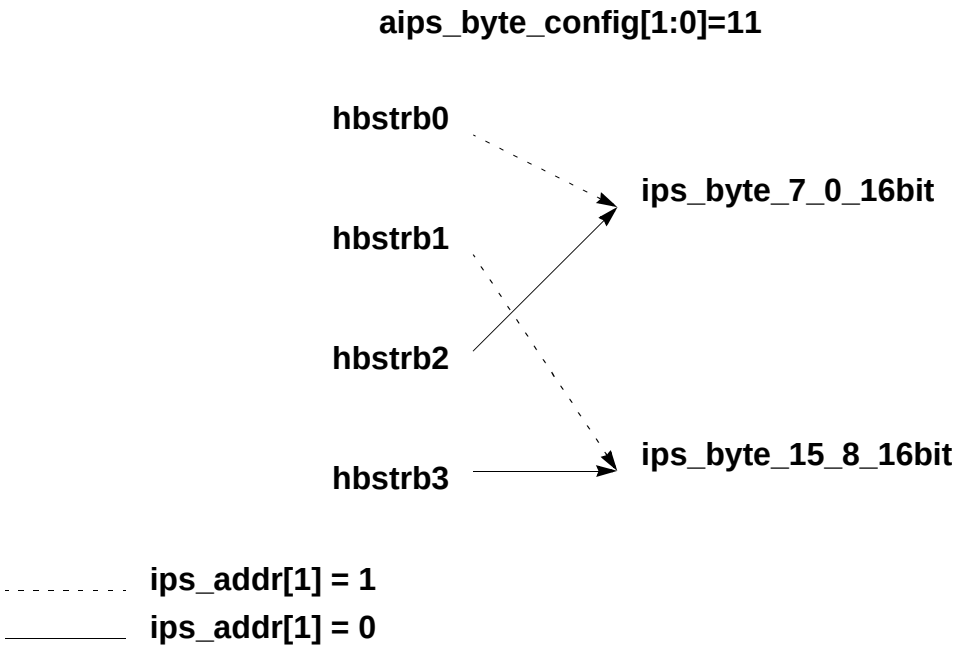


Figure 38-33. Mapping of 16-Bit Byte Strobes, 32-Bit System, aips_byte_config[1:0]= 11

38.4.4.3 8-Bit Peripherals

Data byte lanes of the AHB 2.v6 read and write data buses are mapped to the ips_rdata[7:0] and ips_wdata[7:0] data byte lane of the IPS for 8-bit peripherals based on the access ips_addr[2:0] values and the aips_byte_config[1:0] control signals. Byte enables on the IPS Bus are not needed by 8-bit peripherals.

For maximum flexibility, the AIPS provides control signals aips_byte_config[1:0], which allows the byte ordering of the IPS to be configured to support both byte-invariant (BE8) and word invariant (BE32) Big-Endian operation, as well as Little Endian operation. Figure 38-34 through Figure 38-37 shows the mapping of data byte lanes for 8-bit peripherals based on the state of aips_byte_config[1:0].

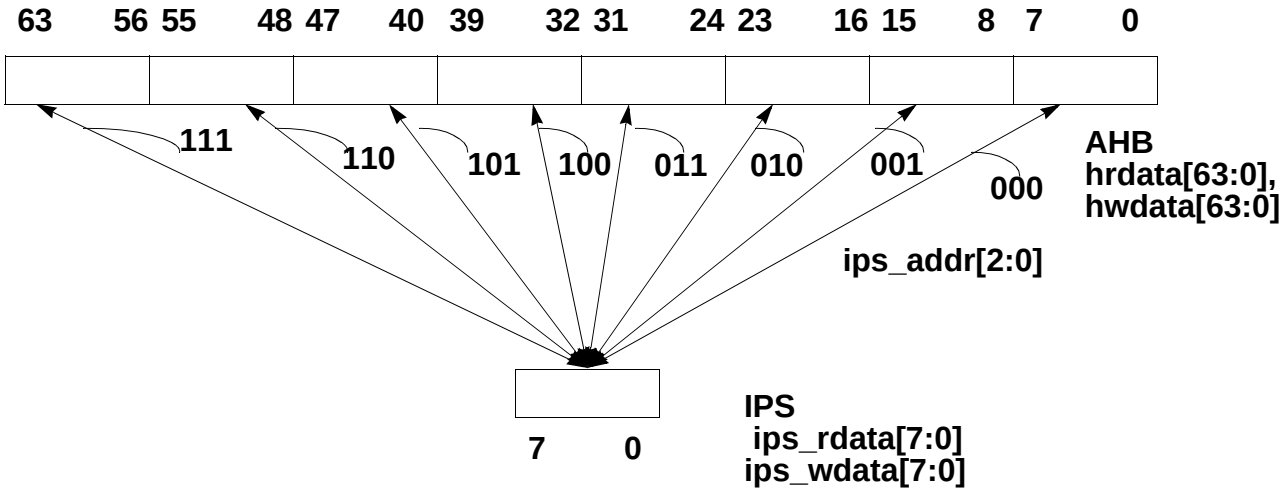


Figure 38-34. Data Bus Byte Lane Mapping for 8-Bit Peripherals, 64-Bit System, aips_byte_config[1:0] = 00, 01

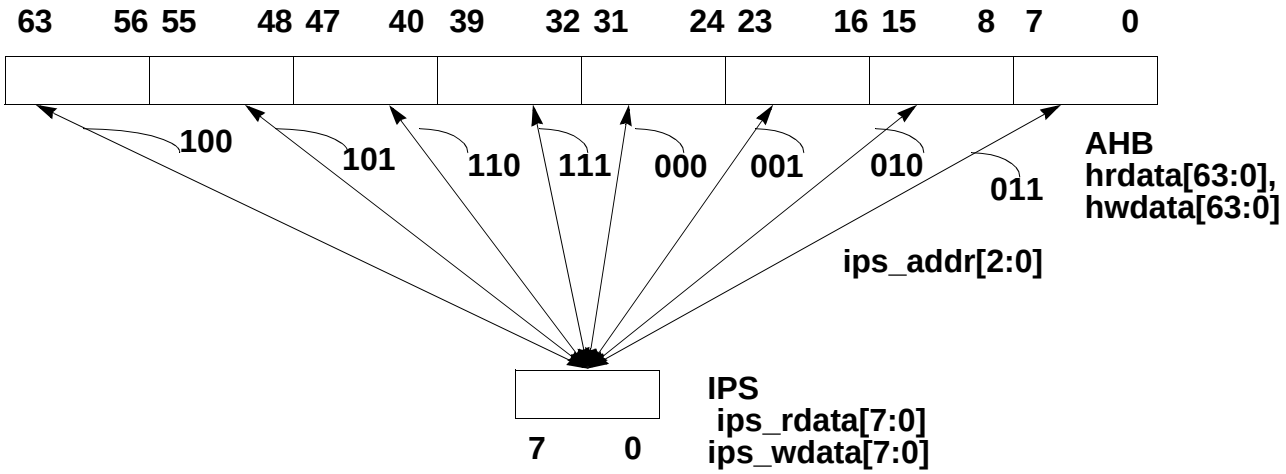


Figure 38-35. Data Bus Byte Lane Mapping for 8-Bit Peripherals, 64-Bit System, aips_byte_config[1:0] = 11

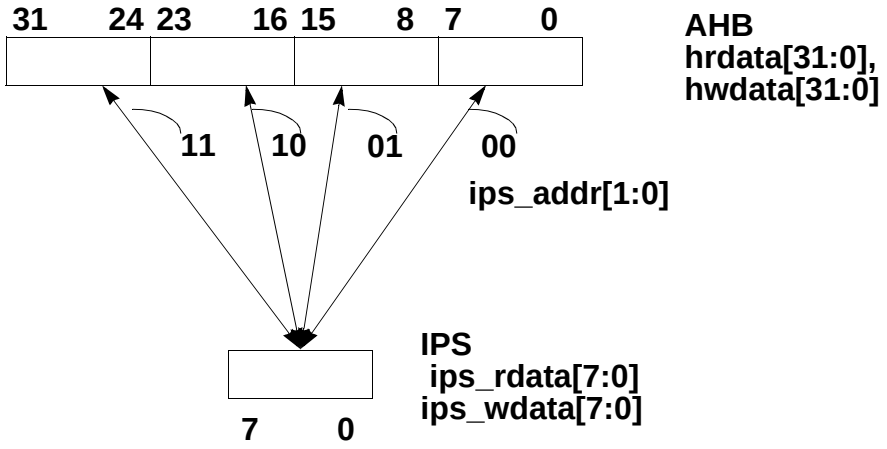


Figure 38-36. Data Bus Byte Lane Mapping for 8-Bit Peripherals, 32-Bit System, aips_byte_config[1:0] = 00, 01

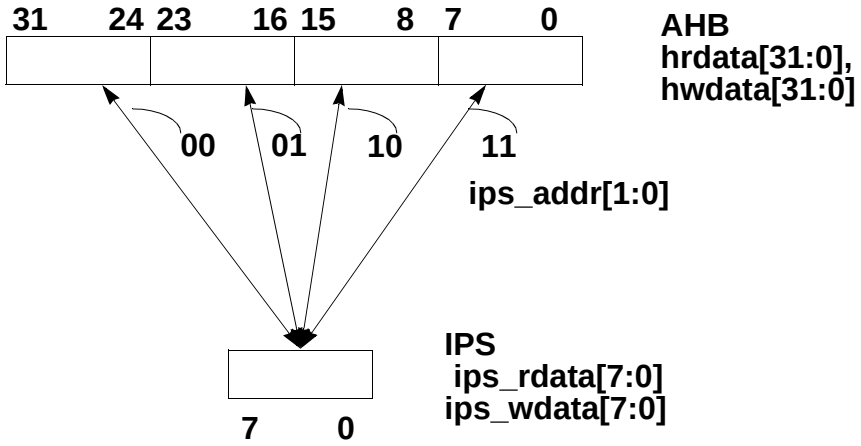


Figure 38-37. Data Bus Byte Lane Mapping for 8-Bit Peripherals, 32-Bit System, aips_byte_config[1:0] = 11

38.4.5 Access Support

Aligned 64-bit accesses, aligned and misaligned word and halfword accesses, as well as byte accesses are supported for 32-bit peripherals. Misaligned accesses are supported to allow memory to be placed on the IPS. Peripheral registers must not be misaligned, although no explicit checking is performed by the AIPS. The AIPS will perform two IPS transfers for 64-bit accesses, word accesses with byte offsets of 1,2, or 3, and for halfword accesses with a byte offset of 3. All other accesses will be performed with a single IPS transfer.

Only aligned halfword and byte accesses are supported for 16-bit peripherals. All other accesses types are unsupported, and are terminated with an error response.

Only byte accesses are supported for 8-bit peripherals. All other accesses types are unsupported, and results of such accesses are terminated with an error response.

Exclusive accesses are signaled by the assertion of hprot5 on the AHB. For read cycles, hprot5 is ignored. Exclusive write cycles (hprot5 asserted) will be terminated with an error response.

Table 38-20 represents read access delay through AIPS for different cases.

Table 38-20. Read Access Delay Through AIPS

S.No.	Access Size	Misaligned Access	AIPS_DLY_CYCLE	No. of Wait States by Peripheral	AIPS Access Delay
1.	64 bit	No	0	0	3 cycle
2.	32 bit	Yes	0	0	3 cycle
3.	32/16/8 bit	No	0	0	2 cycle
4.	32/16/8 bit	No	1	0	3 cycle
5.	32/16/8 bit	No	0	1	3 cycle
6.	32/16/8 bit	No	1	1	4 cycle

Table 38-21 represents write access delay through AIPS for different cases.

Table 38-21. Write Access Delay Through AIPS

S.No.	Access Size	Misaligned Access	No. of Wait States by Peripheral	Buffered Write	AIPS Access Delay
1.	64 bit	No	0	No	4 cycle
2.	32 bit	yes	0	No	4 cycle
3.	32/16/8 bit	No	0	No	3 cycle
4.	32/16/8 bit	No	1	No	4 cycle
5.	32/16/8 bit	No	0	Yes with AIPS_DLY_CYCLE = 0	1 cycle
6.	32/16/8 bit	No	0	Yes with AIPS_DLY_CYCLE = 1	2 cycle

38.4.6 Read Cycles

Two clock read accesses are possible with the AIPS when the requested access size is 32-bits or smaller, and is not misaligned across a 32-bit boundary. If the requested access size is 64-bits, or it is misaligned across a 32-bit boundary, then a minimum of three clocks are required to complete the access. The difference can be seen by examining Figure 38-40 and Figure 38-42.

In a read cycle, all address, control signals, and ips_module_en are launched off the same edge of hclk in which the AIPS becomes selected. The read cycle is terminated when ips_xfr_wait is negated at the rising edge of hclk. On the cycle following the negation of ips_xfr_wait, the AIPS presents the valid data to the AHB port and asserts aips_hready_out to indicate to the AHB that the cycle has terminated.

Once the AIPS asserts aips_hready_out, it remains asserted until the rising edge of hclk when hready_in is asserted. Figure 38-40 shows a case where hready_in has been delayed. This will not be the case for most

systems, but the AIPS supports delayed responses for those systems which need additional propagation time to the master.

When the AIPS is performing a multi-cycle access, the aips_hready_out signal is not asserted until after two read operations have occurred to the targeted IP bus peripheral. Refer to Figure 38-42 for an example.

The AIPS may be configured at synthesis time with the AIPS_DLY_CYCLE parameter to add an additional cycle to read accesses to allow for additional decode time prior to assertion of a module enable. An example read cycle in this configuration is shown in Figure 38-44.

38.4.7 Write Cycles

Three clock write accesses are possible with the AIPS when the requested access size is 32-bits or smaller, and is not misaligned across a 32-bit boundary. If the requested access size is 64 bits, or it is misaligned across a 32-bit boundary, then a minimum of four clocks are required to complete the access. The difference can be seen by examining Figure 38-45 and Figure 38-47.

In a write cycle, all address and control signals are launched off the same edge of hclk, which is the same edge of hclk in which the AIPS becomes selected. The ips_wdata and ips_module_en signals are launched off the following edge of hclk, ensuring relaxed write data timing to the targeted IP bus peripheral. The write cycle is terminated when ips_xfr_wait is negated at the rising edge of hclk. On the cycle following the negation of ips_xfr_wait, the AIPS asserts aips_hready_out to indicate to the AHB port that the cycle has terminated.

When the AIPS is performing a multi-cycle access, the aips_hready_out signal is not asserted until after two write operations have occurred to the targeted IP bus peripheral.

Once the AIPS asserts aips_hready_out, it remains asserted until the rising edge of hclk when hready_in is asserted.

38.4.8 Buffered Write Cycles

Single clock write responses to the AHB are possible with the AIPS when the requested write access is bufferable. If the requested access does not violate the permissions check, and if both master and peripheral are enabled for buffering writes, the AIPS will internally buffer the write cycle. The write cycle is terminated early with zero AHB wait states. The access proceeds normally on the IPS interface, but error responses (ips_xfr_err) are ignored. An example of a buffered write access is shown in Figure 38-49.

Once the AIPS asserts aips_hready_out, it remains asserted until the rising edge of hclk when hready_in is asserted.

The AIPS may be configured at synthesis time with the AIPS_DLY_CYCLE parameter to add an additional cycle to buffered write accesses to allow for additional decode time prior to assertion of aips_hready_out. An example buffered write cycle in this configuration is shown in Figure 38-50.

All accesses are initiated and completed in order on the IPS interface, regardless of buffering. If the buffer is full, a following write cycle will stall until it can either be buffered (if bufferable) or can be initiated. If the buffer has valid entries, a following read cycle will stall until the buffer is emptied and the read cycle can be completed.

38.4.9 Aborted Cycles

The AIPS follows a standard procedure when a system bus cycle is aborted and the abort is initiated by the AIPS itself or the targeted IP bus peripheral. The AIPS either blocks initiation or immediately terminates any IP bus activity that is ongoing. The AIPS then asserts aips_hresp[0] (indicating an error has occurred). On the following clock cycle, the AIPS asserts aips_hready_out and the AIPS holds both aips_hresp[0] and aips_hready_out asserted until hready_in is asserted. At this time the AIPS negates both aips_hresp[0] and aips_hready_out.

There are several conditions that can cause the AIPS to abort the current operation and report an error. The first is the case in which the targeted IP bus peripheral asserts ips_xfr_err while ips_xfr_wait is negated. In this case the AIPS immediately terminates access to the targeted IP bus peripheral and follows the abort procedure described above. Whether the current IP bus access is a multi-cycle access or a single cycle access has no bearing on the behavior of the AIPS. The AIPS responds identically in both cases. Refer to Figure 38-51 and Figure 38-52 for additional clarification.

The second case that can cause an error response to the AHB is when an access is attempted to an IP bus peripheral whose corresponding PACR or OPACR settings do not allow the access, thus causing a permissions violation, or if an unsupported access is attempted (access larger than peripheral size, exclusive accesses, or certain misaligned accesses). In this case the AIPS does not initiate any IP bus activity, but instead responds by following the abort procedure described above. Refer to Figure 38-53 and Figure 38-54 for examples.

The third case that can cause an error response to the AHB is when an access is attempted to a location at which there is no IP bus peripheral. In this case the AIPS does not initiate any IP bus activity but instead responds by following the same abort procedure described above for a permissions violation.

The AIPS can also terminate the current IP bus peripheral access activity if hready_in is asserted before the end of the current IP bus access. While this circumstance should not occur, this will not result in an error condition being reported as this behavior is initiated by the AHB. If the AHB is once again requesting access to an IP bus peripheral, a new access to the targeted peripheral is initiated, otherwise all IP bus activity terminates.

38.4.10 Timing Diagrams

Figure 38-38 through Figure 38-54 illustrate the operations of various cycle types and responses. The operation for 32-bit, 16-bit, and 8-bit peripherals is very similar. Most diagrams illustrate 32-bit peripheral accesses.

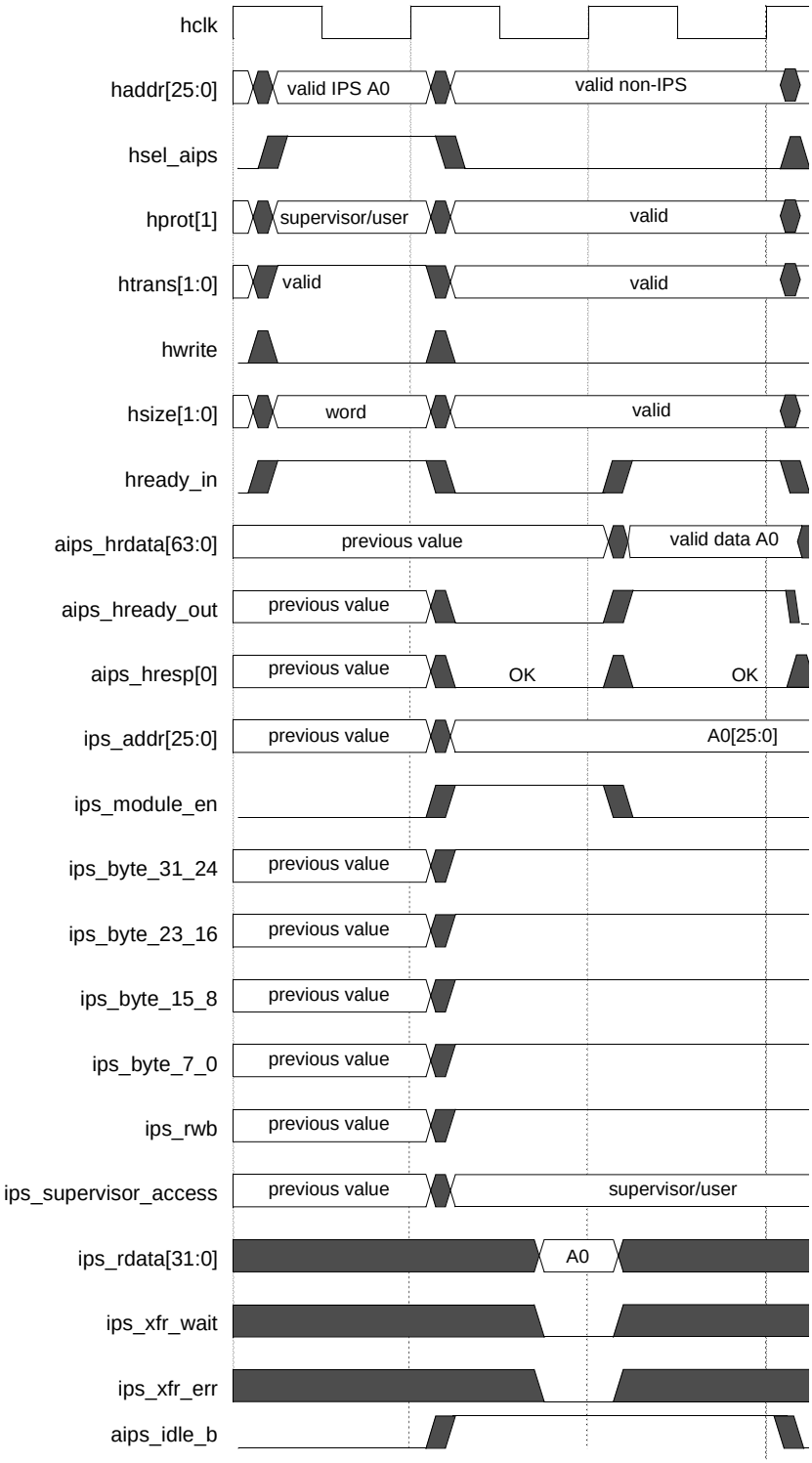


Figure 38-38. Read Access, 32-Bit Peripheral, 2 Cycle AIPS Delay

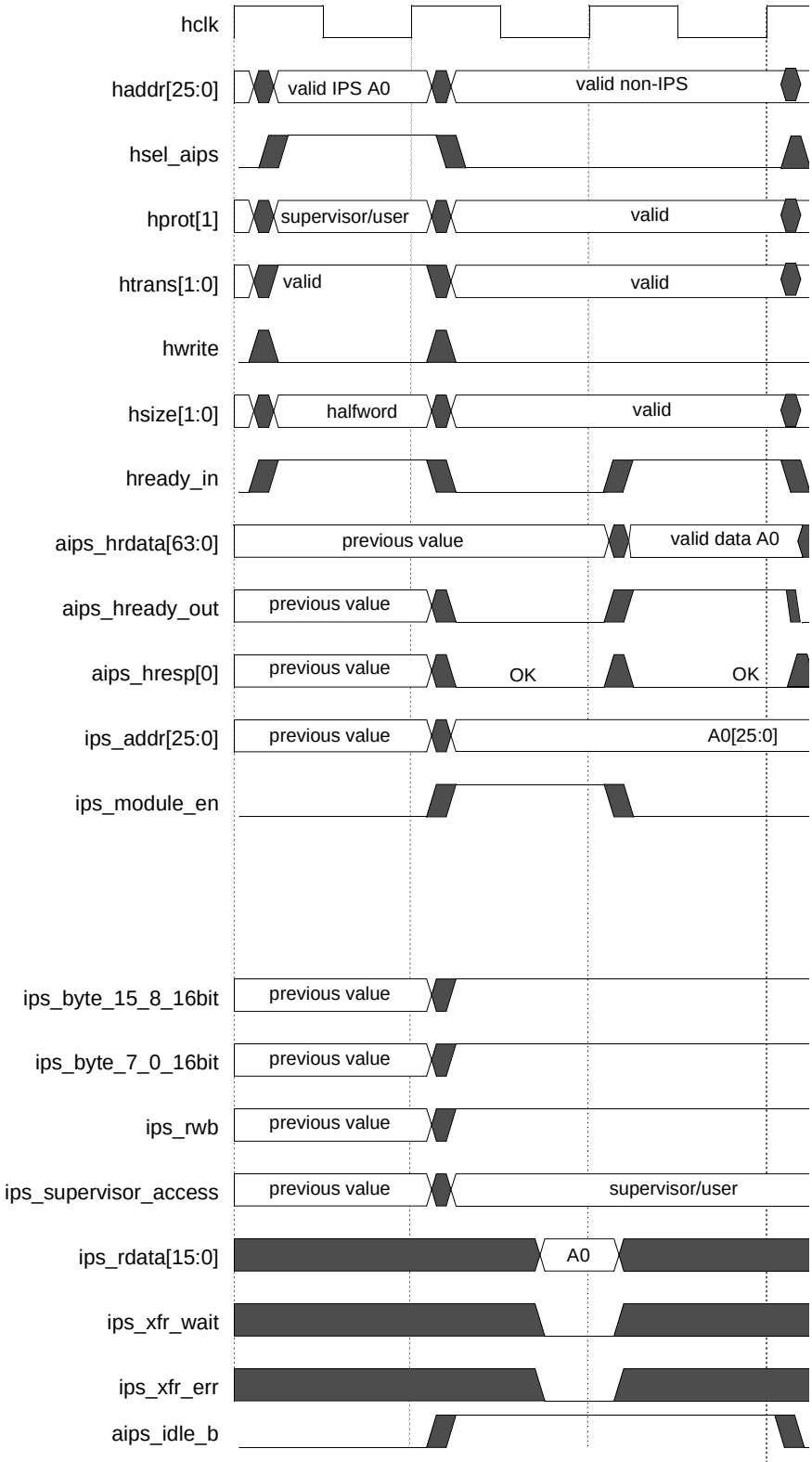


Figure 38-39. Read Access, 16-Bit Peripheral, 2 Cycle AIPS Delay

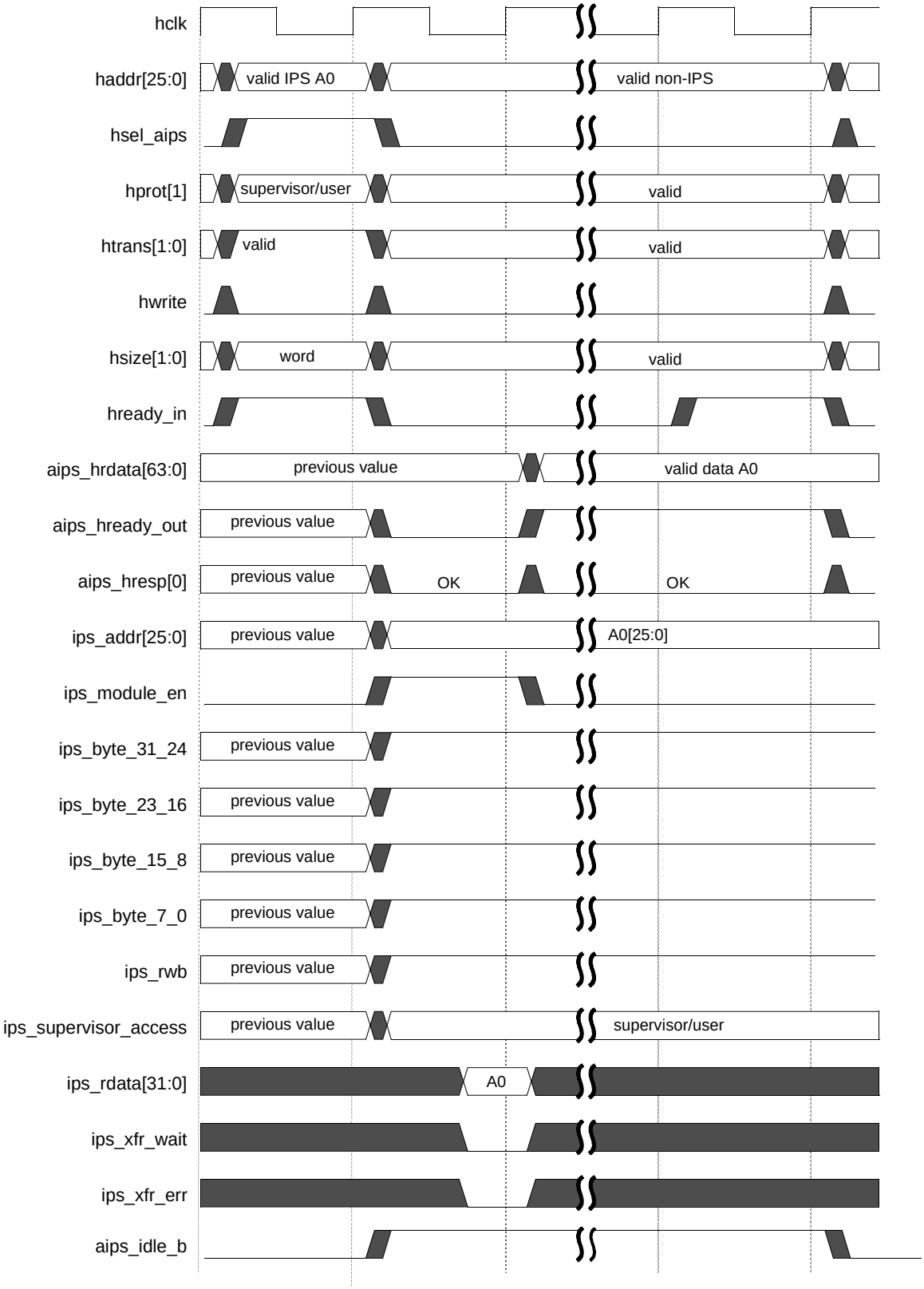


Figure 38-40. Read Access, Delayed hready_in

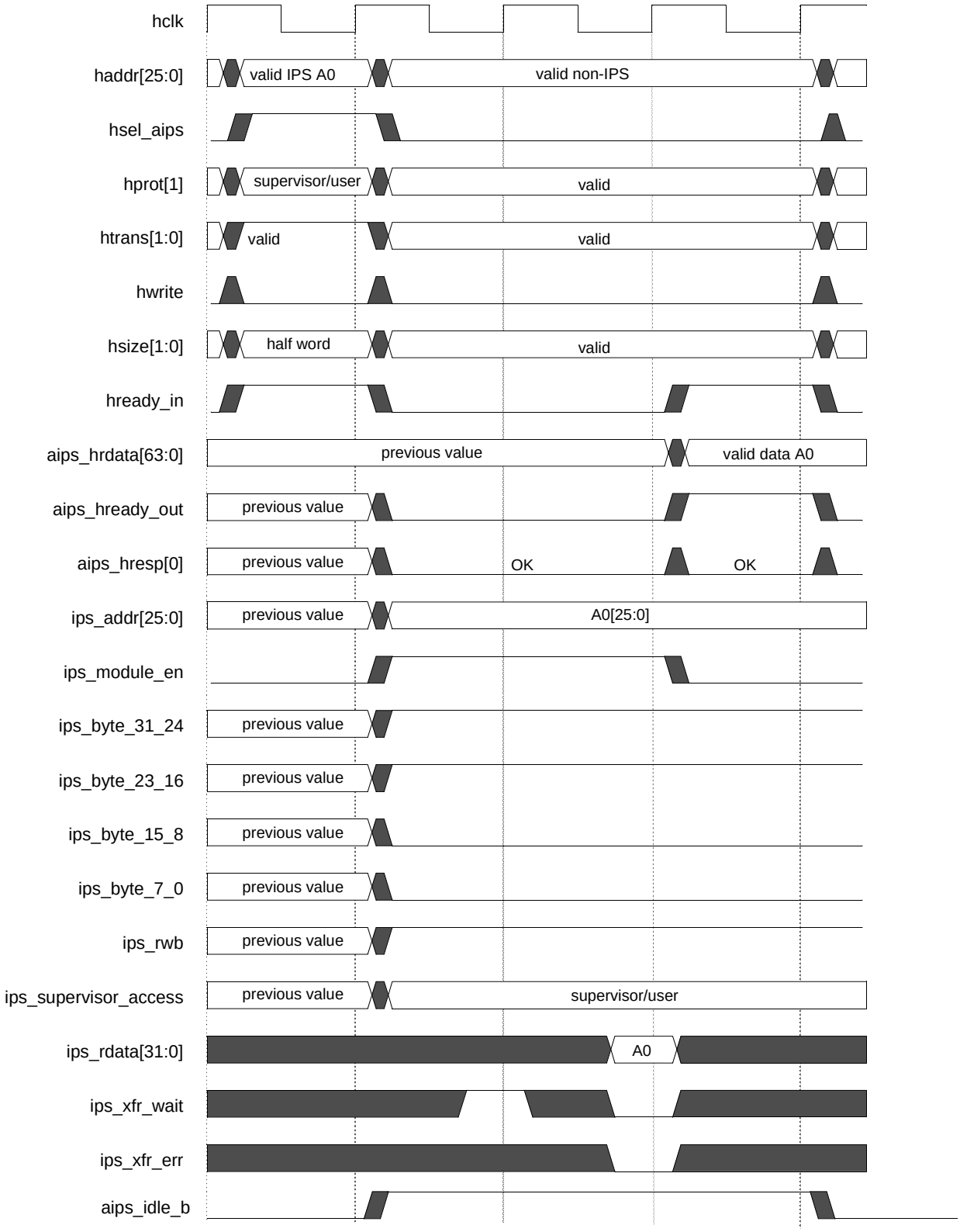


Figure 38-41. Read Access with One IPS Wait-State, Three-Cycle AIPS Delay

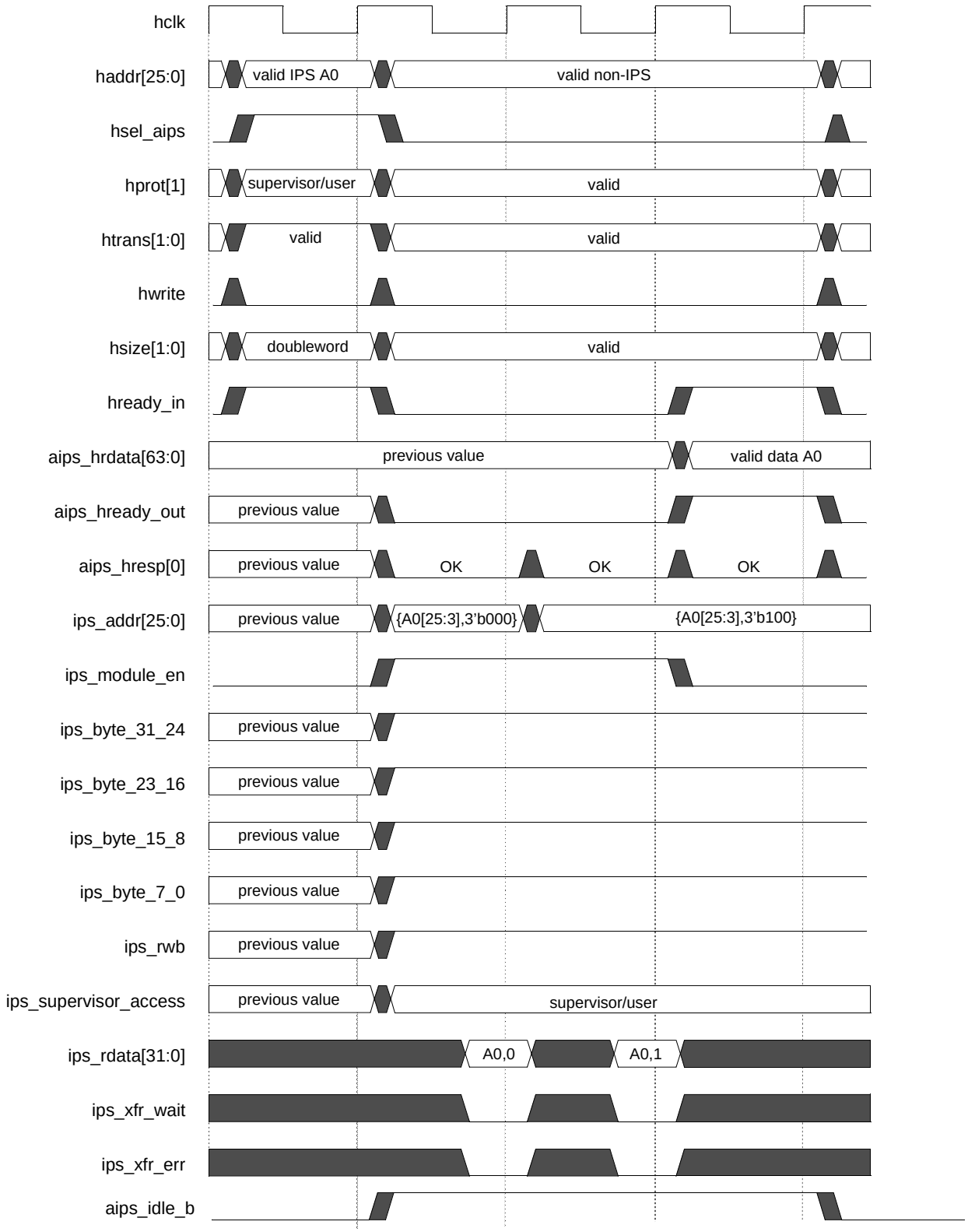


Figure 38-42. Doubleword Read Access, Three-Cycle AIPS Delay

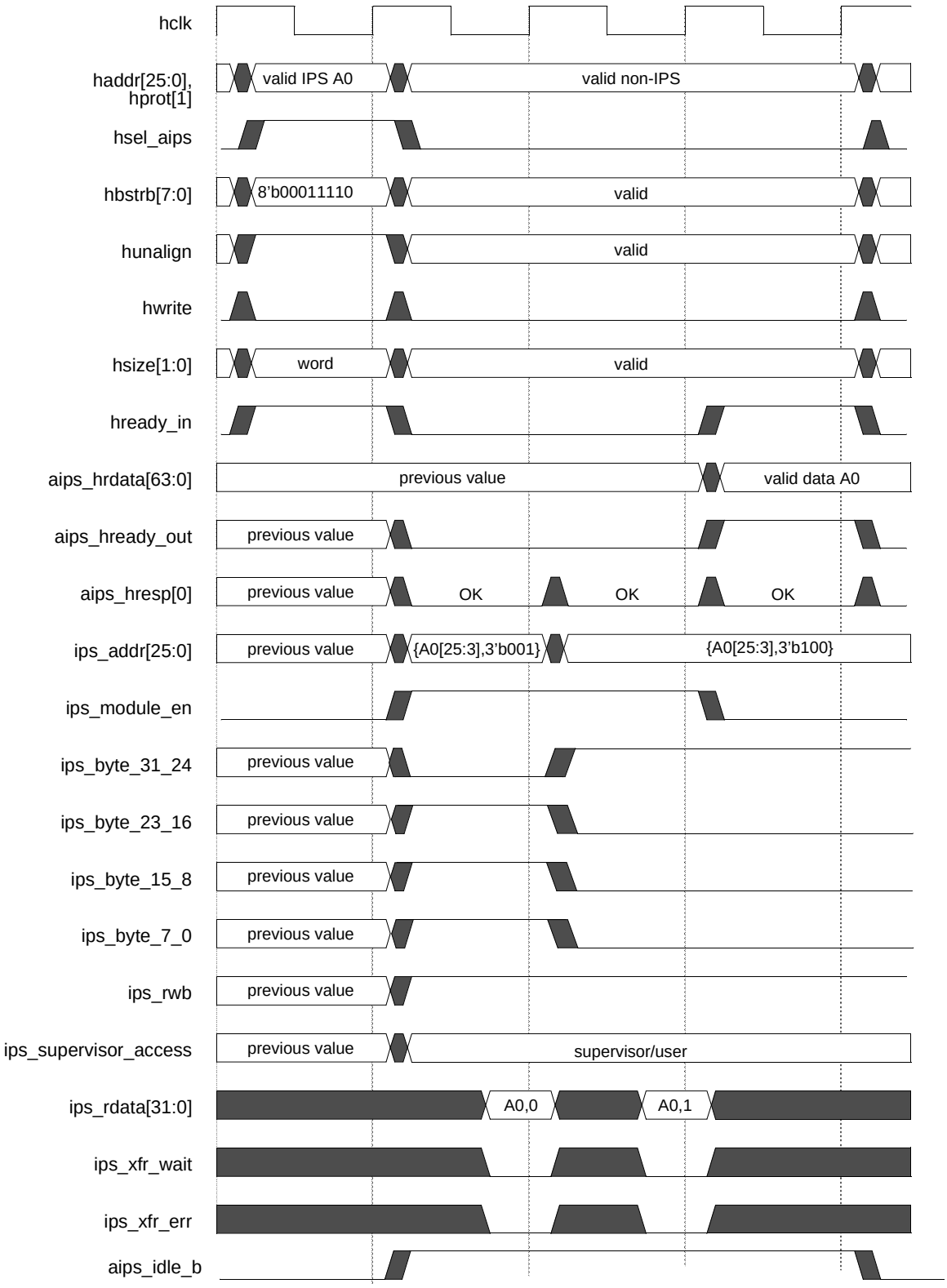


Figure 38-43. Misaligned Read Access (Word Access to Byte Offset 1) Three-Cycle AIPS Delay

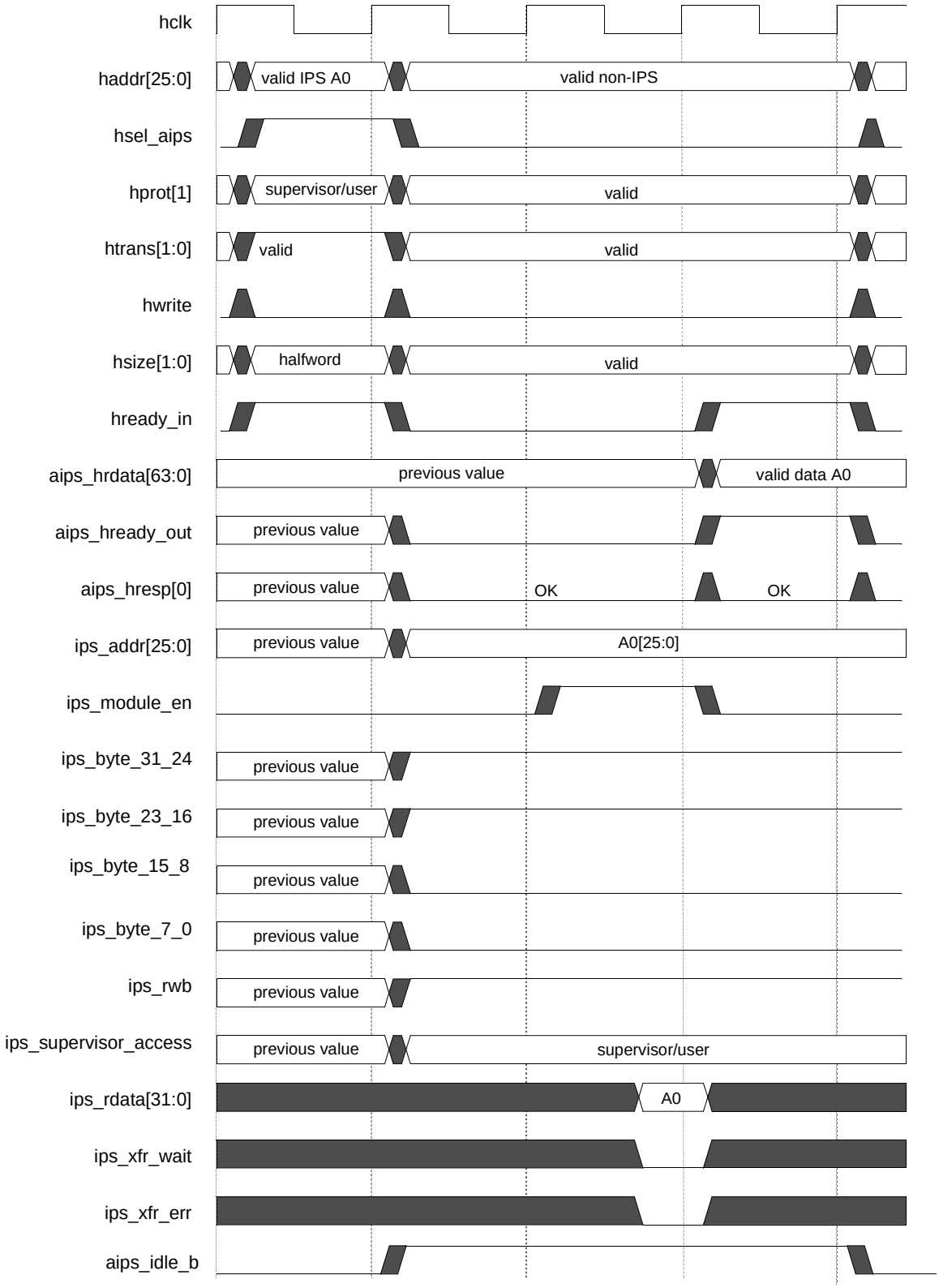


Figure 38-44. Read Access with Delayed Initiation, Three-Cycle AIPS Delay

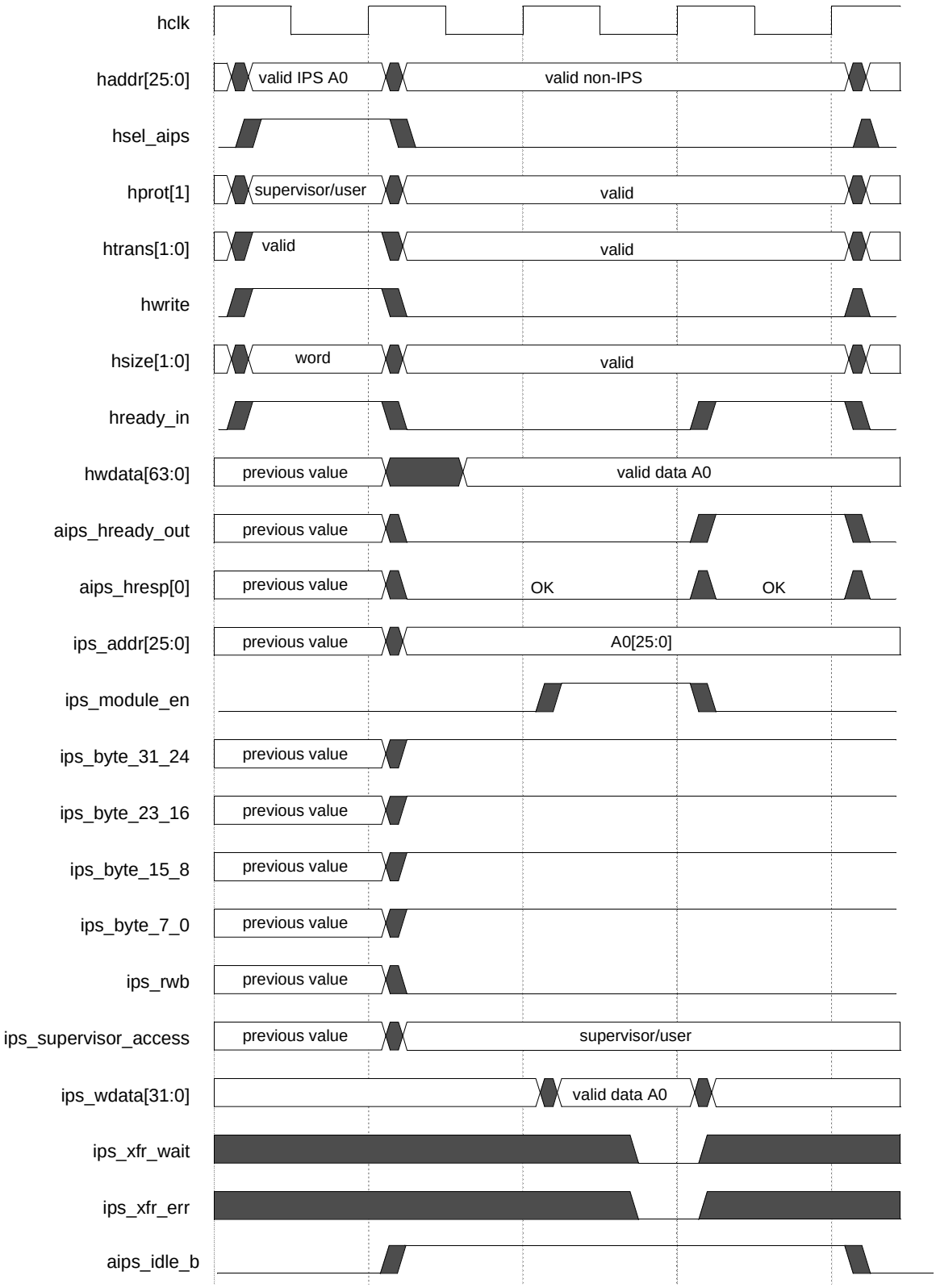


Figure 38-45. Write Access, Three-Cycle AIPS Delay

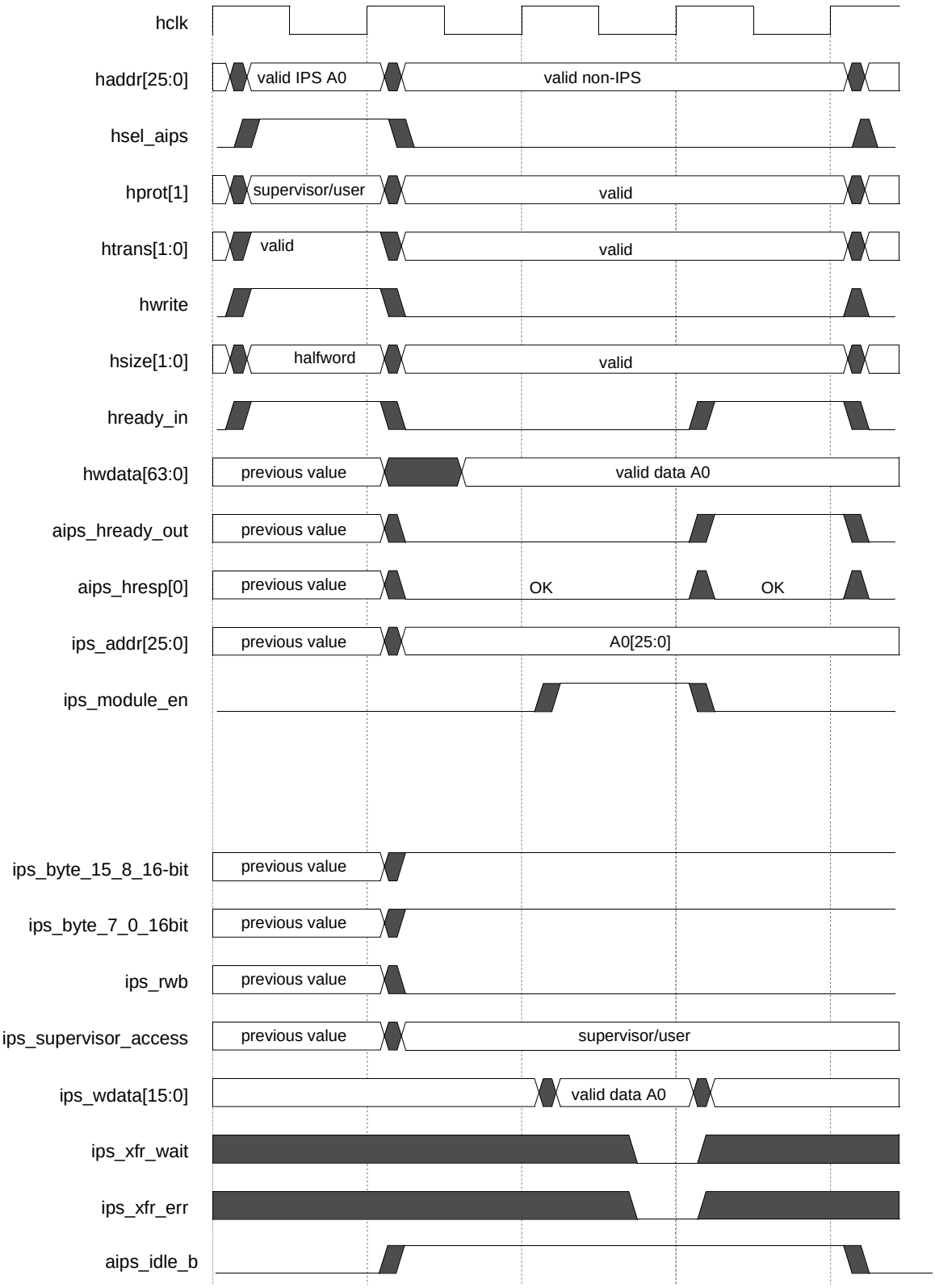


Figure 38-46. Write Access, 16-Bit Peripheral, Three-Cycle AIPS Delay

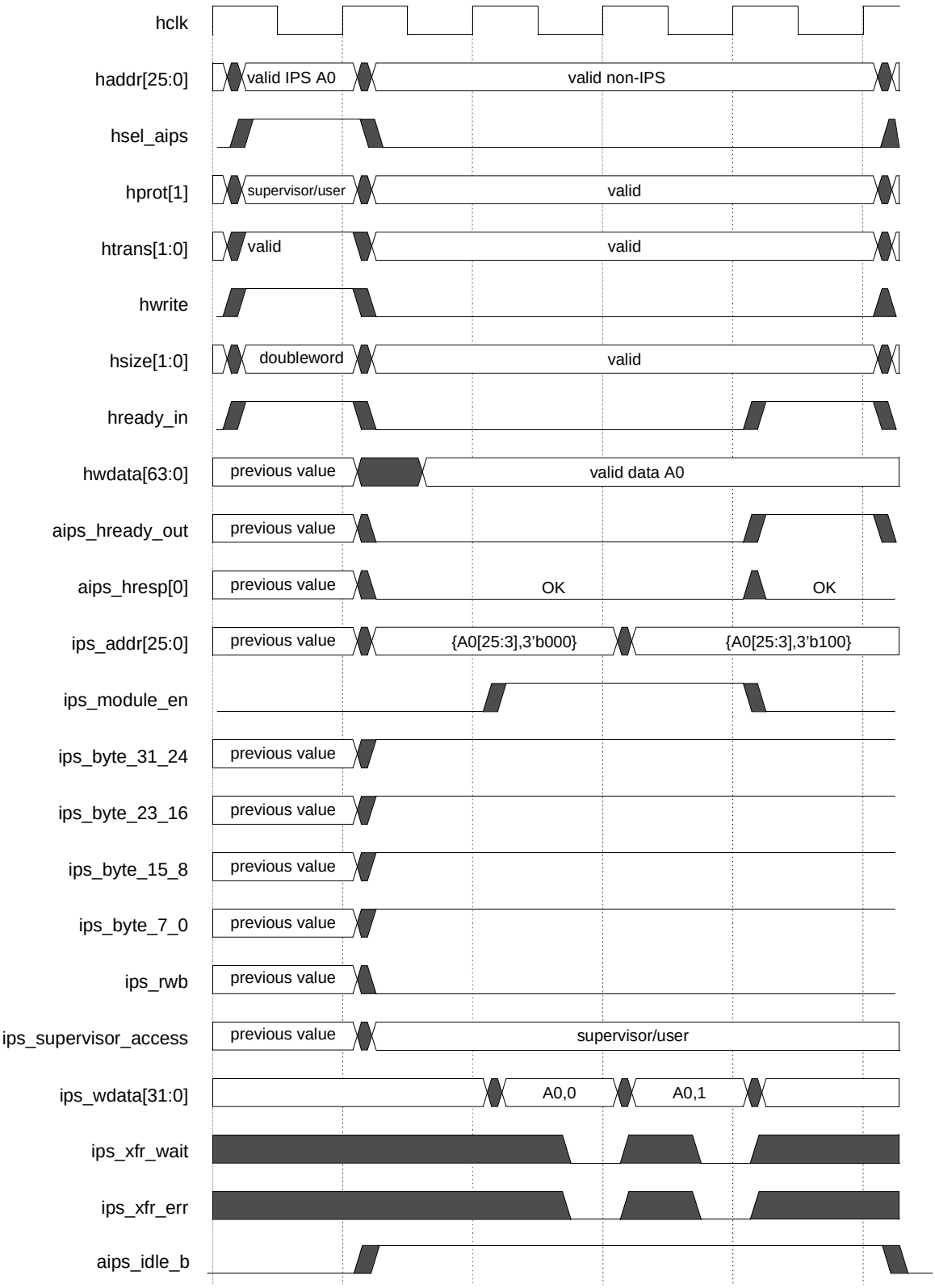


Figure 38-47. Doubleword Write Access, Four-Cycle AIPS Delay

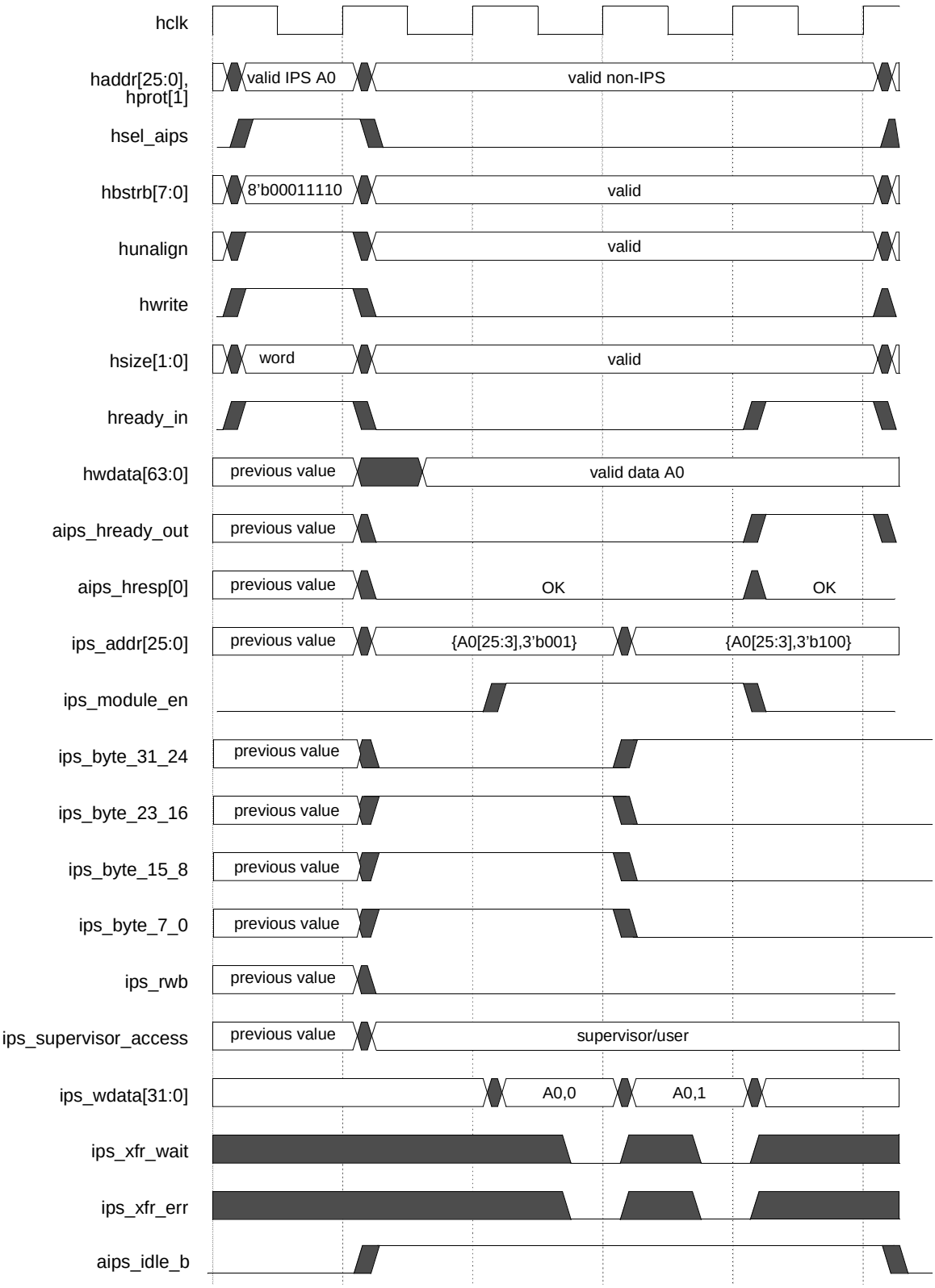


Figure 38-48. Misaligned Write Access (Word Write to Byte Offset 1) Four-Cycle AIPS Delay

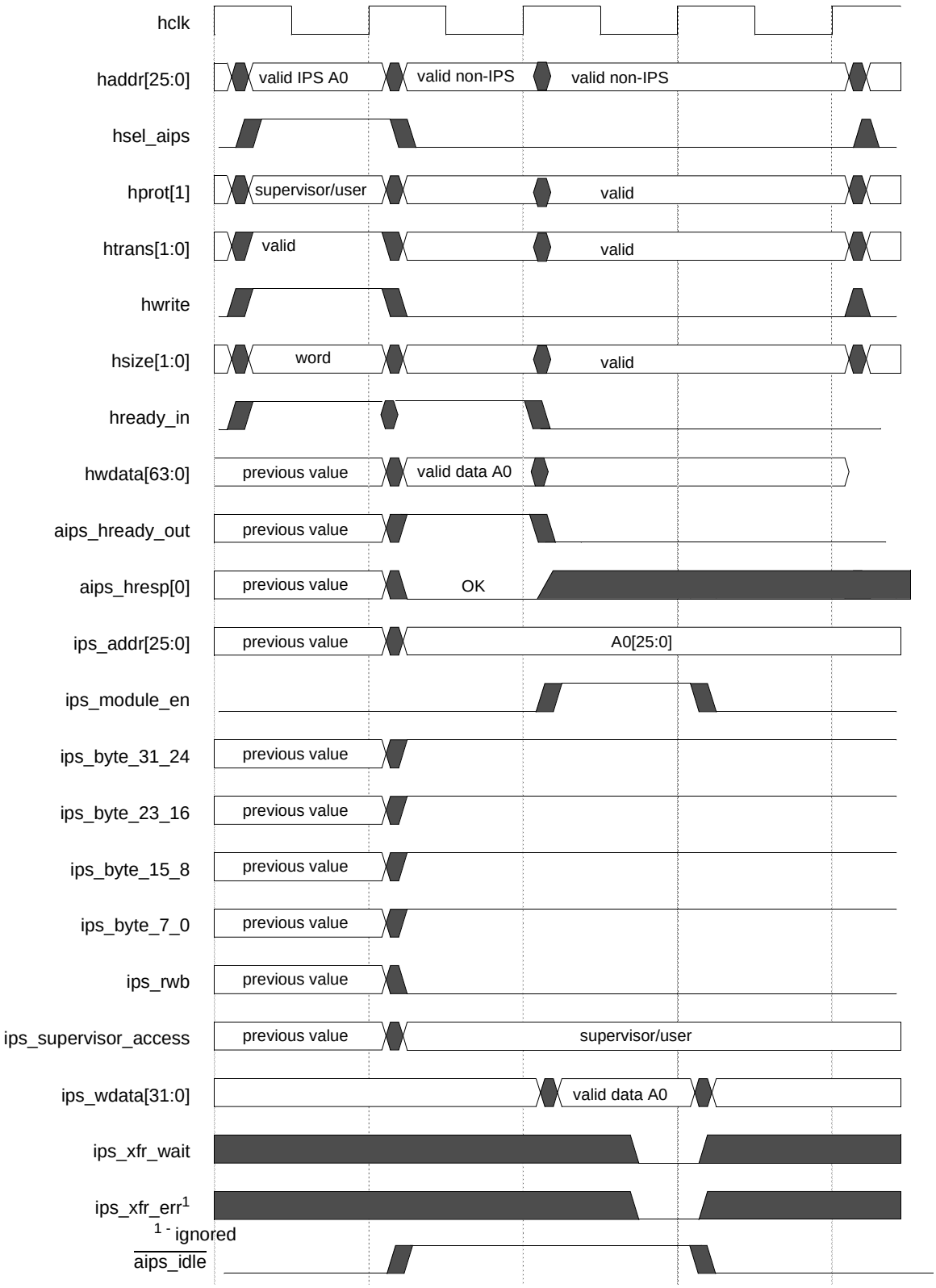


Figure 38-49. Buffered Write Access, Single-Cycle AIPS Delay

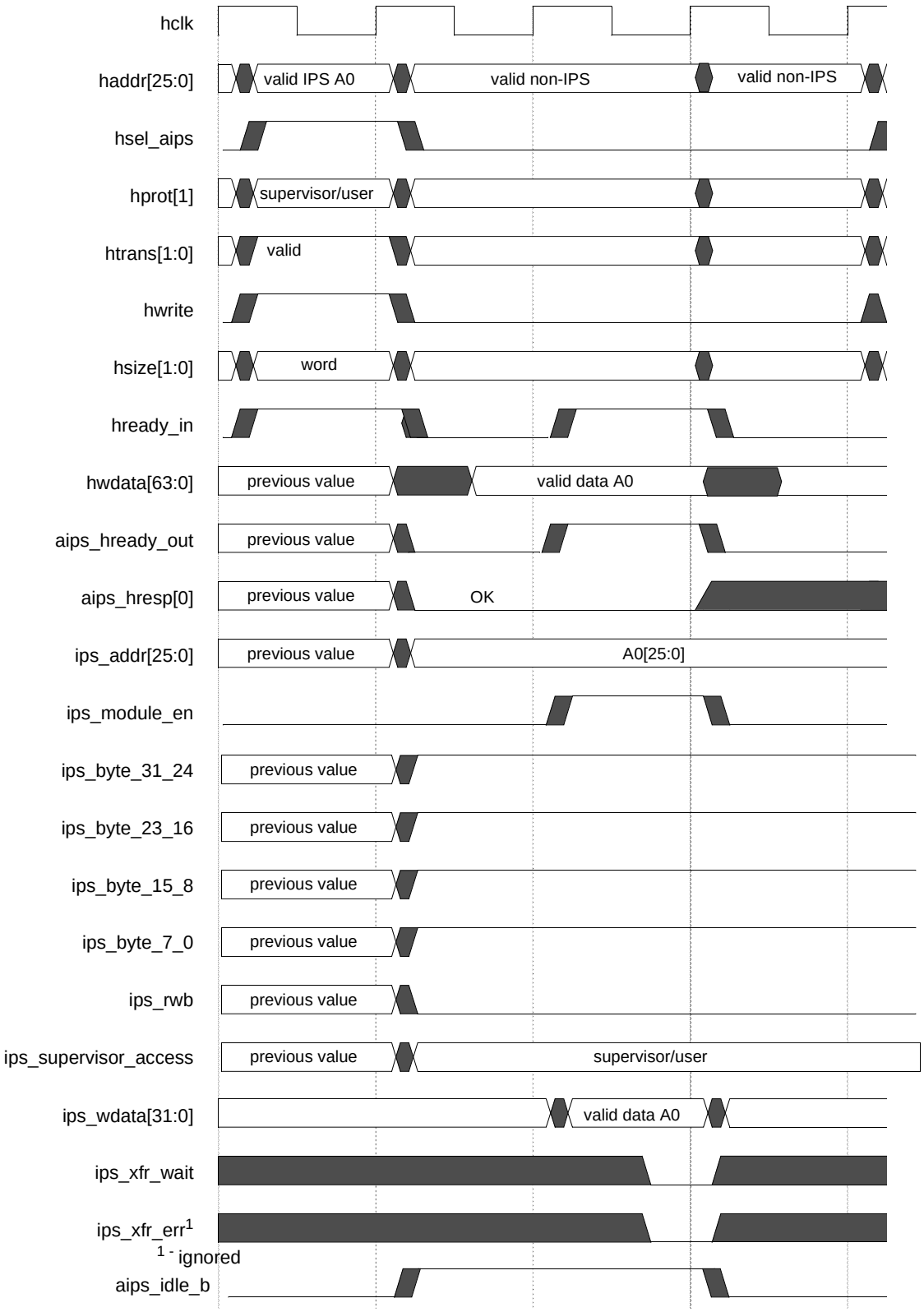


Figure 38-50. Buffered Write Access, Delayed (AIPS_DLY_CYCLE = 1), Two-Cycle AIPS Delay

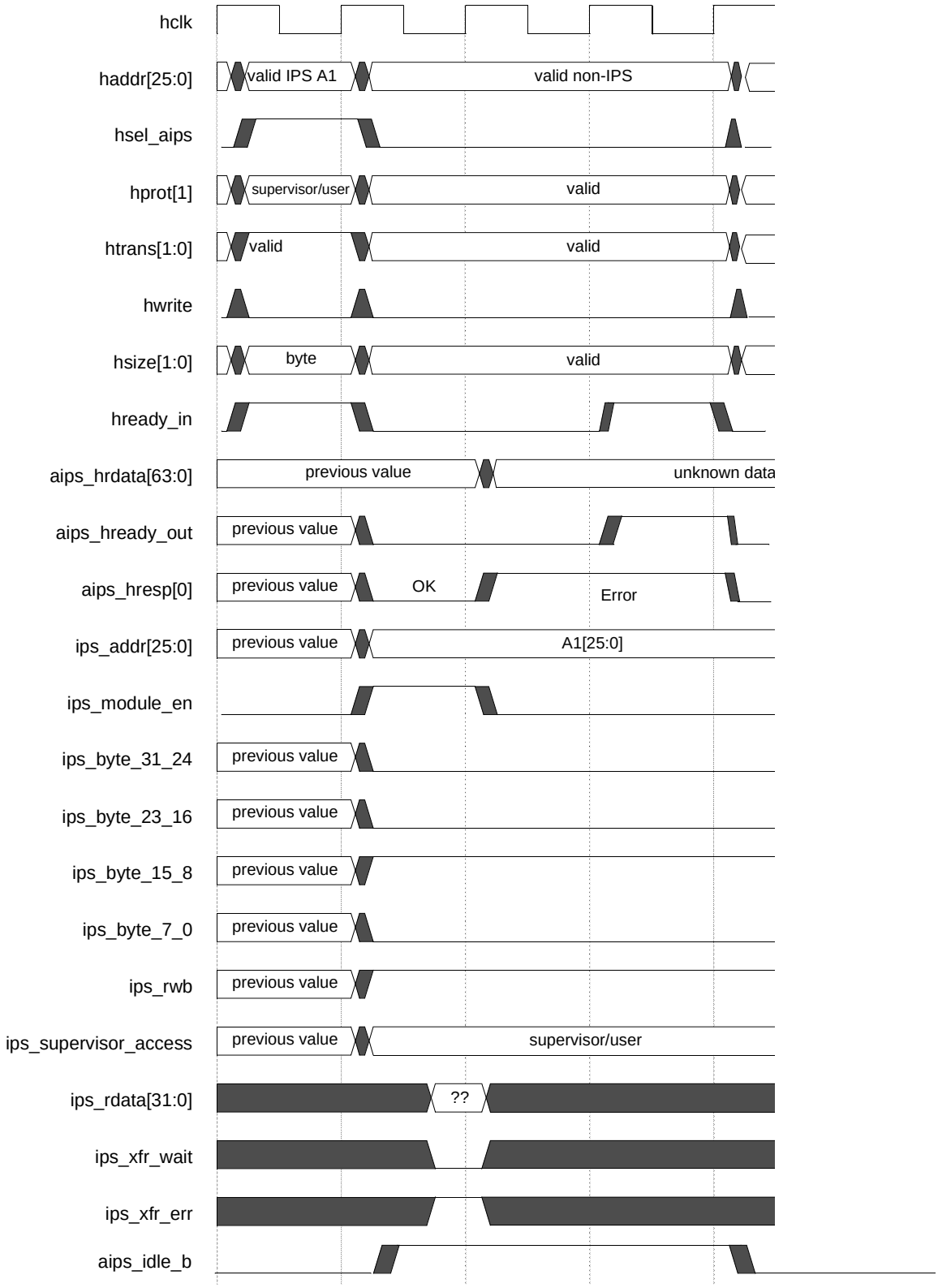


Figure 38-51. Read With Error

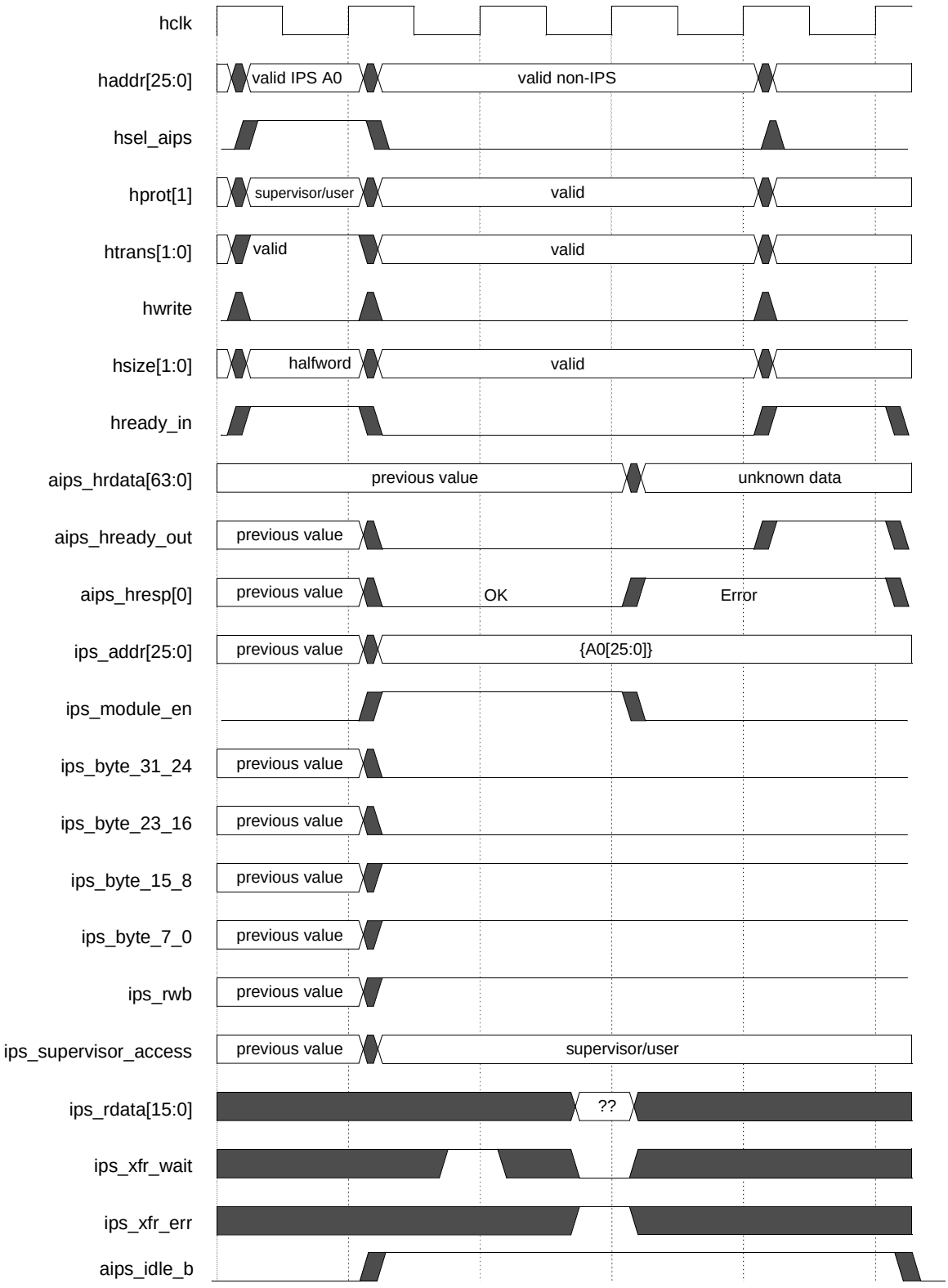


Figure 38-52. Read With Error, One IPS Waitstate

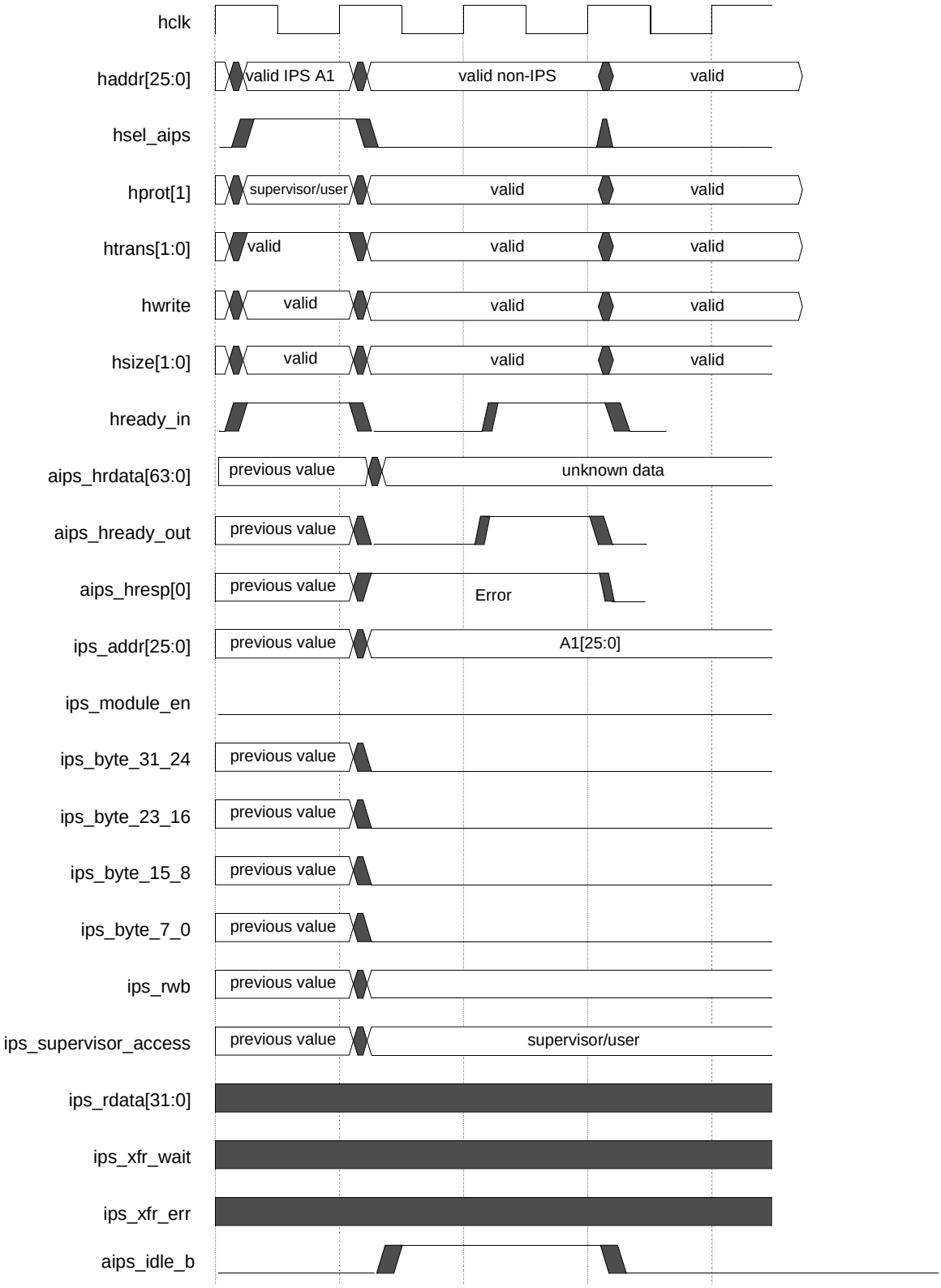


Figure 38-53. Permission Violation Error

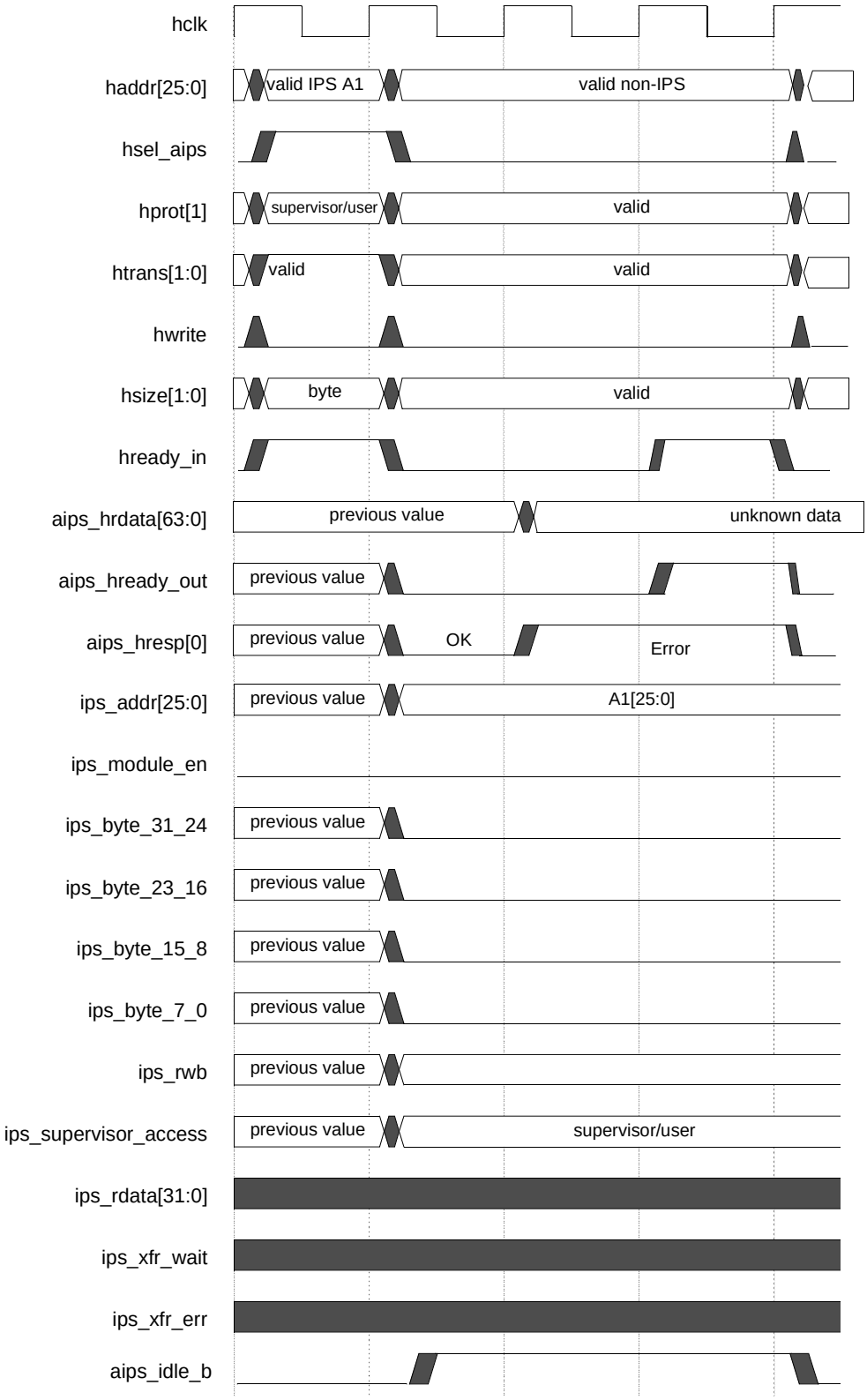


Figure 38-54. Permission Violation Error, AIPS_DLY_CYCLE = 1

38.5 Initialization/Application Information

This section provides initialization and application information for the AIPS.

38.5.1 Software Restrictions

The only AIPS requirement is to initialize the Master Privilege Registers (MPRs), the Peripheral Access Control registers (PACRs), and the Off-platform Peripheral Access Control registers (OPACRs) described in Section 38.3.3.3, “Off-Platform Peripheral Access Control Registers (OPACR_1, OPACR_2, OPACR_3, OPACR_4, and OPACR_5).”

Chapter 39
Multi-Layer AHB Crossbar Switch (MAX)

This chapter provides an overview of the Multi-Layer AHB Crossbar Switch (MAX). The purpose of the MAX is to concurrently support up to five simultaneous connections between master ports and slave ports. The MAX supports a 32-bit address bus width, and a 32-bit data bus width at all master and slave ports. A simplified block diagram is shown in Figure 39-1.

NOTE

The ARM11 Platform implements a 6 master x 5 slave configuration.

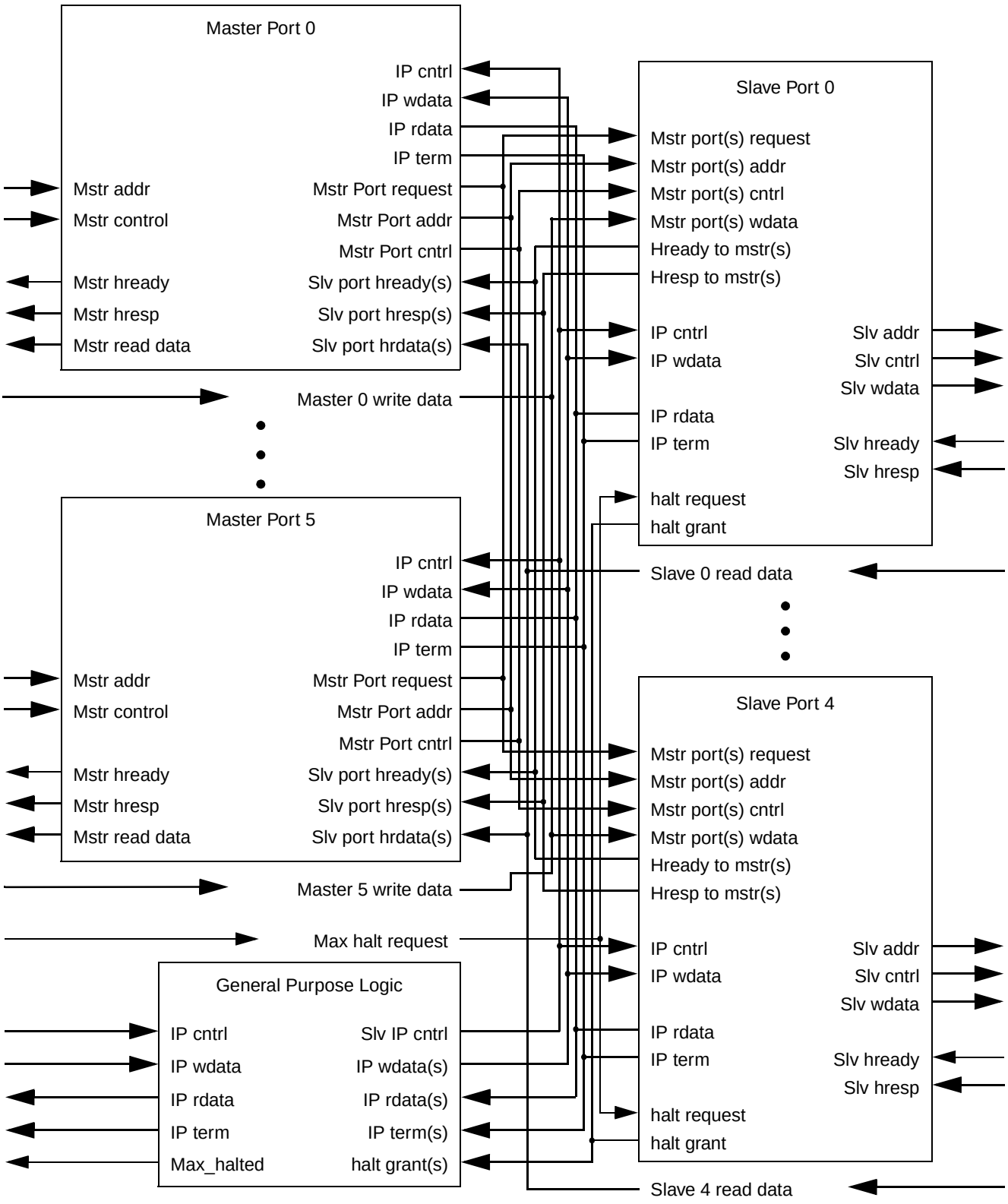


Figure 39-1. MAX Block Diagram

39.1 Features

The MAX has the ability to gain control of all the ARM11 slave ports and prevent any masters from making accesses to the slave ports. This feature is useful to turn off the clocks to the system and to ensure that no bus activity is interrupted.

The MAX can put each slave port into a low power park mode so that slave port will not dissipate any power transitioning address, control or data signals when not being actively accessed by a master port.

Each slave port can also support multiple master priority schemes. Each slave port has a hardware input which selects the master priority scheme so the user can dynamically change master priority levels on a slave port by slave port basis.

The MAX allows for concurrent transactions to occur from any master port to any slave port. It is possible for four master ports and all slave ports to be in use at the same time as a result of independent master requests. If a slave port is simultaneously requested by more than one master port, arbitration logic will select the higher priority master and grant it ownership of the slave port. All other masters requesting that slave port will stalled until the higher priority master completes its transactions.

39.1.1 Limitations

The MAX routes bus transactions initiated on the master ports to the appropriate slave ports. There is no provision included to route transactions initiated on the slave ports to other slave ports or to master ports. Simply put, the slave ports do not support the bus request/bus grant protocol, the MAX assumes it is the sole master of each slave port.

Since the MAX does not support the bus request/bus grant protocol, if multiple masters are to be connected to a single master port an external arbiter will need to be used, this can be seen in Figure 39-12. In the case of a single master connecting to a master port the single master’s bus grant signal must be tied off in the asserted state, this can be seen in Figure 39-13.

Each master and slave port is fully AHB-Lite + AMBA V6 extensions compliant. The ports are not fully AHB compliant because the MAX does not support SPLITs or RETRYs.

39.1.2 General Operation

When a master makes an access to the MAX the access will be immediately taken by the MAX. If the targeted slave port of the access is available then the access will be immediately presented on the slave port. It is possible to make single clock (zero wait state) accesses through the MAX. If the targeted slave port of the access is busy or parked on a different master port the requesting master will simply see wait states inserted (hready held negated) until the targeted slave port can service the master’s request. The latency in servicing the request will depend on each master’s priority level and the responding peripheral’s access time.

Because the MAX appears to be just another slave to the master device, the master device will have no knowledge of whether or not it actually owns the slave port it is targeting. While the master does not have control of the slave port it is targeting it will be wait-stated.

A master is given control of the targeted slave port only after a previous access to a different slave port has completed, regardless of its priority on the newly targeted slave port. This prevents deadlock from occurring when a master has an outstanding request to one slave port that has a long response time, has a pending access to a different slave port, and a lower priority master is also making a request to the same slave port as the pending access of the higher priority master.

Once the master has control of the slave port it is targeting the master will remain in control of that slave port until it gives up the slave port by running an IDLE cycle or by leaving that slave port for its next access. The master could also lose control of the slave port if another higher priority master makes a request to the slave port; however, if the master is running a locked or fixed length burst transfer it will retain control of the slave port until that transfer is completed. Based on the AULB bit in the MGPCR (Master General Purpose Control Register) the master will either retain control of the slave port when doing undefined length incrementing burst transfers or will lose the bus to a higher priority master.

The MAX terminates all master IDLE transfers (as opposed to allowing the termination to come from one of the slave buses). Additionally, when no master is requesting access to a slave port the MAX drives IDLE transfers onto the slave bus, even though a default master may be granted access to the slave port. When the MAX is controlling the slave bus (that is, during low power park or halt mode) the hmaster field indicates 0000.

When a slave bus is being IDLEd by the MAX it can park the slave port on the master port indicated by the PARK bits in the SGPCR (Slave General Purpose Control Register). This can be done in an attempt to save the initial clock of arbitration delay that would otherwise be seen if the master had to arbitrate to gain control of the slave port. The slave port can also be put into low power park mode in attempt to save power.

39.2 MAX Interface Signals

This section provides information on MAX interface signals, including AHB master and slave interface signals as well as IP bus interface signals.

39.2.1 MAX Signal Overview

Table 39-1 provides an overview of MAX module interface signals.

Table 39-1. MAX Signals

Signal	Type	Description
System Signals		
hclk	Input	System clock
$\overline{\text{hreset}}$	Input	System reset
max_halt_request	Input	Transfer lockdown request
max_halted	Output	All transfers halted
Master Port Interface Signals		
m0_hlock	Input	Master port 0 AHB locked transfer signal.
m0_hmastlock	Input	Master port 0 AHB locked transfer signal

Table 39-1. MAX Signals (continued)

Signal	Type	Description
m0_hmaster[3:0]	Input	Master port 0 AHB master identification
m0_htrans[1:0]	Input	Master port 0 AHB transfer type
m0_hprot[3:0]	Input	Master port 0 AHB protection control
m0_hwrite	Input	Master port 0 AHB transfer direction
m0_hsize[1:0]	Input	Master port 0 AHB transfer size
m0_hburst[2:0]	Input	Master port 0 AHB burst type
m0_haddr[31:0]	Input	Master port 0 AHB address bus
m0_hwdata[31:0]	Input	Master port 0 AHB write data bus
m0_hrdata[31:0]	Output	Master port 0 AHB read data bus
m0_hready_out	Output	Master port 0 AHB transfer done out
m0_hresp0	Output	Master port 0 AHB response
m1_hlock	Input	Master port 1 AHB locked transfer signal.
m1_hmastlock	Input	Master port 1 AHB locked transfer signal
m1_hmaster[3:0]	Input	Master port 1 AHB master identification
m1_htrans[1:0]	Input	Master port 1 AHB transfer type
m1_hprot[3:0]	Input	Master port 1 AHB protection control
m1_hwrite	Input	Master port 1 AHB transfer direction
m1_hsize[1:0]	Input	Master port 1 AHB transfer size
m1_hburst[2:0]	Input	Master port 1 AHB burst type
m1_haddr[31:0]	Input	Master port 1 AHB address bus
m1_hwdata[31:0]	Input	Master port 1 AHB write data bus
m1_hrdata[31:0]	Output	Master port 1 AHB read data bus
m1_hready_out	Output	Master port 1 AHB transfer done out
m1_hresp0	Output	Master port 1 AHB response
m2_hlock	Input	Master port 2 AHB locked transfer signal.
m2_hmastlock	Input	Master port 2 AHB locked transfer signal
m2_hmaster[3:0]	Input	Master port 2 AHB master identification
m2_htrans[1:0]	Input	Master port 2 AHB transfer type
m2_hprot[3:0]	Input	Master port 2 AHB protection control
m2_hwrite	Input	Master port 2 AHB transfer direction
m2_hsize[1:0]	Input	Master port 2 AHB transfer size
m2_hburst[2:0]	Input	Master port 2 AHB burst type
m2_haddr[31:0]	Input	Master port 2 AHB address bus

Table 39-1. MAX Signals (continued)

Signal	Type	Description
m2_hwdata[31:0]	Input	Master port 2 AHB write data bus
m2_hrdata[31:0]	Output	Master port 2 AHB read data bus
m2_hready_out	Output	Master port 2 AHB transfer done out
m2_hresp0	Output	Master port 2 AHB response
m3_hlock	Input	Master port 3 AHB locked transfer signal.
m3_hmastlock	Input	Master port 3 AHB locked transfer signal
m3_hmaster[3:0]	Input	Master port 3 AHB master identification
m3_htrans[1:0]	Input	Master port 3 AHB transfer type
m3_hprot[3:0]	Input	Master port 3 AHB protection control
m3_hwrite	Input	Master port 3 AHB transfer direction
m3_hsize[1:0]	Input	Master port 3 AHB transfer size
m3_hburst[2:0]	Input	Master port 3 AHB burst type
m3_haddr[31:0]	Input	Master port 3 AHB address bus
m3_hwdata[31:0]	Input	Master port 3 AHB write data bus
m3_hrdata[31:0]	Output	Master port 3 AHB read data bus
m3_hready_out	Output	Master port 3 AHB transfer done out
m3_hresp0	Output	Master port 3 AHB response
m4_hlock	Input	Master port 4 AHB locked transfer signal.
m4_hmastlock	Input	Master port 4 AHB locked transfer signal
m4_hmaster[3:0]	Input	Master port 4 AHB master identification
m4_htrans[1:0]	Input	Master port 4 AHB transfer type
m4_hprot[3:0]	Input	Master port 4 AHB protection control
m4_hwrite	Input	Master port 4 AHB transfer direction
m4_hsize[1:0]	Input	Master port 4 AHB transfer size
m4_hburst[2:0]	Input	Master port 4 AHB burst type
m4_haddr[31:0]	Input	Master port 4 AHB address bus
m4_hwdata[31:0]	Input	Master port 4 AHB write data bus
m4_hrdata[31:0]	Output	Master port 4 AHB read data bus
m4_hready_out	Output	Master port 4 AHB transfer done out
m4_hresp0	Output	Master port 4 AHB response
m5_hlock	Input	Master port 5 AHB locked transfer signal.
m5_hmastlock	Input	Master port 5 AHB locked transfer signal
m5_hmaster[3:0]	Input	Master port 5 AHB master identification

Table 39-1. MAX Signals (continued)

Signal	Type	Description
m5_htrans[1:0]	Input	Master port 5 AHB transfer type
m5_hprot[3:0]	Input	Master port 5 AHB protection control
m5_hwrite	Input	Master port 5 AHB transfer direction
m5_hsize[1:0]	Input	Master port 5 AHB transfer size
m5_hburst[2:0]	Input	Master port 5 AHB burst type
m5_haddr[31:0]	Input	Master port 5 AHB address bus
m5_hwdata[31:0]	Input	Master port 5 AHB write data bus
m5_hrdata[31:0]	Output	Master port 5 AHB read data bus
m5_hready_out	Output	Master port 5 AHB transfer done out
m5_hresp0	Output	Master port 5 AHB response
Slave Port Interface Signals		
s0_hrdata[31:0]	Input	Slave port 0 AHB read data bus
s0_hready	Input	Slave port 0 AHB transfer done
s0_hresp0	Input	Slave port 0 AHB response
s0_hmastlock	Output	Slave port 0 AHB locked transfer signal
s0_hmaster[3:0]	Output	Slave port 0 AHB master identification
s0_htrans[1:0]	Output	Slave port 0 AHB transfer type
s0_hprot[3:0]	Output	Slave port 0 AHB protection control
s0_hwrite	Output	Slave port 0 AHB transfer direction
s0_hsize[1:0]	Output	Slave port 0 AHB transfer size
s0_hburst[2:0]	Output	Slave port 0 AHB burst type
s0_haddr[31:0]	Output	Slave port 0 AHB address bus
s0_hwdata[31:0]	Output	Slave port 0 AHB write data bus
s1_hrdata[31:0]	Input	Slave port 1 AHB read data bus
s1_hready	Input	Slave port 1 AHB transfer done
s1_hresp0	Input	Slave port 1 AHB response
s1_hmastlock	Output	Slave port 1 AHB locked transfer signal
s1_hmaster[3:0]	Output	Slave port 1 AHB master identification
s1_htrans[1:0]	Output	Slave port 1 AHB transfer type
s1_hprot[3:0]	Output	Slave port 1 AHB protection control
s1_hwrite	Output	Slave port 1 AHB transfer direction
s1_hsize[1:0]	Output	Slave port 1 AHB transfer size

Table 39-1. MAX Signals (continued)

Signal	Type	Description
s1_hburst[2:0]	Output	Slave port 1 AHB burst type
s1_haddr[31:0]	Output	Slave port 1 AHB address bus
s1_hwdata[31:0]	Output	Slave port 1 AHB write data bus
s2_hrdata[31:0]	Input	Slave port 2 AHB read data bus
s2_hready	Input	Slave port 2 AHB transfer done
s2_hresp0	Input	Slave port 2 AHB response
s2_hmastlock	Output	Slave port 2 AHB locked transfer signal
s2_hmaster[3:0]	Output	Slave port 2 AHB master identification
s2_htrans[1:0]	Output	Slave port 2 AHB transfer type
s2_hprot[3:0]	Output	Slave port 2 AHB protection control
s2_hwrite	Output	Slave port 2 AHB transfer direction
s2_hsize[1:0]	Output	Slave port 2 AHB transfer size
s2_hburst[2:0]	Output	Slave port 2 AHB burst type
s2_haddr[31:0]	Output	Slave port 2 AHB address bus
s2_hwdata[31:0]	Output	Slave port 2 AHB write data bus
s3_hrdata[31:0]	Input	Slave port 3 AHB read data bus
s3_hready	Input	Slave port 3 AHB transfer done
s3_hresp0	Input	Slave port 3 AHB response
s3_hmastlock	Output	Slave port 3 AHB locked transfer signal
s3_hmaster[3:0]	Output	Slave port 3 AHB master identification
s3_htrans[1:0]	Output	Slave port 3 AHB transfer type
s3_hprot[3:0]	Output	Slave port 3 AHB protection control
s3_hwrite	Output	Slave port 3 AHB transfer direction
s3_hsize[1:0]	Output	Slave port 3 AHB transfer size
s3_hburst[2:0]	Output	Slave port 3 AHB burst type
s3_haddr[31:0]	Output	Slave port 3 AHB address bus
s3_hwdata[31:0]	Output	Slave port 3 AHB write data bus
s4_hrdata[31:0]	Input	Slave port 4 AHB read data bus
s4_hready	Input	Slave port 4 AHB transfer done
s4_hresp0	Input	Slave port 4 AHB response
s4_hmastlock	Output	Slave port 4 AHB locked transfer signal
s4_hmaster[3:0]	Output	Slave port 4 AHB master identification
s4_htrans[1:0]	Output	Slave port 4 AHB transfer type

Table 39-1. MAX Signals (continued)

Signal	Type	Description
s4_hprot[3:0]	Output	Slave port 4 AHB protection control
s4_hwrite	Output	Slave port 4 AHB transfer direction
s4_hsize[1:0]	Output	Slave port 4 AHB transfer size
s4_hburst[2:0]	Output	Slave port 4 AHB burst type
s4_haddr[31:0]	Output	Slave port 4 AHB address bus
s4_hwdata[31:0]	Output	Slave port 4 AHB write data bus
IP Bus v2.0 Interface Signals		
ipg_clk	Input	IP system clock
ips_addr[11:2]	Input	IP address bus
ips_supervisor_access	Input	Supervisor mode access signal
ips_wdata[31:0]	Input	IP write data bus
ips_module_en	Input	Module enable bus
ips_rwb	Input	Read/write access signal
ips_byte_31_24, ips_byte_23_16, ips_byte_15_8, ips_byte_7_0	Input	Byte enables
ips_xfr_wait	Output	IP bus peripheral wait signal
ips_xfr_err	Output	IP bus peripheral error signal
ips_rdata[31:0]	Output	IP read data bus

39.2.2 MAX Signal Descriptions

Refer to the AMBA Specification Rev 2.0 for a description of the AHB signals in the MAX and the IP Bus Specification Rev 2.0 for a description of the IP Bus signals in the MAX.

39.2.2.1 max_halt_request

This input signal is a request to halt all slave port bus activity (run MAX originated IDLE cycles on each slave port bus, blocking all master port accesses). This signal can be used to gracefully shut down the MAX so the system clock can be stopped for low power mode. This signal is captured by a flop inside the MAX before use.

Once the MAX is halted it remains halted until max_halt_request is negated.

39.2.2.2 max_halted

This output is asserted once the MAX is in control and running IDLE cycles on each slave port.

39.3 Memory Map and Register Definition

There are four registers that reside in each slave port of the MAX and one register that resides in each master port of the MAX. These registers are IP bus compliant registers. Read and write transfers both require two IP bus clock cycles. The registers can only be read from and written to in supervisor mode. Additionally, these registers can only be read from or written to by 32-bit accesses. Section 39.3.3, "MAX Register Descriptions" provides the detailed descriptions for all of the MAX registers.

The registers are fully decoded and an error response is returned if an unimplemented location is accessed within the MAX.

The slave registers also feature a bit, which when written with a 1, will prevent the registers from being written to again. The registers will still be readable, but future write attempts will have no effect on the registers and will be terminated with an error response.

39.3.1 Memory Map

Table 39-2 shows the MAX memory map.

Table 39-2. MAX Memory Map

Address	Register	Access	Reset Value	Section/Page
0x43F0_4000 (MPR0)	Master Priority Register for Slave port 0	R/W	0x0054_3210	39.3.3.1/39-14
0x43F0_4010 (SGPCR0)	General Purpose Control Register for Slave port 0	R/W	0x0000_0000	39.3.3.2/39-15
0x43F0_4100 (MPR1)	Master Priority Register for Slave port 1	R/W	0x0054_3210	39.3.3.1/39-14
0x43F0_4110 (SGPCR1)	General Purpose Control Register for Slave port 1	R/W	0x0000_0000	39.3.3.2/39-15
0x43F0_4200 (MPR2)	Master Priority Register for Slave port 2	R/W	0x0054_3210	39.3.3.1/39-14
0x43F0_4210 (SGPCR2)	General Purpose Control Register for Slave port 2	R/W	0x0000_0000	39.3.3.2/39-15
0x43F0_4300 (MPR3)	Master Priority Register for Slave port 3	R/W	0x0054_3210	39.3.3.1/39-14
0x43F0_4310 (SGPCR3)	General Purpose Control Register for Slave port 3	R/W	0x0000_0000	39.3.3.2/39-15
0x43F0_4400 (MPR4)	Master Priority Register for Slave port 4	R/W	0x0054_3210	39.3.3.1/39-14
0x43F0_4410 (SGPCR4)	General Purpose Control Register for Slave port 4	R/W	0x0000_0000	39.3.3.2/39-15
0x43F0_4800 (MGPCR0)	General Purpose Control Register for Master port 0	R/W	0x0000_0000	39.3.3.3/39-17
0x43F0_4900 (MGPCR1)	General Purpose Control Register for Master port 1	R/W	0x0000_0000	39.3.3.3/39-17

Table 39-2. MAX Memory Map (continued)

Address	Register	Access	Reset Value	Section/Page
0x43F0_4A00 (MGPCR2)	General Purpose Control Register for Master port 2	R/W	0x0000_0000	39.3.3.3/39-17
0x43F0_4B00 (MGPCR3)	General Purpose Control Register for Master port 3	R/W	0x0000_0000	39.3.3.3/39-17
0x43F0_4C00 (MGPCR4)	General Purpose Control Register for Master port 4	R/W	0x0000_0000	39.3.3.3/39-17
0x43F0_4D00 (MGPCR5)	General Purpose Control Register for Master port 5	R/W	0x0000_0000	39.3.3.3/39-17

39.3.2 Register Summary

Figure 39-2 shows the key to the register fields, and Table 39-3 shows the register figure conventions.

Figure 39-2. Key to Register Fields

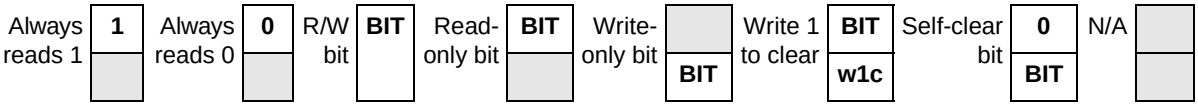


Table 39-3. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writeable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 39-4 shows the MAX register summary.

Table 39-4. MAX Detailed Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x43F0_4000 (MPR0)	R	0	0	0	0	0	0	0	0	0	MSTR_5			0	MSTR_4		
	W																
	R	0	MSTR_3			0	MSTR_2			0	MSTR_1			0	MSTR_0		
	W																
0x43F0_4100 (MPR1)	R	0	0	0	0	0	0	0	0	0	MSTR_5			0	MSTR_4		
	W																
	R	0	MSTR_3			0	MSTR_2			0	MSTR_1			0	MSTR_0		
	W																
0x43F0_4200 (MPR2)	R	0	0	0	0	0	0	0	0	0	MSTR_5			0	MSTR_4		
	W																
	R	0	MSTR_3			0	MSTR_2			0	MSTR_1			0	MSTR_0		
	W																
0x43F0_4300 (MPR3)	R	0	0	0	0	0	0	0	0	0	MSTR_5			0	MSTR_4		
	W																
	R	0	MSTR_3			0	MSTR_2			0	MSTR_1			0	MSTR_0		
	W																
0x43F0_4400 (MPR4)	R	0	0	0	0	0	0	0	0	0	MSTR_5			0	MSTR_4		
	W																
	R	0	MSTR_3			0	MSTR_2			0	MSTR_1			0	MSTR_0		
	W																
0x43F0_4010 (SGPCR0)	R	RO	HLP	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	ARB		0	0	PCTL		0	PARK		
	W																
0x43F0_4110 (SGPCR1)	R	RO	HLP	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	ARB		0	0	PCTL		0	PARK		
	W																
0x43F0_4210 (SGPCR2)	R	RO	HLP	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	ARB		0	0	PCTL		0	PARK		
	W																

Table 39-4. MAX Detailed Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x43F0_4310 (SGPCR3)	R	RO	HLP	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	ARB		0	0	PCTL		0	PARK		
	W																
0x43F0_4410 (SGPCR4)	R	RO	HLP	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	ARB		0	0	PCTL		0	PARK		
	W																
0x43F0_4800 (MGPCR0)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	AULB		
	W																
0x43F0_4900 (MGPCR1)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	AULB		
	W																
0x43F0_4A00 (MGPCR2)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	AULB		
	W																
0x43F0_4B00 (MGPCR3)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	AULB		
	W																
0x43F0_4C00 (MGPCR4)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	AULB		
	W																
0x43F0_4D00 (MGPCR5)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	AULB		
	W																

39.3.3 MAX Register Descriptions

This section contains the detailed register descriptions for the MAX registers.

39.3.3.1 Master Priority Register (MPR0–MPR4)

The Master Priority Register (MPR) sets the priority of each master port on a per slave port basis and resides in each slave port. See Figure 39-5 for an illustration of valid bits in the MPR, and Table 39-6 for its field descriptions.

0x43F0_4000 (MPR0)
0x43F0_4100 (MPR1)
0x43F0_4200 (MPR2)
0x43F0_4300 (MPR3)
0x43F0_4400 (MPR4)

Access: Supervisor Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	MSTR_5			0	MSTR_4		
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	MSTR_3			0	MSTR_2			0	MSTR_1			0	MSTR_0		
W																
Reset	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0

Table 39-5. Master Priority Register (MPR0–MPR4)

Table 39-6. Master Priority Register Descriptions

Field	Description
31–23	Reserved. They are read as zero and should be written with zero for upward compatibility.
22–20 MSTR_5	Master 5 Priority. These bits set the arbitration priority for master port 5 on the associated slave port. These bits are initialized by hardware reset. 000 This master has the highest priority when accessing the slave port. 111 This master has the lowest priority when accessing the slave port.
19	Reserved. They are read as zero and should be written with zero for upward compatibility.
18–16 MSTR_4	Master 4 Priority. These bits set the arbitration priority for master port 4 on the associated slave port. These bits are initialized by hardware reset. 000 This master has the highest priority when accessing the slave port. 111 This master has the lowest priority when accessing the slave port.
15	Reserved. They are read as zero and should be written with zero for upward compatibility.
14–12 MSTR_3	Master 3 Priority. These bits set the arbitration priority for master port 3 on the associated slave port. These bits are initialized by hardware reset. 000 This master has the highest priority when accessing the slave port. 111 This master has the lowest priority when accessing the slave port.
11	Reserved. They are read as zero and should be written with zero for upward compatibility.

Table 39-6. Master Priority Register Descriptions (continued)

Field	Description
10–8 MSTR_2	Master 2 Priority. These bits set the arbitration priority for master port 2 on the associated slave port. These bits are initialized by hardware reset. 000 This master has the highest priority when accessing the slave port. 111 This master has the lowest priority when accessing the slave port.
7	Reserved**.** They are read as zero and should be written with zero for upward compatibility.
6–4 MSTR_1	Master 1 Priority. These bits set the arbitration priority for master port 1 on the associated slave port. These bits are initialized by hardware reset. 000 This master has the highest priority when accessing the slave port. 111 This master has the lowest priority when accessing the slave port.
3	Reserved**.** They are read as zero and should be written with zero for upward compatibility.
2–0 MSTR_0	Master 0 Priority. These bits set the arbitration priority for master port 0 on the associated slave port. These bits are initialized by hardware reset. 000 This master has the highest priority when accessing the slave port. 111 This master has the lowest priority when accessing the slave port.

The Master Priority Register can only be accessed in supervisor mode with 32-bit accesses. Once the RO (Read Only) bit has been set in the Slave General Purpose Control Register the Master Priority Register can only be read from, attempts to write to it will have no effect on the MPR and result in an error response.

Additionally, no two available master ports may be programmed with the same priority level. Attempts to program two or more available masters with the same priority level will result in an error response and the MPR will not be updated.

39.3.3.2 Slave General Purpose Control Register (SGPCR0–SGPCR4)

The Slave General Purpose Control Register (SGPCR) controls several features of each slave port.

The Read Only (RO) bit will prevent any registers associated with this slave port from being written to once set. This bit may be written with 0 as many times as the user desires, but once it is written to a 1 only a reset condition will allow it to be written again.

The Halt Low Priority (HLP) bit will set the priority of the max_halt_request input to the lowest possible priority for initial arbitration of the slave ports. By default it is the highest priority.

NOTE

Setting this bit will not effect the max_halt_request from attaining highest priority once it has control of the slave ports.

The PCTL bits determine how the slave port will park when no master is actively making a request. The available options are to park on the master defined by the PARK bits, park on the last master to use the slave port, or go into a low power park mode which will force all the outputs of the slave port to inactive states when no master is requesting an access. The low power park feature can result in an overall power savings if a the slave port is not saturated; however, it will force an extra clock of latency whenever any master tries to access it when it is not in use because it will not be parked on any master.

The PARK bits determine which master the slave will park on when no master is making an active request and the max_halt_request input is negated.

CAUTION

Only select master ports that are actually present in the design. If you program the PARK bits to a master not present in the current design implementation undefined behavior will result.

See Figure 39-3 for an illustration of valid bits in the SGPCR, and Table 39-7 for its field descriptions.

0x43F0_4010 (SGPCR0)
0x43F0_4110 (SGPCR1)
0x43F0_4210 (SGPCR2)
0x43F0_4310 (SGPCR3)
0x43F0_4410 (SGPCR4)

Access: Supervisor Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RO	HLP	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	ARB		0	0	PCTL		0	PARK		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 39-3. Slave General Purpose Control Register n

Table 39-7. Slave General Purpose Control Register Descriptions

Field	Description
31 RO	Read Only. This bit is used to force all of a slave port's registers to be read only. Once written to 1 it can only be cleared by hardware reset. This bit is initialized by hardware reset. 0 All of this slave port's registers can be written. 1 All of this slave port's registers are read only and cannot be written (attempted writes have no effect and result in an error response).
30 HLP	Halt Low Priority**.** This bit is used to set the initial arbitration priority of the max_halt_request input. This bit is initialized by hardware reset. 0 The max_halt_request input has the highest priority for arbitration on this slave port 1 The max_halt_request input has the lowest initial priority for arbitration on this slave port.
29–10	Reserved**.** They read as zero and should be written with zero for upward compatibility.
9–8 ARB	Arbitration Mode. These bits are used to select the arbitration policy for the slave port. These bits are initialized by hardware reset. 00 Fixed Priority. 01 Round Robin (rotating) Priority 10 Reserved 11 Reserved

Table 39-7. Slave General Purpose Control Register Descriptions (continued)

Field	Description
7–6	Reserved. They read as zero and should be written with zero for upward compatibility.
5–4 PCTL	Parking Control.These bits determine the parking control used by this slave port. These bits are initialized by hardware reset. 00 When no master is making a request the arbiter will park the slave port on the master port defined by the PARK bit field. 01 When no master is making a request the arbiter will park the slave port on the last master to be in control of the slave port. 10 When no master is making a request the arbiter will park the slave port on no master and will drive all outputs to a constant safe state. 11 Reserved
3	Reserved**.** They read as zero and should be written with zero for upward compatibility.
2–0 PARK	Park. These bits are used to determine which master port this slave port parks on when no masters are actively making requests and the PCTL bits are set to 00. These bits are initialized by hardware reset. 000 Park on Master Port 0 001 Park on Master Port 1 010 Park on Master Port 2 011 Park on Master Port 3 100 Park on Master Port 4 101 Park on Master Port 5 110 Reserved 111 Reserved

The SGPCR can only be accessed in supervisor mode with 32-bit accesses. Once the RO (Read-Only) bit has been set in the SGPCR the SGPCR can only be read, attempts to write to it will have no effect on the SGPCR and result in an error response.

39.3.3.3 Master General Purpose Control Register

The Master General Purpose Control Register (MGPCR) presently controls only whether or not the master's undefined length burst accesses will be allowed to complete uninterrupted or whether they can be broken by requests from higher priority masters.

The AULB (Arbitrate on Undefined Length Bursts) bit field determines whether (and when) or not the MAX will arbitrate away the slave port the master owns when the master is performing undefined length burst accesses.

See Figure 39-4 for an illustration of valid bits in the MGPCR, and Table 39-8 for its field descriptions.

0x43F0_4800 (MGPCR0)
0x43F0_4900 (MGPCR1)
0x43F0_4A00 (MGPCR2)
0x43F0_4B00 (MGPCR3)
0x43F0_4C00 (MGPCR4)
0x43F0_4D00 (MGPCR5)

Access: Supervisor Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	AULB		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 39-4. Master General Purpose Control Register n

Table 39-8. Master General Purpose Control Register Descriptions

Name	Description
31–3	Reserved**.** They read as zero and should be written with zero for upward compatibility.
2–0 AULB	Arbitrate on Undefined Length Bursts**.** These bits are used to select the arbitration policy during undefined length bursts by this master. These bits are initialized by hardware reset. 000 No arbitration will be allowed during an undefined length burst. 001 Arbitration will be allowed at any time during an undefined length burst. 010 Arbitration will be allowed after four beats of an undefined length burst. 011 Arbitration will be allowed after eight beats of an undefined length burst. 100 Arbitration will be allowed after 16 beats of an undefined length burst. 101 Reserved 110 Reserved 111 Reserved

The MGPCR can only be accessed in supervisor mode with 32-bit accesses.

39.3.4 Coherency

Since the content of the registers has a real time effect on the operation of the MAX it is important for the user to understand that any register modifications take effect as soon as the register is written. The values of the registers do not track with slave port related AHB accesses but instead track only with IP bus accesses.

The exception to this rule are the AULB bits in the MGPCR. The update of these bits is only recognized when the master on that master port runs an IDLE cycle, even though the IP bus cycle to write them will have long since terminated successfully. If the AULB bits in the MGPCR are written in between two burst

accesses the new AULB encodings will not take effect until an IDLE cycle has been initiated by the master on that master port.

39.4 Detailed Functional Description

This section describes the functionality of the MAX in greater detail.

39.4.1 Arbitration

The MAX supports two arbitration schemes; a simple fixed-priority comparison algorithm, and a simple round-robin fairness algorithm. The arbitration scheme is independently programmable for each slave port.

39.4.1.1 Arbitration During Undefined Length Bursts

Arbitration points during an undefined length burst are defined by the current master's MGPCR AULB field setting. When a defined length is imposed on the burst via the AULB bits the undefined length burst will be treated as a single or series of single back to back fixed length burst accesses.

Example: A master runs an undefined length burst and the AULB bits in the MGPCR indicate arbitration will occur after the fourth beat of the burst. The master runs two sequential beats and then starts what will be an 12 beat undefined length burst access to a new address within the same slave port region as the previous access. The MAX will not allow an arbitration point until the fourth overall access (second beat of the second burst). At that point all remaining accesses will be open for arbitration until the master loses control of the slave port.

Assume the master loses control of the slave port after the fifth beat of the second burst. Once the master regains control of the slave port no arbitration point will be available until after the master has run four more beats of its burst. After the fourth beat of the (now continued) burst (ninth beat of the second burst from the master's perspective) is taken all beats of the burst will once again be open for arbitration until the master loses control of the slave port.

Assume the master again loses control of the slave port on the fifth beat of the third (now continued) burst (10th beat of the second burst from the master's perspective). Once the master regains control of the slave port it will be allowed to complete its final two beats of its burst without facing arbitration.

Note that fixed length burst accesses will not be affected by the AULB bits. All fixed length burst accesses will lock out arbitration until the last beat of the fixed length burst.

39.4.1.2 Fixed Priority Operation

When operating in fixed-priority mode, each master is assigned a unique priority level in the MPR (Master Priority Register). If two masters both request access to a slave port the master with the highest priority in the selected priority register will gain control over the slave port.

Any time a master makes a request to a slave port the slave port checks to see if the new requesting master's priority level is higher than that of the master that currently has control over the slave port (unless the slave

port is in a parked state). The slave port does an arbitration check at every clock edge to ensure that the proper master (if any) has control of the slave port.

If the new requesting master's priority level is higher than that of the master that currently has control of the slave port the new requesting master will be granted control over the slave port at the next clock edge. The exception to this rule is if the master that currently has control over the slave port is running a fixed length burst transfer or a locked transfer. In this case the new requesting master will have to wait until the end of the burst transfer or locked transfer before it will be granted control of the slave port. If the master is running an undefined length burst transfer the new requesting master must wait until an arbitration point for the undefined length burst transfer before it will be granted control of the slave port. Arbitration points for an undefined length burst are defined in the MGPCR for each master.

If the new requesting master's priority level is lower than that of the master that currently has control of the slave port the new requesting master will be forced to wait until the master that currently has control of the slave port either runs an IDLE cycle or runs a non IDLE cycle to a location other than the current slave port.

39.4.1.3 Round-Robin Priority Operation

When operating in round-robin mode, each master is assigned a relative priority based on the master number. This relative priority is compared to the ID of the last master to perform a transfer on the slave bus. The highest priority requesting master will become owner of the slave bus as the next transfer boundary (accounting for locked and fixed-length burst transfers). Priority is based on how far ahead the ID of the requesting master is to the ID of the last master (ID is defined by master port number, not the hmaster field).

Once granted access to a slave port, a master may perform as many transfers as desired to that port until another master makes a request to the same slave port. The next master in line will be granted access to the slave port at the next assertion of sX_hready, or possibly on the next clock cycle if the current master has no pending access request.

As an example of arbitration in round-robin mode, assume the MAX is implemented with master ports 0, 1, 2, 3, 4 and 5. If the last master of the slave port was master 1, and master 0, 4 and 5 make simultaneous requests, (master ports 2 and 3 make no requests), they will be serviced in the order 4, 5 and then 0.

Parking may still be used in a round-robin mode, but will not affect the round-robin pointer unless the parked master actually performs a transfer. Handoff will occur to the next master in line after one cycle of arbitration. If the slave port is put into low power park mode the round-robin pointer will be reset to point at master port 0, giving it the highest priority.

39.4.2 Priority Assignment

Each master port needs to be assigned a unique 3 bit priority level. If an attempt is made to program multiple master ports with the same priority level within a register (MPR) the MAX will respond with an error and the registers will not be updated.

39.4.3 Master Port Functionality

39.4.3.1 General

Each master port consists of two decoders, a capture unit, a register slice, a MUX, and a small state machine.

The first decoder is used to decode the haddr and control signals coming directly from the master, telling the state machine where the master's next access will be and if it is in fact a legal access. The second decoder gets its input from the capture unit, so it may be looking directly at the signals coming from the master or it may be looking at captured signals coming from the master, depending entirely on the state of the targeted slave port. The second decoder is then used to generate the access requests that go to the slave ports.

The capture unit is used to capture the address and control information coming from the master in the event that the targeted slave port cannot immediately service the master. The capture unit is controlled by outputs from the state machine which tell it to either pass through the original master signals or the captured signals.

The register slice contains the registers associated with the specific master port. The registers have a quasi-IP bus interface at this level for reads and writes and the outputs feed directly into the state machine.

The MUX is used simply to select which slave's read data is sent back to the master. The MUX is controlled by the state machine.

The state machine controls all aspects of the master port. It knows which slave port the master wants to make a request to and controls when that request is made. It also has knowledge of each slave port, knowing whether or not the slave port is ready to accept an access from the master port. This will determine whether or not the master may immediately have its request taken by the slave port or whether the master port will have to capture the master's request and queue it at the slave port boundary.

Figure 39-5 shows a block diagram of a master port.

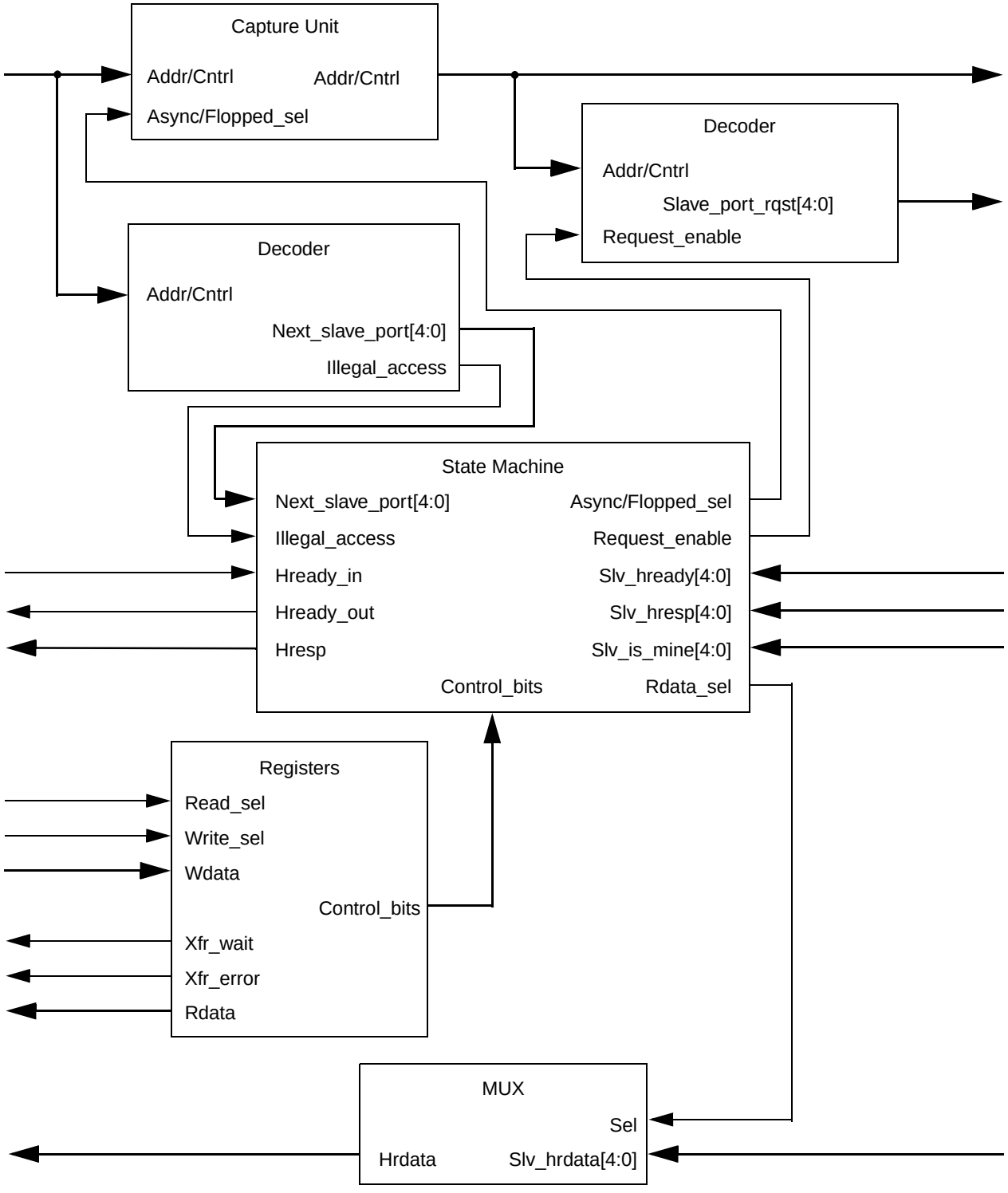


Figure 39-5. MAX Master Port Block Diagram

39.4.3.2 Master Port Decoders

The decoders are very simple as they ensure an access request is allowed to be made and that the slave port targeted is actually present in the design. The decoders feeding the state machine are always enabled. The decoders that select the slave are enabled only when the master port controlling state machine wants to make a request to a slave port. This is necessary so that if a master port is making an access to a slave port and is being wait stated, and its next access is to a different slave port, the request to the second slave port can be held off until the access to the first slave port is terminated.

The decoders also output a “hole decode” or illegal access signal which tells the state machine that the master is trying to access a slave port that does not exist.

39.4.3.3 Master Port Capture Unit

The capture unit simply captures the state of the master’s address and control signals if the MAX cannot immediately pass the master’s request through to the proper slave port. The capture unit consists of a set of flops and a MUX, which selects either the asynchronous path from address and control or the flopped (captured) address and control information.

39.4.3.4 Master Port Registers

The registers in the master port are only those registers associated with this particular master port. The read and write interface for the registers is a quasi-IP bus interface. It is not a full IP bus interface at this level because not all the IP bus signals are routed this deep in the design.

There is a register control block at the same level of the master port and slave port instantiations in the MAX. This control block ensures that all accesses are 32-bit supervisor accesses before passing them on to the master ports.

The register outputs are connected directly to the state machine.

39.4.3.5 Master Port State Machine

39.4.3.5.1 Master Port State Machine States

The master side state machine’s main function is to monitor the activities of the master port. The state machine has six states: busy, idle, stalled, steady state, first cycle error response and second cycle error response.

The busy state is used when the master runs a BUSY cycle to the master port. The master port maintains its request to the slave port if it currently owns the slave port; however, if it loses control of the slave port it will no longer maintain its request. If the master port loses control of the slave port it will not be allowed to make another request to the slave port until it runs a NSEQ or SEQ cycle.

The idle state is used when the master runs a valid IDLE cycle to the master port. The master port makes no requests to the slave ports (disables the slave port decoder) and terminates the IDLE cycle.

The stalled state is used when the master makes a request to a slave port that is not immediately ready to receive the request. In this case the state machine will direct the capture unit to send out the captured

address and control signals and will enable the slave port decoder to indicate a pending request to the appropriate slave port.

The steady state is used when the master port and slave port are in fully asynchronous mode, making the MAX completely transparent in the access. The state machine selects the appropriate slave's hresp0, hready and hrdata to pass back to the master.

The first cycle error response and second cycle error response states are self explanatory. The MAX will respond with an error response to the master if the master tries to access an unimplemented memory location through the MAX (that is, a slave port that does not exist).

39.4.3.5.2 Master Port State Machine Slave Swapping

The design of the master side state machine is fairly straight forward. The one real decision to be made is how to handle the master moving from one slave port access to another slave port access. The approach that was taken was to minimize or eliminate, when possible, any "bubbles" that would get inserted into the access due to switching slave ports.

The state machine does not allow the master to request access to another slave port until the current access being made is terminated. This prevents a single master from owning two slave ports at the same time (the slave port it is currently accessing and the slave port it wishes to access next).

The state machine also maintains watch on the slave port the master is accessing as well as the slave port the master wishes to switch to. If the new slave port is parked on the master then the master will be able to make the switch without incurring any delays. The termination of the current access will also act as the launch of the new access on the new slave port. If the new slave port is not parked on the master then the master will incur a minimum one clock delay before it can launch its access on the new slave port.

This is the same for switching from the busy or idle state to actively accessing a slave port. If the slave port is parked on the master the state machine will go to the steady state and the access will begin immediately. If the slave port is not parked on the master (serving another master, parked on another master or in low power park mode) then the state machine will transition to the stalled state and at least a one clock penalty will be paid.

39.4.4 Slave Port Functionality

39.4.4.1 General

Each slave port consists of a register slice, a bank of muxes, and a state machine.

The register slice contains the registers associated with the specific slave port. The registers have a quasi-IP bus interface at this level for reads and writes and the outputs feed directly into the state machine.

The muxes are a series of 6 to 1 muxes that take in all the address, control and write data information from each of the master ports and then pass the correct master's signals to the slave port. The state machine controls all the muxes.

The state machine is where the main slave port arbitration occurs, it decides which master is in control of the slave port and which master will be in control of the slave port in the next bus cycle.

Figure 39-6 shows a block diagram of a slave port.

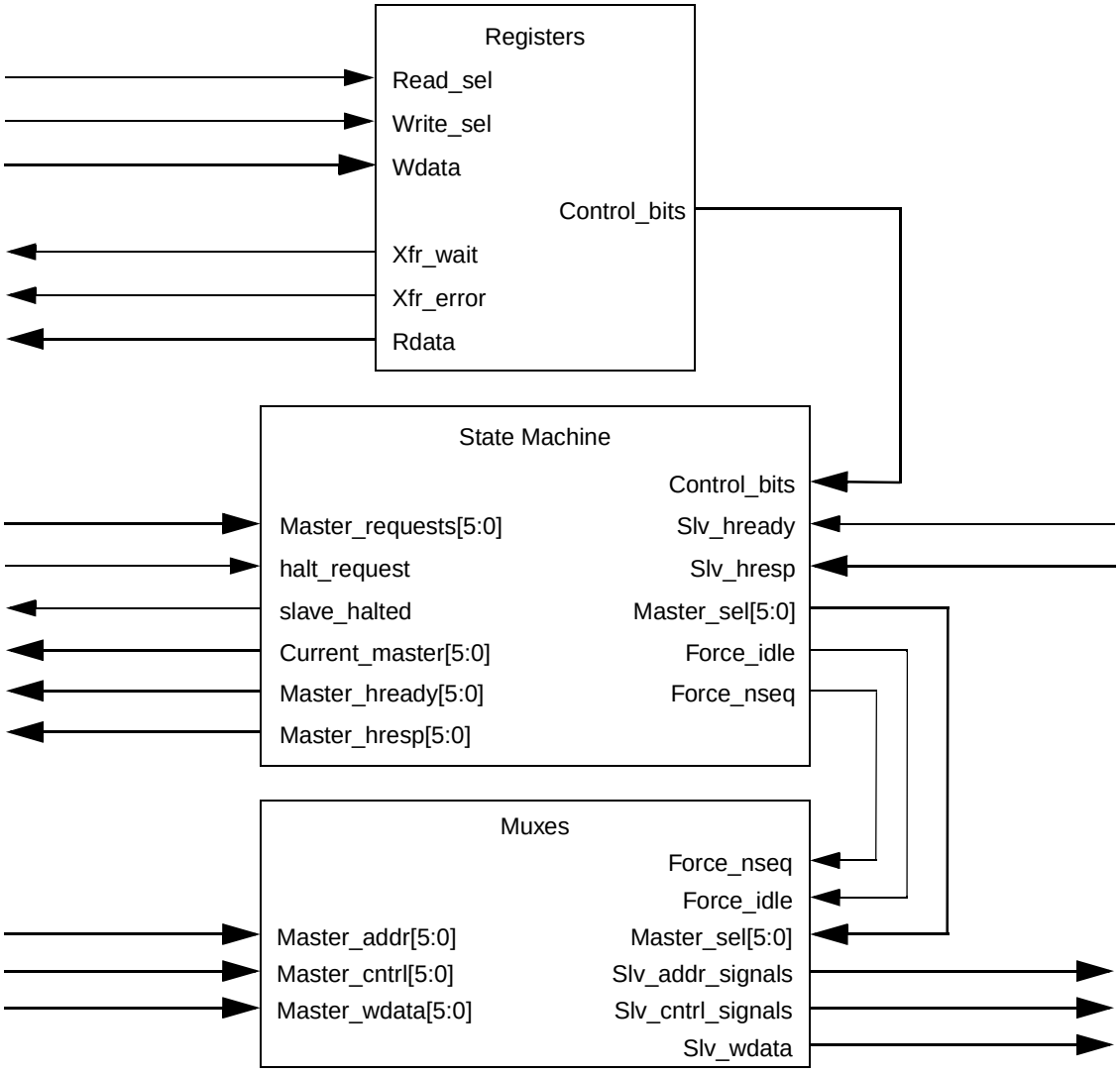


Figure 39-6. MAX Slave Port Block Diagram

39.4.4.2 Slave Port Muxes

The block diagram (Figure 39-6) shows only one block for all the muxes. In reality that block instantiates many 6 to 1 muxes, one for each master-to-slave signal in fact. All the muxes are designed in an AND—OR fashion, so that if no master is selected the output of the muxes will be zero. (This is an important feature for low power park mode.)

The muxes also have an override signal which is used by the slave port to asynchronously force IDLE cycles onto the slave bus. When the state machine forces an IDLE cycle it zeros out htrans and hmastlock, making sure the slave bus sees a valid IDLE cycle being run by the MAX.

The enable to the MUX controlling htrans also contains an additional control signal from the state machine so that a NSEQ transaction can be forced. This is done any time the slave port switches masters to ensure

that no IDLE-SEQ, BUSY-SEQ or NSEQ-SEQ transactions are seen on the slave port when they should not be. If the state machine indicates to run both an IDLE and an NSEQ cycle, the IDLE directive will have priority.

NOTE

IDLE-SEQ is in fact an illegal access, but a possible scenario given the multi-master environment in the MAX unless corrected by the MAX.

39.4.4.3 Slave Port Registers

There is a register control block at the same level of the master port and slave port instantiations in the MAX. This control block ensures that all accesses are 32-bit supervisor accesses before passing them on to the master and slave ports.

The registers in the slave port are only those registers associated with this particular slave port. The read and write interface for the registers is a quasi-IP bus interface. It is not a full IP bus interface at this level because not all the IP bus signals are routed this deep in the design.

The register outputs are connected directly to the slave state machine. The registers can be read from an unlimited number of times. The registers can only be written to as long as the RO bit is written to 0 in the SGPCR, once it is written to a 1 only a hardware reset will allow the registers to be written again.

39.4.4.4 Slave Port State Machine

39.4.4.4.1 Slave Port State Machine States

At the heart of the slave port is the state machine. The state machine is simplicity itself, requiring only four states—steady state, transition state, transition hold state and hold state. Either the slave port is owned by the same master it was in the last clock cycle (either by active use or by parking), it is transitioning to a new master (either for active use or parking), it is transitioning to a new master during wait states or it is being held on the same master pending a transition to a new master.

39.4.4.4.2 Slave Port State Machine Arbitration

The real work in the state machine is determining which master port will be in control of the slave port in the next clock cycle, the arbitration. Each master is programmed with a fixed 3 bit priority level. The MAX uses these bits in determining priority levels when programmed for fixed priority mode of operation.

Arbitration always occurs on a clock edge, but only occurs on edges when a change in mastership will not violate AHB-Lite protocols. Valid arbitrations points include any clock cycle in which sX_hready is asserted (provide the master is not performing a burst or locked cycle) and any wait state in which the master owning the bus indicates a transfer type of IDLE (provided the master is not performing a locked cycle).

Because arbitration can occur on every clock cycle the slave port masks off all master requests if the current master is performing a locked transfer or a protected burst transfer, guaranteeing that no matter how low its priority level it will be allowed to finish its locked or protected portion of a burst sequence.

39.4.4.4.3 Slave Port State Machine Master Handoff

The only times the slave port will switch masters when programmed for fixed priority mode of operation is when a higher priority master makes a request or when the current master is the highest priority and it gives up the slave port by either running and IDLE cycle to the slave port or running a valid access to a location other than the slave port.

If the current master loses control of the slave port because a higher priority master takes it away the slave port will not incur any wasted cycles. The current master will get its current cycle terminated by the slave port at the same time the new master's address and control information will be recognized by the slave port. This will look like a seamless transition on the slave port.

If the current master is being wait stated when the higher priority master makes its request, then the current master is allowed to make one more transaction on the slave bus before giving it up to the new master. Figure 39-7 illustrates the effect of a higher priority master taking control of the bus when the slave port is programmed for a fixed priority mode of operation.

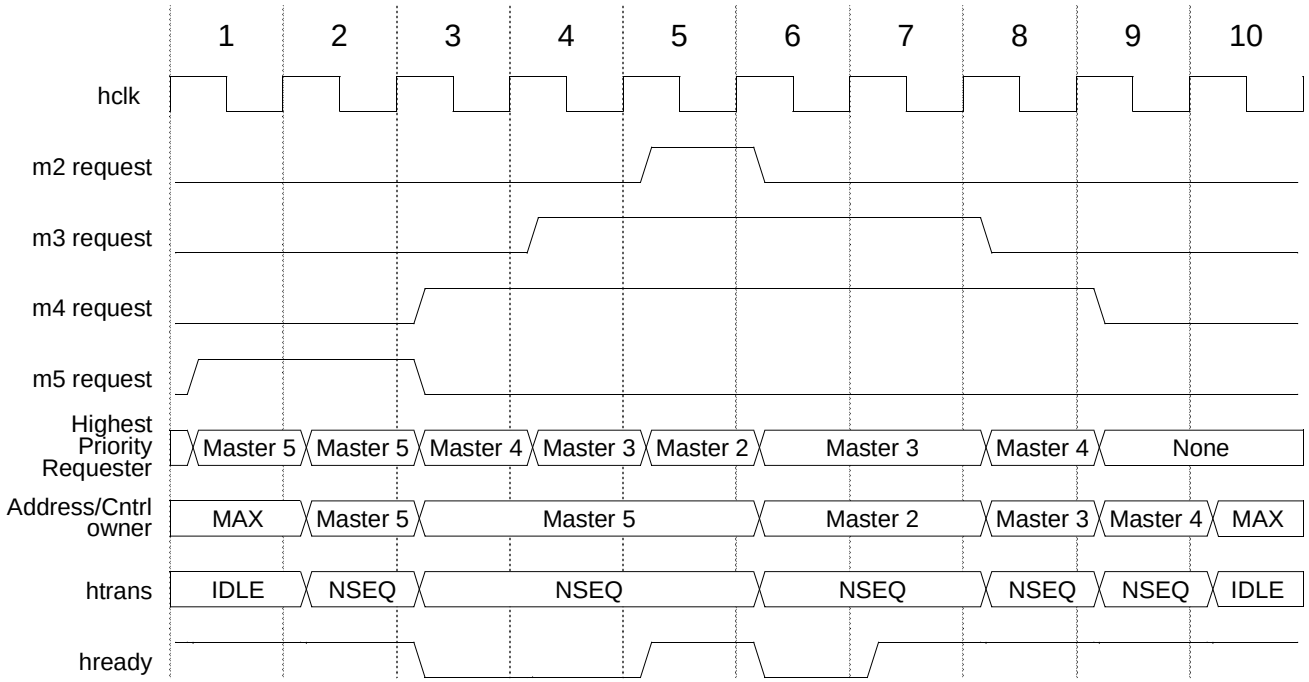


Figure 39-7. Low to High Priority Mastership Change

If the current master is the highest priority master and it gives up the slave port by running an IDLE cycle or by running a valid cycle to another location other than the slave port the next highest priority master will gain control of the slave port. If the current access incurs any wait states then the transition will be seamless and no bandwidth will be lost; however, if the current transaction is terminated without wait states then one IDLE cycle will be forced onto the slave bus by the MAX before the new master will be able to take control of the slave port. If no other master is requesting the bus then IDLE cycles will be run by the MAX but no bandwidth will truly be lost since no master is making a request. Figure 39-8 illustrates the effect of a higher priority master giving up control of the bus.

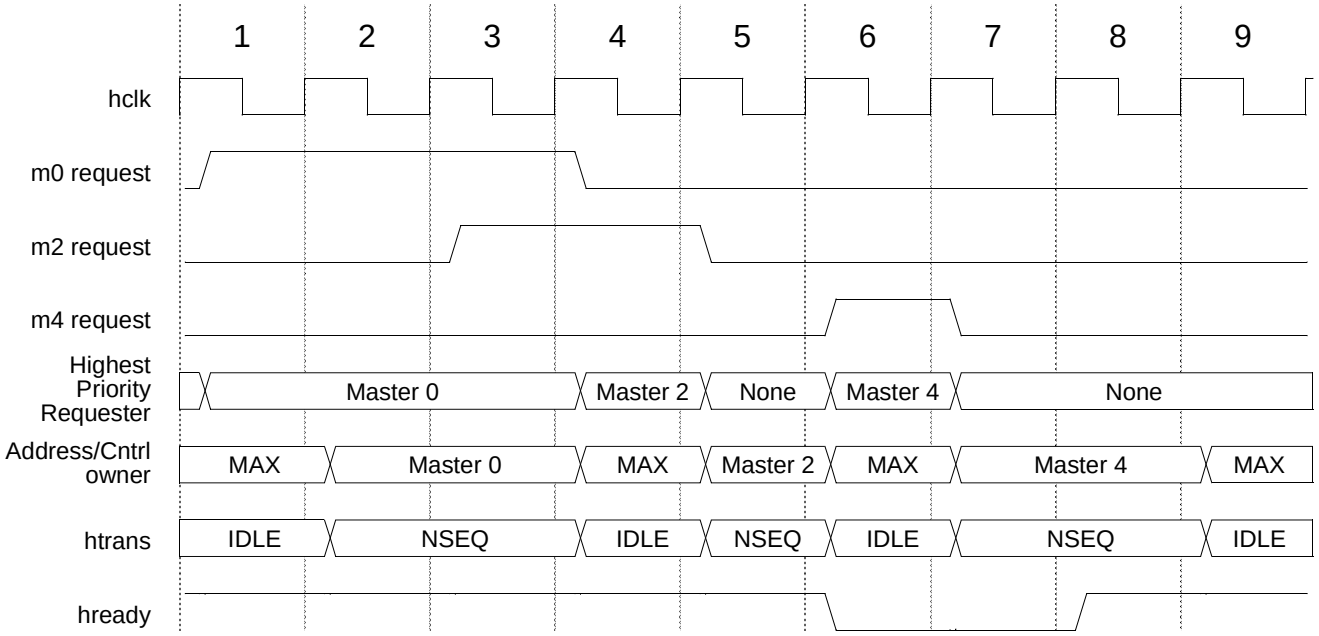


Figure 39-8. High to Low Priority Mastership Change

When the slave port is programmed for round-robin mode of arbitration then the slave port will switch masters any time there is more than one master actively making a request to the slave port. This will happen because any master other than the one which presently owns the bus will be considered to have higher priority. Figure 39-9 shows an example of round-robin mode of operation.

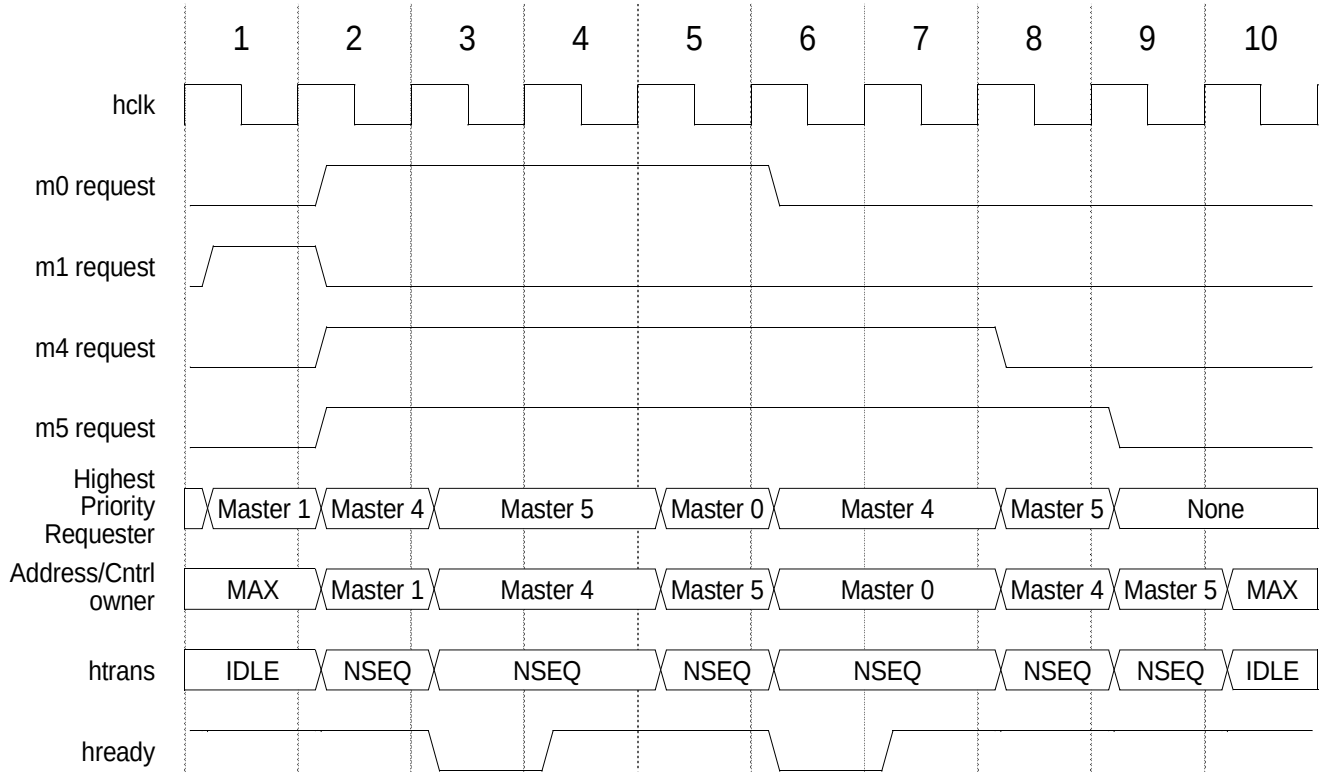


Figure 39-9. Round-Robin Mastership Change

39.4.4.4.4 Slave Port State Machine Parking

If no master is currently making a request to the slave port then the slave port will be parked. It will park in one of four places, dictated by the PCTL and PARK bits in the SGPCR and the locked state of the last master to access it.

If the last master to access the slave port ran a locked cycle and continues to run locked cycles even after leaving the slave port the slave port will park on that master without regard to the bit settings in the SGPCR and without regard to pending requests from other masters. This is done so a master can run a locked transfer to the slave port, leave it, and return to it and be guaranteed that no other master has had access to it (provided the master maintains all transfers are locked transfers). If locking is not an issue for parking the SGPCR bits will dictate the parking method.

If the PCTL bits are set for “low power park” mode then the slave port will enter low power park mode. It will not recognize any master as being in control of it and it will not select any master’s signals to pass through to the slave bus. In this case all slave bus activity will effectively halt because all slave bus signals being driven from the MAX will be 0. This of course can save quite a bit of power if the slave port will not be in use for some time. The down side is that when a master does make a request to the slave port it will be delayed by one clock since it will have to arbitrate to acquire ownership of the slave port.

If the PCTL bits are set to “park on last” mode then the slave port will park on the last master to access it, passing all that masters signals through to the slave bus. The MAX will asynchronously force htrans[1:0], hmaster[3:0], hburst[2:0], and hmastlock to 0 for all access that the master does not run to the slave port.

When that master access the slave port again it will not pay any arbitration penalty; however, if any other master wishes to access the slave port a one clock arbitration penalty will be imposed.

If the PCTL bits are set to “use PARK” mode then the slave port will park on the master designated by the PARK bits. The behavior here is the same as for the “park on last” mode with the exception that a specific master will be parked on instead of the last master to access the slave port. If the master designated by the PARK bits tries to access the slave port it will not pay an arbitration penalty while any other master will pay a one clock penalty. Figure 39-10 illustrates parking on a specific master.

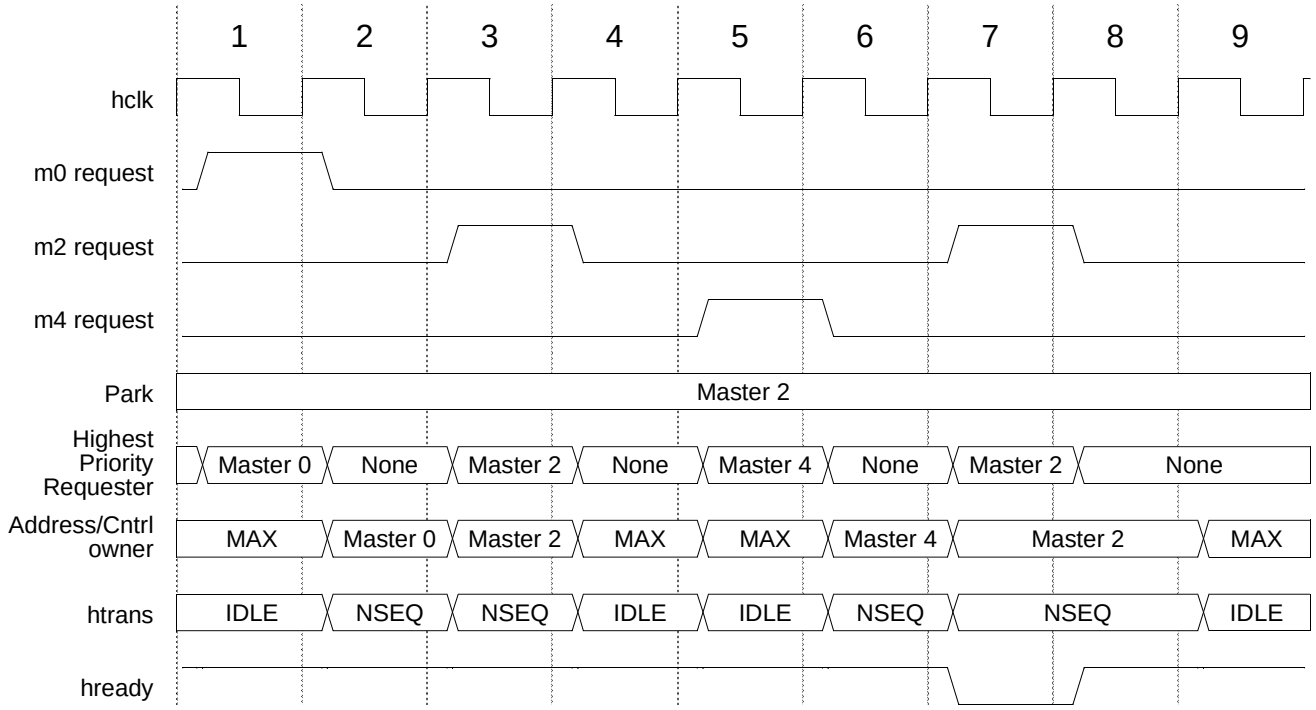


Figure 39-10. Parking on a Specific Master

Figure 39-11 illustrates parking on the last master. Note that in cycle 6 simultaneous requests are made by master 2 and master 4. Although master 2 has higher priority, the slave bus is parked on master 4 so master 4’s access is taken first. The slave port parks on master 2 once it has given control to master 2. This same situation can occur when parking on a specific master as well.

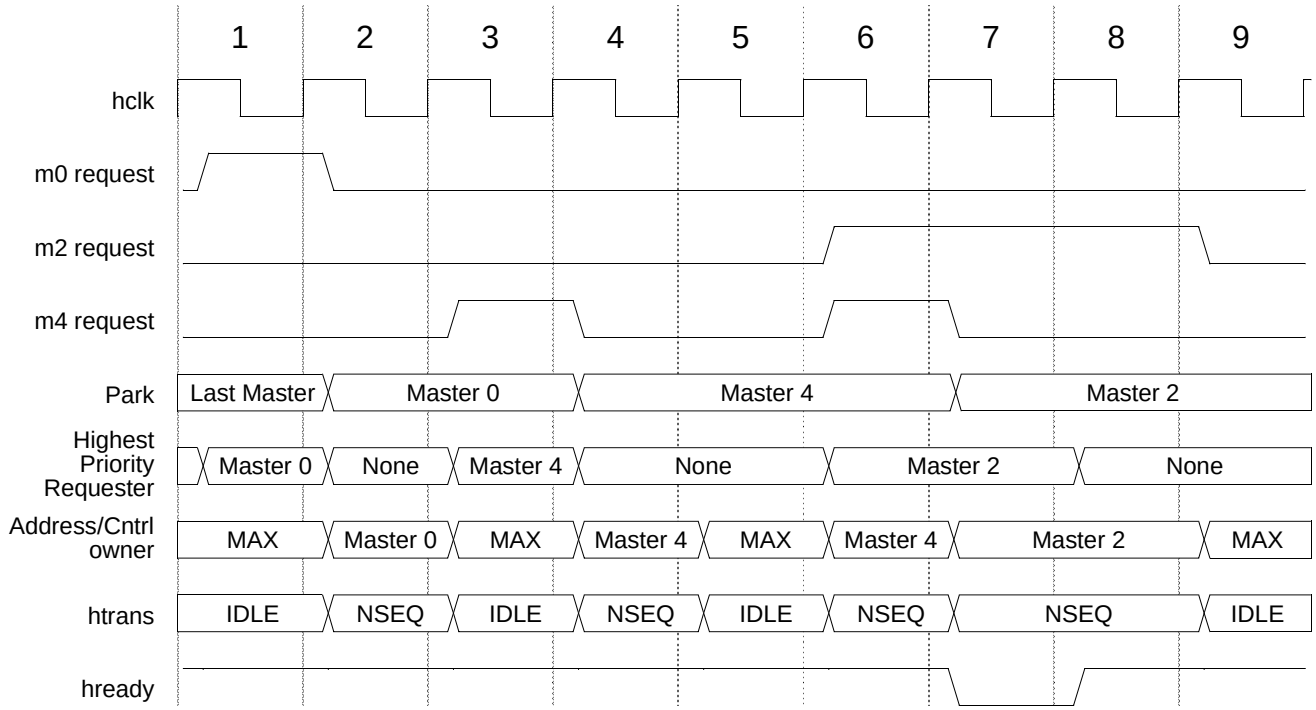


Figure 39-11. Parking on Last Master

39.4.4.4.5 Slave Port State Machine Halt Mode

If the max_halt_request input is asserted the slave port will eventually halt all slave bus activity and go into halt mode, which is almost identical to low power park mode. The HLP bit in the SGPCR controls the priority level of the max_halt_request in the arbitration algorithm. If the HLP bit is cleared then the max_halt_request will have the highest priority of any master and will gain control of the slave port at the next arbitration point (most likely the next bus cycle, unless the current master is running a locked or fixed length burst transfer). If the HLP bit is set then the slave port will wait until no masters are actively making requests before moving to halt mode.

Regardless of the state of the HLP bit, once the slave port has gone into halt mode as a result of max_halt_request being asserted, it will remain in halt mode until max_halt_request is negated, regardless of the priority level of any masters that may make requests.

In halt mode no master is selected to own the slave port so all the outputs of the slave port are set to 0.

39.5 Initialization/Application Information

No initialization is required by or for the MAX. Hardware reset ensures all the register bits used by the MAX are properly initialized.

39.6 MAX Interface

This section provides information on the MAX interface.

39.6.1 Overview

The main goal of the MAX is to increase overall system performance by allowing multiple masters to communicate in parallel with multiple slaves. To maximize data throughput it is essential to keep arbitration delays to a minimum.

This section examines data throughput from the point of view of masters and slaves, detailing when the MAX stalls the masters or insert bubbles on the slave side.

39.6.2 Master Ports

Master accesses receive one of four responses from the MAX. They are either terminated, taken, stalled or responded to with an error.

39.6.2.1 Terminated Accesses

A master access will be terminated if the transfer type is IDLE. The MAX will terminated the access and it will not be allowed to pass through the MAX.

39.6.2.2 Taken Accesses

A master access will be taken if the transfer type is non IDLE and the slave port to which the access decodes is either currently servicing the master or is parked on the master. In this case the MAX will be completely transparent and the master's access will be immediately seen on the slave bus and no arbitration delays will be incurred.

39.6.2.3 Stalled Accesses

A master access will be stalled if the transfer type is non IDLE and the access decodes to a slave port that is busy serving another master, parked on another master or is in low power park mode. The MAX will indicate to the master that the address phase of the access has been taken but will then queue the access to the appropriate slave port to enter into arbitration for access to that slave port.

If the slave port is currently parked on another master or is in low power park mode and no other master is requesting access to the slave port then only one clock of arbitration will be incurred. If the slave port is currently serving another master of a lower priority and the master has a higher priority than all other requesting masters then the master will gain control over the slave port as soon as the data phase of the current access is completed (burst and locked transfers excluded). If the slave port is currently servicing another master of a higher priority then the master will gain control of the slave port once the other master releases control of the slave port if no other higher priority master is also waiting for the slave port.

39.6.2.4 Error Response Terminated Accesses

A master access will be responded to with an error if the transfer type is non IDLE and the access decodes to a location not occupied by a slave port. This is the only time the MAX will respond with an error response. All other error responses received by the master are the result of error responses on the slave ports being passed through the MAX.

See Section 39.7.1, “Address Map” for information on locations that are not occupied by a slave port.

39.6.3 Slave Ports

The goal of the MAX with respect to the slave ports is to keep them 100% saturated when masters are actively making requests. To do this the MAX must not insert any bubbles onto the slave bus unless absolutely necessary.

There is only one instance when the MAX will force a bubble onto the slave bus when a master is actively making a request. This occurs when a higher priority master has control of the slave port and is running single clock (zero wait state) accesses while a lower priority master is stalled waiting for control of the slave port. When the higher priority master either leaves the slave port or runs an IDLE cycle to the slave port the MAX will take control of the slave bus and run a single IDLE cycle before giving the slave port to the lower priority master that was waiting for control of the slave port.

The only other times the MAX will have control of the slave port is when the MAX is halting or when no masters are making access requests to the slave port and the MAX is forced to either park the slave port on a specific master or put the slave port into low power park mode.

In most instances when the MAX has control of the slave port it will indicate IDLE for the transfer type, negate all control signals and indicate ownership of the slave bus via the hmaster encoding of 4’b0000. One exception to this rule is when a master running locked cycles has left the slave port but continues to run locked cycles. In this case the MAX will control the slave port and will indicate IDLE for the transfer type but it will not affect any other signals.

NOTE

When a master runs a locked cycle through the MAX, the master will be guaranteed ownership of all slave ports it accesses while running locked cycles for one cycle beyond when the master finishes running locked cycles.

39.7 Integration

This section describes integrating the MAX into a design.

39.7.1 Address Map

The address map implemented in the MAX is shown in Table 39-9.

Table 39-9. MAX Address Map

haddr[31:28]	Destination
0x00	Slave Port 3
0x01	Slave Port 4
001x	Reserved
0110	Reserved
0111	Slave Port 2

Table 39-9. MAX Address Map (continued)

haddr[31:28]	Destination
10xx	Slave Port 1
11xx	Slave Port 0

39.7.2 Master Ports

Each master port takes on the appearance of a standard AHB-Lite slave device. The master port can only service one master, so if multiple masters are needed the integrator must provide an arbiter to determine which master will be connected to the master port. This configuration is shown in Figure 39-12.

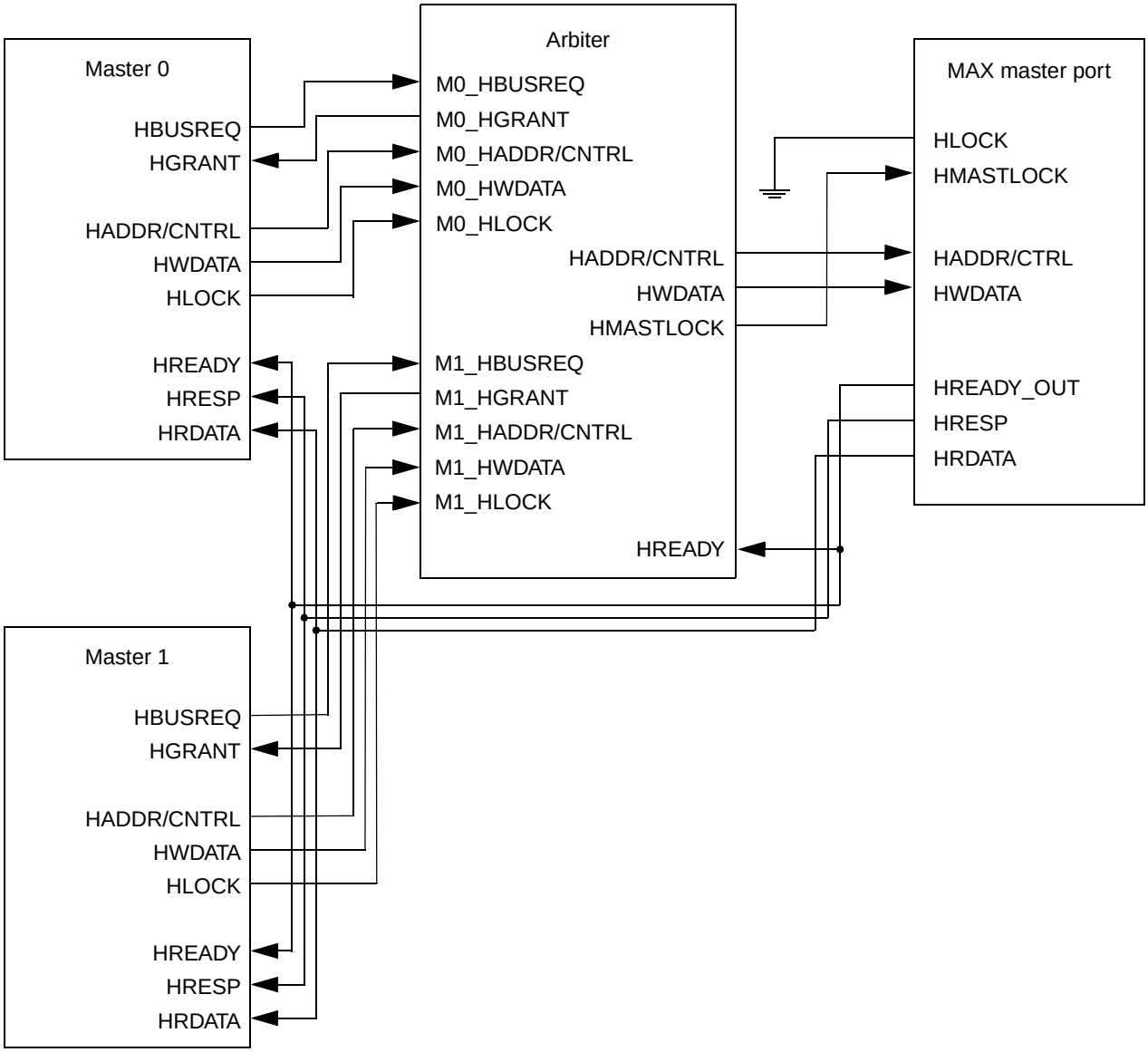


Figure 39-12. Multiple Master Configuration

The simplest approach is connecting a single master to the master port. In this instance all wiring is direct connect; however, since the MAX does not support the bus request/bus grant protocol the master's bus grant input must be tied asserted. This configuration is shown in Figure 39-13.

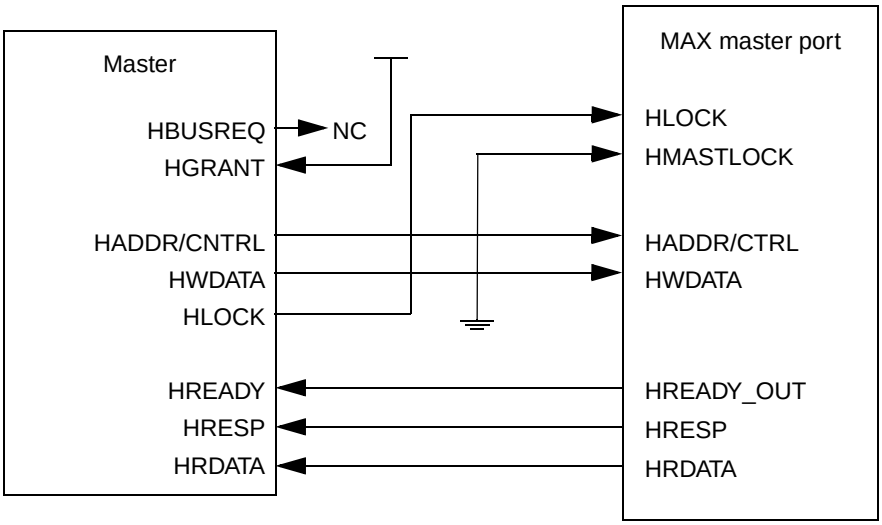


Figure 39-13. Single Master Configuration

39.7.3 Slave Ports

Each slave port takes on the appearance of a standard AHB-Lite master device. Since the slave ports do not support the bus request/bus grant protocol they assume that they are the only master driving the AHB-Lite interface they are connected to.

39.7.4 Registers

The MAX registers are read and written via an IP bus interface. The registers also have their own clock input, ipg_clk. The register clock and the MAX clock must be in phase. The reason for separate clock inputs is to allow for power saving features such as turning off the ipg_clk when no register accesses are being performed. When in doubt connect the ipg_clk to the same clock driving the MAX hclk input.

Chapter 40
Smart Direct Memory Access (SDMA)

The Smart Direct Memory Access (SDMA) architecture offers highly-competitive DMA Controller features combined with software-based virtual-DMA flexibility. It enables data transfers between peripheral I/O devices and internal/external memories.

The Direct Memory Access (DMA) controller is a critical piece of hardware in a highly integrated IC, such as a 3G baseband or multimedia chip. It helps maximize system performance by off-loading the CPU in dynamic data routing. During the SDMA definition phase, special attention was given to the following parameters:

- Data throughput
- Context switching latency
- Channel number scalability
- Single interface peripheral support
- User-friendly programming
- Reduced footprint
- Power-efficient architecture

NOTE

In this document, “AP” refers to interfaces to the ARM (Applications Processor).

40.1 Overview

Figure 40-1 shows a block diagram of the SDMA. It represents the custom RISC core along with its RAM, ROM, DMA units, the CRC unit, and the scheduler.

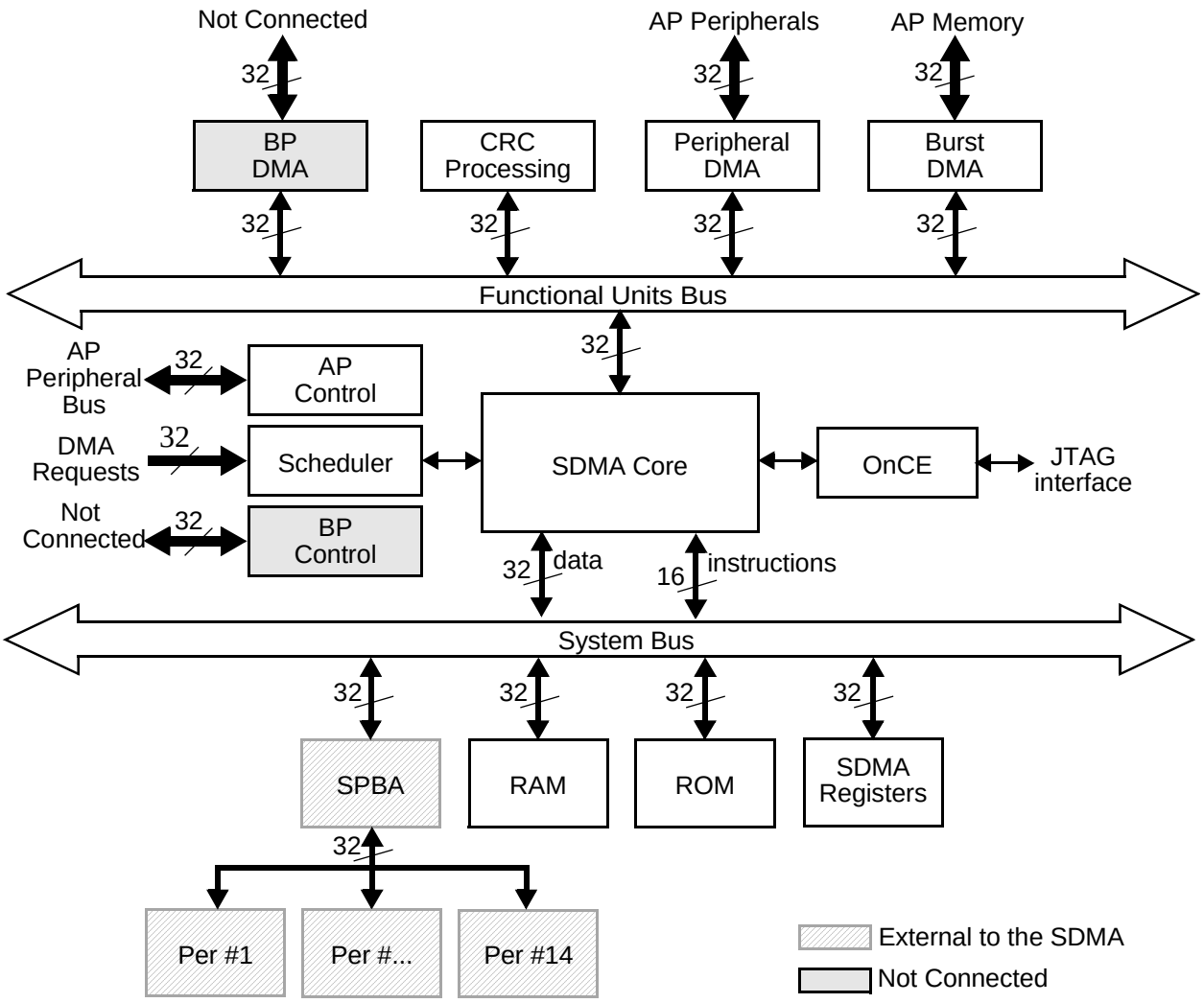


Figure 40-1. SDMA Block Diagram

The SDMA core executes short routines that perform DMA transfers; these routines or programs are called *scripts*. The instruction set comprises single-cycle instructions with the exception of load/store instructions to the internal memory (RAM, ROM, and memory mapped registers), to the registers of the DMA and CRC units, and to branch instructions that may require several cycles to execute. The SDMA core interfaces to its own memory via the SDMA system bus. The SDMA system bus supports a 32-bit data path and a 16-bit address bus. DMA units interface to the core via the Functional Unit Bus and use dedicated registers to perform DMA transfers.

The SDMA memory contains a ROM and a RAM. The ROM contains startup scripts (for example, boot code) and other common utilities, which are referenced by the scripts that reside in the RAM. The internal RAM is divided into a context area and a script area (more details about this mapping are available in Section 40.11.5.1, “Instruction Memory Map” and Section 40.11.5.2, “Data Memory Map”).

Every transfer channel requires one context area to keep the contents of all the core and unit registers while inactive. Channel scripts are downloaded into the internal RAM by the SDMA using a dedicated channel

that is started during the boot sequence. Downloads are invoked using command and pointers provided by the AP. Every channel contains a corresponding channel script located in RAM and/or ROM that can be reconfigured independently on an “as-needed” basis. This permits a wide range of SDMA functionality while using the lowest internal memory footprint possible. Channel scripts can be stored in an external, large capacity FLASH memory and downloaded when needed. The SDMA can be configured with any mixture of scripts to enable an endless combination of supported services.

The scheduler is responsible for monitoring and detecting DMA requests, mapping them to channels, and mapping individual channels to a pre-configured priority. At any given point, the scheduler presents the highest priority channel that requires service to the SDMA core. A special SDMA core instruction is used to “conditionally yield” the current channel being executed to an eligible channel that requires service. If (and only if) there is an eligible channel pending, will the current channel execution be preempted.

There are two `yield` instructions that differently determine the eligible channels: In the first version, eligible channels are pending channels with a strictly higher priority than the current channel priority. In the second version (`yieldge`), eligible channels are pending channels with a priority that is greater or equal to the current channel priority. The scheduler detects devices that need service through its 32 DMA request inputs. After a request is detected, the scheduler determines the channel(s) that is (are) triggered by this request and marks it (them) as pending in the “Channel Pending (EP)” register. The priorities of all the pending channels are combined and continuously evaluated in order to update the highest pending priority. The channel pending flag is cleared by the channel script when the transfer has completed.

The AP Control module contains the control registers used to configure the 32 individual channels. There are 32 Channel Enable registers, and every register maps one DMA request to any desired combination of channels. The 32 Priority registers are used to assign a programmable 1-of-7 level priority to every possible channel. This module also contains all other control registers that the AP can access.

The 32 DMA requests that are connected to the scheduler come from a variety of sources. The “receive register full” and “transmit register empty” signals found in the UART and USB ports are typical examples of DMA requests that can be connected to the SDMA. These requests can be used to trigger a specific SDMA channel, or several channels. This feature can be used to realize a “just-in-time” data exchange between the two processors to relax the requirement to meet critical deadlines.

The embedded nature of the SDMA requires on-chip debug capability to assure product quality and reliability, and to realize the full performance capabilities of the core. The OnCE compatible debug port includes support for setting breakpoints, single-step and trace, and register dump capability. In addition, all memory locations are accessible from the debug port.

Figure 40-2 shows the SDMA structure: The functional unit bus *hooks up* the CRC hardware accelerator and the three DMA units; and the system bus that enables the SDMA core to access its internal memory also supports up to 14 peripherals.

40.2 Features

The following are the SDMA features:

- Multi-channel DMA supporting up to 32 time-division multiplexed DMA channels
- Hardware or software driven triggers for each channel

- 32 hardware driven triggers that can be mapped to any channel.
- Memory accesses including linear addressing, FIFO addressing and 2D addressing
- Fast context-switching with two-level, priority-based preemptive multi-tasking
- 16-bit instruction-set micro-RISC engine (the SDMA core)
- Two DMA units with some or all the following features:
  - — Auto-flush and prefetch capability
  - — Flexible address management (increment, decrement, and no address changes on source and destination address)
  - — Misaligned data-transfer support
  - — Uni-directional and bidirectional flows (copy mode)
  - — Up to eight-word buffers for configurable burst transfers
- Support of byte-swapping and CRC calculations
- An available API and library of scripts
- Little-Endian and Big-Endian modes
- Hardware handshakes for low-power entry sequence
- 4-Kbytes of ROM containing startup scripts (for example, boot code) and other common utilities that can be referenced by RAM-located scripts
- 8-Kbytes of RAM area is divided into a processor context area and a code space area used to store channel scripts that are downloaded from the system memory
- Debug support, including a OnCE port, real-time monitors, and embedded cross-trigger events
- Supported clock frequencies in cmos90lp process:
  - — Configurable clock options for the core and the DMA units that are connected to the AP domain: 1/2 ratio with maximum 66 MHz/133 MHz or 1/1 ratio with maximum 104 MHz/104 MHz
- Peripheral bus interface for configuration register programming by the AP
- The SDMA RISC engine (arithmetic and logic operations, plus CRC acceleration), which is referred to as the “SDMA core.”
- An internal peripheral bus connected to the Shared Peripherals Bus Interface (SPBA) that enables access to up to 14 shared peripherals. SDMA supports 32-bit accesses to word peripherals and 16-bit accesses to halfword peripherals.
- The peripheral DMA unit that is hooked-up to the AP Crossbar Switch to service AP peripherals
- The burst DMA unit is able to perform burst accesses to the external memory
- All the DMA units are 32-bit AHB masters. They are connected to different buses, thus allowing concurrent accesses.

40.3 Functional Description

Figure 40-2 shows the SDMA topology, and is composed of the following components:

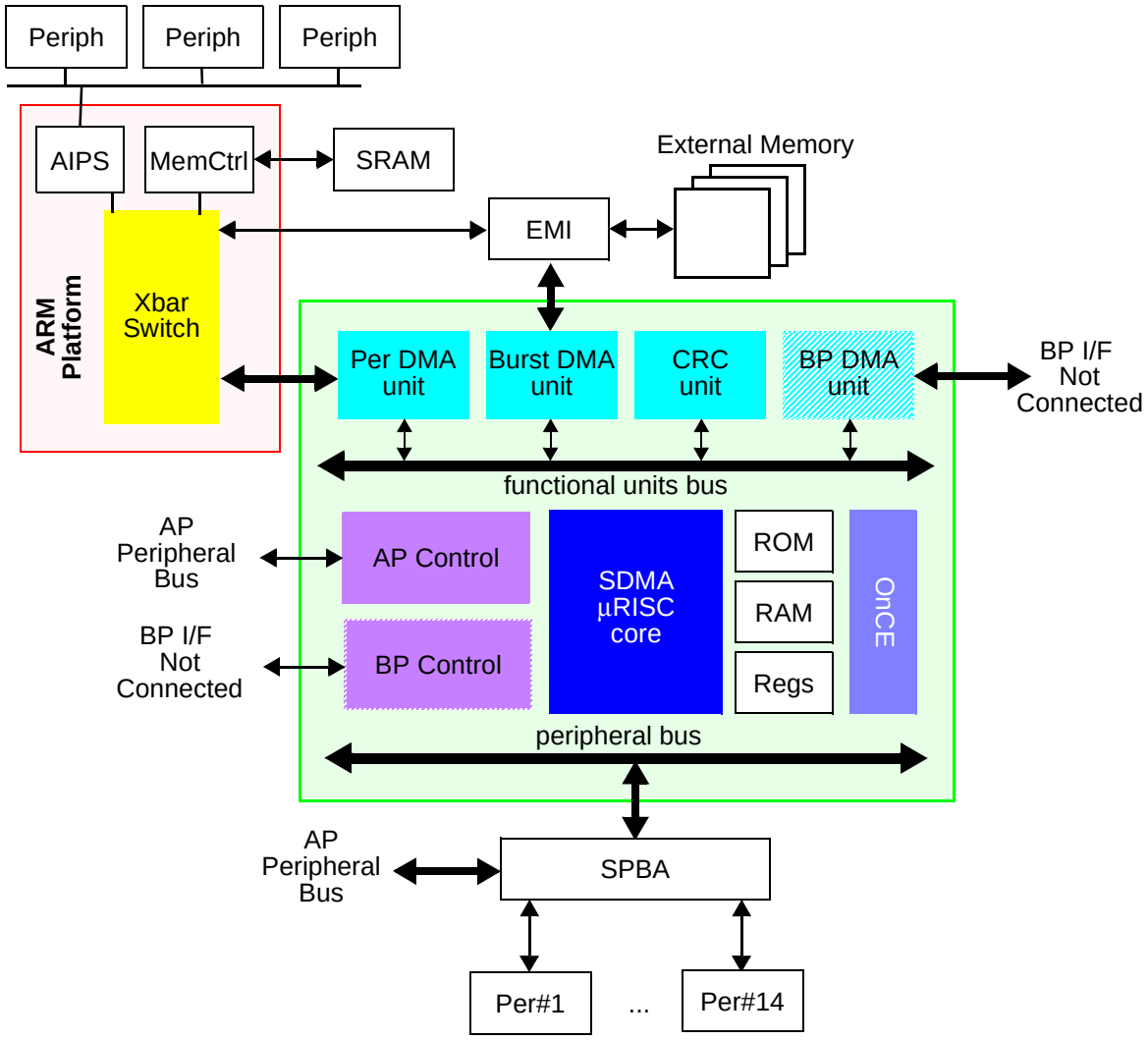


Figure 40-2. SDMA Connections

40.4 SDMA Core

The SDMA core is a customized RISC-like processor that is specifically developed to control DMA units and perform L1 tasks like byte-stuffing or framing. The SDMA core incorporates on-chip debug capability using the OnCE.

The SDMA core is based on a 32-bit register architecture with 16-bit instructions. There are eight general purpose 32-bit registers, four flags (T, LM, SF, and DF), and four PCU registers (PC, RPC, SPC, and EPC) that can address 16,384 16-bit instructions.

40.4.1 Attributes

The following is a summary of key SDMA core attributes:

- Two-stage pipeline
- 32-bit data path architecture
- 14-bit address bus/16-bit instruction bus (up to 16,384 16-bit instructions)
- 16-bit address bus/32-bit data bus (up to 65,536 32-bit data)
- Eight general purpose registers
- Single-entry Return PC Register (RPC) to support subroutine calls
- T flag to reflect the status of some arithmetic and logical instructions
- SF and DF flags to reflect the load/store status on peripheral and functional buses
- Hardware loop capability on branch instructions for enhanced performance
- All the instructions operate on a 32-bit datawidth, including loads and stores. If some external memory objects (for example, I/O registers) are narrower, they are zero-extended to the left when read.
- Special instructions to access the DMA and CRC units specific registers
- Hardware-based, context-switch routine
- Branch instructions do not have a programmer-visible delay slot.

40.4.2 Structure

Figure 40-3 shows the structure of the SDMA core. It also shows the different registers, calculation resources, and possible data movements.

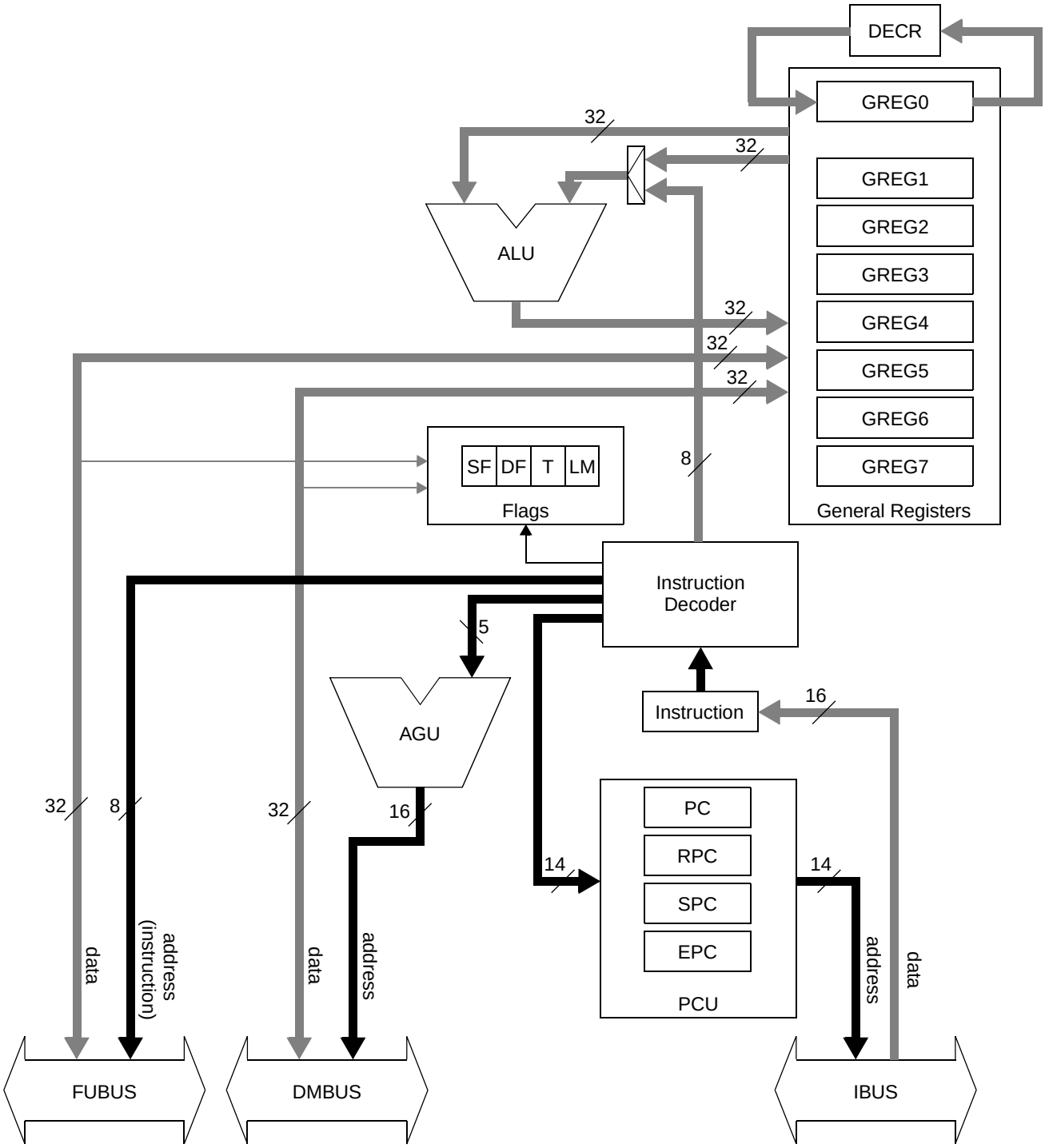


Figure 40-3. SDMA Core

- The Program Control Unit (PCU) is described in Section 40.4.3, “Program Control Unit (PCU).” It handles the state of the core and generates the instruction fetch addresses. Instructions are retrieved from the Instruction Bus (IBUS) and stored in the instruction register prior to their decoding.

The Program Counter (PC) contains the address of the current instruction. The Return Program Counter (RPC) contains the address of the instruction that follows a jump to the subroutine. The Star Program Counter (SPC) and End Program Counter (EPC), respectively, contain the address of the first and last instructions of the current hardware loop.

- The other core registers are the general purpose registers (GREGn) and the flags.

  The general purpose registers can be used to hold data and addresses. They can be loaded with immediate values (for example, 8-bit data that are encoded in the instruction), results of calculations that were performed with the ALU, 32-bit data that comes from the memory or peripherals via the Data Memory Bus (DMBUS), 32-bit data that comes from the DMAs or CRC via the Functional Units Bus (FUBUS) or another general purpose register. Their content can be the operands of the ALU, the data to send on either bus (DMBUS or FUBUS), or a pointer to memory (DMBUS address).

  The general register 0 (GREG0) is also the hardware loop counter. In hardware loops, it cannot be used for any other purpose, and it needs a dedicated decrement unit (DECR).

  The flags reflect the status of operations: SF and DF are set when the last load or store on either bus (FUBUS or DMBUS) received an error response; LM is set when the core is executing instructions inside a hardware loop; and T is set when the ALU operation result was 0 or the loop counter reaches 0 (the latter is preponderant when an ALU operation is the last instruction of a hardware loop).
- The ALU has two operands: any general register and either a second general register or an immediate value. The result is always stored into the first general register; the ALU has an NOP function that can be used to move the second register contents or the immediate value into the first register.
- The 16-bit instructions are fetched via the instruction bus (IBUS) whose address is driven by the PC; the SDMA RAM and ROM are visible to the core as 16-bit devices through this interface.
- The memory (RAM and ROM), memory mapped registers, and external peripherals are accessed via the DMBUS. The address is always taken from a general register whose content is added to a 5-bit immediate value. This is the only available addressing mode. The DMBUS is a 32-bit data bus. Except for the peripherals that are external to the SDMA, the address accuracy is the 32-bit word (for example, adding 1 to an address points to the next word, not the next byte).
- The functional units are accessed via the FUBUS connection. The data is exchanged with any general register, but the address (which in fact is the instruction and the selector of the functional unit) comes from an 8-bit field of the corresponding load or store.

40.4.3 Program Control Unit (PCU)

This part of the SDMA core is dedicated to the control of the RISC engine as implied by the instructions that are executed. Its behavior is determined by the instruction type and the inputs of the SDMA. It contains the PC, RPC, SPC, and EPC registers that are described in Section 40.4.2, “Structure.”

40.4.3.1 Instruction Types

The state sequence and the delay of execution vary according to the type of the instruction. There are six possible categories of instructions, as follows:

1. `standard`—Most of the instructions belong to this category, and always last 1 cycle.
2. `ldf`/`stf`—These are respectively the load and store instructions that access the functional units. They last 1+n cycles where n is the number of wait-states of the targeted functional unit.
3. `ld`/`st`—These are the load and store instructions that access the memory and peripherals. They last 1+n cycles where n is the number of wait-states of the targeted device (1 for the ROM, RAM, and memory mapped registers, 1 + the external peripheral wait-states). These instructions always last at least two cycles, but the core is able to handle them in one cycle. The first wait-state is inserted outside the core.
4. `branch`—These are all the instructions that cause the Program Counter to point to another instruction other than the following one (for example, one that breaks the sequential flow). There are the absolute jumps, the conditional branches, the jump to the sub-routines, and the return from the sub-routine.
5. `loop`-modified load or store—The hardware `loop` instruction modifies the potential behavior of any load or store inside the `loop` (for example, when the LM flag is set). A jump may be implied after any such load or store if it received an error. The error causes an early exit of the loop, which means a jump to the instruction that follows the one that is pointed to by EPC. An additional cycle is required by the PCU to perform the jump (+1 to the `ld`/`st`/`ldf`/`stf` original execution delay). Although there is usually an implicit jump after the last instruction of the loop when the PC goes back to SPC, this is performed at no cycle cost.
6. `done`—The `done`, `yield`, or `yieldge` instructions are used to control channel switching. When no channel switching is performed, these instructions last a single cycle. When there is a change of channel or context switch, the delay is variable and depends on many factors (as detailed in Section 40.5.4, “Context Switching”).

40.4.3.2 PCU States

The PCU state is visible through outputs of the SDMA (see Section 40.18.8.4, “Real-Time Debug Outputs”) or the OnCE status register (see Section 40.17.4.9, “OnCE Status Register (OSTAT)”).

The PCU state is a four-bit field that can take the values shown in Table 40-1. Figure 40-4 shows the possible state transitions and the corresponding conditions.

Table 40-1. PCU States

Value	State	Description
0	Program	The is the usual instruction cycle.
1	Data	This state is inserted when there are wait-states during a load or a store on the data bus (ld/st type).
2	Change of Flow	This is the second cycle of any instruction that breaks the sequence of instructions (branch and done types). This state lasts only a single cycle; it is always followed by the Program state.

Table 40-1. PCU States (continued)

Value	State	Description
3	Error in Loop	This state is used when an error causes a hardware loop exit (loop-modified load or store type). This state only lasts a single cycle; it is always followed by the Program state.
4	Debug	he SDMA is stopped in debug mode.
5	Functional Unit	This state is inserted when there are wait-states during a load or a store on the functional units bus (ldf/stf type).
6	Sleep	No script is running: The core is idle after saving the last channel context.
7	Save	The context switch FSM is saving the current channel.
8	Program in Sleep	Same as Program except there is no associated channel, this state is used when instructions are executed after entering debug mode, whereas the core was in either Sleep mode.
9	Data in Sleep	This is the same as Data except there is no associated channel.
10	Change of Flow in Sleep	This is the same as Change of Flow except there is no associated channel. This state only lasts a single cycle, and is always followed by the Program in Sleep state.
11	Error in Loop in Sleep	This is the same as Error in Loop except there is no associated. channel. This state only lasts a single cycle, and is always followed by the Program in Sleep state.
12	Debug in Sleep	This is the same as Debug except the core was put in debug mode when no channel was active.
13	Functional Unit in Sleep	This is the same as Functional Unit except there is no associated channel.
14	Sleep after Reset	This shows that no script is running, and the core is idle after a reset. When a channel becomes active, no context is restored but the core starts its boot program located at address 0 (or the address available in register in Section 40.10.3.21, “Channel 0 Boot Address (CHN0ADDR)”).
15	Restore	The context switch FSM is restoring the next channel context.

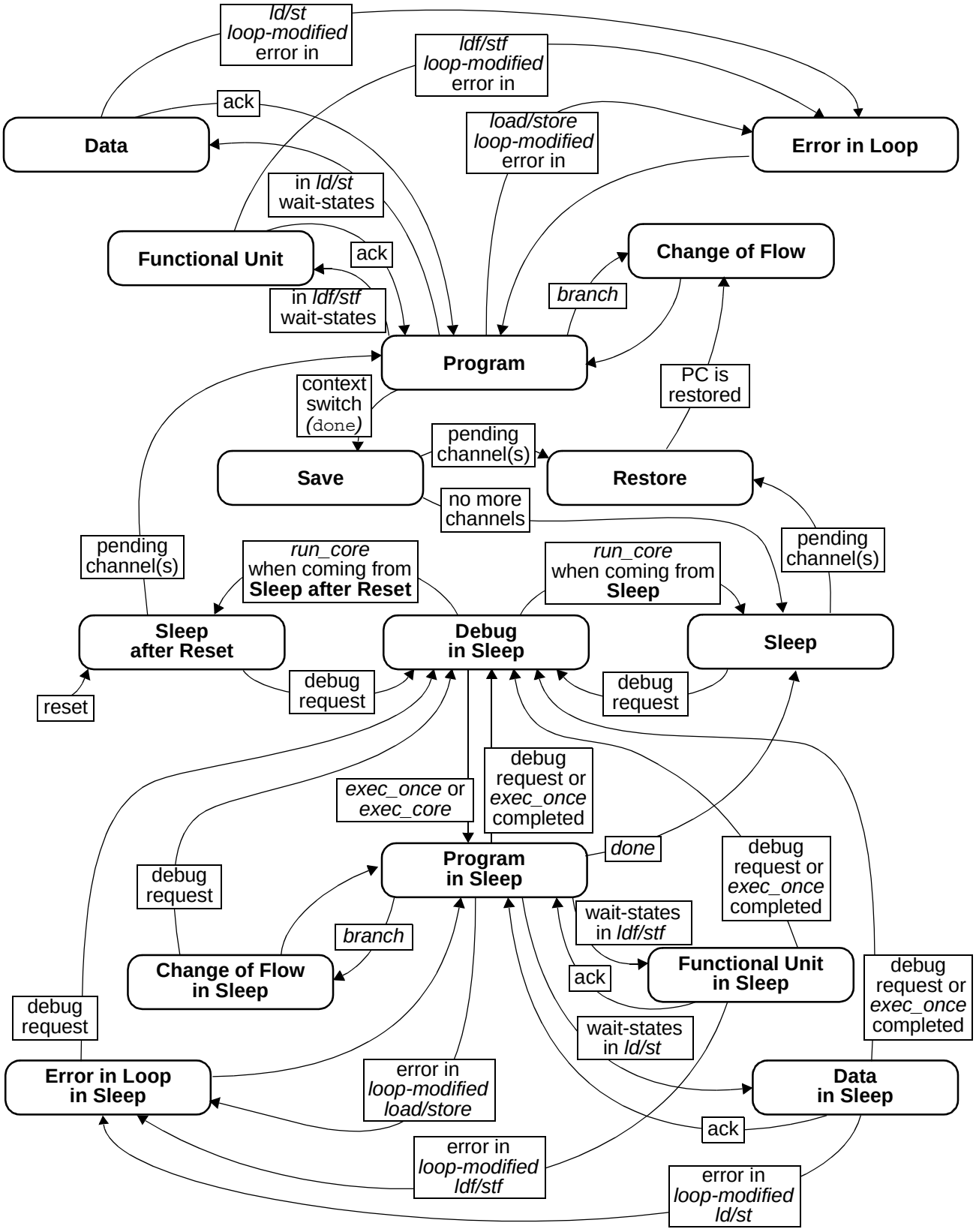


Figure 40-4. PCU State Diagram

40.4.4 SDMA Core Memory

The SDMA has two memory spaces: one for the instructions and one for the data. As both spaces share the same resources (ROM and RAM devices), the system bus manages possible conflicts when the core accesses the same resource for both an instruction read and a data read or write.

Instructions of 16-bit width are stored in 32-bit wide devices and can be accessed as data. The mapping is Big Endian: an even instruction address (terminated by 0) accesses the most significant part of the 32-bit data (bits [31:16]), and an odd instruction address (terminated by 1) accesses the least significant part of the 32-bit data (bits [15:0]).

Instructions can be fetched out of internal ROM or RAM.

Data can be read from ROM, RAM, memory mapped registers, and external peripherals, and written to the same devices (except the ROM).

40.5 Scheduler

All channel scheduling hardware is included in the Scheduler.

40.5.1 Primary Functions

The scheduler is a hardware-based design used to coordinate the timely execution of 32 virtual DMA channels by the SDMA core on the basis of channel status and priority. The scheduler performs the following functions:

- Monitors, detects, and registers the occurrence of any one of the 32 DMA requests
- Links a specific request to a channel or group of channels (channel mapping)
- Ignores requests that are not mapped to a previously configured channel
- Maintains a list of all the channels that are requesting service
- Assigns a pre-programmed priority level (1 of 7) to every channel requesting service
- Detects and flags overrun/underrun conditions

40.5.2 Channels and DMA Requests

40.5.2.1 Channels

A Virtual Channel (hereafter simply called a channel) manages a flow of data through the SDMA. Flows are typically unidirectional. The SDMA can have up to 32 simultaneously operating channels, numbered from 0 to 31. Channel 0 is usually dedicated to control the SDMA script downloading. All the channels can be assigned by the AP software.

40.5.2.2 DMA Requests

A DMA request is caused by externally (for example, external to the SDMA) controlled conditions (for example, UART receive FIFO reaches a threshold). The SDMA currently supports up to 32 DMA requests.

40.5.2.3 Mapping from DMA Requests to Channels and Priorities

A channel can stall waiting on a single DMA request. A single DMA request can awake more than one channel (in fact, any request can awake any combination of channels). The mapping between DMA requests and channels is program-controlled. There is a storage element assigned for each of the 32 requests that contains a bitmap table of the channels that are awakened by the event. Every channel also has a three-bit register that indicates its priority.

40.5.3 Scheduler Functional Description

Section 40.5.3.1, "Scheduler Overview" describes the behavior of the SDMA scheduler—from the channel enabling conditions to the highest priority pending channel selection.

40.5.3.1 Scheduler Overview

The scheduler algorithm is built in hardware. It is provided with possibilities for the AP to control its behavior. Figure 40-5 shows a functional overview. The scheduler processes incoming DMA requests, maps detected requests to 0, one, or several channels, maintains a list of channels that are requesting service (pending channels), identifies the top priority and its associated channel, and selects the next active channel when the current channel yields.

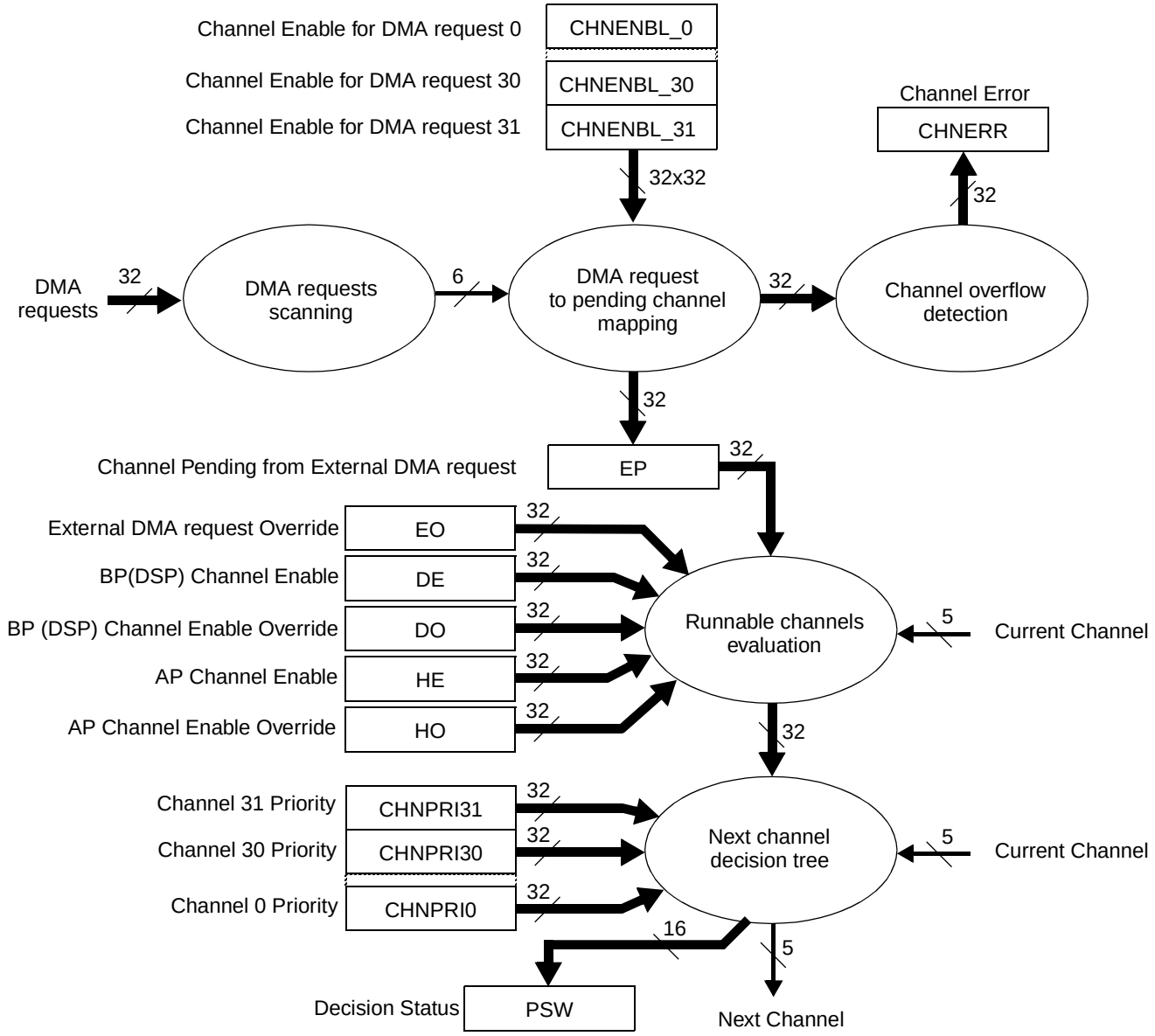


Figure 40-5. SDMA Hardware Scheduler

40.5.3.2 DMA Requests Scanning

The scheduler contains a 32-bit edge detection device that detects the rising edge of every DMA request and transmits the request number to the next stage. The DMA requests are assumed to be generated on the same reference clock as the SDMA core clock; they are detected as soon as the signal goes from a 1-to-n-cycles low state to a 1-to-m-cycles high state.

This system is able to detect single-cycle pulses as well as level-based DMA requests such as a FIFO threshold crossing. In this case, the SDMA provides a memory mapped register that can be used by the channel script to monitor the DMA requests lines, and thus determines whether the data transfer is done or not done, and then continues with the transfer or closes the channel.

When several DMA requests are detected at the same time, they are forwarded to the next scheduler stage at the rate of one request per cycle. No request is lost.

Figure 40-6 shows examples of valid DMA requests.

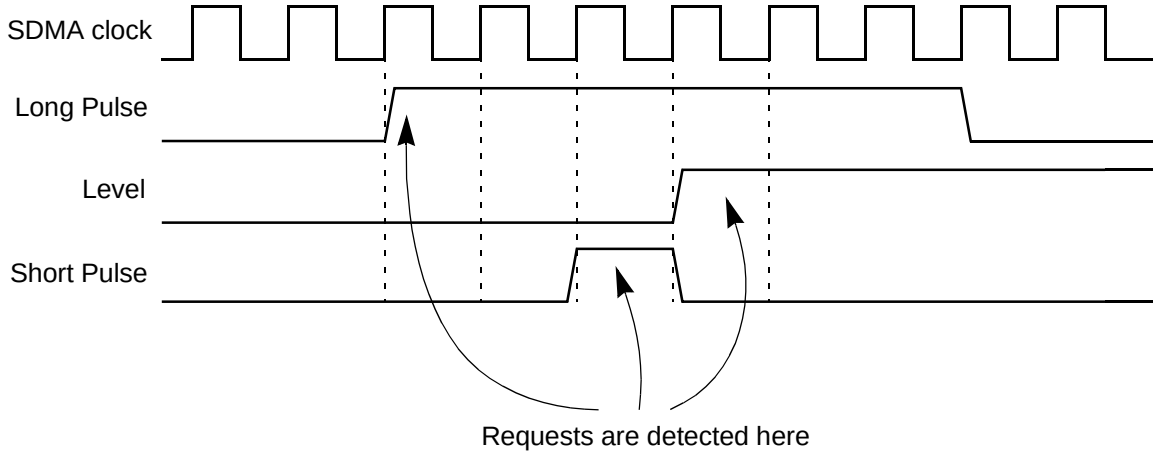


Figure 40-6. Examples of Valid DMA Requests

The DMA request inputs are connected to various sources that depend on the SoC. The exact list of DMA request inputs and their associated number is available in each respective project-specific chapter.

40.5.3.3 Mapping DMA Requests to Pending Channels

Whenever a DMA request is detected by the first stage, its number is used in the second stage to determine the channels that have to be activated. This is performed with an array of 32 registers that are 32 bits wide: There are 32 Channel Enable Registers (CHNENBLn), one register per DMA request. The DMA request number selects the Channel Enable Registers, and every bit of this 32-bit register indicates that the corresponding channel must be activated when it is a 1.

This information is passed on the EP register. For every bit of the Channel Enable Register that is set, the corresponding bit of the EP register is also set, and the remaining bits of EP are left unchanged. The transformation of EP is summarized by Equation 40-1:

$$\text{EP} = \text{EP or CHNENBLn} \qquad \textit{Eqn. 40-1}$$

The EP register is used to know which channels require service because they received a DMA request.

Typical contents of the CHNENBLn registers are all 0s, except for a single bit set. For example, a DMA request triggers one channel, but all 0s or several 1s are possible. One DMA request could activate several channels, and the channel execution sequence can be controlled by the channel priorities and numbers, as explained in the next sections. Table 40-2 illustrates an example configuration.

NOTE

From the table, the DMA request 0 is programmed to simultaneously trigger channels 0, 1, and 31. Also, DMA requests 30 and 31 are not used in this example. The remaining channels 2 to 30, are configured to be triggered by DMA requests 29 to 1, respectively.

Table 40-2. Channel Enable RAM Programming Example

DMA Request Number	Channel (31 … 0)
0	1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
1	0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
2	0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
3	0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
4	0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
5	0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
6	0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
7	0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
8	0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
9	0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
10	0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
11	0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
12	0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
13	0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
14	0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
15	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
16	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
17	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
18	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
19	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
20	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
21	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
22	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
23	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
24	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
25	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
26	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
27	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
28	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
29	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0

Table 40-2. Channel Enable RAM Programming Example (continued)

DMA Request Number	Channel																															
	31																															0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

40.5.3.4 Channel Overflow

A channel overflow occurs when a DMA request requires service from channel *n* by setting bit *n* of the register EP, but this bit is already set, meaning channel *n* is already pending. This can come from an overrun/underrun condition. This detection is possible only when the DMA requests are pulses, because a level-based DMA request stays high until it is serviced, even though an underrun or overrun condition occurs, thus preventing another edge detection of the DMA request.

The channel overflow information is saved in the 32-bit CHNERR register (1 bit per channel). You can configure the SDMA to trigger an interrupt to the AP when there are 1s in CHNERR. Every bit of CHNERR is masked with the corresponding bit of INTRMASK and if it gives a 1, the corresponding bit of INTR is set, triggering the interrupt.

40.5.3.5 Runnable Channels Evaluation

The EP register is used in conjunction with several other 32-bit registers to determine the channels that are runnable. Registers EO, DO, HO, and HE, are controlled by the AP. EP is controlled by the DMA requests and their mapping to channels.

Several channels may be runnable at any given time. The i^{th} channel is runnable if (and only if) the condition below is true:

(HE[i] or HO[i]) and (DE[i] or DO[i]) and (EP[i] or EO[i]) ***Eqn. 40-2***

After reset, the HE[i], HO[i], DE[i], EP[i], and EO[i] bits are all cleared whereas the DO[i] bits are all set. This allows systems, such as this device, where the BP side of the SDMA is not used ignore DE and DO. When DO[i] is set, the scheduler condition becomes:

(HE[i] or HO[i]) and (EP[i] or EO[i]) ***Eqn. 40-3***

NOTE

The SDMA's interfaces to the BP are unconnected on this device. DE[i] will always be zero for this device, and so DO[i] should always be set to 1 so that the relationship in Equation 40-3 effectively controls the runnable channel selection.

The registers in these equations are controlled as follows:

- AP (host) channel enable flag HE[i] may be set or cleared by the AP with the HSTART and STOP_STAT registers. It can also be cleared by the i^{th} channel script.

Typical usage is for the AP to set this flag to activate the channel. The flag is cleared by the SDMA core when the transfer is done.

- BP channel enable flag DE[i] is always zero since the BP interface is not connected for this device.
- Externally triggered channel pending flag EP[i] is set by the scheduler when the channel was activated by a DMA request. It can be cleared by the i^{th} channel script.
- The AP channel override flag HO[i] may be set or cleared by the AP. When set, it enables the i^{th} channel to run without the involvement of the AP.

  Typical usage is for the AP to set this flag for channels that do not need AP supervision such as channels that are controlled by DMA request events (EP).
- BP channel override flag DO[i] may be set or cleared by the AP. The BP interface is not connected on this device, and thus DO should always be set to 1so that the runnable channel evaluation considers only HO, HE, EP, and EO.
- Externally triggered channel override flag EO[i] may be set or cleared by the AP. When set, it prevents the i^{th} channel from stopping and stalling on incoming peripheral DMA requests. This is the case when the channel is not handling data transfers with peripherals (for example, a memory to memory transfer).

The SDMA can clear the HE[i], and EP[i] bits by means of a `done` or `notify` instruction. The `done` instruction causes a reschedule; thus, enabling another channel to preempt the current one, while the `notify` instruction does not. The `done` and `notify` instructions can clear either HE[i] or EP[i] (never more than one at a time).

Table 40-3. Runnable Channel Selection Control

Register	Set by	Cleared By
HO	Write to HOSTOVR register	Write to HOSTOVR register
HE	Write to HSTART register	Write to STOP_STAT register or by the channel script with the `done` or `notify` instructions.
DO	Write to DSPOVER register	Write to DSPOVER register
DE	Never since BP interfaces are not connected.	Always zero since BP interfaces are not connected.
EO	Write to EVTOVER register	Write to EVTOVER register
EP	Set by external DMA request event input.	By the channel script with the `done` or `notify` instructions

40.5.3.6 Next Channel Decision Tree

The next channel number is computed from the runnable channels list, the current channel number, and their respective priorities. It is re-evaluated every cycle, but is only used when the current channel yields or terminates by executing a `yield`, `yieldge`, or `done` instruction.

The decision tree is based on the selection of the runnable channel that has the highest priority. The highest priority channel is selected according to the following rules:

- Runnable channels are sorted by priority.

- If one of the channels with the highest priority had been preempted by a channel with a higher priority, but did not want to yield to a channel of the same priority (for example, it executed a `yield`, not a `yieldge`), it is elected as the next channel.
- The channels that belong to the highest priority group are sorted by their number and the channel that has the highest number in this group becomes the next channel.

When the current channel requires a reschedule with a `yield`(`ge`) or a `done` instruction, the context switch decision is based on the instruction parameter, the current channel number and priority, and the next channel number and priority. The possible cases are all listed in the Table 40-4. The grayed cells in Table 40-4 correspond to unusual cases that should not occur with a typical usage of the SDMA.

Table 40-4. Channel Switching Decision with a yield, yield(ge), or done

Instruction	Current Channel	Next Channel	Priorities Comparison	New Running Channel/Comments
`yield` (`done` 0)	Runnable	Not runnable	none	Current
	Runnable	Runnable	Current > Next	Current
			Current = Next	Current
			Current < Next	Next[1]
	Not runnable	Not runnable	none	none[2] (occurs when the channel was disabled by the AP)
	Not runnable	Runnable	none	Next[1] (occurs when the channel was disabled by the AP)
`yieldge` (`done` 1)	Runnable	Not runnable	none	Current
	Runnable	Runnable	Current > Next	Current
			Current = Next	Next[3]
			Current < Next	Next[1]
	Not runnable	Not runnable	none	none[2] (occurs when the channel was disabled by the AP)
	Not runnable	Runnable	none	Next[1] (occurs when the channel was disabled by the AP)
`done` (`done` >1)	Not runnable	Not runnable	none	none[4]
	Runnable	Not runnable	none	Current[3] (occurs when the `done` instruction does not disable the channel runnable condition)
	Not runnable	Runnable	none	Next[1]
	Runnable	Runnable	none	Current or Next[4] (occurs when the `done` instruction does not disable the channel runnable condition)

1 The operation is a context switch: the current channel script execution is stopped, its context is saved; the next channel context is restored and its script execution resumes
2 Current channel context is saved and SDMA enters IDLE mode
3 Current channel context is saved, then restored, and the current channel script resumes execution
4 Current channel context is saved, then it is included in the list of runnable channels, the highest priority channel is restored and its script resumes execution

Finally, when the SDMA is in IDLE mode and a runnable channel is elected as the next channel, its context is immediately restored and the script execution resumes.

The *combinatorial-decision* tree supports dynamic modifications of the EP, EO, HE, HO, and DO flags as well as dynamic modifications of the channel priorities. The propagation times are detailed in Section 40.5.3.8, "Scheduler Pipeline Timing Diagram."

The decision tree status is available in the PSW register, which is continuously updated. It contains the next channel priority, the next channel number, the current channel priority, and the current channel number. When a priority is read as 0, it means the channel is not runnable.

A few examples of decisions are presented below:

- Channel 31 is running with priority 5, channels 13 and 24 are pending with the same priority 5; channel 24 is eligible as the next channel since 24 > 13.
- Channel 31 is running with priority 7, channels 13 and 24 are pending with priority 5; channel 31 is the next channel because its priority is greater than the other pending channels.
- Channels 7, 23, and 29 are pending with the same priority. Channel 7 is active and runs a `yieldge`; it is preempted by channel 29. After a period of time, channel 29 runs a `yieldge`, it is then preempted by channel 23 that is the selected channel since channel 29 is the current channel. Later, channel 23 runs a `yieldge` and is preempted by channel 29. Channels 23 and 29 will go on switching after every `yieldge` until one of them terminates. It is only at that point that channel 7 becomes eligible again.
- Channel 11 is running with priority 3, and channel 15 is pending with priority 4. When the channel 31 script executes a `yield` instruction, it gets preempted by channel 15; then channels 6 and 18 with priority 3 become pending. Because channel 11 was preempted after executing a `yield` and there is no pending channel with a strictly greater priority, it is eligible as the next channel (although its number 11 < 18).

40.5.3.7 Scheduler State Diagram

The Figure 40-7 summarizes the behavior of the SDMA scheduler with details about the exact mechanism of the priority decision tree. It is important to understand the scheduler is a hardwired pipeline, which means all the stages are performed simultaneously every cycle, but a change on any given stage is reflected on the next stage after the delays presented in Section 40.5.3.8, "Scheduler Pipeline Timing Diagram."

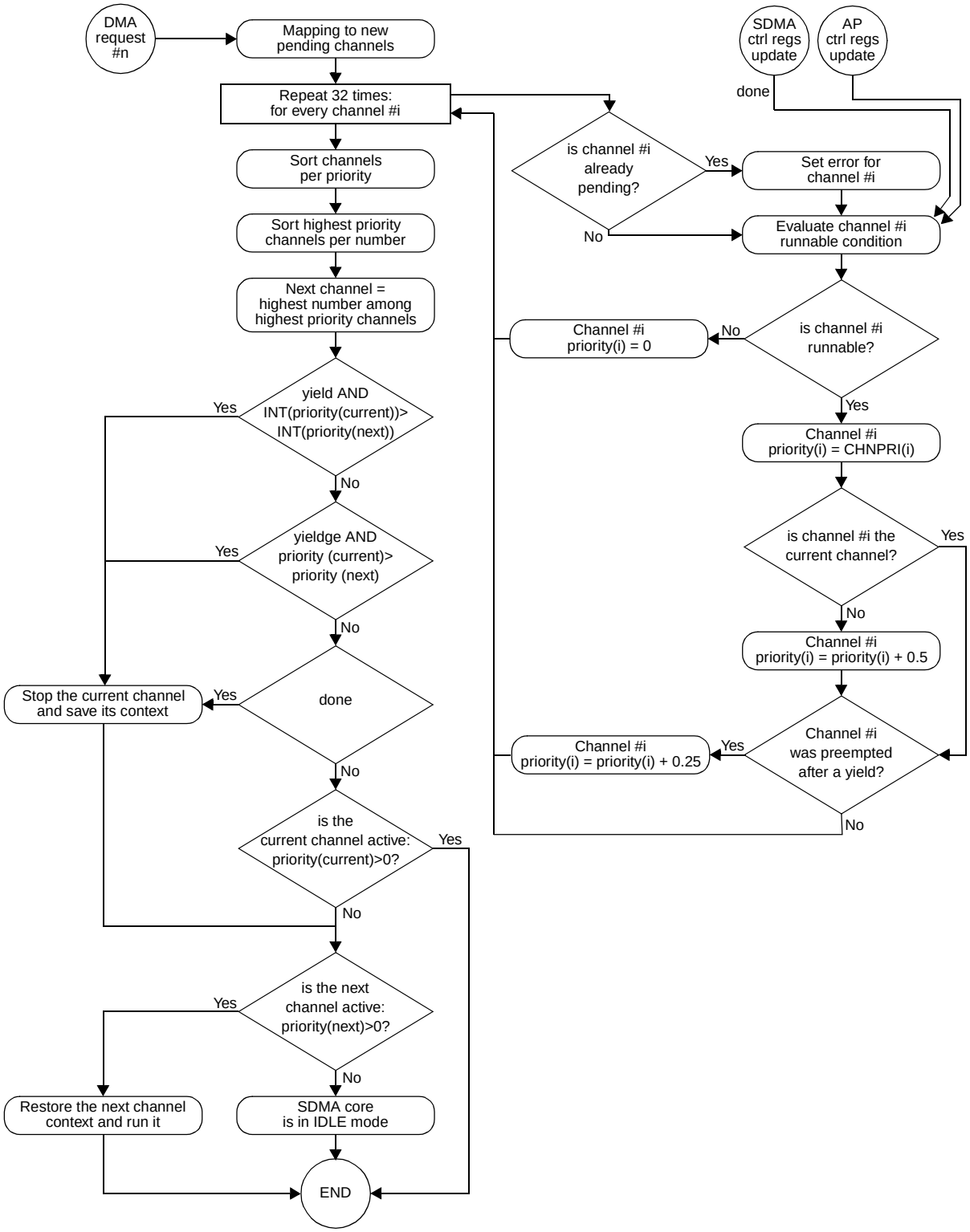


Figure 40-7. Scheduler State Diagram

40.5.3.8 Scheduler Pipeline Timing Diagram

The SDMA scheduler process of DMA-request and control-register modifications is not immediate. Figure 40-8 shows the exact delays of all the tasks. The reference clock is the SDMA core clock.

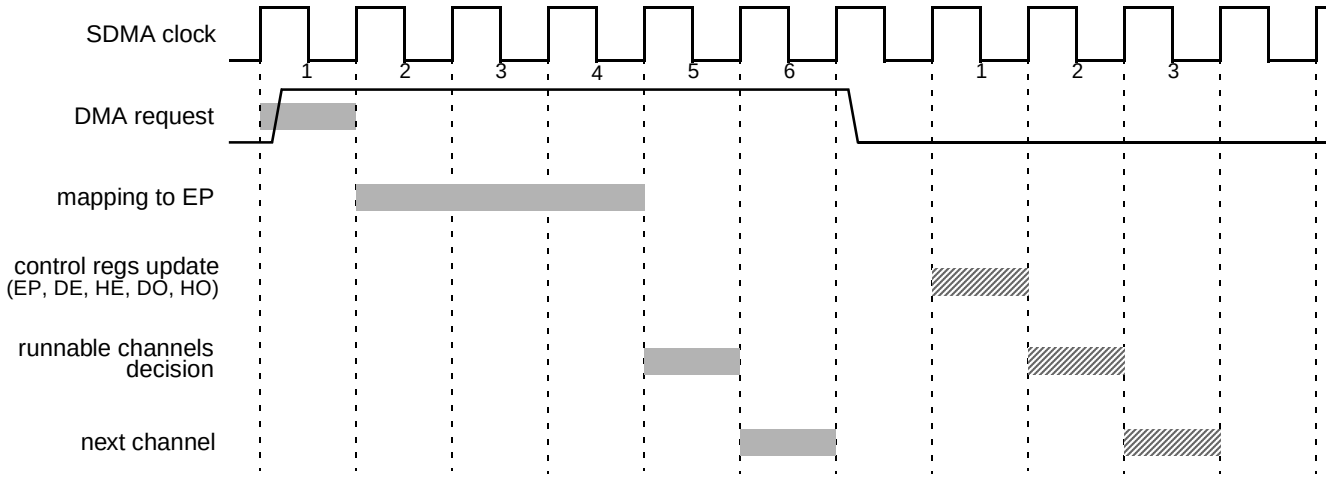


Figure 40-8. Scheduler Timing Diagram

Two numbers can be inferred from this timing diagram. First, it takes six SDMA core clock cycles to update the next channel from a DMA request. Second, it takes three SDMA core clock cycles to update the next channel from a direct modification of the condition registers (EP, DO, HE, or HO) by any processor. The processors that can modify these bits include SDMA with a `done` instruction or the AP with a write access through the corresponding control port on their respective peripheral bus).

40.5.3.9 Channel-DMA Request Mapping

The 32 DMA request inputs to the SDMA scheduler are listed in project-specific chapters. Refer to the respective chapters for this information.

40.5.3.10 Examples: How to Start a Channel

A channel can be started when Equation 40-2 is true for channel *i*:

(HE[i] or HO[i]) and (DE[i] or DO[i]) and (EP[i] or EO[i])

Once this equation is true, the scheduler can start this channel according to the priority of all pending channels. Several examples of configuration are listed below:

1. To start a channel triggered by AP software:
   - Initially, configure HO[i]=0, DO[i]=1, and EO[i]=1 using registers indicated in Table 40-3.
   - AP software triggers the channel by writing to the HSTART register to set HE[i]=1, thereby setting Equation 40-2 true.
2. To start a channel triggered by DMA request event.
   - Initially, configure HO[i]=1, DO[i]=1, and EO[i]=0 using registers indicated in Table 40-3.

— The DMA request is asserted to trigger the channel by setting EP[i]=1, which makes Equation 40-2 true.

40.5.4 Context Switching

On execution of a `done` or `yield`(`ge`) instruction, the current channel may be changed either because it has finished (which necessarily happens when the `done` instruction is executed), or it was preempted by a higher priority channel (which is possible but not systematic when the `yield`(`ge`) is executed).

40.5.4.1 Context Switch Modes

The exact procedure to save the context of the old channel, and to restore the context of the new channel depends on the context switch mode selected by the AP in the CONFIG control register. The following are the context switch modes:

- By default, the “dynamic” context switch is set. This mode provides the most efficient context switch for an average of eight cycles to stop the current channel, save its context, restore the next channel context, and resume its execution. It consists of saving modified registers of the current channel in the background (for example, during the channel execution)—which leaves very few registers to save when the switch is decided—resuming execution of the next channel as soon as possible (for example, when the minimal set of registers is restored), and continuing the restore phase during this execution.
- In “dynamic with no loop” mode, the same principle is followed except the modified registers are only saved in the background when the loop flag is not set. This mode offers almost the same effectiveness as the previous one, but it prevents the system from accessing the RAM during loops to save power. This is the recommended mode for an efficient context-switch when the loop bodies are short.
- In “dynamic power” mode, no background saving is performed, which reduces power consumption to the minimum. The modified registers are only saved when the context switch starts. The restore phase is the same as before. This is the mode that achieves the optimal power consumption at the cost of a slower context-switch.
- In a “static” context switch, all the registers are saved when a context switch is decided, and all the registers are restored before starting the execution of the new channel. This mode enables a predictable behavior of the context switch since all the registers are restored prior to the channel start and all registers are saved after the channel termination.

40.5.4.2 Context Switch Procedure

The Program Control Unit goes into the *save* state, the current context is spilled into memory, and the next channel context is restored according to the context-switch mode that was selected by the AP.

The context switch procedure is as follows:

1. Load the current context’s spill base address.
2. Spill the modified registers of the current channel to memory according to the selected context switch mode while the channel is running.

On a `done` or `yield`(`ge`) that causes the channel preemption, the PCU goes into the *save* state. In *static* mode, all the registers are saved; whereas, in either *dynamic* mode, the registers that were modified but not yet saved are then saved, and the PCU registers and flags are finally saved.

3. Put the SDMA core into *sleep* and wait for new channels to be serviced. This step is skipped if there are pending channels when the current channel is saved.

   As soon as there is at least one pending channel, the PCU goes into its *restore* state to restore the context of the channel that was elected by the scheduler.

   Once a channel is elected, it remains the current channel until its script requests a rescheduling operation with a `done` or `yield`(`ge`) instruction. That means the current channel cannot be modified by the AP, even if it is no more runnable or if its priority is modified.

   The AP can however force a reschedule by writing the corresponding bit in the CONFIG register, which has the same effect as if the script had executed a `done` instruction. That feature should only be used to stop the SDMA in emergency cases.

4. Load the context base-address of the new channel.

   In “static” mode, all the registers are restored. In either “dynamic” modes, only the PCU registers are restored.

   The new channel is running. In “static” mode, no more activity regarding context restoring or saving is performed. In either “dynamic” modes, the registers are restored in the background every time an access to the context RAM is possible, and priority is given to restoring the registers that are required by the next instruction to be executed. When a register has not been restored and the next instruction needs it, this instruction gets stalled until the register was restored.

   In “dynamic” and “dynamic with no loop” modes, background saving of dirty registers is performed every time an access to the context RAM is possible and allowed by the context switch mode.

NOTE

The contents of a channel context space in the context RAM depends on the selected context switch mode. In “dynamic” and “dynamic with no loop” modes, the contents of the context RAM tend to match the contents of the SDMA registers (except for the PCU registers and flags that are never saved in the background). In “dynamic power” and “static” modes, the contents of the context RAM remain unchanged until the channel terminates with a `done` or gets preempted.

40.5.4.3 Context Map in Memory

Refer to Section 40.11.4, “Context Switching.”

40.6 Functional Units

The functional units are small systems that are used by the SDMA core to perform complex calculations like the CRC unit, or to handle data transfers between the core and a bus domain external to the SDMA.

The SDMA core is able to control and exchange data with these systems by sending instructions and reading or writing data from/to the functional units' registers via the FUBUS. This is done with the `ldf` and `stf` instructions.

The following sections provide introductions to the available functional units. Section 40.16, "Functional Units Programming Model" provides descriptions the functional units' behaviors.

40.6.1 CRC Calculation Unit

The Cyclic Redundancy Check (CRC) unit can perform CRC calculation. A single byte of data can be processed every cycle, but up to four bytes can be simultaneously loaded.

The CRC unit supports the following set of polynomials:

```
CRC32:   X^32+X^26+X^23+X^22+X^16+X^12+X^11+X^10+X^8+X^7+X^5+X^4+X^2+X+1
CRC16:   X^16+X^15+X^2+1
CCITT16: X^16+X^12+X^5+1
IS136:   X^12+X^10+X^8+X^5+X^4+X^3+1
CRC10:   X^10+X^9+X^5+X^4+X+1
CRC8:    X^8+X^2+X+1
parity:  X^8+1
```

40.6.1.1 CRC Structure

Figure 40-9 describes the overall structure of the CRC unit and introduces its registers that are accessible by the SDMA core via the FUBUS.

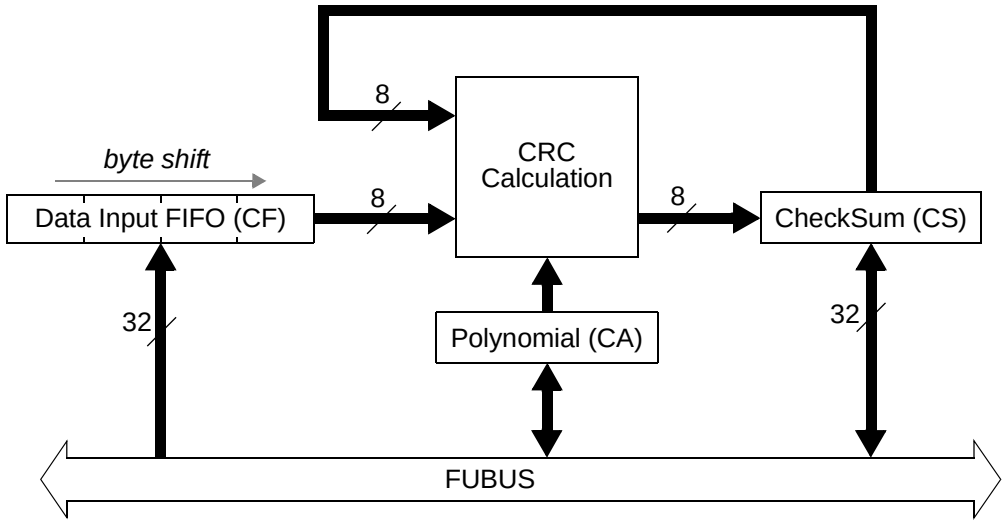


Figure 40-9. CRC Structure

40.6.1.2 CRC Data Processing

CRC processing requires the following three stages:

1. The preliminary initialization stage consists of selecting the desired polynomial, and storing the initialization pattern into the checksum register.

2. Data processing itself, which comes to feeding incoming bytes to the CRC input FIFO — bytes can be written one at a time (8-bit write access from the SDMA core), by pairs (16-bit write), or groups of four (32-bit write). One byte is processed every cycle: Any subsequent access may stall the SDMA core until completion of the previous calculation.
3. The CRC result (or checksum) can be retrieved at any time, but all data must be processed before it is made available.

40.6.1.3 CRC Registers

The CRC enables reading and writing to three register addresses, which trigger the operations described in Section 40.6.1.2, "CRC Data Processing." The following are the three registers:

- *CA (CRC algorithm)*—The CA selects the desired polynomial. Reading and writing this register correspond to retrieving and changing the polynomial.
- *CS (CRC checksum)*—Writing to this register initializes the checksum accumulator, and reading this register yields the result from the CRC calculation (the result width depends on the polynomial size).
- *CF (CRC FIFO)*—Writing to this register loads the data into the input FIFO and triggers the CRC calculation process. It is not possible to read this register.

40.6.1.4 CRC Summary

Every operation mentioned in Section 40.6.1.3, "CRC Registers" takes time to be executed by the CRC unit. When the CRC receives a command, it immediately acknowledges the SDMA core provided it is not busy completing a previous operation. Therefore, wait-states are inserted by the CRC when a command succeeds another command that is not yet competed. Table 40-5 lists the number of cycles that the CRC takes to execute every possible command. When an operation lasts one cycle, it means the CRC is able to process another command in the next clock cycle (for example, no wait-state is inserted by the CRC because of the former operation that is processing).

Table 40-5. CRC Processing Summary

Operation	Command	Delay	Comments
Set the polynomial	Write CA	1	
Retrieve the polynomial	Read CA	1	
Initialize the checksum	Write CS	1	
Retrieve the checksum	Read CS	1	
Store 1 byte to process	Write CF	1	
Store 2 bytes to process	Write CF	2	CRC is busy for 1 cycle
Store 4 bytes to process	Write CF	4	CRC is busy for 3 cycles

40.6.2 Burst DMA Unit

The burst DMA unit enables the SDMA core to perform data transfers to and from the AP memory. It is optimized for accessing SDRAM-like devices. It does not provide control to assign a privilege level to the DMA access. The burst DMA unit provides the SDMA with means to do the following:

- Perform up to 8-beat read and write bursts to the AP memory, which optimizes throughput when accessing SDRAM-type devices because of an internal, 36-byte FIFO
- Access the AP memory at once or twice the SDMA core frequency
- Copy data from one AP memory location to another AP memory location at the AP bus speed, which provides a very high throughput
- Control the method for addressing the AP memory (automatic increment of addresses or frozen addresses—the former aimed at accessing RAM-like memory and the latter aimed at accessing single-address FIFOs)
- Enable or disable automatic prefetch when reading data from the AP memory. When the prefetch mode is selected, the burst DMA automatically triggers external bursts to fill its FIFO without waiting for the SDMA core to request the corresponding data, greatly improving throughput.
- Rely on the DMA to automatically flush its FIFO content when there is enough data to generate an 8-beat burst to the AP memory. Or, it forces a flush when a data transfer must terminate.

  In the former case, the SDMA core may only be stalled when it tries writing data and there is not enough room left in the FIFO. In the latter case, the core is stalled until the data is effectively written to the AP memory.

  In automatic flush mode, the core receives an acknowledge that does not reflect the actual error status when the data is effectively written into the AP memory. This error status is retrieved by a later access to the burst DMA. Terminating a write data transfer with a forced flush command guarantees that any bus error to the AP memory is caught.
- Handle address alignment issues between the AP memory map and the SDMA core data. This enables the core to read or write 32-bit data from the burst DMA, whereas the corresponding AP address is not 32-bit aligned. This drastically improves the SDMA scripts' efficiency since the same loop that transfers 32 bits at a time can be used regardless of the start and end addresses in the AP memory space.

This unit structure and registers are described in Section 40.6.2.1, "Burst DMA Structure" and Section 40.6.2.2, "Burst DMA Registers."

40.6.2.1 Burst DMA Structure

The burst DMA is depicted in Figure 40-10. It is essentially made up of a 36-byte FIFO, address registers, and a controlling state-machine. The 36-byte FIFO enables eight-word buffering with address alignment, and the state-machine manages clock adaptation when required.

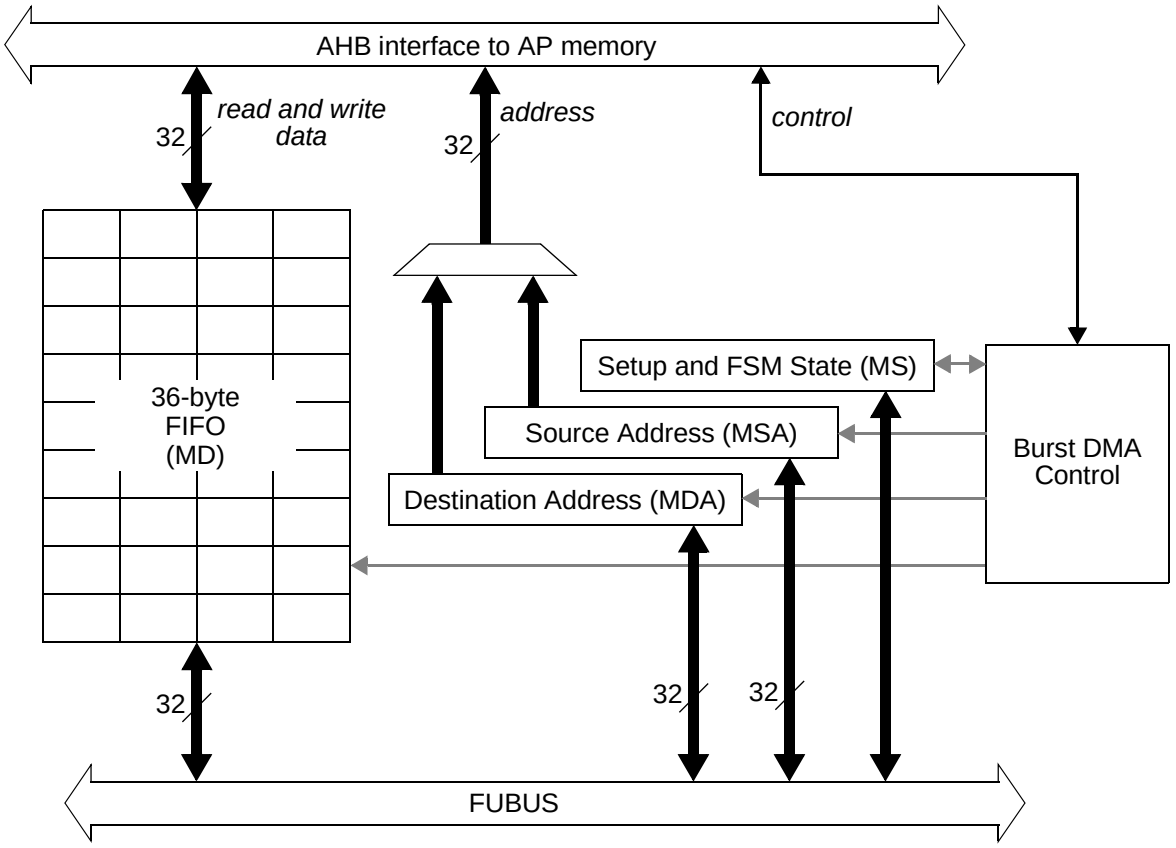


Figure 40-10. Burst DMA Structure

40.6.2.2 Burst DMA Registers

There are four registers, as follows, that may be accessed from the SDMA core:

- MSA (Memory Source Address) — Holds the source byte address in the AP memory map for reading data from this location. This register is automatically modified every time the core reads new data from the FIFO.
- MDA (Memory Destination Address) — Holds the destination byte address in the AP memory map for writing data to this location. This register is automatically modified every time the core writes new data into the FIFO.
- MD (Memory Data) — Labels the 36-byte FIFO access point: Reading a byte, halfword, or word from MD respectively retrieves the first 1, 2, or 4 bytes of the FIFO (for example, the bytes that were stored first by the DMA state-machine when transferring data from the AP memory).

  When the FIFO does not hold as many bytes as required by the SDMA core, the core is stalled until the missing bytes are read from the AP memory. In the case of prefetch mode, the DMA controller decides when it should start a burst to AP memory in order to reduce the risk to not have the required data for the future accesses of the core. When there is no prefetching, a burst is triggered when the required data is not available in the FIFO.

  Writing a byte, halfword, or word to MD stores 1, 2, or 4 bytes, respectively, at the end of the FIFO (for example, these bytes are transmitted to the AP memory after all the other bytes that were

previously stored in the FIFO). When the FIFO does not have enough room left to hold the written data, the SDMA core is stalled until a sufficient amount of FIFO contents are flushed out to the AP memory. Flushing is decided by the DMA controller when there are enough bytes in the FIFO to perform the largest allowed burst to AP memory (the exact size depends on the burst start address and the AHB 1 Kbyte boundary rule). However, the SDMA core has the ability to force the flushing operation at any time, for example, when at the end of the data transfer, prior to channel closure.

- MS (Memory Setup) — Contains the state of the burst DMA control, the two flags that define whether each address register is incremented after every access to the external memory, and another flag that is set when a bus error occurred.

40.6.2.3 Data Transfers

Three typical usages have been identified that involve the burst DMA: the data transfer startpoint, the endpoint, or both. Every case requires a different procedure, as listed in Section 40.6.3.3.1, "Data Retrieval from the AP Memory or Peripheral," Section 40.6.3.3.2, "Storing Data into the AP Memory or Peripheral," and Section 40.6.3.3.2, "Storing Data into the AP Memory or Peripheral."

40.6.2.3.1 Data Retrieval from the AP Memory

The following steps retrieve data from AP memory using the burst DMA unit:

- Set up the MS flags to reflect the mode for the source address (incremented or frozen according to the type of accessed device: memory or peripheral FIFO), then initialize the source address register itself (DSA).
- Read data from the FIFO using the *ldf MD* instruction as many times as needed. If an error occurred during the fetch from AP memory, the DMA control tags the error status on the data and the SDMA core SF flag is set when reading this data from the FIFO.

40.6.2.3.2 Storing Data Into the AP Memory

The following steps store data from AP memory using the burst DMA unit:

- Set up the MS flags to reflect the mode for the destination address (incremented or frozen according to the type of accessed device: memory or peripheral FIFO), then initialize the destination address register itself (MDA).
- Store data into the FIFO using the *stf MD* instruction as many times as needed.
- When the transfer is finished and if the DMA worked in automatic flush mode, force the flush of the FIFO. This instruction is stalled until all the FIFO data is effectively sent to the AP memory and the error status of the transfer is available in the DF flag.

40.6.2.3.3 Transferring Data Between Two AP Memory Locations

The following steps copy data between two AP memory locations using the burst DMA unit:

- Set up the MS flags to reflect the modes for the source and destination addresses (all the combinations are possible), then initialize the source address register (MSA) and the destination address register (MDA). Both addresses must be word-aligned.

- Use as many *stf MD* instructions with the *COPY* flag as needed. Every instruction triggers a burst read of a given number of words from the source address (this number is provided to the burst DMA via the SDMA core general purpose register, which is referenced in the *stf* instruction). Once all the data is loaded into the FIFO, the DMA empties it with a write burst of the same count to the destination address. The DMA acknowledges prior to instruction completion, which frees the SDMA core for other tasks at no delay cost.
- Once the transfer is done, there should be a final access to the burst DMA to check the error status.

40.6.3 Peripheral DMA Unit

The peripheral DMA unit is the second functional unit that connects the SDMA to the AP memory. Unlike the burst DMA, it does not support burst transfers and is optimized for accessing peripherals. It does not provide control to assign a privilege level to the DMA access. Its feature list comprises the following:

- Access to the AP peripherals or memory at once or twice the SDMA core frequency
- Data copy from one AP memory location to another AP memory location at memory bus speed, improving throughput
- Control of the method for addressing the AP memory (automatic increment or decrement of addresses or frozen addresses, the first ones aimed at accessing RAM-like memory and the last one aimed at accessing single-address FIFOs)
- Selectable automatic prefetch when reading data from the AP memory. In prefetch mode, the peripheral DMA automatically fetches another data—without waiting for the SDMA core to request it—when its data register is empty, which improves the throughput
- Selectable automatic flush. In this mode, the SDMA core may only be stalled when it tries writing data and the previous write operation is not finished yet; whereas, in forced flush mode, the core is stalled until the data is effectively written to the AP memory.

  In automatic flush mode, the core receives an acknowledge that does not reflect the actual error status when the data is effectively written into the AP memory or the peripheral. This error status is retrieved by a later access to the peripheral DMA. Terminating a write data transfer with a forced flush command guarantees that any bus error to the AP memory has been caught.

This unit structure and registers are described in Section 40.6.3.1, “Peripheral DMA Structure” and Section 40.6.3.2, “Peripheral DMA Registers.”

40.6.3.1 Peripheral DMA Structure

The peripheral DMA is shown in Figure 40-11. It is made up of a 32-bit data register, two address registers, and a controlling state-machine. The state-machine manages clock adaptation, when required.

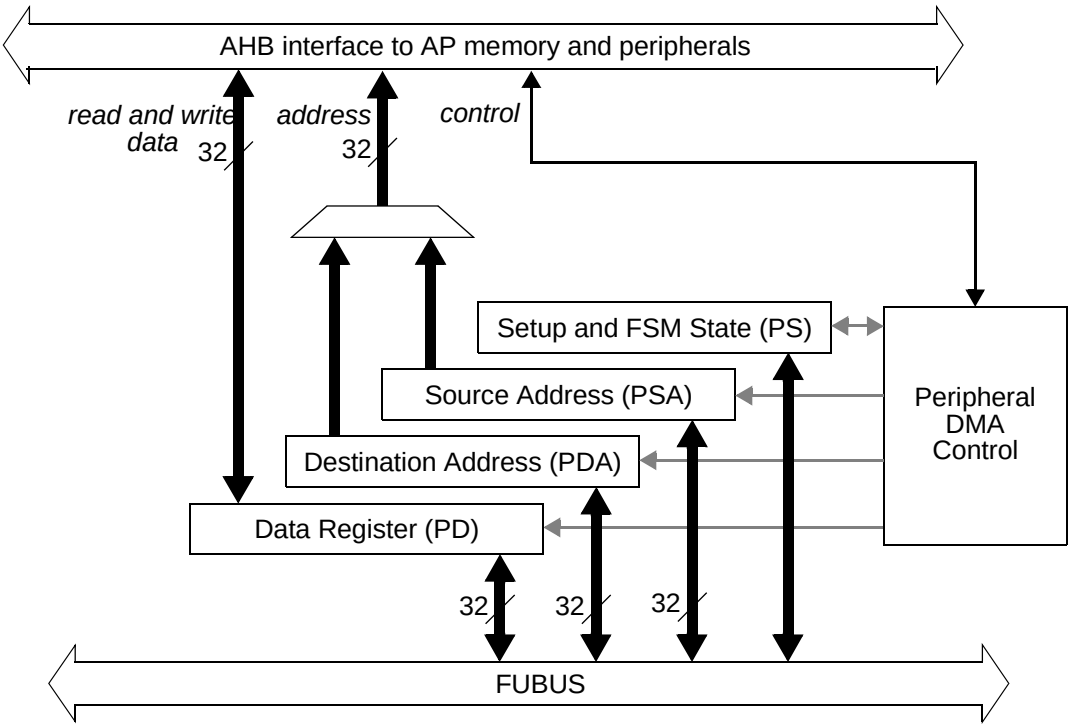


Figure 40-11. Peripheral DMA structure

40.6.3.2 Peripheral DMA Registers

According to Figure 40-11, the peripheral DMA has four registers that may be read or written by the SDMA core:

- *PD (Peripheral Data)* is the DMA 32-bit data register.
- *PSA (Peripheral Source Address)* holds the source byte address in the AP memory map for reading data from this location. This register is automatically modified every time the core reads a new data from PD.
- *PDA (Peripheral Destination Address)* holds the destination byte address in the AP memory map for writing data to this location. This register is automatically modified every time the core writes a new data into PD.
- *PS (Peripheral Setup)* contains the state of the peripheral DMA control, two configuration fields that define the way address registers are modified after every data access, two additional configuration fields that define the data size to access the source and destination devices, and another field that contains the latest transfer error status.

40.6.3.3 Peripheral DMA Data Transfers

There are three typical usages that involve the peripheral DMA, whether it is the data transfer start-point, endpoint, or both. Every case requires a different procedure, as described in Section 40.6.3.3.1, “Data Retrieval from the AP Memory or Peripheral,” Section 40.6.3.3.2, “Storing Data into the AP Memory or Peripheral,” and Section 40.6.3.3.3, “Transferring Data Between Two AP Memory Locations.”

40.6.3.3.1 Data Retrieval from the AP Memory or Peripheral

The following steps retrieve data from AP memory using the peripheral DMA unit:

- Set up the PS fields to reflect the mode and data size for the source (incremented, decremented, or frozen address register; 8-bit, 16-bit, or 32-bit data transfers), then initialize the source address register itself (PSA) with an address that is aligned to the programmed data size.
- Read data from PD using the `ldf PD` instruction as many times as needed. If an error occurs during the fetch from the AP memory or peripheral, the DMA control tags the error status on the data and the SDMA core SF flag is set when reading this data from PD.

40.6.3.3.2 Storing Data into the AP Memory or Peripheral

The following steps store data to AP memory using the peripheral DMA unit:

- Set up the PS fields to reflect the mode and data size for the destination (incremented, decremented, or frozen address register; 8-bit, 16-bit, or 32-bit data transfers), then initialize the destination address register itself (PDA) with an address that is aligned to the programmed data size.
- Store data into PD using the *stf PD* instruction as many times as needed.
- When the transfer is finished and if the peripheral DMA worked in automatic flush mode, force the flush of PD. This instruction is stalled until PD contents are effectively sent to the AP memory or peripheral, and the error status of the transfer is available in the DF flag.

40.6.3.3.3 Transferring Data Between Two AP Memory Locations

The following steps copy data between two AP memory locations using the peripheral DMA unit:

- Set up the PS fields to reflect the modes and data size for the source and destination addresses (all the combinations of addressing modes are possible, but both data sizes must be identical), then initialize the source address register (PSA) and the destination address register (PDA). Both addresses must be aligned with the programmed data size.
- Use as many *stf PD* instructions with the *COPY* flag as needed. Every instruction triggers a single read from the source address; a single write of the received data immediately follows. The DMA acknowledges prior to instruction completion, which frees the SDMA core for other tasks at no delay cost.
- Once the transfer is done, there should be a final access to the peripheral DMA to check the error status.

40.7 OnCE and PCU Debug States

The SDMA has two different debug modes in which the OnCE performs debug instructions. Refer to Figure 40-4 for an example of the PCU states in debug. The following are the two debug states:

- When a channel is running (that is, when CCR and CCPRI are different from 0, which can be read in the PSW register), SDMA can execute a `SoftBkpt` instruction from the channel script or receive a debug request. When either happens, the SDMA enters its “Classical” *Debug* state, which is described in Section 40.16.5, “OnCE and Real-Time Debug.”

- When a channel is not running, the SDMA can be in *Sleep* state or in *Sleep after Reset* state. If a debug request is sent to the core, it enters its *Debug in Sleep* state. This debug mode works similarly to the "Classical" *Debug* state, except it returns to the original state (*Sleep* or *Sleep after Reset*) when the debug mode is left via the `exec_core` instruction of the OnCE. From this *Debug in Sleep* state, the SDMA can execute a program whereas no channel is running. If a new debug request is sent to the core or if a `SoftBkpt` is executed, it comes back to this *Debug in Sleep* state.

The OnCE is provided with several instructions that can be executed when the core is in either debug state. Table 40-6 summarizes the behavior of these OnCE debug instructions. There exists other secondary OnCE instructions that are described in Section 40.16.5, "OnCE and Real-Time Debug."

Table 40-6. SDMA in Debug Mode

Instruction	Debug	Debug in Sleep
`exec_once`	`exec_once` <instruction> SDMA executes the <instruction> and returns to the *Debug* state. The Program Counter (PC) is not incremented. This command must not be used with an instruction that modifies the PC value.	`exec_once` <instruction> SDMA executes the <instruction> and returns to the *Debug in Sleep* state. The Program Counter (PC) is not incremented. This command must not be used with an instruction that modifies the PC value.
`run_core`	`run_core` <instruction> SDMA executes the <instruction>, leaves the *Debug* state and continues executing the channel script from the position where it stopped. This command must not be used with an instruction that modifies the PC value.	`run_core` <instruction> SDMA executes the <instruction> and returns to its *Sleep* or *Sleep after Reset* initial state. This command must not be used with an instruction that modifies the PC value.
`exec_core`	`exec_core` <instruction> It is similar to `run_core` except it requires an instruction that changes the PC value (jump, branch...): the SDMA jumps to the new PC value, leaves the *Debug* state and starts executing instructions from this new PC value.	`exec_core` <instruction> If the previous state was *Sleep after Reset*, the SDMA returns to this state, and Chn0Addr value overrides the PC value. Otherwise, the SDMA jumps to the new PC value and starts executing instructions from this new PC.

NOTE

The feature `exec_core` in *Debug in Sleep* after *Sleep after Reset* was added for the Channel boot (channel 0) to allow the debugger to return to *Sleep after Reset* state with a new PC value. The SDMA will be ready to boot at the Chn0Addr address.

40.8 SDMA Clocks and Low Power Modes

The SDMA receives several root clocks from the SoC clock controller module and performs adaptive clock gating to optimize its power consumption. From a user standpoint, clock gating and power mode selection are fully automatized inside the SDMA. Root clock control is available from the SoC clock controller module. Refer to Section 40.8.1, "Root and Derived Clocks."

40.8.1 Root and Derived Clocks

The SDMA receives five clock sources, listed in Table 40-7, from the SoC with comments regarding their usage.

Table 40-7. Clocking Scheme

Source Clock	Typical Frequency	Comments
main	66 MHz	Source clock for the core and all its operations; this clock is thus used by most of the SDMA submodules.
AP AHB	133 MHz	AHB interface for the peripheral DMA and the burst DMA. It is balanced with the main clock source, and its frequency is either once or twice the main clock frequency.
AP peripheral	66 MHz	Connection to the AP peripheral bus. It is a sub-frequency of the main clock frequency.

40.8.2 Clock Gating and Low Power Modes

The SDMA automatically performs power saving without requiring user involvement. It implements two levels of clock gating.

40.8.2.1 Coarse Clock Gating

Every submodule clock comes from one of the five available sources, and is gated with the submodule specific enabling condition. Table 40-8 displays the submodule clocks and their source. It also indicates the relations that may exist between different submodules clock enables.

Table 40-8. Submodules Clocks

Submodule	Source Clocks	Enabling Condition and Comments	Related Enabling Conditions
Core	main	The core clock is running when the core is not in one of its sleep states (*Sleep* or *Sleep after Reset*) or there is a pending channel. Typically, the core clock is stopped once all the channels are processed and the core enters its sleep state. A new pending channel awakes the core clock.	None
Memories	main	The clock activation only occurs during a core access.	Disabled when Core clock is disabled
Scheduler	main	Its clock only runs when scheduling is needed: for example, when there are pending channels, upon reception of a DMA request, and anytime the AP modifies the channel running conditions.	None
AP Control	main AP peripheral	The AP *peripheral* clock is solely used to determine the frequency ratio with the *main* clock. The control registers' clock is based on *main*; it is active when the AP or the SDMA modifies the contents of one of these registers.	None
CRC	main	The CRC clock is based on *main* and is only active during data processing.	Disabled when Core clock is disabled

Table 40-8. Submodules Clocks (continued)

Submodule	Source Clocks	Enabling Condition and Comments	Related Enabling Conditions
Burst DMA	main AP AHB	The burst DMA has two clocks: The first clock is derived from *main* and drives registers that are connected to the FUBUS. The second clock is derived from AP *AHB* and drives registers that are connected to the AP AHB bus outside the SDMA. Both clocks are enabled during active phases of data transfers (for example, these clocks are turned off when the burst DMA is not used by the running channel script).	Disabled when Core clock is disabled
Peripheral DMA	main AP AHB	The peripheral DMA has two clocks: The first clock is derived from *main* and drives registers that are connected to the FUBUS. The second clock is derived from AP *AHB* and drives registers that are connected to the AP AHB bus outside the SDMA. Both clocks are enabled during active phases of data transfers (for example, these clocks are turned off when the peripheral DMA is not used by the running channel script).	Disabled when Core clock is disabled
OnCE	main	The OnCE clock is derived from *main* source clock. It is disabled by default. In order to use the OnCE, its clock must be explicitly turned on, either by enabling the OnCE access from the AP peripheral bus (register ONCE_ENB), or by driving the *clk_gating_off* input pin high. This is a SDMA input whose driver depends on the SoC implementation (typically a JTAG controller). The OnCE also receives the TCK input, which is the JTAG clock. It does not use it as a functional clock; the TCK input is sampled instead. Refer to Section 40.17.5.2, “Synchronization Implementation.”	When enabled, all other clocks are systematically on (clock gating is off)

40.8.2.2 Refined Clock Gating

The SDMA implements a second level of clock gating on a register-per-register basis. Unlike the first level that covers all the SDMA flip-flops, except the synchronizers (only five flip-flops are always running), the second level is only available for eligible registers, which amounts to about 90% of the SDMA flip-flops.

These gated registers are only clocked when the hardware logic detects a new data loading. This additional gating further reduces dynamic power consumption.

40.8.2.3 Low Power Modes and User Control

Power savings are automatically managed by the SDMA hardware without any user involvement; however, one can distinguish three different power modes: SLEEP, RUN, and DEBUG. Table 40-9 describes these modes, and shows how to switch from one mode to another.

Table 40-9. Power Modes

Power Mode	Submodules										Comments
	Core	Memories	Scheduler	AP Control	BP Control	CRC	Burst DMA	Peripheral DMA	BP DMA	OnCE	
SLEEP	off[1]	off	wait[2]	wait	wait	off	off	off	off	off	Set when the PCU state is either *Sleep* or *Sleep after Reset* and the SDMA is not in DEBUG mode. This is the default mode after reset.
RUN	on[3]	wait	wait	wait	wait	wait	wait	wait	wait	off	Set for the other PCU states that are reachable out of debug: *Program*, *Data*, *Change of Flow*, *Error in Loop*, *Debug*, *Functional Unit*, *Save,* or *Restore*.
DEBUG	on	on	on	on	on	on	on	on	on	on	Set regardless of the PCU state when clock gating is turned off to use the OnCE features (either *clk_gating_off* pin high or ONCE_ENB[0] set).

1 *off*: no clock

2 *wait*: only clocked when accessed or stimulated

3 *on*: clock is always running

It is possible to control the SDMA power mode. The procedures to force the SDMA into either mode are described in Section 40.8.2.3.1, "SLEEP Mode."

40.8.2.3.1 SLEEP Mode

This is the default mode after reset; therefore, resetting the SDMA forces this mode. However, the common procedure is as follows:

- Ensure the *clk_gating_off* pin is low and ONCE_ENB[0] is cleared.
- Disable all channels (via the STOP_STAT control register, and the HO, DO, EO if necessary).
- Wait for the active channels to complete or force a reschedule via the reschedule bit in the RESET register.
- The SDMA is in SLEEP mode making it possible to completely shut off its clock from the chip level clock controller using the procedure described in Section 40.8.2.4, "Stop Mode Response."

40.8.2.3.2 RUN Mode

This is the default mode when a channel is running:

- Ensure the *clk_gating_off* pin is low and ONCE_ENB[0] is cleared.
- Activate at least one channel (via the HSTART or DSTART control registers, a DMA request, and/or the HO, DO, EO register bits).

40.8.2.3.3 DEBUG Mode

The DEBUG mode must be set when one needs to use the debugging facilities of the SDMA:

- Ensure the SDMA clocks are running from the CCM.
- Set the *clk_gating_off* pin high or use the SDMA to set ONCE_ENB[0].

40.8.2.4 Stop Mode Response

The SDMA receives a stop request from the chip level clock controller. This request may be asserted when the chip enters the stop low power mode. If the SDMA is running when the request is received, then the SDMA will complete all pending channels before returning to the SLEEP state. The SDMA sends an acknowledgement to the clock controller when the SLEEP state is entered indicating that the SDMAs clocks can be turned off.

40.8.3 Reset

After reset (either received from the reset module or a software reset required by the AP), the SDMA is in IDLE mode. It will start its boot code located at address 0 once a channel is activated. Activating a channel can be done by the AP after programming a positive priority and setting the channel bit in the EVTPEND register.

40.9 Software Interface

A separate document exists that fully describes the SDMA Application Programming Interface (API): Refer to the latest revision of document MOT-SFS-IAPI-SAS-001 (version 0.04).

40.10 AP Memory Map and Control Register Definitions

The AP controls the SDMA by means of several interface registers. Those registers are described in Section 40.10, “AP Memory Map and Control Register Definitions.”

40.10.1 AP Memory Map

Table 40-10 provides the memory map for the AP control SDMA registers.

- DSPOVR Register (Section 40.10.3.6, “Channel DSP Override (DSPOVR)”) must be set to all 1’s (0xFFFF_FFFF).
- The DSPDMA bit in the CONFIG register (Section 40.10.3.14, “Configuration Register (CONFIG)”) should be set to zero.

Table 40-10. AP Memory Map

Address	Register	Access	Reset Value	Section/Page
General Registers				
0x53FD_4000 (MC0PTR)	AP (MCU) Channel 0 Pointer	R/W	0x0000_0000	40.10.3.1/40-43
0x53FD_4004 (INTR)	Channel Interrupts	R/W	0x0000_0000	40.10.3.2/40-44
0x53FD_4008 (STOP_STAT)	Channel Stop/Channel Status	R	0x0000_0000	40.10.3.3/40-44
0x53FD_400C (HSTART)	Channel Start	R/W	0x0000_0000	40.10.3.4/40-45
0x53FD_4010 (EVTOVR)	Channel Event Override	R/W	0x0000_0000	40.10.3.5/40-46
0x53FD_4014 (DSPOVR)	Channel DSP(BP) Override	R/W	0xFFFF_FFFF	40.10.3.6/40-46
0x53FD_4018 (HOSTOVR)	Channel AP Override	R/W	0x0000_0000	40.10.3.7/40-47

Table 40-10. AP Memory Map (continued)

Address	Register	Access	Reset Value	Section/Page
0x53FD_401C (EVTPEND)	Channel Event Pending	R	0x0000_0000	40.10.3.8/40-48
0x53FD_4024 (RESET)	Reset Register	R	0x0000_0000	40.10.3.9/40-48
0x53FD_4028 (EVTERR)	DMA Request Error Register	R	0x0000_0000	40.10.3.10/40-49
0x53FD_402C (INTRMASK)	Channel AP Interrupt Mask	R/W	0x0000_0000	40.10.3.11/40-50
0x53FD_4030 (PSW)	Schedule Status	R	0x0000_0000	40.10.3.12/40-51
0x53FD_4034 (EVTERRDBG)	DMA Request Error Register	R	0x0000_0000	40.10.3.13/40-52
0x53FD_4038 (CONFIG)	Configuration Register	R/W	0x0000_0003	40.10.3.14/40-53
0x53FD_4040 (ONCE_ENB)	OnCE Enable	R/W	0x0000_0000	40.10.3.15/40-54
0x53FD_4044 (ONCE_DATA)	OnCE Data Register	R/W	0x0000_0000	40.10.3.16/40-54
0x53FD_4048 (ONCE_INSTR)	OnCE Instruction Register	R/W	0x0000_0000	40.10.3.17/40-55
0x53FD_404C (ONCE_STAT)	OnCE Status Register	R	0x0000_E000	40.10.3.18/40-56
0x53FD_4050 (ONCE_CMD)	OnCE Command Register	R/W	0x0000_0000	40.10.3.19/40-58
0x53FD_4054 (EVT_MIRROR)	DMA Requests	R	0x0000_0000	40.10.3.22/40-60
0x53FD_4058 (ILLINSTADDR)	Illegal Instruction Trap Address	R/W	0x0000_0001	40.10.3.20/40-59
0x53FD_405C (CHN0ADDR)	Channel 0 Boot Address	R/W	0x0000_0050	40.10.3.21/40-59
0x53FD_4070 (XTRIG_CONF1)	Cross-Trigger Events Configuration Register 1	R/W	0x0000_0000	40.10.3.23/40-61
0x53FD_4074 (XTRIG_CONF2)	Cross-Trigger Events Configuration Register 2	R/W	0x0000_0000	40.10.3.23/40-61
0x53FD_4100+n*4 (CHNPRIn)[1]	Channel Priority Registers	R/W	0x0000_0000	40.10.3.24/40-64
0x53FD_4080+n*4 (CHNENBLn)[2]	Channel Enable RAM	R/W	Undefined	40.10.3.25/40-65

1 CHNPRIn: For n= 0 to 31

2 CHNENBLn: For n = 0 to 31

40.10.2 Register Summary

Figure 40-12 provides the definitions that serve as a key for the AP control SDMA register summary.

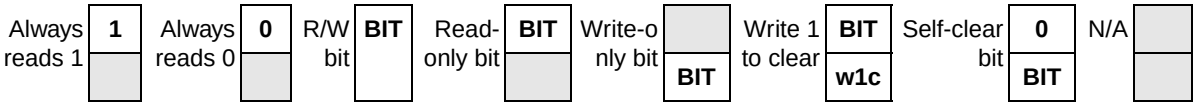


Figure 40-12. Key to Register Fields

Table 40-11 provides a key for register figures.

Table 40-11. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writeable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

All registers are clocked with the SDMA clock (which means the AP must ensure that the SDMA clock is running when it wants to access any register).

Table 40-12. AP SDMA Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x53FD_4000 (MC0PTR)	R	MCOPTR[31:16]															
	W																
	R	MCOPTR[15:0]															
	W																
0x53FD_4004 (INTR)	R	HI[31:16]															
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
	R	HI[15:0]															
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c

Table 40-12. AP SDMA Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x53FD_4008 (STOP_STAT)	R	HE[31:16]															
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
	R	HE[15:0]															
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
0x53FD_400C (HSTART)	R	HSTART[31:16]/HE[31:16]															
	W	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr
	R	HSTART[15:0]/HE[15:0]															
	W	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr
0x53FD_4010 (EVTOVR)	R	EO[31:16]															
	W																
	R	EO[15:0]															
	W																
0x53FD_4014 (DSPOVR)	R	DO[31:16]															
	W																
	R	DO[15:0]															
	W																
0x53FD_4018 (HOSTOVR)	R	HO[31:16]															
	W																
	R	HO[15:0]															
	W																
0x53FD_401C (EVTPEND)	R	EP[31:16]															
	W	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr
	R	EP[15:0]															
	W	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr
0x53FD_4024 (RESET)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RESCHED	RESET
	W																
0x53FD_4028 (EVTERR)	R	CHNERR[31:16]															
	W																
	R	CHNERR[15:0]															
	W																

Table 40-12. AP SDMA Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x53FD_402C (INTRMASK)	R	HIMASK[31:16]															
	W																
	R	HIMASK[15:0]															
	W																
0x53FD_4030 (PSW)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	NCP[2:0]			NCR[4:0]					CCP[2:0]			CCR[4:0]				
	W																
0x53FD_4034 (EVTERRDBG)	R	CHNERR[31:16]															
	W																
	R	CHNERR[15:0]															
	W																
0x53FD_4038 (CONFIG)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	DSPDMA	RTDOBS	0	0	0	0	0	0	ACR	0	0	CSM[1:0]	
	W																
0x53FD_4040 (ONCE_ENB)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ENB
	W																
0x53FD_4044 (ONCE_DATA)	R	DATA[31:16]															
	W																
	R	DATA[15:0]															
	W																
0x53FD_4048 (ONCE_INSTR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	INSTRUCTION[15:0]															
	W																

Table 40-12. AP SDMA Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x53FD_404C (ONCE_STAT)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	PST[3:0]				RCV	EDR	ODR	SWB	MST	0	0	0	0	ECDR[2:0]		
	W																
0x53FD_4050 (ONCE_CMD)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	CMD[3:0]			
	W																
0x53FD_4054 (EVT_MIRROR)	R	EVENTS[31:16]															
	W																
	R	EVENTS[15:0]															
	W																
0x53FD_4058 (ILLINSTADDR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	ILLINSTADDR[13:0]													
	W																
0x53FD_405C (CHN0ADDR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	SMSZ	CHN0ADDR[13:0]													
	W																
0x53FD_4070 (XTRIG_CONF1)	R	0	CNF3	0	NUM3					0	CNF2	0	NUM2				
	W																
	R	0	CNF1	0	NUM1					0	CNF0	0	NUM0				
	W																
0x53FD_4074 (XTRIG_CONF2)	R	0	CNF7	0	NUM7					0	CNF6	0	NUM6				
	W																
	R	0	CNF5	0	NUM5					0	CNF4	0	NUM4				
	W																

Table 40-12. AP SDMA Register Summary (continued)

Name		31 15	30 14	29 13	28 12	27 11	26 10	25 9	24 8	23 7	22 6	21 5	20 4	19 3	18 2	17 1	16 0
0x53FD_4080+n*4 (CHNENBLn)[1]	R	ENBLn[31:16]															
	W																
	R	ENBLn[15:0]															
	W																
0x53FD_4100+n*4 (CHNPRIn)[2]	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	CHNPRIn[2:0]		
	W																

1 CHNENBLn: For n=0 to 31

2 CHNPRIn: For n=0 to 31

40.10.3 Register Descriptions

The following sections provide figures and detailed field descriptions of the SDMA registers.

40.10.3.1 AP Channel 0 Pointer (MC0PTR)

Figure 40-13 presents the register; Table 40-13 provides its field descriptions.

0x53FD_4000 (MC0PTR) Access: User Read/Write
Wait State: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	MC0PTR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MC0PTR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 40-13. AP Channel 0 Pointer (MC0PTR)

Table 40-13. MC0PTR Field Descriptions

Field	Description
31–0 MC0PTR	**Channel 0 Pointer** contains the 32-bit address, in AP memory, of channel 0 control block (the boot channel). Refer to the API document MOT-SFS-IAPI-SAS-001 (version 0.04) for the use of this register. The AP has a read/write access and the SDMA has a read-only access.

40.10.3.2 Channel Interrupts (INTR)

Figure 40-14 presents the register; Table 40-14 provides its field descriptions.

0x53FD_4004 (INTR) Access: User Read/Write
Wait State: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	HI[31:16]															
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HI[15:0]															
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 40-14. Channel Interrupts (INTR) Register

Table 40-14. INTR Field Descriptions

Field	Description
31–0 HI[31:0]	The AP Interrupts register contains the 32 HI[i] bits. If any bit is set, it will cause an interrupt to the AP. This register is a “write-ones” register to the AP. When the AP sets a bit in this register the corresponding HI[i] bit is cleared. The interrupt service routine should clear individual channel bits when their interrupts are serviced, failure to do so will cause continuous interrupts. The SDMA is responsible for setting the HI[i] bit corresponding to the current channel when the corresponding `done` instruction is executed.

40.10.3.3 Channel Stop/Channel Status (STOP_STAT)

Figure 40-15 presents the register; Table 40-15 provides its field descriptions.

0x53FD_4008 (STOP_STAT) Access: User Read/Write
Wait State: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	HE[31:16]															
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HE[15:0]															
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 40-15. Channel Stop/Channel Status (STOP_STAT) Register

Table 40-15. STOP_STAT Field Descriptions

Field	Description
31–0 HE	This 32-bit register gives access to the AP Enable bits. There is one bit for every channel. This register is a “write-ones” register to the AP. When the AP writes 1 in bit i of this register, it clears the HE[i] and HSTART[i] bits. Reading this register yields the current state of the HE[i] bits.

40.10.3.4 Channel Start (HSTART)

Figure 40-16 presents the register; Table 40-16 provides its field descriptions.

0x53FD_400C (HSTART) Access: User Read-Only
Wait State: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	HSTART[31:16]/HE[31:16]															
W	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HSTART[15:0]/HE[15:0]															
W	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 40-16. Channel Start (HSTART) Register

Table 40-16. HSTART Field Descriptions

Field	Description
31–0 HSTART/HE	The HSTART/HE registers are 32 bits wide with one bit for every channel. When a bit is written to 1, it enables the corresponding channel. Two physical registers are accessed with that address (HSTART and HE), which enables the AP to trigger a channel a second time before the first trigger is processed. • This register is a “write-ones” register to the AP. Neither HSTART[i] bit can be set while the corresponding HE[i] bit is cleared. • When the AP tries to set the HSTART[i] bit by writing a one (if the corresponding HE[i] bit is clear), the bit in the HSTART[i] register will remain cleared and the HE[i] bit will be set. • If the corresponding HE[i] bit was already set, the HSTART[i] bit will be set. The next time the SDMA channel *i* attempts to clear the HE[i] bit by means of a `done` instruction, the bit in the HSTART[i] register will be cleared and the HE[i] bit will take the old value of the HSTART[i] bit. • Reading this register yields the current state of the HSTART[i] bits. This mechanism enables the AP to pipeline two HSTART commands per channel.

40.10.3.5 Channel Event Override (EVTOVR)

Figure 40-17 presents the register; Table 40-17 provides its field descriptions.

0x53FD_4010 (EVTOVR) Access: User Read/Write
Wait State: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	EO[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	EO[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 40-17. Channel Event Override (EVTOVR) Register

Table 40-17. EVTOVR Field Descriptions

Field	Description
31–0 EO	The Channel Event Override register contains the 32 EO[i] bits. A bit set in this register causes the SDMA to ignore DMA requests when scheduling the corresponding channel.

40.10.3.6 Channel DSP Override (DSPOVR)

Figure 40-18 presents the register; Table 40-18 provides its field descriptions.

0x53FD_4014 (DSPOVR) Access: User Read/Write
Wait State: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DO[31:16]															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DO[15:0]															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 40-18. Channel DSP Override (DSPOVR) Register

Table 40-18. DSPOVR Field Descriptions

Field	Description
31–0 DO	The Channel BP Override register contains the 32 DO[i] bits. A bit set in this register causes the SDMA to ignore a BP enable condition when scheduling the corresponding channel. By default, all DO[i] bits are set to 1, which means that register can be ignored when the BP connections are not used. The BP interfaces are not connected for this device, so **all DO bits should be set to 1.**

40.10.3.7 Channel AP Override (HOSTOVR)

Figure 40-19 presents the register; Table 40-19 provides its field descriptions.

0x53FD_4018 (HOSTOVR) Access: User Read/Write
Wait State: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	HO[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	HO[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 40-19. Channel AP Override (HOSTOVR) Register

Table 40-19. HOSTOVR Field Descriptions

Field	Description
31–0 HO	The Channel AP Override register contains the 32 HO[i] bits. A bit set in this register causes the SDMA to ignore the AP enable bit when scheduling the corresponding channel.

40.10.3.8 Channel Event Pending (EVTPEND)

Figure 40-20 presents the register; Table 40-20 provides its field descriptions.

0x53FD_401C (EVTPEND) Access: User Read-Only
Wait State: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	EP[31:16]															
W	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	EP[15:0]															
W	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 40-20. Channel Event Pending (EVTPEND) Register

Table 40-20. EVTPEND Field Descriptions

Field	Description
31–0 EP	The Channel Event Pending register contains the 32 EP[i] bits. Reading this register enables the AP to determine what channels are pending after the reception of a DMA request. • Setting a bit in this register causes the SDMA to reevaluate scheduling as if a DMA request mapped on this channel had occurred. This is useful for starting up channels, so that initialization is done before awaiting the first request. The scheduler can also set bits in the EVTPEND register according to the received DMA requests. • The EP[i] bit may be cleared by the `done` instruction when running the channel *i* script. This a “write-ones” mechanism: Writing a ‘0’ does not clear the corresponding bit.

40.10.3.9 Reset Register (RESET)

Figure 40-21 presents the register; Table 40-21 provides its field descriptions.

0x53FD_4024 (RESET) Access: User Read-Only
Wait State: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RESCHED	RESET
W															slfclr	slfclr
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 40-21. Reset Register

Table 40-21. RESET Field Descriptions

Field	Description
31–2	Reserved
1 RESCHED	When set, this bit forces the SDMA to reschedule as if a script had executed a `done` instruction. This enables the AP to recover from a runaway script on a channel by clearing its HE[i] bit via the STOP register, and then forcing a reschedule via the RESCHED bit. The RESCHED bit is cleared when the context switch starts.
0 RESET	When set, this bit causes the SDMA to be held in a software reset. The internal reset signal is held low 16 cycles; the RESET bit is automatically cleared when the internal reset signal rises.

40.10.3.10 DMA Request Error Register (EVTERR)

Figure 40-22 presents the register; Table 40-22 provides its field descriptions.

0x53FD_4028 (EVTERR) Access: User Read-Only
Wait State: 0

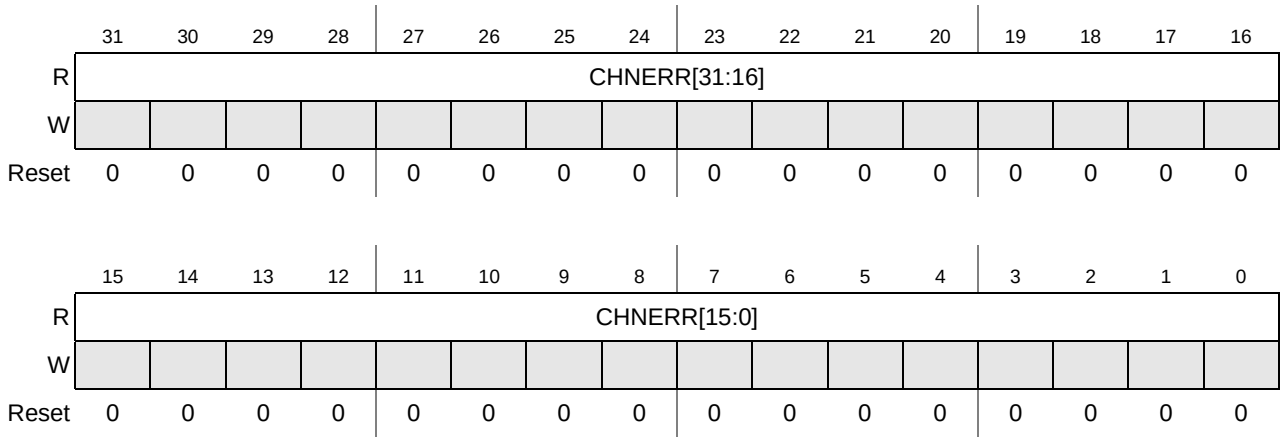


Figure 40-22. DMA Request Error (EVTERR) Register

Table 40-22. EVTERR Field Descriptions

Field	Description
31–0 CHNERR	This register is used by the SDMA to warn the AP when an incoming DMA request was detected and it triggers a channel that is already pending or being serviced. This probably means there is an overflow of data for that channel. • An interrupt is sent to the AP if the corresponding channel bit is set in the INTRMASK register. • This is a “write-ones” register for the scheduler. It is only able to set the flags. The flags are cleared when the register is read by the AP or during SDMA reset. • The CHNERR[i] bit is set when a DMA request that triggers channel *i* is received through the corresponding input pins and the EP[i] bit is already set; the EVTERR[i] bit is unaffected if the AP tries to set the EP[i] bit, whereas, that EP[i] bit is already set.

40.10.3.11 Channel AP Interrupt Mask Flags (INTRMASK)

Figure 40-23 presents the register; Table 40-23 provides its field descriptions.

0x53FD_402C (INTRMASK) Access: User Read/Write
Wait State: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	HIMASK[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	HIMASK[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 40-23. Channel AP Interrupt Mask Flags (INTRMASK) Register

Table 40-23. INTRMASK Field Description

Field	Description
31–0 HIMASK	The Interrupt Mask Register contains 32 interrupt generation mask bits. If bit HIMASK[i] is set, the HI[i] bit is set and an interrupt is sent to the AP when a DMA request error is detected on channel *i* (for example, EVTERR[i] is set).

40.10.3.12 Schedule Status (PSW)

Figure 40-24 presents the register; Table 40-24 provides its field descriptions.

0x53FD_4030 (PSW) Access: User Read-Only
Wait State: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	NCP[2:0]			NCR[4:0]					CCP[2:0]			CCR[4:0]				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 40-24. Schedule Status (PSW) Register

Table 40-24. PSW Field Descriptions

Field	Description
31–16	Reserved
15–13 NCP[2:0]	The Next Channel Priority gives the next pending channel priority. When the priority is 0, it means there is no pending channel and the NCR value has no meaning. 0 No running channel 1–7 Active channel priority
12–8 NCR[4:0]	The Next Channel Register indicates the number of the next scheduled pending channel with the highest priority.
7–4 CCP[2:0]	The Current Channel Priority indicates the priority of the current active channel. When the priority is 0, no channel is running: The SDMA is idle and the CCR value has no meaning. 0 No running channel 1–7 Active channel priority
3–0 CCR[4:0]	The Current Channel Register indicates the number of the channel that is being executed by the SDMA.

40.10.3.13 DMA Request Error Register for Debug (EVTERRDBG)

Figure 40-25 presents the register; Table 40-25 provides its field descriptions.

0x53FD_4034 (EVTERRDBG)
Wait State: 0

Access: User Read-Only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CHNERR[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CHNERR[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 40-25. DMA Request Error for Debug (EVTERRDBG) Register

Table 40-25. EVTERRDBG Field Descriptions

Field	Description
31–0 CHNERR	This register is the same as EVTERR, except reading it does not clear its contents. This address is meant to be used in debug mode. The AP OnCE may check this register value without modifying it.

40.10.3.14 Configuration Register (CONFIG)

Figure 40-26 presents the register; Table 40-26 provides its field descriptions.

0x53FD_4038 (CONFIG) Access: User Read/Write
Wait State: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	DSP DMA	RTD OBS	0	0	0	0	0	0	ACR	0	0	CSM[1:0]	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Figure 40-26. Configuration Register (CONFIG)

Table 40-26. CONFIG Register Field Descriptions

Field	Description
31–13	Reserved
12 DSPDMA	Indicates if the BP DMA is used. Default is unused. This bit determines if the BP DMA registers are saved/restored in the channel context. Must be set if the BP connections are utilized. The BP interfaces are not connected for this device, so *this bit should be configured as zero.* 0 BP DMA is not used 1 BP DMA is used
11 RTDOBS	Indicates if Real-Time Debug pins are used: They do not toggle by default in order to reduce power consumption. 0 RTD pins disabled 1 RTD pins enabled
10–8	Reserved
7–5	Reserved
4 ACR	AHB/SDMA Core Clock Ratio. Selects the clock ratio between AHB interfaces (burst DMA and peripheral DMA on AP domain) and the internal SDMA core clock. This bit has to match the configuration of the chip clock controller that generates the clocks used in the SDMA. 0 AHB interface frequency equals twice core frequency 1 AHB interface frequency equals core frequency

Table 40-26. CONFIG Register Field Descriptions (continued)

Field	Description
3–2	Reserved
1–0 CSM	Selects the Context Switch Mode. The AP has a read/write access. The SDMA cannot modify that register. The value at reset is 3, which selects the dynamic context switch by default. That register can be modified at anytime but the new context switch configuration will only be taken into account at the start of the next restore phase. 0 static 1 dynamic low power 2 dynamic with no loop 3 dynamic

40.10.3.15 OnCE Enable (ONCE_ENB)

Figure 40-27 presents the register; Table 40-27 provides its field descriptions.

0x53FD_4040 (ONCE_ENB) Access: User Read/Write
Wait State: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ENB
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 40-27. OnCE Enable (ONCE_ENB) Register

Table 40-27. ONCE_ENB Field Descriptions

Field	Description
31–1	Reserved
0 ENB	The OnCE Enable register selects the OnCE control source: When cleared (0), the OnCE registers are accessed through the JTAG interface; when set (1), the OnCE registers may be accessed by the AP through the addresses described, as follows. • After reset, the OnCE registers are accessed through the JTAG interface. • Writing a 1 to ENB enables the AP to access the ONCE_* as any other SDMA control register. • When cleared (0), all the ONCE_xxx registers cannot be written.

40.10.3.16 OnCE Data Register (ONCE_DATA)

Figure 40-28 presents the register; Table 40-28 provides its field descriptions.

0x53FD_4044 (ONCE_DATA) Access: User Read/Write
Wait State: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DATA[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATA[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 40-28. OnCE Data Register (ONCE_DATA)

Table 40-28. ONCE_DATA Field Descriptions

Field	Description
31–0 DATA	Data register of the OnCE JTAG controller. Refer to Section 40.16.5, “OnCE and Real-Time Debug” for information on this register.

40.10.3.17 OnCE Instruction Register (ONCE_INSTR)

Figure 40-29 presents the register; Table 40-29 provides its field descriptions.

0x53FD_4048 (ONCE_INSTR) Access: User Read/Write
Wait State: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INSTR[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 40-29. OnCE Instruction Register (ONCE_INSTR)

Table 40-29. ONCE_INSTR Field Descriptions

Field	Description
31–16	Reserved
15–0 INSTR	Instruction register of the OnCE JTAG controller. Refer to Section 40.16.5, “OnCE and Real-Time Debug” for information on this register.

40.10.3.18 OnCE Status Register (ONCE_STAT)

Figure 40-30 presents the register; Table 40-30 provides its field descriptions.

0x53FD_404C (ONCE_STAT) Access: User Read-Only
Wait State: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PST[3:0]				RCV	EDR	ODR	SWB	MST	0	0	0	0	ECDR[2:0]		
W																
Reset	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 40-30. OnCE Status Register (ONCE_STAT)

Table 40-30. ONCE_STAT Field Descriptions

Field	Description
31–16	Reserved
15–12 PST[3:0]	The Processor Status bits reflect the state of the SDMA RISC engine. Its states are as follows: • The “Program” state is the usual instruction execution cycle. • The “Data” state is inserted when there are wait-states during a load or a store on the data bus (ld or st). • The “Change of Flow” state is the second cycle of any instruction that breaks the sequence of instructions (jumps and channel switching instructions). • The “Change of Flow in Loop” state is used when an error causes a hardware loop exit. • The “Debug” state means the SDMA is in debug mode. • The “Functional Unit” state is inserted when there are wait-states during a load or a store on the functional units bus (ldf or stf). • In “Sleep” modes, no script is running (this is the RISC engine idle state). The “after Reset” is slightly different because no context restoring phase will happen when a channel is triggered: The script located at address 0 will be executed (boot operation). • The “in Sleep” states are the same as above except they do not have any corresponding channel: They are used when entering debug mode after reset. The reason is that it is necessary to return to the “Sleep after Reset” state when leaving debug mode. 0 Program 1 Data 2 Change of Flow 3 Change of Flow in Loop 4 Debug 5 Functional Unit 6 Sleep 7 Save 8 Program in Sleep 9 Data in Sleep 10 Change of Flow in Sleep 11 Change Flow in Loop in Sleep 12 Debug in Sleep 13 Functional Unit in Sleep 14 Sleep after Reset 15 Restore
11 RCV	After each write access to the real time buffer (RTB), the RCV bit is set. This bit is cleared after execution of an `rbuffer` command and on a JTAG reset.
10 EDR	This flag is raised when the SDMA has entered debug mode after an external debug request.
9 ODR	This flag is raised when the SDMA has entered debug mode after a OnCE debug request.
8 SWB	This flag is raised when the SDMA has entered debug mode after a software breakpoint.
7 MST	This flag is raised when the OnCE is controlled from the AP peripheral interface. 0 The JTAG interface controls the OnCE. 1 The AP peripheral interface controls the OnCE.

Table 40-30. ONCE_STAT Field Descriptions (continued)

Field	Description
6–3	Reserved
2–0 ECDR	Event Cell Debug Request. If the debug request comes from the event cell, the reason for entering debug mode is given by the EDR bits. If all three bits of the EDR are reset, then it did not generate any debug request. If the cell did generate a debug request, then at least one of the EDR bits is set (the meaning of the encoding is given below). The encoding of the EDR bits is useful to find out more precisely why the debug request was generated. A debug request from an event cell is generated for a specific combination of the addra_cond, addrb_cond, and data_cond conditions. The value of those fields is given by the EDR bits. EDR[0] 1 matched addra_cond EDR[1] 1 matched addrb_cond EDR[2] 1 matched data_cond

40.10.3.19 OnCE Command Register (ONCE_CMD)

Figure 40-31 presents the register; Table 40-31 provides its field descriptions.

0x53FD_4050 (ONCE_CMD) Access: User Read/Write
Wait State: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	CMD[3:0]			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 40-31. OnCE Command Register (ONCE_CMD)

Table 40-31. ONCE_CMD Field Descriptions

Field	Description
31–4	Reserved
3–0 CMD	Writing to this register will cause the OnCE to execute the command that is written. When needed, the ONCE_DATA and ONCE_INSTR registers should be loaded with the correct value before writing the command to that register. For a list of the OnCE commands and their usage, see Section 40.16.5, "OnCE and Real-Time Debug." 0 `rstatus` 1 `dmov` 2 `exec_once` 3 `run_core` 4 `exec_core` 5 `debug_rqst` 6 `rbuffer` 7–15 *reserved*

40.10.3.20 Illegal Instruction Trap Address (ILLINSTADDR)

Figure 40-32 presents the register; Table 40-32 provides its field descriptions.

0x53FD_4058 (ILLINSTADDR) Access: User Read/Write
Wait State: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	ILLINSTADDR[13:0]													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 40-32. Illegal Instruction Trap Address (ILLINSTADDR)

Table 40-32. ILLINSTADDR Field Descriptions

Field	Description
31–14	Reserved
13–0 ILLINSTADDR	The Illegal Instruction Trap Address is the address where the SDMA jumps when an illegal instruction is executed. It is 0x0001 after reset.

40.10.3.21 Channel 0 Boot Address (CHN0ADDR)

Figure 40-33 presents the register; Table 40-33 provides its field descriptions.

0x53FD_405C (CHN0ADDR) Access: User Read/Write
Wait State: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	SMSZ	CHN0ADDR[13:0]													
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0

Figure 40-33. Channel 0 Boot Address (CHN0ADDR) Register

Table 40-33. CHN0ADDR Register Field Descriptions

Field	Description
31–15	Reserved
14 SMSZ	The bit 14 (Scratch Memory Size) determines if scratch memory must be available after every channel context. After reset, it is equal to 0, which defines a RAM space of 24 words for each channel. All of this area stores the channel context. By setting this bit, 32 words are reserved for every channel context, which gives eight additional words that can be used by the channel script to store any type of data. Those words are never erased by the context switching mechanism. 0 24 words per context 1 32 words per context
13–0 CHN0ADDR	This 14-bit register is used by the boot code of the SDMA. After reset, it points to the standard boot routine in ROM (channel 0 routine). By changing this address, you can perform a boot sequence with your own routine. The very first instructions of the boot code fetch the contents of this register (it is also mapped in the SDMA memory space) and jump to the given address. The reset value is 0x0050 (decimal 80).

40.10.3.22 DMA Requests (EVT_MIRROR)

Figure 40-34 presents the register; Table 40-34 provides its field descriptions.

0x53FD_4054 (EVT_MIRROR) Access: User Read-Only
Wait State: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	EVENTS[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	EVENTS[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 40-34. DMA Requests (EVT_MIRROR)

Table 40-34. EVT_MIRROR Field Descriptions

Field	Description
31–0 EVENTS	This register reflects the DMA requests received by the SDMA. The AP and the SDMA have a read-only access. There is one bit associated with each of 32 DMA request events. This information may be useful during debug of the modules that generate the DMA requests. The EVT_MIRROR register is cleared following read access. 0 DMA request event not pending 1 DMA request event pending

40.10.3.23 Cross-Trigger Events Configuration Register (1) and (2) (XTRIG_CONF1 and XTRIG_CONF2)

Figure 40-35 presents the XTRIG_CONF1 register; Table 40-35 provides its field descriptions.

0x53FD_4070 (XTRIG_CONF1) Access: User Read/Write
Wait State: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	CNF3	0	NUM3[5:0]					0	CNF2	0	NUM2[5:0]				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	CNF1	0	NUM1[5:0]					0	CNF0	0	NUM0[5:0				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 40-35. Cross-Trigger Events Configuration Register (1) (XTRIG_CONF1)

Table 40-35. XTRIG_CONF1 Field Descriptions

Field	Description
31	Reserved
30 CNF3	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by the reception of a DMA request or by the starting of a channel script execution. When a pulse is generated by starting of a channel script execution, it is in fact generated by the restore part of the context switch mechanism. Therefore, no pulse is generated the first time channel 0 (boot channel) is executed after reset (on AP demand). 0 channel 1 DMA request
29	Reserved
28–24 NUM3[4:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.
23	Reserved
22 CNF2	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by receiving a DMA request or by starting a channel script execution. When a pulse is generated by starting of a channel script execution, it is in fact generated by the restore part of the context switch mechanism. Therefore, no pulse is generated the first time channel 0 (boot channel) is executed after reset (on AP demand). 0 channel 1 DMA request
21	Reserved
20–16 NUM2[4:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.
15	Reserved
14 CNF1	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by receiving a DMA request or by starting a channel script execution. When a pulse is generated by starting of a channel script execution, it is in fact generated by the restore part of the context switch mechanism. Therefore, no pulse is generated the first time channel 0 (boot channel) is executed after reset (on AP demand). 0 channel 1 DMA request
13	Reserved
12–8 NUM1[4:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.
7	Reserved
6 CNF0	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by receiving a DMA request or by starting a channel script execution. When a pulse is generated by starting of a channel script execution, it is in fact generated by the restore part of the context switch mechanism. Therefore, no pulse is generated the first time channel 0 (boot channel) is executed after reset (on AP demand). 0 channel 1 DMA request

Table 40-35. XTRIG_CONF1 Field Descriptions (continued)

Field	Description
5	Reserved
4–0 NUM0[4:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.

Figure 40-36 presents the XTRIG_CONF2 register; Table 40-36 provides its field descriptions.

0x53FD_4074 (XTRIG_CONF2) Access: User Read/Write
Wait State: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	CNF7	0	NUM7[4:0]					0	CNF6	0	NUM6[4:0]				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	CNF5	0	NUM5[4:0]					0	CNF4	0	NUM4[4:0]				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 40-36. Cross-Trigger Events Configuration Register (2) (XTRIG_CONF2)

Table 40-36. XTRIG_CONF2 Field Descriptions

Field	Description
31	Reserved
30 CNF7	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by receiving a DMA request or by starting a channel script execution. When a pulse is generated by starting of a channel script execution, it is in fact generated by the restore part of the context switch mechanism. Therefore, no pulse is generated the first time channel 0 (boot channel) is executed after reset (on AP demand). 0 channel 1 DMA request
29	Reserved
28–24 NUM7[4:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.
23	Reserved

Table 40-36. XTRIG_CONF2 Field Descriptions (continued)

Field	Description
22 CNF6	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by receiving a DMA request or by starting a channel script execution. When a pulse is generated by starting of a channel script execution, it is in fact generated by the restore part of the context switch mechanism. Therefore, no pulse is generated the first time channel 0 (boot channel) is executed after reset (on AP demand). 0 channel 1 DMA request
21	Reserved
20–16 NUM6[4:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.
15	Reserved
14 CNF5	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by receiving a DMA request or by starting a channel script execution. When a pulse is generated by starting of a channel script execution, it is in fact generated by the restore part of the context switch mechanism. Therefore, no pulse is generated the first time channel 0 (boot channel) is executed after reset (on AP demand). 0 channel 1 DMA request
13	Reserved
12–8 NUM5[4:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.
7	Reserved
6 CNF4	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by receiving a DMA request or by starting a channel script execution. When a pulse is generated by starting of a channel script execution, it is in fact generated by the restore part of the context switch mechanism. Therefore, no pulse is generated the first time channel 0 (boot channel) is executed after reset (on AP demand). 0 channel 1 DMA request
5	Reserved
4–0 NUM4[4:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.

40.10.3.24 Channel Priority Registers (CHNPRIn)

Figure 40-37 presents the register; Table 40-37 provides its field descriptions.

0x53FD_4100+n*4 (CHNPRIn[1])
Wait State: 0

Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	CHNPRIn[2:0]		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 40-37. Channel Priority Registers (CHNPRIn)

1 for n = 1 to 31

Table 40-37. CHNPRIn Field Descriptions

Field	Description
31–3	Reserved
2–0 CHNPRIn	This contains the priority of channel number *n*. Useful values are between 1 and 7; 0 is reserved by the SDMA hardware to determine when there is no pending channel. Reset value is 0, which prevents the channels from starting.

40.10.3.25 Channel Enable RAM (CHNENBLn)

Figure 40-38 presents the register; Table 40-38 provides its field descriptions.

0x53FD_4080+n*4 (CHNENBLn[1])
Wait State: 1

Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ENBLn[31:16]															
W																
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ENBLn[15:0]															
W																
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Figure 40-38. Channel Enable RAM (CHNENBLn) Register

[1] for n = 0 to 31

Table 40-38. CHNENBLn Field Descriptions

Field	Description
31–0 ENBLn	This 32-bit value selects the channels that are triggered by the DMA request number *n*. If ENBLn[i] is set to 1, bit EP[i] will be set when the DMA request *n* is received. These 32 32-bit registers are physically located in a RAM, with no known reset value. It is thus essential for the AP to program them before any DMA request is triggered to the SDMA, otherwise an unpredictable combination of channels may be started.

40.11 SDMA Programming Model

The following section describes the programming model for the SDMA RISC engine, including its processor, memory, and internal control registers.

All addresses are related to the internal SDMA memory map, which is completely different from the AP memory maps. The AP processor has no access to any hardware resource described, except when those resources are described in Section 40.10, "AP Memory Map and Control Register Definitions."

40.11.1 State and Registers Per Channel

The SDMA can be seen as a set of 32 identical devices that are able to perform one data transfer channel each. Only one channel can work at a time, but every channel state is available at any time. This chapter lists the components of every channel state.

40.11.2 General Purpose Registers

Each channel has eight general purpose registers of 32 bits for use by scripts. General register 0 has a dedicated function for the `loop` instruction, but otherwise can be used for any purpose.

40.11.3 Functional Unit State

Each channel context has some state that is part of the functional units. The specific allocation of this state is part of the functional unit definition that is described in Section 40.16.1, "Burst DMA Unit," Section 40.16.2, "Peripheral DMA Unit," and Section 40.16.3, "BP DMA Unit." This state must be saved/restored on context switches.

40.11.3.1 Program Counter Register (PC)

The PC is 14 bits. Since instructions are 16 bits in width and all memory in the SDMA is 32 bits in width, the low order bit of the PC selects which half of the 32-bit word contains the current instruction. A low order bit of zero selects the most significant half of the word.[1]

1. For example, Big-Endian.

40.11.3.2 Flags

Each channel has the following four flags:

- The T bit reflects the status of some arithmetic and test instructions. It is set when the result of an addition or a subtraction is zero and cleared otherwise. It is also the copy of the tested bits. Finally, it can also be set when the loop counter (GReg0) reaches zero. When the last instruction of the hardware loop is an operation that can modify the T flag, its effect on T is discarded and replaced by the GReg0 status.
- Two additional bits, SF and DF, are used to indicate error conditions resulting from loading data sources and storing to destinations, respectively. Access errors set these bits, and successful transactions clear them. They can also be cleared by specific instructions (`CLRF` and `loop`). The source fault (SF) is updated by the loads LD and LDF; the destination fault (DF) is updated by the stores ST and STF.

  Access errors are caused by several conditions including writing to the ROM, writing to a read-only memory mapped register, accessing an unmapped address, or any transfer error received by a peripheral when it is accessed.

  The SF and DF flags have a major impact on the behavior of the hardware loop: If SF or DF is set when starting a hardware loop and it is not masked by the `loop` instruction, the loop body will not be executed. Inside the loop body, if a load or store sets the corresponding SF or DF flag, the loop exits immediately. Testing the status of the T flag at the end of the loop (as well as testing both SF and DF) tells if the loop exited abnormally as any anticipated exit prevents GReg0 from reaching the zero value and thus setting the T flag. This is also valid if the fault occurs at the last instruction of the last loop.
- The last flag is the loop mode flag, LM, which is composed of two bits. The most significant bit indicates when the processor is currently operating in loop mode. It is set by the `loop` instruction and is cleared after execution of the last instruction of the last loop. The least significant bit is set when the program counter points to the last instruction of a loop on the last path. It is used for a channel that is restored with this configuration to know that the next program counter is EPC. As with the dynamic context switch GReg0, which indicates when the program must get out of the loop, it can be restored only on the last instruction of the loop. This, however, is too late to fetch the next instruction after the loop.

40.11.3.3 Return Program Counter (RPC)

The RPC is 14 bits. It is set by the jump to the subroutine instructions and used by the return from the subroutine instructions. Instructions are available to transfer its contents to and from a general register.

40.11.3.4 Loop Mode Start Program Counter (SPC)

The SPC is 14 bits. It is set by the `loop` instruction to the location immediately following it.

40.11.3.5 Loop Mode End Program Counter (EPC)

The EPC is 14 bits. It is set by the `loop` instruction to the location of the next instruction after the loop.

40.11.4 Context Switching

Each channel has a separate context consisting of the eight general purpose registers and additional registers representing the state of the functional units. The active registers and functional units contain the context of the active channel. The context of inactive channels is stored in SDMA RAM, which is part of the SDMA address space.

In a function of the selected context switching mode (Section 40.5.4, “Context Switching”), modified registers by the program can be saved in the channel RAM space while the program is going on. In every cycle, a write access to the RAM is possible.

On a `done` or `yield`(`ge`) instruction, SDMA goes into “real” context switching. In one of the dynamic modes, modified registers not previously saved, as well as the PC-Loop registers, are stored into the context area of the channel that will be closed. The new PC-Loop registers are loaded from the context area of the new channel. All other registers are restored while the program is executed, giving priority to registers used by the decoded instruction. Therefore, in the best case, only the PC and Loop registers should be saved and restored during this context-switching phase, which only requires five SDMA cycles.

In static mode, the context switch stores all registers in the old channel RAM space, and restores all registers from the new channel RAM space. It requires 26 SDMA cycles.

The address of the context memory for channel *i* is *CONTEXT_BASE* + 24**i* or *CONTEXT_BASE + 32*i* where CONTEXT_BASE equals 0x0800. Table 40-39 presents the layout of a channel context in memory:

Table 40-39. Layout of a Channel Context in Memory for SDMA

OFFSET	31	30	29–16	15	14	13–0
0	SF	—	RPC	T	—	PC
1	LM		EPC	DF	—	SPC
2	GR0					
3	GR1					
4	GR2					
5	GR3					
6	GR4					
7	GR5					
8	GR6					
9	GR7					
10	MDA (burst DMA)					
11	MSA (burst DMA)					
12	MS (burst DMA)					
13	MD (burst DMA)					
14	PDA (peripheral DMA)					
15	PSA (peripheral DMA)					

Table 40-39. Layout of a Channel Context in Memory for SDMA (continued)

OFFSET	31	30	29–16	15	14	13–0
16	PS (peripheral DMA)					
17	PD (peripheral DMA)					
18	CA (CRC)					
19	CS (CRC)					
20	DDA[1] (BP DMA)					
21	DSA[1] (BP DMA)					
22	DS[1] (BP DMA)					
23	DD[1] (BP DMA)					
24	Scratch RAM (optional)					
25	Scratch RAM (optional)					
26	Scratch RAM (optional)					
27	Scratch RAM (optional)					
28	Scratch RAM (optional)					
29	Scratch RAM (optional)					
30	Scratch RAM (optional)					
31	Scratch RAM (optional)					

1 These registers are not saved in the context when the DSPDMA bit in the CONFIG register is zero.

40.11.5 Address Space

The SDMA has four internal buses, as follows:

- The Instruction bus reads instructions from the memory. Its address map is described in Section 40.11.5.1, “Instruction Memory Map.”
- The Data bus (DMBUS) accesses the same memories as those visible on the Instruction bus, some memory-mapped registers (scheduler status and OnCE registers), and up to 14 peripherals. Its address map is described in Section 40.11.5.2, “Data Memory Map.”
- The Functional Units bus (FUBUS) accesses the burst DMA, peripheral DMA, and CRC internal registers. The addressing mechanism is further detailed in Section 40.16, “Functional Units Programming Model.”
- The Context Switch bus reads/writes registers into context-switch RAM space. It is a 64-bit bus dedicated for accessing this RAM space for updating the context of the running channel. While the program is going on, this bus has the lowest priority compared to the Instruction and Data buses, except for restoring a register needed for the decoded instruction to be executed. On the save part of a context switch (when the PCU is in its slave state), this is the only one used. On the restore part, the Instruction bus has the priority to read the next instruction at the restored PC and otherwise the Context Switch bus is used. It is not possible to control the actual data transfers that occur on this bus.

40.11.5.1 Instruction Memory Map

The instruction memory map is based on a 14-bit address bus and a 16-bit data (instruction) bus. Each address corresponds to a 16-bit data location. Instructions are fetched from either program ROM or program RAM. An SDMA script is able to change the contents of the program RAM, which is also visible from the data bus.

The first two instruction locations (at 0 and 1) are special. Location 0 is where the PC is set on reset. Location 1 is where the PC is set upon the execution of an illegal instruction. It is expected that both of these locations will contain a `jmp` to handle routines.

Table 40-40. SDMA Instruction Memory Space

Device	SDMA Address (Hex)	Base Address Label	Module Name	WS	Description
ROM	0x0000 ↓ 0x07FF	SDMA_IBUS_ROM_ADDR	—	0	4 Kbytes internal ROM with boot code and standard routines.
RAM	0x1000 ↓ 0x1FFF	SDMA_IBUS_RAM_ADDR	—	0	8 Kbytes internal RAM with channels context and user data/routines.

40.11.5.2 Data Memory Map

All of the data accessible to SDMA scripts make up the data memory space of the SDMA. This address space has several components:

- ROM (also visible on the Instruction bus)
- RAM (also visible on the Instruction bus)
- Shared Peripherals Registers
- SDMA Internal Registers (scheduler, OnCE, and registers that are also accessible by the AP)

SDMA scripts can read and write to the context RAM, data RAM, shared peripheral registers, and internal registers.

The address range is 16 bits and the data width is 32 bits. Each address corresponds to a 32-bit data word. When accessing peripheral registers (USB and so on), the data width may be different. The exact address map for the peripherals depends on the project (as presented in each respective chapter).

The SDMA can perform only 32-bit access to shared peripheral registers. For this device, shared peripherals registers are aliased as if byte addressed. This is a consequence of connections through the SPBA shared peripheral bus outside of the SDMA. The result is that although address space 4 Kwords (16 Kbyte) is allocated for each peripheral, only the first 4 Kbyte of the peripheral's register space can accessed. For example, the shared peripheral register at address 0x3000 is mapped also to addresses 0x3001, 0x3002, 0x3003. A read or write access to any of any of these 4 addresses will respond as if the access was to address 0x3000.

Data access is performed with *ld* and *st* instructions that take the address from a general purpose register in the core (GReg*n*). The mapping between the general purpose register contents and the address bus is given in Figure 40-39:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
sz	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
address															

Figure 40-39. GReg*n* to DMBUS Address Mapping

Grayed bits are simply discarded but they must be cleared to ensure forward-script compatibility.

- sz (bit 31) indicates the peripheral data width: 0 is used for a 32-bit peripheral and 1 is used for a 16-bit peripheral.
- Address (bits 15 down to 0) is the address of the accessed resource (internal memory, internal register, or shared peripheral).

Table 40-41 shows the SDMA data memory space.

Table 40-41. SDMA Data Memory Space

Device	SDMA Address (Hex)	Description
ROM	0x0000 → 0x03FF	4 Kbyte internal ROM with boot code and standard routines
RAM	0x0800 → 0x0FFF	8 Kbyte internal RAM with channels contexts and user data/routines
per1	0x1000 → 0x1FFF	*per1* memory space (4 K address space)
per2	0x2000 → 0x2FFF	*per2* memory space (4 K address space)
per3	0x3000 → 0x3FFF	*per3* memory space (4 K address space)
per4	0x4000 → 0x4FFF	*per4* memory space (4 K address space)
per5	0x5000 → 0x5FFF	*per5* memory space (4 K address space)
per6	0x6000 → 0x6FFF	*per6* memory space (4 K address space)
Registers	0x7000 → 0x7FFF	Memory mapped registers (4 Kwords or 16 Kbyte are reserved)
per7	0x8000 → 0x8FFF	*per7* memory space (4 K address space)
per8	0x9000 → 0x9FFF	*per8* memory space (4 K address space)
per9	0xA000 → 0xAFFF	*per9* memory space (4 K address space)
per10	0xB000 → 0xBFFF	*per10* memory space (4 K address space)
per11	0xC000 → 0xCFFF	*per11* memory space (4 K address space)
per12	0xD000 → 0xDFFF	*per12* memory space (4 K address space)
per13	0xE000 → 0xEFFF	*per13* memory space (4 K address space)
per14	0xF000 → 0xFFFF	*per14* memory space (4 K address space)

40.12 SDMA Internal (Core) Memory Map and Internal Register Definitions

The actual SDMA memory mapped registers are summarized in the following sections; for peripherals' memory maps, refer to the respective chapters.

40.12.1 SDMA Internal (Core) Registers Memory Map

Table 40-42. SDMA Internal Registers Memory Map

Address	Register	Access	Reset Value	Section/Page
General Registers				
0x7000 (MC0PTR)	AP (MCU) Channel 0 Pointer	R	0x0000_0000	40.12.3.1/40-76
0x7002 (CCPTR)	Current Channel Pointer	R	0x0000_0000	40.12.3.2/40-77
0x7003 (CCR)	Current Channel Register	R	0x0000_0000	40.12.3.3/40-77
0x7004 (NCR)	Highest Pending Channel Register	R	0x0000_0000	40.12.3.4/40-78
0x7005 (EVENTS)	External DMA Requests Mirror	R	0x0000_0000	40.12.3.5/40-79
0x7006 (CCPRI)	Current Channel Priority	R	0x0000_0000	40.12.3.6/40-80
0x7007 (NCPRI)	Next Channel Priority	R	0x0000_0000	40.12.3.7/40-81
0x7009 (ECOUNT)	OnCE Event Cell Counter	R/W	0x0000_0000	40.12.3.8/40-81
0x700A (ECTL)	OnCE Event Cell Control Register	R/W	0x0000_0000	40.12.3.9/40-82
0x700B (EAA)	OnCE Event Address Register A	R/W	0x0000_0000	40.12.3.10/40-83
0x700C (EAB)	OnCE Event Cell Address Register B	R/W	0x0000_0000	40.12.3.11/40-84
0x700D (EAM)	OnCE Event Cell Address Mask	R/W	0x0000_0000	40.12.3.12/40-85
0x700E (ED)	OnCE Event Cell Data Register	R/W	0x0000_0000	40.12.3.13/40-85
0x700F (EDM)	OnCE Event Cell Data Mask	R/W	0x0000_0000	40.12.3.14/40-86
0x7018 (RTB)	OnCE Real-Time Buffer	R/W	0x0000_0000	40.12.3.15/40-87
0x7019 (TB)	OnCE Trace Buffer	R	0x0000_0000	40.12.3.16/40-87
0x701A (OSTAT)	OnCE Status	R	0x0000_0000	40.12.3.17/40-88
0x701C (MCHN0ADDR)	Channel 0 Boot Address	R	0x0000_0000	40.12.3.18/40-91
0x701D (ENDIANESS)	ENDIAN Mode Status Register	R	0x0000_0000	40.12.3.19/40-92

40.12.2 Register Summary

The following definitions serve as a key for the SDMA internal register summary.

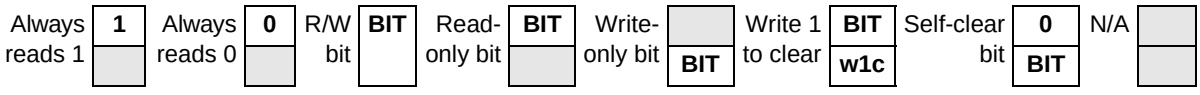


Figure 40-40. Key to Register Fields

Table 40-43 provides a key for register figures.

Table 40-43. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writeable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Note: for n = 0 to 31

Table 40-44 presents a summary of the SDMA internal (core) registers.

Table 40-44. SDMA Internal Registers Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x7000 (MC0PTR)	R	MC0PTR[31:16]															
	W																
	R	MC0PTR[15:0]															
	W																
0x7002 (CCPTR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	CCPTR[15:0]															
	W																

Table 40-44. SDMA Internal Registers Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x7003 (CCR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	CCR[4:0]				
	W																
0x7004 (NCR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	NCR[4:0]				
	W																
0x7005 (EVENTS)	R	EVENTS[31:16]															
	W																
	R	EVENTS[15:0]															
	W																
0x7006 (CCPRI)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	CCPRI[2:0]		
	W																
0x7007 (NCPRI)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	NCPRI[2:0]		
	W																
0x7009 (ECOUNT)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	ECOUNT[15:0]															
	W	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm
0x700A (ECTL)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	EN	CNT	ECTC[1:0]		DTC[1:0]		ATC[1:0]		ABTC[1:0]		AATC[1:0]		ATS[1:0]	
	W																
0x700B (EAA)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	EAA[15:0]															
	W																

Table 40-44. SDMA Internal Registers Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x700C (EAB)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	EAB[15:0]															
	W																
0x700D (EAM)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	EAM[15:0]															
	W																
0x700E (ED)	R	ED[31:16]															
	W																
	R	ED[15:0]															
	W																
0x700F (EDM)	R	EDM[31:16]															
	W																
	R	EDM[15:0]															
	W																
0x7018 (RTB)	R	RTB[31:16]															
	W																
	R	RTB[15:0]															
	W																
0x7019 (TB)	R	0	0	0	TBF	TADDR[13:2]											
	W																
	R	TADDR[1:0]		CHFADDR[13:0]													
	W																
0x701A (OSTAT)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	PST[3:0]				RCV	EDR	ODR	SWB	MST	0	0	0	0	ECDR[2:0]		
	W																

Table 40-44. SDMA Internal Registers Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x701C (MCHN0ADDR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	SMSZ	CHN0ADDR[13:0]													
	W																
0x701D (ENDIANESS)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	BPEND	AP-END
	W																

40.12.3 SDMA Core Register Descriptions

The SDMA core has access to several memory mapped registers through its internal data bus. They are described in the following sections.

40.12.3.1 AP (MCU) Channel 0 Pointer (MC0PTR)

Figure 40-41 shows the register; Table 40-45 provides its field descriptions.

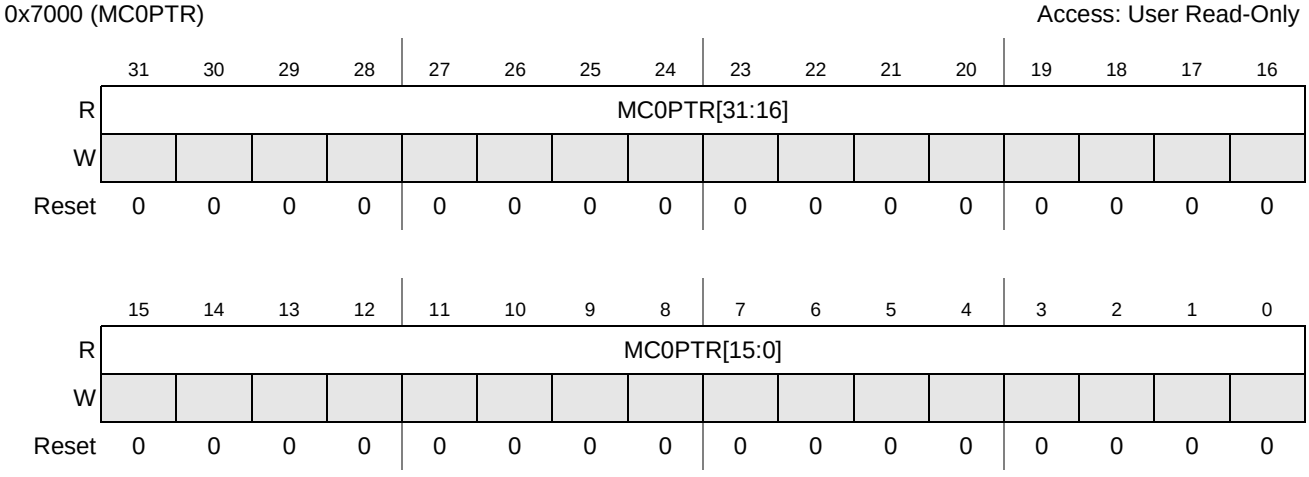


Figure 40-41. AP (MCU) Channel 0 Pointer (MC0PTR) Register

Table 40-45. MC0PTR Field Descriptions

Field	Description
31–0 MC0PTR	Contains the address—in the AP memory space—of the initial SDMA context and scripts that are loaded by the SDMA boot script running on channel 0.

40.12.3.2 Current Channel Pointer (CCPTR)

Figure 40-42 shows the register; Table 40-46 provides its field descriptions.

0x7002 (CCPTR) Access: User Read-Only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CCPTR[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 40-42. Current Channel Pointer (CCPTR) Register

Table 40-46. CCPTR Field Descriptions

Field	Description
31–16	Reserved
15–0 CCPTR	Contains the start address of the context data for the current channel: Its value is *CONTEXT_BASE* + 24**CCR* or *CONTEXT_BASE* + 32**CCR* where CONTEXT_BASE = 0x0800. The value 24 or 32 is selected according to the programmed channel scratch RAM size in the register shown in Section 40.10.3.21, "Channel 0 Boot Address (CHN0ADDR)."

40.12.3.3 Current Channel Register (CCR)

Figure 40-43 shows the register; Table 40-47 provides its field descriptions.

0x7003 (CCR) Access: User Read-Only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	CCR[4:0]				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 40-43. Current Channel Register (CCR)

Table 40-47. CCR Field Descriptions

Field	Description
31–5	Reserved
4–0 CCR	Contains the number of the current running channel whose context is installed.

40.12.3.4 Highest Pending Channel Register (NCR)

Figure 40-44 shows the register; Table 40-48 provides its field descriptions.

0x7004 (NCR) Access: User Read-Only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	NCR[4:0]				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 40-44. Highest Pending Channel Register (NCR)

Table 40-48. NCR Field Descriptions

Field	Description
31–5	Reserved
4–0 NCR	Contains the number of the pending channel that the scheduler has selected to run next.

40.12.3.5 External DMA Requests Mirror (EVENTS)

Figure 40-45 shows the register; Table 40-49 provides its field descriptions.

0x7005 (EVENTS) Access: User Read-Only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	EVENTS[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	EVENTS[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 40-45. External DMA Requests Mirror (EVENTS)

Table 40-49. EVENTS Field Descriptions

Field	Description
31–0 EVENTS	Reflects the status of the SDMA's external DMA requests. It is meant to allow any channel to monitor the states of these SDMA inputs. This register displays EVENTS 0-31.

NOTE

This register is very useful in the case of DMA requests that are active when a peripheral FIFO level is above the programmed watermark. The activation of the DMA request (rising edge) is detected by the SDMA logic and it can enable one or several channels. One of the channels accesses the peripheral and reads or writes a number of data that matches the watermark level (for example, if the watermark is four words, the channel reads or writes four words).

If the channel is effectively executed long after the DMA request was received, reading or writing the watermark number of data may not be sufficient to reset the DMA request (for example, if the FIFO watermark is four and at the channel execution it already contains nine pieces of data). This means no new rising edge may be detected by the SDMA, although there still remains transfers to perform. Therefore, if the channel were terminated at that time, it would not be restarted, causing potential overrun or underrun of the peripheral.

The proposed mechanism is for the channel to check this register after it has performed the “watermark” number of accesses to the peripheral. If the bit for the DMA request that triggers this channel is set, it means there is still another watermark number of data to transfer. This goes on until the bit is cleared. The same script can be used for multiple channels that require this behavior. The script can determine its channel number from the CCR register and infer the corresponding DMA request bit to check. It needs a reference table that is coherent with the request-channel matrix that the AP programmed.

40.12.3.6 Current Channel Priority (CCPRI)

Figure 40-46 shows the register; Table 40-50 provides its field descriptions.

0x7006 (CCPRI) Access: User Read-Only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	CCPRI[2:0]		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 40-46. Current Channel Priority (CCPRI) Register

Table 40-50. CCPRI Field Descriptions

Field	Description
31–3	Reserved
2–0 CCPRI	Contains the 3-bit priority of the channel whose context is installed. It is 0 when no channel is running. 0 no running channel 1–7 current channel priority

40.12.3.7 Next Channel Priority (NCPRI)

Figure 40-47 shows the register; Table 40-51 provides its field descriptions.

0x7007 (NCPRI) Access: User Read-Only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	NCPRI[2:0]		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 40-47. Next Channel Priority (NCPRI) Register

Table 40-51. NCPRI Field Descriptions

Field	Description
31–3	Reserved
2–0 NCPRI	Contains the 3-bit priority of the channel the scheduler has selected to run next. It is 0 when no other channel is pending.

40.12.3.8 OnCE Event Cell Counter (ECOUNT)

Figure 40-48 shows the register; Table 40-52 provides its field descriptions.

0x7009 (ECOUNT) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ECOUNT[15:0]															
W	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 40-48. OnCE Event Cell Counter (ECOUNT) Register

Table 40-52. ECOUNT Field Descriptions

Field	Description
31–16	Reserved
15–0 ECOUNT	The event cell counter contains the number of times minus one that an event detection must occur before generating a debug request. • This register should be written before any attempt to use the event detection counter during an event detection process. • The counter is cleared on a JTAG reset.

40.12.3.9 OnCE Event Cell Control Register (ECTL)

Figure 40-49 shows the register; Table 40-53 provides its field descriptions.

0x700A (ECTL) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	EN	CNT	ECTC[1:0]		DTC[1:0]		ATC[1:0]		ABTC[1:0]		AATC[1:0]		ATS[1:0]	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 40-49. OnCE Event Cell Control Register (ECTL)

Table 40-53. ECTL Field Descriptions

Field	Description
31–14	Reserved
13 EN	Event Cell Enable. If the EN bit is set, the event cell is allowed to generate debug requests (the cell is awakened). If it is cleared, the event detection unit is disabled and no hardware breakpoint is generated, but matching conditions are still reflected on the emulation pin. 0 Cell is disabled. 1 Cell is enabled.
12 CNT	Event Counter Enable. The event counter enable bit determines if the cell counter is used during the event detection. In order to use the event counter during an event detection process, the event cell counter register should be loaded with a value equal to the number of times minus one that an event occurs before a debug request is sent. After every event detection, the counter is decreased. When the counter reaches the value 0, the event detection cell sends a debug request to the core. The event counter register should be written and the EN bit should be set before each new event detection process uses the event counter. 0 Counter is disabled. 1 Counter is enabled.

Table 40-53. ECTL Field Descriptions (continued)

Field	Description
11–10 ECTC[1:0]	The event cell trigger condition bits select the combination of address and data matching conditions that generate the final address/data condition. During program execution, if this event cell trigger condition goes to 1, a debug request is sent to the SDMA. The EN bit must be set to enable the debug request generation. 00 address ONLY 01 data ONLY 10 address AND data 11 address OR data
9–8 DTC[1:0]	The data trigger condition bits define when data is considered matching after comparison with the data register of the event detection unit. The operations are performed on unsigned values. 00 equal 01 not equal 10 greater than 11 less than
7–6 ATC[1:0]	The address trigger condition bits select how the two address conditions (addressA and addressB) are combined to define the global address matching condition. The supported combinations are described, as follows. 00 addressA ONLY 01 addrA AND addrB 10 addrA OR addrB 11 reserved
5–4 ABTC[1:0]	The Address B Trigger Condition (ABTC) controls the operations performed by address comparator B. All operations are performed on unsigned values. This comparator B outputs the addressB condition. 00 equal 01 not equal 10 greater than 11 less than
3–2 AATC[1:0]	The Address A Trigger Condition (AATC) controls the operations performed by address comparator A. All operations are performed on unsigned values. This comparator A outputs the addressA condition. 00 equal 01 not equal 10 greater than 11 less than
1–0 ATS[1:0]	The access type select bits define the memory access type required on the SDMA memory bus. 00 read ONLY 01 write ONLY 10 read or write 11 —

40.12.3.10 OnCE Event Address Register A (EAA)

Figure 40-50 shows the register; Table 40-54 provides its field descriptions.

0x700B (EAA) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	EAA[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 40-50. OnCE Event Address Register A (EAA)

Table 40-54. EAA Field Descriptions

Field	Description
31–16	Reserved
15–0 EAA	Event Cell Address Register A computes an address A condition. It is cleared on a JTAG reset.

40.12.3.11 OnCE Event Cell Address Register B (EAB)

Figure 40-51 shows the register; Table 40-55 provides its field descriptions.

0x700C (EAB) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	EAB[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 40-51. OnCE Event Cell Address Register B

Table 40-55. EAB Field Descriptions

Field	Description
31–16	Reserved
15–0 EAB	Event Cell Address Register B computes an address B condition. It is cleared on a JTAG reset.

40.12.3.12 OnCE Event Cell Address Mask (EAM)

Figure 40-52 shows the register; Table 40-56 provides its field descriptions.

0x700D (EAM) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	EAM[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 40-52. OnCE Event Cell Address Mask (EAM)

Table 40-56. EAM Field Descriptions

Field	Description
31–16	Reserved
15–0 EAM	The Event Cell Address Mask contains a user-defined address mask value. This mask is applied to the address value latched from the memory address bus before performing the address comparison. **Note:** There is a common address mask value for both address comparators. If bit *i* of this register is set, then bit *i* of the address value latched from the memory bus does not influence the result of the address comparison. The register is cleared on a JTAG reset.

40.12.3.13 OnCE Event Cell Data Register (ED)

Figure 40-53 shows the register; Table 40-57 provides its field descriptions.

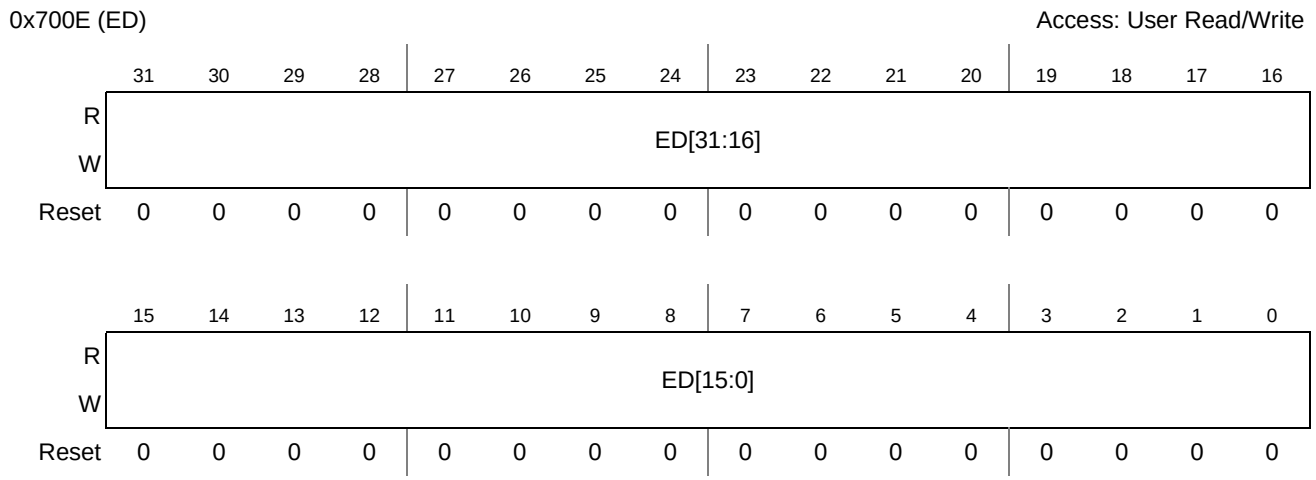


Figure 40-53. OnCE Event Cell Data (ED) Register

Table 40-57. ED Field Descriptions

Field	Description
31–0 ED	The event cell data register contains a user defined data value. This data value is an input for the data comparator which generates the data condition. It is cleared on a JTAG reset.

40.12.3.14 OnCE Event Cell Data Mask (EDM)

Figure 40-54 shows the register; Table 40-58 provides its field descriptions.

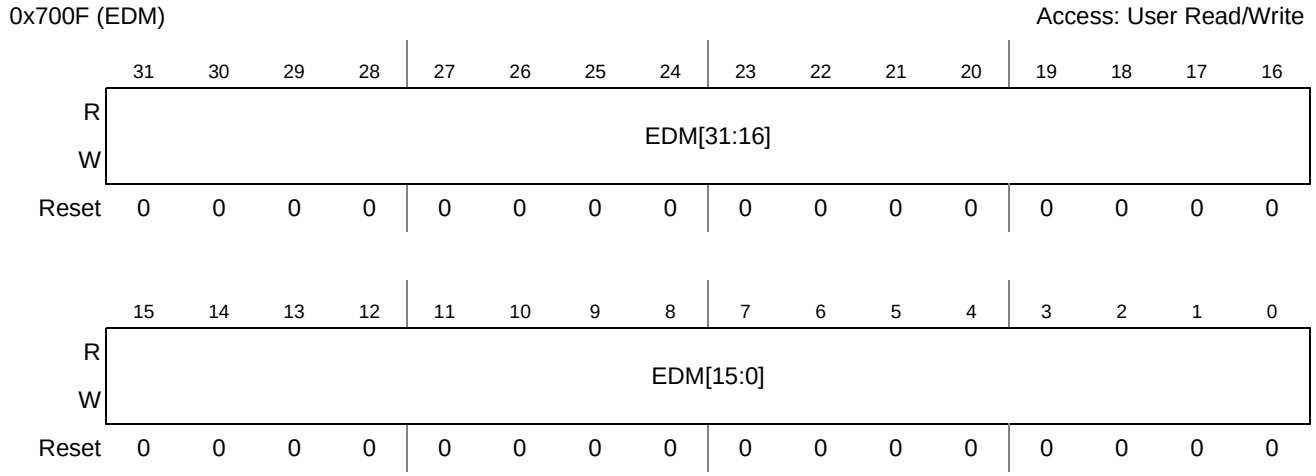


Figure 40-54. OnCE Event Cell Data Mask (EDM) Register

Table 40-58. EDM Field Descriptions

Field	Description
31–0 EDM	The event cell data mask register contains the user-defined data mask value. • This mask is applied to the data value latched from the memory bus before performing the data comparison. • Setting bit *i* of the event cell data mask register means that bit *i* of the data value latched from the address bus does not influence the result of the data comparison. • The data mask is cleared on a JTAG reset.

40.12.3.15 OnCE Real-Time Buffer (RTB)

Figure 40-55 shows the register; Table 40-59 provides its field descriptions.

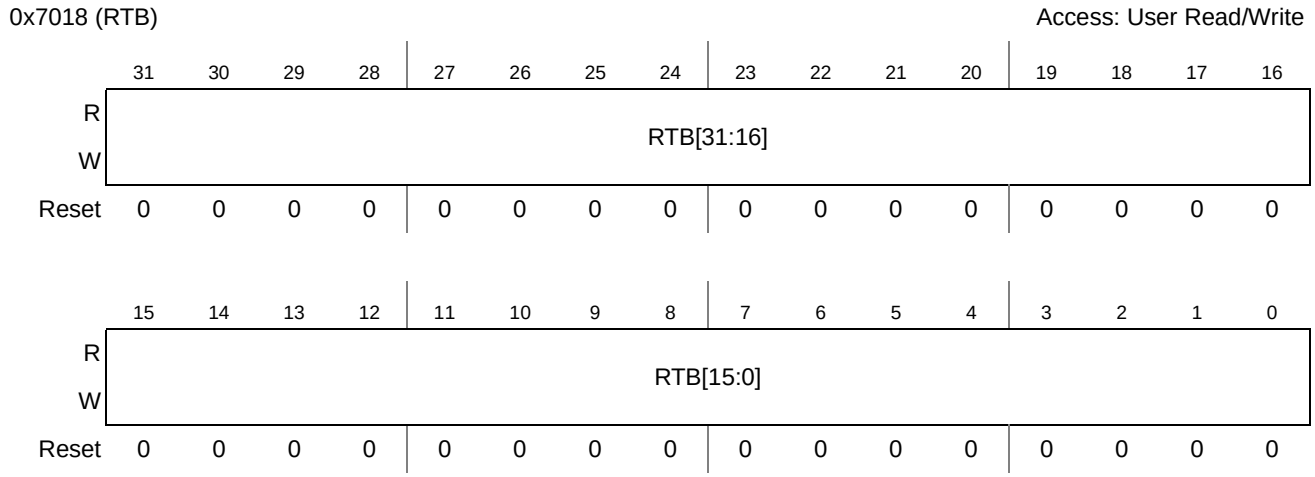


Figure 40-55. OnCE Real-Time Buffer (RTB) Register

Table 40-59. RTB Field Descriptions

Field	Description
31–0 RTB	The Real Time Buffer register stores and retrieves run-time information without putting the SDMA in debug mode. Writing to that register triggers a pulse on a specific real-time debug pin whose connection depends on the chip implementation. The RTB value can be accessed by the OnCE under AP or JTAG control using the rbuffer command.

40.12.3.16 OnCE Trace Buffer (TB)

Figure 40-56 shows the register; Table 40-60 provides its field descriptions.

0x7019 (TB) Access: User Read-Only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	TBF	TADDR[13:2]											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TADDR[1:0]		CHFADDR[13:0]													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 40-56. OnCE Trace Buffer (TB) Register

Table 40-60. TB Field Descriptions

Field	Description
31–29	Reserved
28 TBF	The Trace Buffer Flag is set when the buffer contains the addresses of a valid change of flow. The contents of the buffer should be ignored otherwise. 0 Invalid information 1 Valid information
27–14 TADDR	The target address is the address taken after the execution of the change of flow instruction.
13–0 CHFADDR	The change of flow address is the address where the change of flow is taken when executing a change of flow instruction.

40.12.3.17 OnCE Status (OSTAT)

Figure 40-57 shows the register; Table 40-61 provides its field descriptions.

0x701A (OSTAT) Access: User Read-Only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PST[3:0]				RCV	EDR	ODR	SWB	MST	0	0	0	0	ECDR[2:0]		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 40-57. OnCE Status (OSTAT) Register

Table 40-61. OSTAT Field Descriptions

Field	Description
31–16	Reserved
15–12 PST[3:0]	The Processor Status bits reflect the state of the SDMA RISC engine. • The “Program” state is the usual instruction execution cycle. • The “Data” state is inserted when there are wait-states during a load or a store on the data bus (ld or st). • The “Change of Flow” state is the second cycle of any instruction that breaks the sequence of instructions (jumps and channel-switching instructions). • The “Change of Flow in Loop” state is used when an error causes a hardware loop exit. • The “Debug” state means the SDMA is in debug mode. • The “Functional Unit” state is inserted when there are wait-states during a load or a store on the functional units bus (ldf or stf). • In “Sleep” modes, no script is running (this is the RISC engine idle state). The “after Reset” is slightly different because no context restoring phase will happen when a channel is triggered: The script located at address 0 will be executed (boot operation). • The “in Sleep” states are the same as above except they do not have any corresponding channel. They are used when entering debug mode after reset; the reason is that it is necessary to return to the “Sleep after Reset” state when leaving debug mode. 0 Program 1 Data 2 Change of Flow 3 Change of Flow in Loop 4 Debug 5 Functional Unit 6 Sleep 7 Save 8 Program in Sleep 9 Data in Sleep 10 Change of Flow in Sleep 11 Change Flow Loop Sleep 12 Debug in Sleep 13 Functional Unit in Sleep 14 Sleep after Reset 15 Restore
11 RCV	After each write access to the real time buffer (RTB), the RCV bit is set. This bit is cleared after execution of an `rbuffer` command and on a JTAG reset.
10 EDR	This flag is raised when the SDMA has entered debug mode after an external debug request.
9 ODR	This flag is raised when the SDMA has entered debug mode after a OnCE debug request.
8 SWB	This flag is raised when the SDMA has entered debug mode after a software breakpoint.

Table 40-61. OSTAT Field Descriptions (continued)

Field	Description
7 MST	This flag is raised when the OnCE is controlled from the AP peripheral interface. 0 JTAG interface controls the OnCE. 1 AP peripheral interface controls the OnCE.
2–0 ECDR[2:0]	Event Cell Debug Request. If the debug request comes from the event cell, the reason for entering debug mode is given by the EDR bits. If all three bits of the EDR are reset, then it did not generate any debug request. If the cell did generate a debug request, then at least one EDR bit is set; the meaning of the encoding is as follows: EDR[0] 1 matched addressA condition EDR[1] 1 matched addressB condition EDR[2] 1 matched data condition The encoding of the EDR bits is useful to find out more precisely why the debug request was generated. A debug request from an event cell is generated for a specific combination of the addressA, addressB, and data conditions; the value of those fields is given by the EDR bits.

40.12.3.18 Channel 0 Boot Address (MCHN0ADDR)

Figure 40-58 shows the register; Table 40-62 provides its field descriptions.

0x701C (MCHN0ADDR) Access: User Read-Only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	SMSZ	CHN0ADDR[13:0]													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 40-58. Channel 0 Boot Address (MCHN0ADDR) Register

Table 40-62. MCHN0ADDR Field Descriptions

Field	Description
31–15	Reserved
14 SMSZ	The bit 14 (Scratch Memory Size) determines if scratch memory must be available after every channel context. After reset, it is equal to 0, which defines a RAM space of 24 words for each channel. All of this area stores the channel context. By setting this bit, 32 words are reserved for every channel context, which gives eight additional words that can be used by the channel script to store any type of data. Those words are never erased by the context switching mechanism. 0 24 words per context 1 32 words per context
13–0 CHN0ADDR[13:0]	Contains the address of the channel 0 routine programmed by the AP; it is loaded into a general register at the very start of the boot and the SDMA jumps to the address it contains. By default, it points to the standard boot routine in ROM.

40.12.3.19 ENDIAN Mode Status Register (ENDIANESS)

Figure 40-59 shows the register; Table 40-63 provides its field descriptions.

0x701D (ENDIANESS) Access: User Read-Only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	BPEND	APEND
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 40-59. Endian Mode Status Register (ENDIANESS) Register

Table 40-63. ENDIANESS Field Descriptions

Field	Description
31–2	Reserved

Table 40-63. ENDIANESS Field Descriptions (continued)

Field	Description
1 BPEND	BPEND indicates the Endian mode of the BP DMA interface. The BP interface is not connected for this device. 0 BP is in Little Endian mode. 1 BP is in Big Endian mode.
0 APEND	APEND indicates the Endian mode of the Peripheral and Burst DMA interfaces. 0 AP is in Little Endian mode. 1 AP is in Big Endian mode.

40.13 SDMA Peripheral Registers

Refer to the respective peripherals' chapters more information.

40.14 SDMA Initialization

The API document MOT-SFS-IAPI-SAS-001 (version 0.04) describes the setup of the SDMA. This section provides a quick description of several initialization procedures.

NOTE

There may be differences with the actual implementation in the API.

40.14.1 Hardware Reset

After reset, the RAM that holds contexts, data, scripts, and the DMA request-channels matrix has unpredictable content. The core registers are all reset to 0, including the PC; the PCU state is *Sleep after Reset*. No channel can be activated because all of the priorities are also reset to 0.

40.14.2 Standard Boot Sequence

The following is the standard boot sequence:

1. Initialize the CONFIG register—detailed in Section 40.10.3.14, "Configuration Register (CONFIG)"—to determine the AHB/core clock ratio (1 or 2).
2. Initialize the DMA request-channels matrix (see Section 40.10.3.25, "Channel Enable RAM (CHNENBLn)").
3. Program the channel control registers—Channel Event Override (EVTOVR), Channel DSP Override (DSPOVR), Channel AP Override (HOSTOVR), and Channel Event Pending (EVTPEND)—according to the channel allocation.
4. Perform any necessary setup as required by the standard boot script in ROM (this is described in MOT-SFS-IAPI-SAS-001 (version 0.04)).
5. Trigger channel 0 with the Channel Start (HSTART) register, which starts the execution of the ROM script starting at address 0. This boot downloads channel scripts and contexts in RAM.

40.14.3 User-Defined Boot Sequence

The following is a user-defined boot sequence:

1. Initialize the Configuration Register (CONFIG), Channel Enable RAM (CHNENBLn), Channel Event Override (EVTOVR), Channel DSP Override (DSPOVR), Channel AP Override (HOSTOVR), and Channel Event Pending (EVTPEND).
2. Use the OnCE (either via its JTAG interface or its AP control registers) to download any code in the SDMA RAM. Section 40.18.5.4, "Accessing the Memory" describes how to write data to the RAM via the OnCE.
3. Use the OnCE instructions to make the PC default value point to the new boot script start address, or rely on the ROM startup script, which first jumps to the address in Channel 0 Boot Address (CHN0ADDR). (This register default address points to the standard boot script.)

40.14.4 Script Loading and Context Initialization

The execution of an SDMA script depends on both the instructions that make up the script and the data context upon which it operates. Both must be initialized before the script is allowed to execute. Each of the 32 channels has a separate data context, but may share scripts and locations in the data RAM.

The AP manages the space in program RAM and data RAM. It also manages the assignment of SDMA channels to the device drivers that need them. Channels are initialized by the AP via the channel 0 boot script. The boot channel downloads any required scripts with their data and the channels' initial contexts. Every context contains all the initial values of the registers, including the PC. Then the AP can enable any channel that becomes active and begins fetching and executing instructions from its script.

40.15 Instruction Description

The following sections introduce the instruction of the SDMA. Instruction set details are available in Section 40.19, "Instruction Set."

40.15.1 Scheduling Instructions

The following are scheduling instructions:

- `done`—The instruction causes certain scheduling or interrupt bits to be set or cleared, which may cause a change in the schedule-ability of the running channel. Then the instruction causes the SDMA to evaluate the current scheduling priorities and to choose the highest priority ready channel. If this channel is not the current channel, a context switch will take place. If there are no runnable channels, the SDMA will enter the stopped mode. The `done` 5 has a special usage reserved for debug, as explained in Section 40.15.17, "Debug Instructions."
- `yield`—These instructions are special cases of the `done` instruction. They do not modify the scheduling bits, but allow the highest pending channel (if it exists) to preempt the current channel if the pending channel priority is strictly greater than the current channel priority.
- `yieldge`—These instructions are special cases of the `done` instruction. They do not modify the scheduling bits, but allow the highest pending channel (if it exists) to preempt the current channel if the pending channel priority is strictly greater or equal to the current channel priority.

- notify—The notify instruction affects the scheduling bits, but does not cause rescheduling.

40.15.2 Conditional Branch Instructions

The conditional branch instructions of an 8-bit displacement, which is sign-extended and added to the current PC (which points to the next instruction) if the condition is satisfied. Otherwise, control passes to the next sequential instruction.

- BF—Branch if False. The branch is taken if the T bit in the processor status is zero (false).
- BT—Branch if True. The branch is taken if the T bit in the processor status is one (true).
- BSF—Branch if Source Fault. The branch is taken if the SF bit in the processor status is one.
- BDF—Branch if Destination Fault. The branch is taken if the DF bit in the processor status is one.

40.15.3 Unconditional Jump Instructions

There are two varieties of unconditional control transfers: an absolute transfer and a through-register transfer. Absolute transfers have a 14-bit address field that replaces the current PC.

- JMP—Jump. Causes the processor to jump to an absolute address encoded in the instruction itself.
- JSR—Jump to Subroutine. Causes the processor to jump to a subroutine, the address of which is encoded in the instruction itself.
- JMPR—Jump through Register. Causes the processor to jump to an absolute address contained in a General register. This instruction is meant to be used when more than one level of subroutines are required.
- JSRR—Jump to Subroutine through Register. Causes the processor to jump to a subroutine, the address of which is contained in a General register. This instruction is meant to be used when more than one level of subroutines are required.

40.15.4 Subroutine Return Instructions

The following are subroutine return instructions:

- RET—Return from Subroutine. The RET restores the contents of RPC to PC.
- LDRPC—Load from RPC to Register. THe LDRPC instruction is meant to be used when more than one level of subroutines are required. It stores the contents of RPC in any General register.

40.15.5 Loop Instruction

The following is a loop instruction:

LOOP—Enters Loop Mode. Before entering loop mode, the loop instruction can optionally clear the fault flags (SF and/or DF) based on a 2-bit field in the instruction. This feature is linked to the fact that setting SF or DF in loop mode will cause an immediate exit of the loop.

40.15.6 Miscellaneous Instructions

The following are miscellaneous instructions:

- CLRF—Clear Fault Flags. This instruction clears any combination of SF and DF.
- MOV r,s—This moves data from GReg[s] to GReg[r].
- LDI r,immediate—This loads GReg[r] with a zero-extended immediate value.

40.15.7 Logic Instructions

The following are logic instructions:

- XOR r,s—This performs an exclusive or between GReg[r] and GReg[s], and stores the result in GReg[r].
- XORI r,immediate—This performs an exclusive or between GReg[r] and a zero-extended immediate value, and stores the result in GReg[r].
- OR r,s—This performs an or between GReg[r] and GReg[s], and stores the result in GReg[r].
- ORI r,immediate—This performs an or between GReg[r] and a zero-extended immediate value and, stores the result in GReg[r].
- ANDN r,s—This performs an and between GReg[r] and the negated GReg[s], and stores the result in GReg[r].
- ANDNI r,immediate—This performs an and between GReg[r] and the negated zero-extended immediate value, and stores the result in GReg[r].
- AND r,s—This performs an and between GReg[r] and GReg[s], and stores the result in GReg[r].
- ANDI r,immediate—This performs an and between GReg[r] and a zero-extended immediate value, and stores the result in GReg[r].

40.15.8 Arithmetic Instructions

Arithmetic instructions modify the T bit in the processor status according to the result of the operation. The T bit is set if the result is zero, otherwise it is cleared.

- ADD r,s—This performs the addition of GReg[r] and GReg[s], and stores the result in GReg[r].
- ADDI r,immediate—This performs the addition of GReg[r] and a zero-extended immediate value, and stores the result in GReg[r].
- SUB r,s—This performs the subtraction of GReg[s] from GReg[r], and stores the result in GReg[r].
- SUBI r,immediate—This performs the subtraction of a zero-extended immediate value from GReg[r], and stores the result in GReg[r].

40.15.9 Compare Instructions

Compare instructions modify the T bit in the processor status according to the result of the operation. The T bit is set if the comparison is true, otherwise it is cleared.

NOTE

Only one version of the immediate form is implemented. Non-equality comparisons to immediate values will require two instructions.

- `CMPEQ r,s`—This sets T when registers `GReg[r]` and `GReg[s]` are equal.
- `CMPEQI r,immediate`—This sets T when register `GReg[r]` and the zero-extended immediate value are equal.
- `CMPLT r,s`—This sets T when register `GReg[r]` is less than and not equal to `GReg[s]`. The comparison is signed.
- `CMPHS r,s`—This sets T when register `GReg[r]` is greater than or equal to `GReg[s]`. The comparison is signed.

40.15.10 Test Instructions

Test instructions modify the T bit in the processor status according to the result of the operation. The T bit is set if any bit in the result is one, otherwise it is cleared.

- `TST r,s`—This performs an `and` between `GReg[r]` and `GReg[s]`, and sets T if the result is not zero.
- `TSTI r,immediate`—This performs an `and` between `GReg[r]` and a zero-extended immediate value, and sets `T` if the result is not zero.

40.15.11 Byte Permutation Instructions

These instructions shuffle the bytes in a register. For the purpose of describing these instructions, have the bytes in a register be numbered from the most significant as b_3, b_2, b_1, b_0.

- `RORB r`—The rotate right byte. The result is b_0, b_3, b_2, b_1.
- `REVB r`—The reverse bytes in word. The result is b_0, b_1, b_2, b_3.
- `REVBLO r`—The reverse, two low-order bytes. The result is b_3, b_2, b_0, b_1.

40.15.12 Bit Shift Instructions

The following are bit shift instructions:

- `ROR1 r`—The rotate right 1 bit. This instruction does a circular right shift of 1 bit.
- `LSR1 r`—The logical shift right 1 bit. This instruction shifts all bits to the right by 1. The high order bit is replaced by a 0.
- `ASR1 r`—The arithmetic shift right 1 bit. This instruction shifts all bits to the right by 1. The high order bit is replaced by itself.
- `LSL1 r`—The logical shift left 1 bit. This instruction shifts all bits to the left by 1. The low order bit is replaced by zero.

40.15.13 Bit Manipulation Instructions

- `BCLRI r,n`—The bit clear is immediate; clears bit number *i* in register *r.*
- `BSETI r,n`—The bit set is immediate; sets bit number *i* in register *r.*
- `BTSTI r,n`—The bit test is immediate; tests bit number *i* in register *r* (T becomes equal to the selected register bit).

40.15.14 SDMA Memory Access Instructions

All memory accesses are 32 bits. Any memory location that is implemented with less than 32 bits (for example, peripheral registers) causes unimplemented bits to be read as 0s. All memory accesses will cause either the SF or DF flags in the processor status to be set if they cause a fault. What constitutes a fault, especially when accessing peripheral registers, is a property of the memory location.

- `LD r,(b,d)`—The load instruction creates an address by adding the displacement field (d) to the contents of the base register (b). The SDMA location at the resulting address is read and placed in the destination register (r).
- `ST r,(b,d)`—The store instruction creates an address in the same manner as the load instruction. The register (r) is stored in the SDMA location at the resulting address.

40.15.15 Functional Unit Instructions

The functional unit instructions have an 8-bit field that is placed on the functional unit bus. Some of these bits are used to select which functional unit should be involved in the transfer. The remaining bits are decoded by the selected functional unit so their specific use depends on the functional unit. See Section 40.16, “Functional Units Programming Model.”

There are two functional unit instructions, as follows:

- `LDF r,fub`—The 8-bit field is placed on the functional unit bus and a read is issued to the selected functional unit. As a result of this instruction, the SF may be set in the processor status.
- `STF r,fub`—The 8-bit field is placed on the functional unit bus and a write is issued to the selected functional unit. As a result of this instruction, the DF may be set in the processor status.

40.15.16 Illegal Instructions

All instruction encodings that are illegal cause the following actions:

- The current PC (which points to one beyond the offending instruction) is put in the EPC register.
- The loop mode bit is cleared.
- The PC is set to the value stored in the Illegal Instruction Trap Address (ILLINSTADDR) register (the default value is 0x0001).

ILLEGAL—Although any instruction other than those indicated in the SDMA specification will trigger the illegal instruction mechanism, the ILLEGAL instruction code is preferred as it will always be kept as *illegal* in the possible future versions of the SDMA core.

40.15.17 Debug Instructions

The following are debug instructions:

- SOFTBKPT—The software breakpoint instruction causes the core to stop and enter debug mode. The core can then be accessed and started by the OnCE debug module only.
- done 5—This instruction is used for debugging, as it copies the contents of the PCU registers and flags to the context memory. Information on this instruction is described in Section 40.18.5.2, "Saving the Context."
- CpShReg—This instruction copies the context memory into the PCU registers and flags. Modifying the corresponding memory location before executing this instruction enables you to have the channel continue from a new instruction address. This instruction is described in Section 40.18.5.3, "Restoring the Context."

40.16 Functional Units Programming Model

The functional unit instructions cause an 8-bit code, found in the low eight bits of the instruction, to be asserted on the functional unit control bus. Some of these bits are used to select one of several functional units. Functional units which can be selected include SDMA registers such as MSA and MSD which are not mapped in the SDMA memory map, and are accessible only through the functional unit bus. These Functional Unit Registers are listed in Table 40-64. In order to establish a programming convention, assume the selection bits are some number of the most significant bits of the 8-bit code. Furthermore, some number of the least significant bits is decoded by a given functional unit to establish the type of operation to perform.

Table 40-64. Functional Unit Registers

Functional Unit	Register	Register Name	Section/Page
Burst DMA Unit	MSA	Memory Source Address Register	40.16.1.1/40-100
	MDA	Memory Source Address Register	40.16.1.2/40-100
	MD	Memory Data Buffer Register	40.16.1.3/40-101 (Write) 40.16.1.5/40-103 (Read) 40.16.1.6/40-105
	MS	Memory State Register	40.16.1.4/40-101
Peripheral DMA Unit	PSA	Peripheral Source Address Register	40.16.2.1/40-113
	PDA	Peripheral Source Address Register	40.16.2.2/40-114
	PD	Peripheral Data Buffer Register	40.16.2.3/40-114 (Write) 40.16.2.5/40-116 (Read) 40.16.2.6/40-119
	PS	Peripheral State Register	40.16.2.4/40-115

Table 40-64. Functional Unit Registers (continued)

Functional Unit	Register	Register Name	Section/Page
CRC Unit	CA	Polynomial Register	40.16.4.1/40-124
	CS	Accumulator Register	40.16.4.2/40-125

More information regarding the functional units can be found in Section 40.6.1, "CRC Calculation Unit," Section 40.6.2, "Burst DMA Unit," and Section 40.6.3, "Peripheral DMA Unit."

40.16.1 Burst DMA Unit

The DMA instructions control the DMA state machine and may cause a DMA cycle on the associated memory bus. There are four registers associated with the burst DMA unit, a Memory Source Address register (MSA), a Memory Destination Address register (MDA), a Memory Data buffer (MD), and a state register (MS).

The burst DMA has two different uses:

- A data transfer between External Memory Interface and SDMA general register
- A data transfer in copy mode where blocks of data are transferred from the source address to the destination address

40.16.1.1 Memory Source Address Register (MSA)

The source address register contains the pointer into EMI memory associated with the next read data transfer. It has byte granularity.

Reading the register with the `ldf` instruction has no side effects, and gives the address value in the EMI memory of the next data that is read by the SDMA during an `ldf MD` instruction.

Writing the source address register has two side effects: If the prefetch bit is set, a DMA read cycle (8-word read access) is issued with the new address. Any data still located in the buffer is lost. If there is valid write data in the buffer, it is necessary to force the DMA to completely flush it out before modifying MSA to guarantee all the data is effectively written to memory.

The MSA register has two modes of programming:

- Frozen—In frozen mode, the MSA register is not modified after DMA accesses.
- Incremented (default mode)—In incremental mode, MSA is incremented by the number of bytes transferred during read cycles.

40.16.1.2 Memory Destination Address Register (MDA)

The destination address register contains the pointer into EMI memory associated with the next write data transfer. It has byte granularity.

Reading the MDA register with the `ldf` instruction has no side effects. It gives the address value in the EMI memory where the next SDMA data (`stf r,MD` instruction) is stored when MD FIFO is flushed.

Writing the destination address register has one side effect. Any data still located in the buffer is lost. If there is valid write data in the buffer, it is necessary to force the DMA to completely flush it out before modifying MSA to guarantee all the data is effectively written to memory.

The MDA register has two modes of programming:

- Frozen—In frozen mode, the MDA register is not modified after DMA accesses.
- Incremented (default mode)—The MDA register is incremented by the number of bytes transferred during write cycles.

40.16.1.3 Memory Data Buffer Register (MD)

The data buffer register consists of a bank of 36 bytes that behave like FIFO. This FIFO stores the eight words received when a read burst is triggered by the DMA (DMA is in read mode). The MD register is in write mode after a writing in MDA or after an `stf MD` instruction. In that case, a burst write access is automatically triggered when there are more than eight words in MD. For bandwidth optimization, any transfers between DMA and the EMI (M3IF) controller are based on burst accesses.

An `ldf r,MD|SIZE` instruction that reads the data buffer may cause a DMA cycle, as follows:

- If there are less bytes in the FIFO than the size parameter of the instruction. For instance, if only two bytes are available in MD and a 4-byte read is requested, a burst read access is executed to complete the two bytes.
- If the prefetch bit is set, and after reading there is enough space in the FIFO to store a full burst, a burst read access is triggered.

An `stf r,MD|SIZE` instruction that writes to the data buffer may cause a DMA cycle if the number of written bytes in MD is higher than 32 (eight words) or if the flush bit is set.

When DMA is used for data transfer between SDMA and EMI (reading or writing), no immediate error is possible because the module manages a data misalignment issue; therefore, it is allowed to read/write a word to/from a halfword address. However, the addresses (source or destination) must belong to the EMI memory mapping. The only potential error, in this mode, would be the error sent back by the EMI controller when an access to a super-user page is detected. If no error is returned for the first data, it means the whole transfer on the DMA associated bus will be successful. In copy mode, an immediate error could be returned to SDMA as described in Section 40.16.1.10, “Error Management.”

40.16.1.4 State Register (MS)

The state register contains the DMA state-machine value. It can be accessed in case of an error received during a transfer. MS is also accessed to set-up the conditional yielding feature.

The initialization value of this register is 0 and it consists of the following:

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	stype	0	0	0	dtype
W																
R	0	0	0	0	y	d	e		0	0	n					
W																

Figure 40-60. MS Structure

Table 40-65. MS Field Descriptions

Field	Description
31–21	Reserved
20 stype	Source Mode. Indicates if MSA has to be incremented (or not) during accesses. 0 Frozen—MSA is not modified. 1 Incremented—MSA is incremented by the number of transferred bytes during write access.
19–17	Reserved
16 dtype	Destination Mode. Indicates if MDA has to be incremented (or not) during accesses. 0 Frozen—MDA is not modified. 1 Incremented—MDA is incremented by the number of transferred bytes during write access.
15–12	Reserved
11 y	Conditional Yielding selector. When selected, the `yield`/`yieldge` instructions will not switch channels if the Burst DMA is in Write Mode, and it has less than four bytes in its FIFO. This is aimed at reducing the number of inefficient FIFO flushes due to context switches. 0 Always yields 1 Yields conditionally (when there are less than four bytes in the FIFO in write mode)
10 d	Access Direction or DMA Mode. DMA is in write mode when data was written into MD by `stf MD` instructions, or if a previous DMA cycle on the external bus was a write access. Writing MDA or MSA does not change the DMA mode.DMA is in read mode when a previous DMA cycle was a read access, and DMA stays in read mode when data is read by SDMA with an `ldf MD` instruction. Reading MDA or MSA does not change the DMA mode. 0 Read Mode 1 Write Mode
9–8 e	Error. Indicates if the previous access was acknowledged with a bus error. 00 No error was received. 01 *reserved* 10 Error mode 11 error read burst

Table 40-65. MS Field Descriptions (continued)

Field	Description
7–6	Reserved
5–0 n	Number of bytes in the MD FIFO.

40.16.1.5 Burst DMA Write (stf)

When received from a `stf` instruction, the function code bits are interpreted as follows, depending on the addressed register:

Register	7	6	5	4	3	2	1	0
MSA	s		p	freeze	r			
MDA								
MD			f	cpy			sz	
MS								

Figure 40-61. STF Code Bits

Table 40-66. STF Code Bit Field Descriptions

Field	Description
7–6 s	Functional Unit selector 00 for Burst DMA
5 p (MSA)	Prefetch Flag 0 No prefetch 1 Prefetch required from new MSA
5 f (MD)	Forced Flush Flag 0 Automatic flush 1 FIFO contents are flushed (including the new written data).
4 freeze (MSA/MDA)	Address Freeze Mode 0 Address is normally incremented. 1 Address is frozen.
4 cpy (MD)	Copy Mode selection 0 Write Mode 1 Copy Mode
3–2 r	Register selection 00 MSA 01 MDA 10 MD 11 MS
1–0 sz (MD/MS)	Transfer Size 00 size 0 (no data stored in the FIFO) 01 byte (8 bits) 10 halfword (16 bits) 11 word (32 bits)

The possible write instructions are listed in Table 40-67 (unused bits should always be cleared).

Table 40-67. Burst DMA STF Instruction List

Binary	Assembly	Comments
00_0_0_00_00	stf r,MSA	Writes content of the SDMA general register (r) to the source address register. MSA is in incremented mode.
00_0_1_00_00	stf r,MSA|FR	Writes content of the SDMA general register (r) to the source address register. MSA is in frozen mode.
00_1_0_00_00	stf r,MSA|PF	Writes content of the SDMA general register (r) to the source address register, and starts a read burst access. MSA is in incremented mode.
00_1_1_00_00	stf r,MSA|PF|FR	Writes content of the SDMA general register (r) to the source address register, and starts a read burst access.
00_0_0_01_00	stf r,MDA	Writes content of the SDMA general register (r) to the destination address register. MDA is in incremented mode.
00_0_1_01_00	stf r,MDA|FR	Writes content of the SDMA general register (r) to the destination address register. MDA is in frozen mode.
00_1_0_10_00	stf r,MD|SZ0|FL	No data transfers between the SDMA and MD, but all valid written data of the MD is flushed to the memory. An acknowledge or error is sent back to the SDMA core on transfer completion.
00_0_0_10_01	stf r,MD|SZ8	8-bit (byte) transfer to write buffer MD
00_1_0_10_01	stf r,MD|SZ8|FL	8-bit (byte) transfer to write buffer MD and flush after transfer. All valid written data of the MD is flushed to memory.
00_0_0_10_10	stf r,MD|SZ16	16-bit (halfword) transfer to write buffer MD
00_1_0_10_10	stf r,MD|SZ16|FL	16-bit (halfword) transfer to write buffer MD and flush after transfer. All valid written data of the MD is flushed to memory.
00_0_0_10_11	stf r,MD|SZ32	32-bit (word) transfer to write buffer MD
00_1_0_10_11	stf r,MD|SZ32|FL	32-bit (word) transfer to write buffer MD and flush after transfer. All valid written data of MD is flushed to memory.
00_0_1_10_00	stf r,MD|CPY	No data transfer between SDMA and MD but starts a copy transfer whose length is given by the 4 LSB of *r* register. (Maximum burst length is eight words.)
00_0_0_11_11	stf r,MS	32-bit (word) transfer to status register MS
00_0_0_11_00	stf r,MS|SZ0	Clears the error flag (if set). Other MS bits are unchanged; this instruction is also known as `clref` MS.

NOTE

When a flush bit is set, the SDMA flushes the FIFO including the newly written data. An acknowledge is sent to the core before the flush completes (except if size 0 is used). The goal of this flush bit is to force a flush, but it is recommended to use it only when needed (for example, when finishing a row of pixels during 2D data transfers). Indeed, if this bit is omitted and if there are more than 32 bytes in the FIFO, a burst write access is automatically triggered.

Since all the `stf r,MD` instructions (including the copy mode) acknowledge the SDMA core before the store is effective (except if size 0 is used), it is recommended to perform an `ldf` from MS before terminating a channel in order to check the final error status. (The `ldf` from MS will stall the core until all the data was flushed out and the transfer status is known.)

After every `stf MD` instruction, the MDA is incremented by the number of bytes that are written in MD, except when it is programmed in frozen mode.

40.16.1.6 Burst DMA Read (ldf)

When received from an `ldf` instruction, the function code bits are interpreted as follows, depending on the addressed register:

Register	7	6	5	4	3	2	1	0
MSA	s				r			
MDA								
MD			p				sz	
MS								

Figure 40-62. LDF Code Bits

Table 40-68. LDF Code Bit Field Descriptions

Field	Description
7–6 s	Functional Unit selector 00 for Burst DMA
5 p (MD)	Prefetch Flag 0 no prefetch 1 automatic prefetch
3–2 r	Register selection 00 MSA 01 MDA 10 MD 11 MS
1–0 sz (MD)	Transfer Size 00 reserved 01 byte (8 bits) 10 halfword (16 bits) 11 word (32 bits)

Table 40-69 lists the possible write instructions (unused bits should always be cleared).

Table 40-69. Burst DMA LDF Instruction List

Binary	Assembly	Comments
00_0_0_00_00	ldf r,MSA	Copies the source address register value into an SDMA general register. It gives the M3IF address of the next data that will be read with an `ldf` MD instruction.
00_0_0_01_00	ldf r,MDA	Copies the destination address register value into an SDMA general register. It gives the M3IF address where the next incoming data will be flushed.
00_0_0_10_01	ldf r,MD|SZ8	8-bit (byte) read
00_1_0_10_01	ldf r,MD|SZ8|PF	8-bit (byte) read. If after this reading and the MD FIFO is empty, a burst read access at the MSA address is triggered.
00_0_0_10_10	ldf r,MD|SZ16	16-bit (halfword) read
00_1_0_10_10	ldf r,MD|SZ16|PF	16-bit (halfword) read. If after this reading, and the MD FIFO is empty, a burst read access at the MSA address is triggered.
00_0_0_10_11	ldf r,MD|SZ32	32-bit (word) read
00_1_0_10_11	ldf r,MD|SZ32|PF	32-bit (word) read. If after this reading and the MD FIFO is empty, a burst read access at the MSA address is triggered.
00_0_0_11_00	ldf r,MS	Copy the status register value into an SDMA general register.

NOTE

Read data is 0-extended before writing in the SDMA general registers. When reading the MD register, the DMA takes data from the FIFO if it is available. If part or whole data is not in the FIFO, an external burst read access is performed to provide the missing data. The SDMA is stalled as long as the required read data is not complete.

After every reading, MSA is incremented by the number of read bytes from MD FIFO, except when MSA is programmed in frozen mode.

40.16.1.7 Prefetch/Flush and Auto-Flush Management

The prefetch and auto-flush management enables the SDMA RISC machine to go on while a DMA access is performed. When the RISC core requires a prefetch (p = 1) to the Burst DMA, it will receive an immediate transfer acknowledge before the DMA has finished the external access. This enables the RISC core to do other things like accessing another DMA machine.

The basic principle in prefetch mode is for the DMA to anticipate data reads from the SDMA RISC engine by fetching external bursts of data as soon as there is enough space in the DMA FIFO to store it. If ever the RISC engine required data that is not available in the FIFO, the read acknowledge is delayed until the data is available, but it does not have to wait until the burst completes.

The auto-flush basic principle is similar: An automatic flush is triggered every time there are eight words to be written in the FIFO. If the FIFO is full and the RISC engine requires another write, it is stalled until the burst has started and enough space was freed in the FIFO to store that new data. This means the SDMA

RISC engine does not have to wait for the completion of a burst to receive its acknowledge and continue its processing.

In particular, an auto-flush is executed when DMA is in write mode and if the following is true:

- If the FIFO is empty and the first write is to a word-aligned address of any size (for example, the 2 LSB of MDA[1:0]= 0x0), the auto-flush is triggered immediately after the write of the 32'nd byte.
- If the FIFO is empty, and if MDA is an odd byte address (1, 3, 5, 7, and so on) and an `stf MD|SZ8` is executed, the byte is flushed to memory. Once MDA increments to a word aligned address, the auto-flush will be triggered every 32 bytes.
- If the FIFO is empty, and if MDA is a halfword address (2, 6, 0xA, and so on) and an `stf MD|SZ16` is executed, the two bytes of the incoming data are flushed to memory. Once MDA increments to a word aligned address, the auto-flush will be triggered every 32 bytes.
- If the FIFO is empty, and if MDA is not a word-aligned address (for example, 1, 2, 3, 5, 6, 7, 9, and so on), and an `stf MD|SZ32` is executed, the first 1 to 3 bytes will be flushed up to the next word aligned address. Afterwards, an auto-flush will be triggered each time the FIFO receives 32-bytes.

  Therefore, if an `stf MD|SZ32` is executed with MDA equal to 0x1 and with an empty MD FIFO, the bytes located at addresses 1, 2, and 3 are flushed, and the byte located at address 4 remains in MD FIFO. This solves the misalignment issue. Additionally, the next write instructions (`stf`) complete the FIFO until it contains eight words; then a burst write is executed by the DMA to empty the FIFO. Protocol on the external bus does not support bursts of different data types (byte, halfword, or word).

For example, consider the case where data is written using a byte access, `stf MD|SZ8`. The value of MDA during the very first byte write determines when the auto-flush will occur as follows:

- If MDA=0x0, the flush occurs following the write of byte 32
- If MDA=0x1, the flush occurs following the write of byte 1, byte 3 and byte 35.
- If MDA=0x2, the flush occurs following the write of byte 2 and byte 34.
- If MDA=0x3, the flush occurs following the write of byte 1 and byte 33.
- If MDA=0x4, the flush occurs following the write of byte 32

The flush command forces the DMA to flush all MD valid bytes to the EMI controller. An acknowledge is sent immediately to the SDMA, and any potential error is reported on a future access. It is thus essential to conclude a transfer with a last read from MS, which will stall the core until all data was flushed out and returned to the transfer status (acknowledge or error).

NOTE

During this kind of auto-flush—which occurs only at the beginning of a misaligned write transfer—no acknowledge is sent back to the SDMA, which is stalled until a flush is completed.

40.16.1.8 Data Alignment and Endianness

40.16.1.8.1 Burst DMA in Read Mode

For every read access to MD, the data returned to the SDMA core and the new FIFO state depends on the MSA status and the access size. The FIFO is considered as a stack of 36 bytes: Data is fetched externally on a 32-bit bus, but the valid bytes only are stored in the FIFO and left-aligned (for a transfer of consecutive words, it is only the first word that may be truncated). Table 40-70 shows the FIFO byte alignment strategy and the corresponding MSA, the returned data, and the new FIFO state for any access size of an internal read from MD.

Table 40-70. FIFO Read Configuration

Before read		Internal read access size	Read data	After read	
MSA[1:0]	FIFO state			MSA[1:0]	FIFO state
00	x0 x1 x2 x3 y0 y1 y2 y3 z0 z1 z2 z3 and so on...	sz8	00 00 00 x0	01	x1 x2 x3 y0 y1 y2 y3 z0
		sz16	00 00 x0 x1	10	x2 x3 y0 y1 y2 y3 z0 z1
		sz32	x0 x1 x2 x3	00	y0 y1 y2 y3 z0 z1 z2 z3
01	x1 x2 x3 y0 y1 y2 y3 z0 z1 z2 z3 t0 and so on...	sz8	00 00 00 x1	10	x2 x3 y0 y1 y2 y3 z0 z1
		sz16	00 00 x1 x2	11	x3 y0 y1 y2 y3 z0 z1 z2
		sz32	x1 x2 x3 y0	01	y1 y2 y3 z0 z1 z2 z3 t0
10	x2 x3 y0 y1 y2 y3 z0 z1 z2 z3 t0 t1 and so on...	sz8	00 00 00 x2	11	x3 y0 y1 y2 y3 z0 z1 z2
		sz16	00 00 x2 x3	00	y0 y1 y2 y3 z0 z1 z2 z3
		sz32	x2 x3 y0 y1	10	y2 y3 z0 z1 z2 z3 t0 t1
11	x3 y0 y1 y2 y3 z0 z1 z2 z3 t0 t1 t2 and so on...	sz8	00 00 00 x3	00	y0 y1 y2 y3 z0 z1 z2 z3
		sz16	00 00 x3 y0	01	y1 y2 y3 z0 z1 z2 z3 t0
		sz32	x3 y0 y1 y2	11	y3 z0 z1 z2 z3 t0 t1 t2

40.16.1.8.2 Burst DMA in Write Mode

For every write access to the MD, the new FIFO state depends on the MDA status and the access size. The FIFO is considered as a stack of 36 bytes: Data is stored in the FIFO according to the internal access size

and the former MDA value. Table 40-71 shows the FIFO byte alignment strategy corresponding to MDA, as well as the new FIFO state for any access size of an internal write to MD.

Table 40-71. FIFO Write Configuration

Before Write		Internal Write Access Size	Written Data	After Write	
MDA[1:0]	FIFO state			MDA[1:0]	FIFO state
00	tt uu vv ww ?? ?? ?? ?? ?? ?? ?? ?? and so on...	sz8	?? ?? ?? x0	01	tt uu vv ww x0 ?? ?? ?? ?? ?? ?? ??
		sz16	?? ?? x0 x1	10	tt uu vv ww x0 x1 ?? ?? ?? ?? ?? ??
		sz32	x0 x1 x2 x3	00	tt uu vv ww x0 x1 x2 x3 ?? ?? ?? ??
01	tt uu vv ww xx ?? ?? ?? ?? ?? ?? ?? and so on...	sz8	?? ?? ?? x0	10	tt uu vv ww xx x0 ?? ?? ?? ?? ?? ??
		sz16	?? ?? x0 x1	11	tt uu vv ww xx x0 x1 ?? ?? ?? ?? ??
		sz32	x0 x1 x2 x3	01	tt uu vv ww xx x0 x1 x2 x3 ?? ?? ??
10	tt uu vv ww xx yy ?? ?? ?? ?? ?? ?? and so on...	sz8	?? ?? ?? x0	11	tt uu vv ww xx yy x0 ?? ?? ?? ?? ??
		sz16	?? ?? x0 x1	00	tt uu vv ww xx yy x0 x1 ?? ?? ?? ??
		sz32	x0 x1 x2 x3	10	tt uu vv ww xx yy x0 x1 x2 x3 ?? ??
11	tt uu vv ww xx yy zz ?? ?? ?? ?? ?? and so on...	sz8	?? ?? ?? x0	00	tt uu vv ww xx yy zz x0 ?? ?? ?? ??
		sz16	?? ?? x0 x1	01	tt uu vv ww xx yy zz x0 x1 ?? ?? ??
		sz32	x0 x1 x2 x3	11	tt uu vv ww xx yy zz x0 x1 x2 x3 ??

NOTE

If the FIFO mode changes from a write to a read mode, all remaining written bytes in MD are lost but no error is returned. Typically, this happens if an `ldf` MD is executed after `stf` MD instructions. Before a mode change, it is recommended to force the flush of a potential remaining byte by a `stf MD|SZ0|FL` instruction. In the same way, if a FIFO mode changes from a read to a write mode, all prefetched data present in the FIFO is lost and no error is returned.

40.16.1.8.3 Endianness

Big and Little Endian are supported by the M3IF DMA, but data is always stored in MD in Big Endian. Byte manipulation is performed when data is exchanged with an M3IF controller (for example, during read or write burst accesses).

40.16.1.9 Copy Mode

A mechanism is available to perform fast AP-to-AP transfers. Data does not flow through the SDMA core: It is kept in the DMA FIFO. This mechanism is selected when writing MD with a special option in the instruction code (copy flag). It is possible to transfer up to eight words in one SDMA instruction (this does not mean in one cycle). In this mode, every time an `stf MD|CPY` is executed, a read burst is executed and directly followed by a write burst transfer. Burst transfers are limited to eight words. The size of the transfer (in words)—given by the SDMA general register (4 LSB)—is also limited to eight. The following SDMA code shows how 100 bytes could be copied from the MSA address to the MDA address. This is sample code only.

```
        ldi r0,@src
        stf r0,MSA                  // Source address setup
        ldi r1,@dst
        stf r1,MSA                  // Destination address setup
        ldi r0,0x64                 // data transfer counter
        ldi r1,0x8
MAIN_XFER:
        cmphs r0,r1                 // Is r0 >= 0x8
        bf LAST_XFER                // If not, jump to last transfer label
        stf r1,MD|CPY               // Copy 8 words from MSA to MDA address.
        subi r0,0x8                 // Decrement counter
        jmp MAIN_XFER               // return to main transfer loop
LAST_XFER:
        stf r0,MD|CPY
```

The main transfer loop is executed 12 times; then `r0` equals 4 and the last transfer loop is run.

In this mode, an acknowledge is transmitted to the core as soon as the read burst can start; thus, a first copy instruction returns an immediate acknowledge and subsequent copy instructions will be acknowledged as soon as the previous copy has finished.

40.16.1.10 Error Management

Another point to consider is the management of errors. Because the DMA immediately sends an acknowledge to the RISC core (except for the `stf MS|SZ0|FLS` instruction), it assumes no error will occur. If an error occurs, it is flagged (transfer error acknowledge) for the following DMA access. This should not be a problem if the DMA is used properly. The MD accesses are meant to stall the SDMA as little as possible to optimize throughput and hide calculation time. Therefore, final access to MS should be performed before closing a channel. This access waits until any pending operation is finished in the burst DMA and gather any remaining error.

In copy mode, an error could be immediately returned to the SDMA on execution of the `ldf` copy or `stf` copy instruction. It happens when MSA or MDA are not word addresses (for example, 0[4]). This is because copy mode must only be used for transferring a large packet of aligned data.

When an error is received during a *read* transfer to the external bus, which may occur during the burst accesses, the MD FIFO contains the valid beats of the burst, and the error flag of MS is set to 2'b11 (error read burst). It is possible to read MS ("n" field) to know how much valid data remains in MD and when MD is empty (after `ldf` instructions). The next read MD instruction sets the MS error flag to 2'b10 (error mode), and an error is sent back to the SDMA core. In error mode, it is possible to read MSA, which gives the address of the error data. Any attempt to read or write MD, or to modify MDA or MSA in error mode, gives rise to an error; therefore, an error flag must be reset by clearing MS at the end of the SDMA code section responsible for error management.

In "error read burst" mode, writing MDA, MSA, or MD, or starting a copy transfer by a `stf MD|COPY` instruction will cancel the error mode. Table 40-72 shows when an immediate error is sent back according to the executed instruction.

Table 40-72. Possibilities in ERROR READ BURST Mode

DMA Instruction	Immediate Error	Comments
stf rn, MD stf rn, MSA (|U |PF) stf rn, MDA stf rn,MD|COPY	NO	Error mode is reset. MSA, MDA, or MD are updated and a DMA cycle may start. For the `stf MD|COPY`, a copy loop is executed.
stf rn, MS	NO	MS is updated.

ldf rn, MS
ldf rn, MSA
ldf rn, MDA
	NO	MS, MSA, and MDA could be read in ERROR READ mode without any side effects (for example, no DMA cycle is triggered).

ldf rn, MD	YES/NO	Immediate error if there is no more data available for read in the FIFO.

When an error is received during a *write* transfer, the error is reported to the next DMA access. In this case, an error is sent to the SDMA core and the DMA goes to its error mode. Reading MS gives the number of bytes that remain in MD; reading MDA gives the address of the error data. Any attempt to read or write MD, or to modify MDA or MSA in error mode, give rise to an error; therefore, an error flag must be reset by clearing MS at the end of the SDMA code section responsible for error management.

Table 40-73. Possibilities in ERROR Mode

DMA Instruction	Immediate Error	Comments
stf rn, MD stf rn, MSA stf rn, MDA	Yes	Any attempt to modify MD, MSA, MDA will raise an immediate error and burst DMA remains in error mode. When address registers are write-accessed, an error is returned.
stf rn, MS	No	This is the only way to exit error mode. MS[9:8] must be reset by an `stf MS|SZ0` instruction.

ldf rn, MS
ldf rn, MSA
ldf rn, MDA
	No	MS, MSA, and MDA could be read in error mode without any side effects (for example, no DMA cycle is triggered).

ldf rn, MD	Yes	Whatever the DMA direction (read or write), an `ldf rn` triggers an immediate error.

40.16.1.11 Conditional Yielding

The standard SDMA transfer is based upon a hardware loop that has the following structure:

```
loop
load Rn,source  // can be ldf or ld
<computation>   // can be done through functional units
store Rn,dest   // can be st or stf
done 0          // yield
```

This structure needs to be kept independent of the functional units' particularities regarding the context switch. However, there can be variations in the context switch's efficiency, which can depend on the number of data received up to that point, and on the data itself.

The DMA, with its 8-word burst capability, has a preferable context switch period when its address register is 8-word aligned: It is the only moment that occurs once every eight loops when the succession of bursts is not broken by the context switch. When this is not the case, a context switch requires the storing (or loading) of less than eight words, which requires separate accesses and is far less efficient. The rest of the 8-word packet is stored (or loaded) after the context restore, and this is done as separate accesses.

The proposed solution is a conditional yielding, which occurs only when the DMA is in an optimum state. It does not require any modification to the scripts. The condition is decided at the DMA level.

The DMA can be programmed in two modes—conditional or always-true—for every channel, which provides complete flexibility. By default, the DMA is not in conditional mode.

The DMA condition is computed from the FIFO fill level and the various modes, as follows:

- When copy mode is selected, regardless of the transfer direction ('read' or 'write'), the condition is always true.
- In read mode, the condition is always true.
- In write mode, the condition is true when there are four bytes or less in the FIFO; it is false when there are more than four bytes. The 4-byte limit comes from the possibility of saving those bytes as MD with absolutely no impact on the bus accesses.

The aim at conditional yielding is to avoid splitting bus accesses (especially bursts).

40.16.2 Peripheral DMA Unit

The peripheral DMA unit is connected to the Multi-Layer AHB Crossbar Switch of the AP platform. Its goal is to perform data transfers between any modules connected to the AHB bus of this platform. These modules are either peripherals or memories. The peripheral DMA could be seen as the AP DMA controller.

The DMA performs data transfers in three modes:

- Read mode, where data is read from peripherals or from memory connected to the AP and copied in a SDMA general register.
- Write mode, where data of a general register has to be written in a peripheral or a memory.
- Copy mode, where data is read from a peripheral (or memory) at a source address (PSA) and automatically written to a peripheral (or memory) at a destination address (PDA).

In copy mode, no SDMA general register is involved as transferred data only goes through the data register of the DMA.

The peripheral DMA has three addressing modes: frozen, incremented, and decremented, as follows:

- Frozen mode—When source or destination addresses are frozen, their value is not modified after a transfer. This mode is typically used for addressing peripheral FIFOs located at a fixed address.
- Incremented mode—When source or destination addresses are in incremented mode, after every transfer they are incremented by the number of bytes transferred.
- Decremented mode—In decremented mode, addresses are decremented by the number of bytes transferred.

The peripheral DMA registers are as follows:

- Two, 32-bit address registers (PSA and PDA) that respectively contain the source address for a read access and the destination address for a write access
- A 32-bit status register (PS) that contains information on the peripheral DMA configuration, such as the number of valid bytes in the data register, the error flag, the source and destination address mode, and so on.
- A 32-bit data register (PD) that stores data involved in a data transfer

40.16.2.1 Peripheral Source Address Register (PSA)

The source address register contains a pointer to a source peripheral or a memory associated with the next read data transfer. It has byte granularity. It is based on the following:

- A 32-bit register (PSA) to store the address value
- A 2-bit register (stype) to store the source address mode (frozen, incremented, or decremented)
- A 2-bit register (ssize) to store the source target data path size (byte, halfword, or word)

Reading the register with the `ldf` instruction has no side effects and gives the address value of the next data that will be read by the SDMA during an `ldf MD` instruction. Writing the source address register may

have side effects. If there is valid write data in the data register and the source address is changed, the write data is discarded. If the prefetch bit is set, a DMA read cycle is issued with the new address.

When PSA is to be written, you must specify the source target address mode, providing its size (byte, halfword, or word). This enables omission of the size field in all `ldf MD` instructions. When DMA performs a read cycle, its size is given by the value of the PSA source size register (ssize). If source is a memory in incremented mode, first programmed in word mode (`stf PSA|SZ32|I`), and if an SDMA script needs to read bytes from this memory, the size of the source target must be updated before executing new accesses. The source address mode and its size are given by labels added to the `stf PSA` instruction as described in the write section. The ssize and stype registers are part of the DMA status register (PS).

Writing to PSA may issue an immediate error if the source size is not compatible with the value to be written into the PSA register. For instance, writing a 2 in PSA and specifying that it is memory-accessed in word mode creates an immediate error.

40.16.2.2 Peripheral Destination Address Register (PDA)

The destination address register contains a pointer to a source peripheral or a memory associated with the next write data transfer. It has byte granularity. It is based on the following:

- A 32-bit register (PDA) to store the address value
- A 2-bit register (dtype) to store the destination address mode (frozen, incremented, or decremented)
- A 2-bit register (dsize) to store the destination target data path size (byte, halfword, or word)

Reading the register with the `ldf` instruction has no side effects, and gives the address value of the next data that will be written by SDMA during an `stf MD` instruction. Writing the destination register has no side effect. Similar to the PSA register, the destination address mode and source are specified in the `stf PDA` instruction and may also generate an error in case of incorrect programming.

40.16.2.3 Peripheral Data Register (PD)

The data register of the peripheral DMA is a 32-bit register. When the destination address is correctly set up, any writing to PD will automatically flush the new input data. The number of SDMA bytes that will be transferred is given by the PDA size register. Unlike other SDMA DMAs, PD is not a FIFO: It is not used to accumulate bytes that from the SDMA and must be packed before being sent to external memories. In read mode, and if the source address is correctly set up, an `ldf` instruction will empty PD. If a prefetch is required along with the instruction, the DMA will initiate a new read transfer.

Reading PD in prefetch mode only stalls the SDMA when the prefetched data is not yet available. Writing PD only stalls the SDMA if the previous write operation was not completed. As soon as the previous operation is over, the acknowledge is sent back to the SDMA RISC engine.

An error flag—part of PS—is set when an external access fails. The error is thus reported to the next SDMA instruction that involves the peripheral DMA.

40.16.2.4 Peripheral State Register (PS)

The state register contains the DMA state-machine value. It can be accessed in case of an error received during a transfer. Although all PS fields can be written by an `stf` instruction, it is recommended to access only the error bit (to reset it). Modifying other PS fields will provide an un-guaranteed DMA behavior.

The initialization value of PS is 0, and it consists of the following structure:

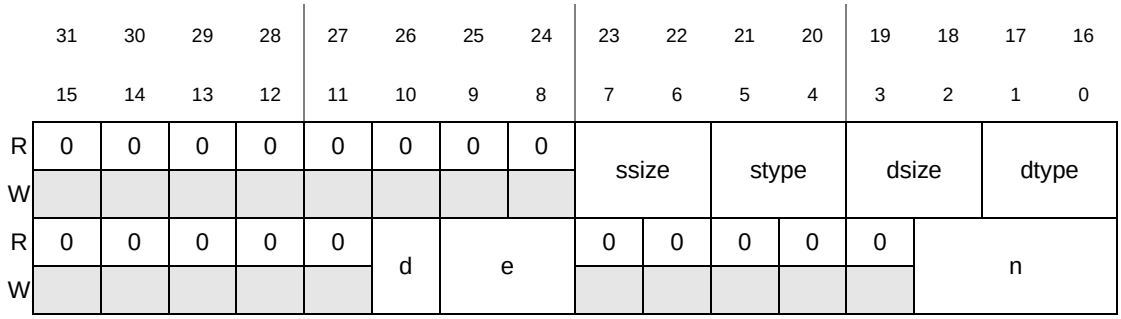


Figure 40-63. PS Structure

Table 40-74. PS Field Descriptions

Field	Description
31–24	Reserved
23–22 ssize	Source Target Size. Determines the size of the read transfers on the external bus. It should match the accessed device characteristics. 00 *reserved* 01 Byte (8 bits) 10 halfword (16 bits) 11 word (32 bits)
21–20 stype	Source address Mode. Determines whether PSA is incremented, decremented, or kept unmodified after every read from the external bus. 00 Frozen Mode 01 Incremented Mode 10 Decremented Mode 11 *reserved*
19–18 dsize	Destination Target Size. Determines the size of the write transfers on the external bus. It should match the accessed device characteristics. 00 *reserved* 01 Byte (8 bits) 10 halfword (16 bits) 11 word (32 bits)
17–16 dtype	Destination address Mode. Determines whether PDA is incremented, decremented, or kept unmodified after every write on the external bus. 00 Frozen Mode 01 Incremented Mode 10 Decremented Mode 11 *reserved*
15–11	Reserved

Table 40-74. PS Field Descriptions (continued)

Field	Description
10 d	Direction Flag or DMA Mode. DMA is in write mode when data was written into PD by `stf PD` instructions, or if a previous DMA cycle on the external bus was a write access. Writing PDA or PSA does not change the DMA mode. DMA is in read mode when a previous DMA cycle was a read access, and DMA stays in read mode when data is read by the SDMA with an `ldf PD` instruction. Reading PDA or PSA does not change the DMA mode. 0 Read Mode 1 Write Mode
9–8 e	Error. Indicates if the previous access was acknowledged with a bus error. 00 No error was received. 01 *reserved* 10 Error mode 11 Error read
7–3	Reserved
2–0 n	number of bytes in PD

NOTE

dtype, dsize, stype, and ssize are updated when PSA and PDA are written.

40.16.2.5 Peripheral DMA Write (stf)—Write Mode

When written by an `stf` instruction, the function code bits are interpreted as follows:

Register	7	6	5	4	3	2	1	0
PSA	s		p	ar	am		sz	
PDA								
PD			pdsel					
PS			pssel					

Figure 40-64. STF Code Bits

Table 40-75. STF Code Bits Field Descriptions

Field	Description
7–6 s	Functional Unit selector 11 for Peripheral DMA
5 p (PSA)	Prefetch Flag 0 no prefetch 1 automatic prefetch
4 ar (PSA/PDA)	Address Register Selector 0 PSA 1 PDA

Table 40-75. STF Code Bits Field Descriptions (continued)

Field	Description
3–2 am (PSA/PDA)	Address Mode. Determines how PSA or PDA is modified after every read or write access to the PD. 00 Frozen—Address registers are not modified after the transfer. 01 Incremented—Address registers are incremented by the number of transferred bytes. 10 Decremented—Address registers are decremented by the number of transferred bytes. 11 Updated—PSA and PDA are not modified. Either address mode is not modified, but the width of the data path is updated by the sz field.
1–0 sz	Transfer Size 00 *reserved* 01 byte (8 bits) 10 halfword (16 bits) 11 word (32 bits)
5–0 pdsel	PD access selector 001000 is the only valid option
5–0 pssel	PS access selector 111111 writes to PS 001100 only clears the error flag in PS

Due to the large number of possible `stf` instructions, Table 40-76 provides only a short list of all the possible write instructions:

Table 40-76. Peripheral DMA STF Instruction List

Binary	Assembly	Comments
11_00_00_01 11_00_00_10 11_00_00_11	stf Rn, PSA|SZ8 |F stf Rn, PSA|SZ16|F stf Rn, PSA|SZ32|F	• Source is a byte, halfword, or word target at the Rn address. Any further PD read instructions will trigger a byte, halfword, or word access to the source. • Source address is frozen.
11_10_00_01 11_10_00_10 11_10_00_11	stf Rn,PSA|SZ8 |F|PF stf Rn,PSA |SZ16|F|PF stf Rn,PSA |SZ32|F|PF	• Source is a byte, halfword, or word target at the Rn address. Any further PD read instructions will trigger a byte, halfword, or word access to the source. • 1, 2, or 4 bytes are *fetched* from the peripheral source. • Source address is frozen.
11_00_01_01 11_00_01_10 11_00_01_11	stf Rn, PSA|SZ8 |I stf Rn, PSA|SZ16|I stf Rn, PSA|SZ32|I	• Source is a byte, halfword, or word target at the Rn address. Any further PD read instructions will trigger a byte, halfword, or word access to the source. • Source address is in incremented mode: PSA = PSA + 1,2 or 4 after read PD.
11_10_01_01 11_10_01_10 11_10_01_11	stf Rn, PSA|SZ8 |I|PF stf Rn, PSA|SZ16|I|PF stf Rn, PSA|SZ32|I|PF	• Source is a byte, halfword, or word target at the Rn address. Any further PD read instructions will trigger a byte, halfword, or word access to the source. • Source address is in incremented mode: PSA = PSA + 1, 2, or 4 after read PD. • 1, 2, or 4 bytes are *fetched* from the peripheral source.

Table 40-76. Peripheral DMA STF Instruction List (continued)

Binary	Assembly	Comments
11_00_10_01 11_00_10_10 11_00_10_11	stf Rn, PSA|SZ8 |D stf Rn, PSA|SZ16|D stf Rn, PSA|SZ32|D	• Source is a byte, halfword, or word target at the Rn address. Any further PD read instructions will trigger a byte, halfword, or word access to the source. • Source address is in incremented mode: PSA = PSA—1,2, or 4 after read PD.
11_10_10_01 11_10_10_10 11_10_10_11	stf Rn, PSA|SZ8 |D|PF stf Rn, PSA|SZ16|D|PF stf Rn, PSA|SZ32|D|PF	• Source is a byte, halfword, or word target at the Rn address. Any further PD read instructions will trigger a byte, halfword, or word access to the source. • Source address is in incremented mode: PSA = PSA—1,2, or 4 after read PD. • 1, 2, or 4 bytes are *fetched* from the peripheral source.
11_00_11_01 11_00_11_10 11_00_11_11	stf Rn, PSA|SZ8 |U stf Rn, PSA|SZ16 |U stf Rn, PSA|SZ32 |U	• *Update* source pointer to memory, which becomes a pointer to a memory accessed in byte, halfword, or word. • PSA value is not modified by Rn. • Bytes present in PD are lost.
11_10_11_01 11_10_11_10 11_10_11_11	stf Rn, PSA|SZ8 |PF|U stf Rn, PSA|SZ16 |PF|U stf Rn, PSA|SZ32 |PF|U	• *Update* source pointer, which becomes a pointer to a target accessed in byte, halfword, or word. • PSA value is not modified by Rn. • Bytes present in PD are lost. • 1, 2, or 4 bytes are *fetched* from the memory source.
11_01_00_01 11_01_00_10 11_01_00_11	stf Rn, PDA|SZ8 |F stf Rn, PDA|SZ16|F stf Rn, PDA|SZ32|F	• Destination is a byte, halfword, or word target at the Rn address, and any further PD write instructions will trigger byte, halfword, or word access to the destination. • Destination address is frozen.
11_01_01_01 11_01_01_10 11_01_01_11	stf Rn, PDA|SZ8 |I stf Rn, PDA|SZ16|I stf Rn, PDA|SZ32|I	• Destination is a byte, halfword, or word target at the Rn address, and any further PD write instructions will trigger byte, halfword, or word access to the destination. • Destination address is in incremented mode: PDA = PDA + 1, 2, or 4 after write PD.
11_01_10_01 11_01_10_10 11_01_10_11	stf Rn, PDA|SZ8 |D stf Rn, PDA|SZ16|D stf Rn, PDA|SZ32|D	• Destination is a byte, halfword, or word target at the Rn address, and any further PD write instructions will trigger byte, halfword, or word access to the destination. • Destination address is in incremented mode: PDA = PDA—1, 2, or 4 after write PD.
11_01_11_01 11_01_11_10 11_01_11_11	stf Rn, PDA|SZ8 |U stf Rn, PDA|SZ16 |U stf Rn, PDA|SZ32 |U	• *Update* destination pointer to memory, which becomes a pointer to a memory accessed in byte, halfword, or word. • PDA value is not modified by Rn • bytes present in PD are lost
11_00_10_00	stf Rn, PD	• Write “dsize” bytes of Rn in PD and automatically flush to destination target
11_11_11_11	stf Rn, PS	• Write status register
11_00_11_00	stf Rn,clrefPS	• Clear error flag if set

NOTE

When writing PD, size information is not important: It is embedded in the dsize field of PDA register. If dsize is 1, 2, or 4, then one, two, or four bytes from Rn is written to the PD register, and automatically flushed out to the destination target.

40.16.2.6 Peripheral DMA Read (ldf)—Read Mode

When received from an `ldf` instruction, the function code bits are interpreted as follows.

<table>
<tr><th>Register</th><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>PSA</td><td colspan="2" rowspan="4">s</td><td></td><td rowspan="2">ar</td><td rowspan="3">a</td><td></td><td></td><td></td></tr>
<tr><td>PDA</td><td></td><td></td><td></td><td></td></tr>
<tr><td>PD</td><td>p</td><td>cpy</td><td></td><td></td><td></td></tr>
<tr><td>PS</td><td colspan="6">pssel</td></tr>
</table>

Figure 40-65. LDF Code Bits

Table 40-77. LDF Code Bits Descriptions

Field	Description
7–6 s	Functional Unit selector 11 for Peripheral DMA
5 p (PD)	Prefetch Flag 0 no prefetch 1 automatic prefetch
4 ar (PSA/PDA)	Address Register Selector 0 PSA 1 PDA
4 cpy (PD)	Copy Mode 0 standard access 1 copy mode access
3 a	Register Set selection 0 PSA or PDA 1 PD or PS
5–0 pssel	PS access selector 111111 is the only valid option to read PS

Table 40-78 provides a list of supported `ldf` instructions.

Table 40-78. Peripheral DMA LDF Instruction List

Binary	Assembly	Comments
11_0_0_0_000	ldf Rn, PSA	Reads 32-bit of PSA value
11_0_1_0_000	ldf Rn, PDA	Reads 32-bit of PDA value
11_0_0_1_000	ldf Rn, PD	Reads programmed source size bytes of PD (0-extended)

Table 40-78. Peripheral DMA LDF Instruction List (continued)

Binary	Assembly	Comments
11_1_0_1_000	ldf Rn, PD|PF	Reads programmed source size bytes of PD (0-extended), and starts a prefetch at PSA address.
11_0_1_1_000	ldf Rn, PD|COPY	Starts a copy transfer from the source target at the PSA address to the destination target at the PDA address. No data transmits through Rn, but Rn contents are lost (Rn is loaded with PD temporary contents that are *not* the copied data).
11_111111	ldf Rn, PS	Reads 32-bit of PS value

NOTE

When reading PD, size information is not important: It is embedded in the ssize field of the PSA register. If ssize is 1, 2, or 4, the one, two, or four bytes is transferred from PD to Rn. Read data is 0-extended.

40.16.2.7 Copy Mode

Like burst DMA, the peripheral DMA unit has a copy mode that is used when data transfers do not involve SDMA general registers. Data is read from the source target at a PSA address, stored in PD, and then automatically flushed to the destination target at the PDA address. Copy mode is only available for transfers that involve two targets of the same data path width. Since copy mode is invoked with an `ldf` instruction, the *loaded* general purpose register loses its previous contents. (However, the new contents are unpredictable as they depend on temporary values that are seen on the external AHB bus.)

40.16.2.8 Error Management

Peripheral DMA generates two kinds of errors: the immediate error that sanctioned incorrect register programming; and the error triggered by the previous access and stored in the error flag of PS until a DMA instruction is executed.

40.16.2.8.1 Immediate Errors

Table 40-79 lists all incorrect DMA register setups.

Table 40-79. Immediate Errors with Peripheral DMA

Rn[1:0] values	DMA instruction	Comments
0x01 0x11	stf Rn, PSA|SZ16|F stf Rn, PSA|SZ16|I stf Rn, PDA|SZ16|F stf Rn, PDA|SZ16|I	If PSA points to a halfword peripheral or to a halfword address in memory, its value must be 0 modulo 2.
0x01 0x10 0x11	stf Rn, PSA|SZ32|F stf Rn, PSA|SZ32|I stf Rn, PDA|SZ32|F stf Rn, PDA|SZ32|I	If PSA points to a word peripheral or to a word address in memory, its value must be 0 modulo 4.

Table 40-79. Immediate Errors with Peripheral DMA (continued)

PSA[1:0]–PDA[1:0]	DMA instruction	Comments
0x01 0x10 0x11	stf Rn, PSA|SZ32|U stf Rn, PDA|SZ32|U	When PDA or PSA is updated and becomes a pointer to a word address in memory, its content must be 0 modulo 4.
0x01 0x11	stf Rn, PSA|SZ16|U stf Rn, PDA|SZ16|U	When PDA or PSA is updated and becomes a pointer to a halfword address in memory, its content must be 0 modulo 2.
Read/Write PD instruction	**Comments**	
stf Rn,PD ldf Rn,PD	If PDA size (dsize) has never been set up before an `stf PD` instruction (dsize=0) If PSA size (ssize) has never been set up before an `ldf PD` instruction (ssize=0)	
ldf Rn,PD|CPY	Copy mode is possible only between two targets whose data path width is identical. It is P8↔P8, P16↔P16, or P32↔P32 regardless of the way the address registers are incremented.	

40.16.2.8.2 Data Transfer Errors

When PSA and PDA are correctly set up, the only error that may arise for an `ldf PD` or `stf PD` instruction would be the error of the previous DMA cycle. Error handling is driven by a single consideration: When an error occurred during a data read on the AHB interface, this error should appear as a transfer error to the core when the core attempts to retrieve the data that was not successfully read from the accessed device (memory or peripheral). When an error occurred during a write access to the AHB interface, the data is still available in PD and should not be destroyed by subsequent core accesses: The core must be warned about the error issue.

There are three error handling mechanisms for each case: Read Error (First Phase), Write Error and Read Error (Second Phase), and Copy Mode Errors handling.

40.16.2.8.3 Read Error (First Phase)

If an error occurred during a prefetch command, the peripheral DMA enters its ERROR READ mode (PS[9:8]=11). In this mode, the error is reported on the next `ldf PD` instruction and writing PSA, PDA, or PD will cancel the error flag. The module returns no error mode and instructions are normally executed (a DMA cycle may be triggered). Similarly, initiating a copy transfer will reset the error flag and start a copy transfer. Table 40-80 details which instructions can be executed in this mode.

Table 40-80. Possibilities in ERROR READ Mode

DMA Instruction	Immediate Error	Comments
stf rn, PD stf rn, PSA (|U |PF) stf rn, PDA ldf rn,PD|COPY	NO	Error mode is reset, PSA or PDA are updated, or a write cycle is started. For the `ldf PD|COPY`, a copy loop is executed.
stf rn, PS	NO	PS is updated.

Table 40-80. Possibilities in ERROR READ Mode (continued)

DMA Instruction	Immediate Error	Comments

ldf rn, PS
ldf rn, PSA
ldf rn, PDA
	NO	PS, PSA, and PDA could be read in ERROR READ mode without any side effects (for example, no DMA cycle is triggered).

ldf rn, PD	YES	Error of the previous read access is reported here and the peripheral DMA enters its ERROR mode.

40.16.2.8.4 Write Error and Read Error (Second Phase)

The peripheral DMA enters its ERROR mode (PS[9:8]=10) when the previous DMA write cycle failed, or, as explained in Read Error (First Phase), when an `ldf PD` is executed while the module is in ERROR READ mode. When a DMA cycle failed, address registers (PSA, PDA) are not modified and continue to point to the problematic address. In ERROR mode, `stf` instructions may raise an immediate error, and `ldf` instructions will not (as detailed in Table 40-81).

Table 40-81. Possibilities in ERROR Mode

DMA Instruction	Immediate Error	Comments
stf rn, PD stf rn, PSA stf rn, PDA	YES	Any attempt to modify PD, PSA, or PDA will raise an immediate error, and the peripheral DMA stays in ERROR mode. When address registers are write accessed, an error is returned.
stf rn, PS	NO	This is the only way to exit the ERROR mode. PS[3] must be reset by an `stf PS` instruction.

ldf rn, PS
ldf rn, PSA
ldf rn, PDA
	NO	PS, PSA, and PDA could be read in ERROR mode without any side effects (for example, no DMA cycle is triggered).

ldf rn, PD	YES	Whatever the DMA direction (read or write), an `ldf rn, PD` instruction will show an immediate error.

40.16.2.8.5 Copy Mode Errors

Because copy mode is a write access that follows a read access, there are two possible cases of bus error.

When the read access incurs a bus error, the peripheral DMA behaves exactly as described in Read Error (First Phase) on page 40-121 and Write Error and Read Error (Second Phase) on page 40-122: It enters its ERROR READ mode, and so on.

When the error occurred during the write access of the copy transfer, the DMA enables the core to retrieve the data that was read because it is assumed the read from the peripheral removed the data from its source device. Therefore, the data to be flushed is still in PD. Any subsequent access to PD triggers an error to the core, which should execute its error handling procedure.

Once the ERROR mode is left (after writing to PS), it is possible for the core to retrieve the data in PD with an `ldf` instruction or try to flush PD contents once again (for example, when the error was due to a full FIFO and the script waited for the FIFO to be emptied) with another `ldf` instruction in copy mode. This latter instruction detects that there is valid data in PD, tries to flush it, and thus skips the read phase

of the copy instruction. This is a different behavior from the usual `stf PD` instruction that overwrites PD with the selected General Purpose register contents. The same mechanism can be used any time PD holds data that is not written because of a bus error on the AHB interface; when the data was written via a copy instruction, or via the usual `stf PD` instruction.

40.16.2.8.6 Error Check Example

Example 40-1 illustrates an example checking for both immediate and data transfer errors on a store to the PD register. The first `bdf` instruction checks for an immediate error, but if a data transfer error occurred it is reported until the next instruction to access the Peripheral DMA. A second check of the error flags is done after the stf PDA instruction. This will capture an error due either to a data transfer error from the `stf PD` instruction, or an immediate error from the stf PDA instruction.

Example 40-1. Peripheral DMA Error Check

```
clrf     0                 // Clear SF and DF flags
stf      R4, PD            // Write data
bdf      error_routine     // Check for immediate error from write to PD.
stf      r3, PDA|U|SZ32    // Change PDA
bdf      error_routine     // Check for:
                           //       1) data transfer error from stf PD
                           //       2) or Immediate error from stf PDA
```

40.16.2.9 Prefetch/Flush Management

There is no flush bit because every time data is stored in PD by a `stf PD` instruction—assuming PDA is correctly programmed—it is automatically flushed to the destination. An acknowledge is returned in the cycle of the DMA instruction, and the SDMA is only stalled by an instruction that addresses the peripheral DMA when the previous DMA access is not over.

40.16.3 BP DMA Unit

NOTE

The BP DMA is not connected for this device. The BP DMA and the BP control submodules can be ignored. However, the BP DMA registers can be accessed from the SDMA core, which may hang the SDMA. At which point, a software reset via the RESET[0] bit would be required. The BP DMA registers should not be accessed to avoid this situation.

The DMA instructions control the DMA state machine and may cause a DMA cycle on the associated memory bus. There are four registers associated with the BP (DSP) DMA unit which may be accessed on the functional unit bus:

- A source address register (DSA)
- A destination address register (DDA)
- A data buffer (DD)
- A state register (DS)

The BP DMA is not connected on this device, but its registers are still accessible. An attempt to initiate a BP DMA memory read or write might hang the SDMA since the access cannot be completed. The SDMA core must avoid accessing the DSA, DDA, DD, and DS registers to avoid situations that can cause the SDMA to hang.

40.16.4 CRC Unit

The Cyclic Redundancy Check (CRC) unit is connected to the SDMA core via the FUBUS. This unit can perform a CRC calculation for a set of given polynomials from degree 8 to 32.

When all CRC unit registers are loaded, the unit can process one byte of data every clock cycle. When loading new data to compute the CRC, the SDMA can perform 32-bit, 16-bit, or 8-bit accesses.

Two 32-bit registers comprise the unit:

- The CRC algorithm CA that describes the polynomial
- The CRC checksum CS to accumulate the data after each processing

40.16.4.1 Polynomial Register (CA)

This register defines the CRC algorithm currently used in the calculation, and the management of data ordering to/from the data register CS. Before starting any CRC calculation, it must be loaded with the chosen polynomial reference number and data ordering mode as indicated in Figure 40-66 and Table 40-82.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
R	0	0	0	0	0	0	0	0	0	0	0	ri	ro	p		
W																

Figure 40-66. CA Structure

Table 40-82. CA Descriptions

Field	Description
31–5	Reserved
4 ri	Reverse bit order of data in 0 Must be used when the peripheral receives bytes as bit streams in LSB-first order (UART case). 1 Selects the reverse mode, which must be used when the peripheral receives bytes as bit streams in MSB-first order (MMC case).

Table 40-82. CA Descriptions (continued)

Field	Description
3 ro	Reverse bit order of data out 0 Must be used when the peripheral transmits bytes as bit streams in LSB-first order (UART case). 1 Selects the reverse mode, which must be used when the peripheral transmits bytes as bit streams in MSB-first order (MMC case).
2–0 p	Polynomial selection 000 CRC32 ($X^{32}+X^{26}+X^{23}+X^{22}+X^{16}+X^{12}+X^{11}+X^{10}+X^8+X^7+X^5+X^4+X^2+X+1$) 001 CRC16 ($X^{16}+X^{15}+X^2+1$) 010 CCITT16 ($X^{16}+X^{12}+X^5+1$) 011 IS136 ($X^{12}+X^{10}+X^8+X^5+X^4+X^3+1$) 100 CRC10 ($X^{10}+X^9+X^5+X^4+X+1$) 101 CRC8 ($X^8+X^2+X+1$) 110 Parity ($X^8+1$) 111 Reserved

40.16.4.2 Accumulator Register (CS)

The accumulator register accumulates the division remainder during the CRC processing.

When writing the accumulator register (process bit is not set) if the *ro* bit is set, the write data has its order reversed whatever the data length (the length is specified by the selected polynomial).

When computing new CRC (writing CS and process bit set) if the *ri* bit is set in CA, any byte of write data will have its bit order reversed before storage and calculation. In the first byte, bit 7 is replaced by bit 0, bit 6 is replaced by bit 1, and so on. In the second byte, bit 7 is replaced by bit 0, bit 6 is replaced by bit 1, and so on, which means in the write data, bit 15 is replaced by bit 8, bit 14 will replaced by bit 9, and so on.

If the *ro* bit is set in CA, any read data will have its bit order reversed whatever the data length. If the valid data length is 16 bits, bit 15 is replaced by bit 0, bit 14 is replaced by bit 1, and so on.

When loading new data to compute the CRC, the SDMA can perform 32-bit, 16-bit, or 8-bit accesses. The CRC is computed in four clock cycles when performing a 32-bit access, in two clock cycles when performing a 16-bit access, and in one clock cycle when performing an 8-bit access. In all cases, an immediate acknowledge is sent back to the SDMA. A wait state is inserted if the SDMA tries to perform a new access before the end of the computation.

When performing a multi-bit access, the first byte used to compute the CRC is the most significant byte of the valid data.

40.16.4.3 Write Instruction (stf)

Table 40-83 shows the `stf` instructions.

Table 40-83. Stf Instructions for CRC

Byte Command								Description
1	0	0	0	0	0	0	0	Write polynomial encoded in the General Register (CA)
1	0	0	0	0	1	0	0	Write accumulator register (right-aligned)
1	0	0	1	0	1	0	0	Compute the CRC with the new incoming byte (b0)
1	0	0	1	0	1	0	1	Compute the CRC with the new incoming byte (b0)
1	0	0	1	0	1	1	0	Compute the CRC with the new incoming halfword (b1 b2) The bytes are used in the following order: b1 and b0.
1	0	0	1	0	1	1	1	Compute the CRC with the new incoming word (b3 b2 b1 b0) The bytes are used in the following order: b3, b2, b1, and b0.

40.16.4.4 Read Instruction (ldf)

Table 40-84 shows the `ldf` instructions.

Table 40-84. ldf Instructions for CRC

Byte Command								Description
1	0	0	0	0	0	0	0	read the polynomial register encoded
1	0	0	0	0	1	0	0	read the accumulator register, right aligned

40.16.4.5 Operating Mode

The following is the operating mode example:

- Load the polynomial register to select the algorithm and preset the accumulator, if required.
- Data is right-aligned in the general register.
- Input data is fed byte-by-byte into the accumulator through `stf` instructions.
- When all data is fed into the CRC unit, the CRC checksum is read with `ldf ca, sz`.

Example 40-2. 16-bit CCITT CRC with P(X) = $X^{16}+X^{12}+X^5+1$

```
.equ rL,0          // GReg[0] = rL; loop count register
.equ rA,1          // GReg[1] = rA; CCITT16 polynomial
.equ rP,2          // GReg[2] = rP; initial CRC value
.equ rT,3          // GReg[3] = rT; transferred byte register
.equ aT,4          // GReg[4] = aT; transferred byte address
                   // (MMC DAT_RX - $A003)
.equ CA,8          // CRC unit CA register
.equ CS,9          // CRC unit CS register
ldi rA,2           // init register with CCITT16 polynomial
stf rA,CA          // updates the selected polynomial
ldi rP,$ff         // initializes register with $000000ff
stf rP,CS,0        // presets the accumulator with $000000ff
ldi aT,$A0         // initializes aT with $A003: aT = $000000A0
revblo aT          // initializes aT with $A003: aT = $0000A000
ori aT,$03         // initializes aT with $A003: aT = $0000A003
ldi rL,200         // expects to read 200 bytes from MMC DAT_RX
loop 2             // executes 200 times the next 2 instructions
```

```
ld rT,(aT,0)      // loads next byte from MMC DAT_RX
stf rT,CS,1       // process checksum with new incoming byte
ldf rT,CS         // read the final checksum at the end
```

40.16.5 OnCE and Real-Time Debug

The On-Chip Emulation module (OnCE) is the debug interface to the SDMA. It supports the access to all core internal devices (registers, memory, and so on), and provides a set of mechanisms that control the core. The OnCE is accessed by JTAG ports at the chip's board level, or by the host via its peripheral bus.

To reduce the size of the hardware material involved, all tasks supported by the OnCE are performed on the SDMA core. The architecture of the SDMA OnCE is relatively simple and very flexible.

The commands supported by the SDMA OnCE are listed in the following sections.

40.16.5.1 Memory and Register Access

A set of mechanisms is provided to access SDMA memory and register locations. Both reading and writing are allowed. The access is supported if the processor is in debug mode.

Those registers can also be accessed through the AP Control interface when the OnCE is controlled by the AP, as described in Section 40.17.2.2, "Using the AP."

40.16.5.2 Hardware Breakpoints

An event detection unit is implemented to support memory breakpoints. The unit watches the data exchanged between the SDMA memory bus and the core. A debug request is sent to the core when matching conditions occur. The unit supports mixed conditions based on address range, access type, and data value. Event detection unit configuration registers are memory mapped in the SDMA space (see Section 40.12.3, "SDMA Core Register Descriptions"): You can modify them through a regular memory access or the AP control interface.

40.16.5.3 Watchpoints

One output pin is provided to monitor matching trigger conditions that are defined in the event detection unit.

40.16.5.4 Software Breakpoints

The SDMA instruction set contains a software breakpoint. Upon executing a software breakpoint instruction, the core suspends normal execution and enters debug mode. No hardware step execution mode is implemented in the OnCE module, but this feature may be implemented at the software level with this instruction.

40.16.5.5 Core Control

Commands are provided to monitor and control processor activity. You can halt the core, rerun the core from another address location, and get processor status. Any hardware breakpoint on the instruction bus is not supported, but this feature may be implemented by inserting a software breakpoints program.

40.17 The OnCE Controller

The OnCE controller receives commands from the AP or from the JTAG controller. Each command is interpreted before being sent to the core.

40.17.1 OnCE Commands

A small set of commands supports the communication between the OnCE module and the external world. This command set enables you to perform any of the following tasks: control processor activity, save core context, and execute an SDMA instruction from the OnCE module. Combined together, these tasks perform more complex commands.

A full OnCE command contains a 4-bit instruction (the OnCE command opcode) and a variable length data field (the OnCE data). During command execution, the OnCE data is transferred in a OnCE internal register before being exchanged with the SDMA. Some data values are also exported. This mechanism creates a link between the processor and the external world. Nine commands are defined: Table 40-85 presents their formats.

Table 40-85. OnCE Command Opcode Values

Instruction Opcode	Name	Action	Register	Data Field Size	Mode
0000	rstatus	Reads the OnCE status register	STATUS	16-bit	Normal/Debug
0001	dmov	Updates general register GReg1	GREG1	32-bit	Debug
0010	exec_once	Runs the instruction from the SDMA instruction register	INSTRUCTION	16-bit	Debug
0011	run_core	Returns to normal execution	BYPASS	1-bit	Debug
0100	exec_core	Returns to normal execution via a jump instruction that specifies the new address	INSTRUCTION	16-bit	Debug
0101	debug_rqst	Stops the core after execution of current instruction	BYPASS	1-bit	Normal
0110	rbuffer	Reads the real time buffer	RTB	32-bit	Normal/Debug
0111–1110	reserved	Reserved	BYPASS	1-bit	Normal/Debug
1111	bypass	Bypasses TAP controller	BYPASS	1-bit	Normal/Debug

Each instruction corresponds to a specific action performed on the OnCE module. The nature of the associated data field is clearly identified. The `dmov` command is followed by a 32-bit data value (which is a data value for the SDMA); the `exec_once` and the `exec_core` commands are followed by a 16-bit data value (which is an instruction for the SDMA); the `rstatus` command is followed by a 16-bit control value (which is the content of the OnCE status register); the `rbuffer` command is followed by a 32-bit data value.

The debug_rqst and the run_core commands are followed by a single bit data field (this is a bypass value). Finally, the bypass instruction enables the SDMA JTAG TAP controller to be daisy-chained with another JTAG TAP controller. This is a JTAG-only feature. The set of commands is simple, but enables you to perform any possible task on the SDMA during a debug process.

40.17.2 Sending Commands to the OnCE Controller

The JTAG access is the standard access to the OnCE, but sometimes the JTAG is not available to fix some bugs (if the chip is in production for instance), an additional access is then required. Therefore, an AP access to the OnCE is provided.

40.17.2.1 Using the JTAG Interface

A serial access is performed through the five JTAG pins TCK, TRST, TMS, TDI, and TDO. A Test Access Port controller is provided to decode the TMS control signal. It produces shift-enable signals (shift_ir and shift_dr), and updates enable signals (update_ir and update_dr). It is fully compliant with the IEEE 1149.1 testability (JTAG) standard.

During the shift_ir state, the command opcode is shifted into the OnCE controller (for example, the signal from the TDI pin is shifted into the command register and the TDO pin receives the signal shifted out). After transferring the four bits of the command, an update_ir signal is asserted and the command is decoded. The target data register is now clearly identified and the corresponding control signal is produced, as follows: bypass enable signal (bp_en), instruction enable signal (inst_en), data enable (data_en), and status enable signal (stat_en).

During the shift_dr state, the TDI signal is shifted into one of the following target registers: bypass register (1 bit), SDMA instruction register (16 bits), SDMA data register (32 bits), or OnCE status register (16 bits). The TDO pin is connected to the output of the selected register to receive the signals shifted out.

The JTAG access is disabled when the AP access is enabled.

40.17.2.2 Using the AP

The AP access to the OnCE is not the standard access, but it is required if the JTAG is not available. For example, if the SDMA ROM is out of use on a chip in production, and the AP needs to download new code and restart the SDMA, the OnCE can easily perform this operation. This type of debug operation justifies the use of an AP access to the OnCE.

To drive the OnCE, the AP uses some registers contained in the AP Control module of the SDMA. These registers are accessed through the AP peripheral bus. Most of these registers are connected to another register in the OnCE controller. Thus, accessing one of these registers is equivalent to accessing the associated register in the OnCE controller.

The set of registers in the AP Control module is listed below:

- ONCE_ENB register (1 bit, read/write)—This 1-bit register enables the AP access to the OnCE. When this bit is set, the signals from the JTAG are ignored. When it is cleared, all writing operations to the following registers through the Host Control interface are ignored. This register is reset on a JTAG reset.

- ONCE_CMD register (4 bits, read/write)—This 4-bit register receives the command opcode. It is connected to the command register in the controller. A write access to this register causes the associated command to be executed on the OnCE. For example, after writing “0001” in this register, a `dmov` command is executed.

  NOTE

  On the AP side, the `rstatus` and `bypass` commands are not supported. This register is reset on a JTAG reset.

- ONCE_DATA register (32 bits, read/write)—This 32-bit register is connected to the SDMA data register. This register is used when executing a `dmov` or `rbuffer` command.

  NOTE

  Before requesting a `dmov` command, the 32-bit data to transfer must be written in the ONCE_DATA register. At the end of the execution, the register is updated with GReg1 former value. This register is reset on a JTAG reset.

- ONCE_INSTR register (16 bits, read/write)—This 16-bit register is connected to the SDMA instruction register. This register is used when executing an `exec_core` or an `exec_once` command.

  NOTE

  Before requesting an `exec_core` or an `exec_once` command, the appropriate instruction must be written in the ONCE_INSTR register. This register is reset on a JTAG reset.

- ONCE_STAT register (16 bits, read only)—A read access to the ONCE_STAT register returns the content of the OnCE status register (OSTAT). This register is read only.
- The bypass register is not useful when the AP controls the OnCE, therefore no register is defined in the AP Control module to access the bypass register.

40.17.2.3 Conflicts Between the JTAG and the AP Accesses

When AP access to the SDMA OnCE is enabled (that is, when the bit in the ONCE_ENB register is set), the JTAG access is disabled. This guarantees that the module is not accessed at the same time on both sides.

It is possible to check whether the JTAG access to the SDMA OnCE is enabled from the JTAG port. When the JTAG access is disabled, the SDMA TDO always returns 1. The check requires the following steps:

- Execute a `dmov` command from debug mode (with neither 0xffffffff nor 0x0 as `dmov` value: 0x5a5a5a5a is good).
- Execute another `dmov` command (the value here is not important).

  The returned value from the latter `dmov` command should be the original one if the JTAG access is enabled; if it is 0xffffffff instead of the original input value, this means the JTAG access is disabled.

40.17.3 Executing a Command from the OnCE

All the commands defined in Section 40.17.1, "OnCE Commands" can be accessed through the JTAG. The AP can access all these commands except the rstatus command. On the AP side, the OnCE status is directly accessed by reading the ONCE_STAT register.

40.17.3.1 Nature of the Commands

Two types of commands may be distinguished. First, there are two commands that do not interact with the core: rstatus and rbuffer. Those commands may be requested at any time: They do not depend on the core status.

NOTE

Each of these commands exports a data value or a status value from the SDMA.

There are also commands that interact with the core: dmov, run_core, exec_core, exec_once, and debug_rqst. These commands are core status dependent, as follows:

- During user mode only the debug_rqst is taken into account.
- During debug mode, all these commands are taken into account except the debug_rqst. For example, an exec_once command requested while not in debug mode has no effect.

40.17.3.2 Execution Request

The SDMA starts executing a task in debug mode when requested by the OnCE controller. The execution starting time depends on the type of access used to communicate with the OnCE.

If the JTAG is used, the request is send after decoding the update_dr state in the TAP controller. Therefore, always cross this state when sending a command through the JTAG. If the OnCE is driven from the AP side, the request is sent after detecting a write access to the ONCE_CMD register. All the registers involved in this operation must be loaded first.

The following is an example of an exec_core command execution from the AP side: After writing '010' in the ONCE_CMD register, the OnCE controller asks the SDMA to execute the instruction contained in the ONCE_INSTR register. The instruction involved should be available in the ONCE_INSTR register before the beginning of the execution.

40.17.3.3 Command Execution

The following list shows the commands and details how each command is executed:

- rstatus command execution—The rstatus command exports the content of the OnCE status register (OSR). If the JTAG is used, the status information is captured in the OnCE status register during the capture_dr state, and shifted out after 16 TCK clock cycles in the shift_dr state. The rstatus command is not supported on the AP side, but a status register is provided instead. The rstatus may be performed in both debug and user modes.

- `dmov` command execution—The `dmov` command accesses SDMA internal registers. Executing a `dmov` instruction exchanges the 32-bit data values between the SDMA data register and the general register GReg[1].

  If the JTAG is used, the content of GReg1 is captured in the SDMA data register during the capture_dr state, then it is shifted out after 32 TCK clock cycles in the shift_dr state. During the update_dr state, GReg1 is updated with the new, shifted-in 32-bit data value. If the OnCE is driven from the AP side, the data values contained in GReg1 and the SDMA data register are exchanged after detecting a write access to the ONCE_CMD register. The ONCE_DATA register must therefore be loaded first.

- `exec_once` command execution—The `exec_once` command executes the instruction loaded in the SDMA instruction register. The command may only be requested from debug mode. The SDMA returns to debug mode at the end of the execution.

  Change of flow instructions as well as instructions that may cause a context switch are not supported: The comprehensive list comprises `done/yield/yiedge` (except `done` 5), BF, BT, BSF, BDF, JMP, JSR, JMPR, JSRR, RET, and LOOP, as well as all the illegal instructions.

  No other command should be requested before the SDMA returns to debug mode. The SDMA status (for example, whether it is in debug mode or not) can be detected by polling with the `rstatus` OnCE command, monitoring the debug_mode pin, or checking the Section 40.10.3.18, "OnCE Status Register (ONCE_STAT)" register via the AP control interface.

  NOTE

  Most of the instructions are single-cycle, which omits the step of polling the status. Loads and stores to DMA units are typical instructions that might require this polling.

  If the JTAG is used, the 16-bit instruction is shifted in the SDMA instruction register after 16 TCK clock cycles in the shift_dr state. A request is sent to the core when the update_dr state is decoded in the TAP controller. If the OnCE is driven from the AP side, the request is sent to the SDMA when detecting a write access to the ONCE_CMD register. The ONCE_INSTR register must be therefore be loaded first.

- `run_core` command execution—The `run_core` command leaves debug mode and resume normal program execution. The next instruction executed is the last instruction decoded before entering debug mode. Be sure to restore core context before re-running the core. This procedure is detailed in Section 40.18.5.3, "Restoring the Context."

  If the JTAG is used, a 1-bit bypass value is shifted in the bypass register in the shift_dr state. The SDMA is rerun when the update_dr state is decoded in the TAP controller. If the OnCE is driven from the AP side, the core is rerun when detecting a write access to the ONCE_CMD register.

- `exec_core` command execution—The `exec_core` command resumes program execution from any address. The 16-bit instruction provided with the `exec_core` overwrites the last instruction decoded before entering debug mode. This command is designed to support change of flow instructions, so that a program execution can be restarted from any address. After executing an `exec_core` command, the SDMA leaves debug mode. The `exec_core` command is usually used with a `jmp` instruction.

If the JTAG is used, the 16-bit branch instruction is shifted in the SDMA instruction register after 16 TCK clock cycles in the shift_dr state. The SDMA is rerun when the update_dr state is decoded in the TAP controller. If the OnCE is driven from the AP side, the SDMA reruns when detecting a write access to the ONCE_CMD register. The ONCE_INSTR register must therefore be loaded first. For example, to restart the SDMA from the program address 0x100, the instruction loaded should be a `jump to address 0x100` instruction.

- `debug_rqst` command execution—The `debug_rqst` command puts the SDMA in debug mode. If the JTAG is used, a 1-bit bypass value is shifted in the bypass register during the shift_dr state. A debug request is sent to the SDMA when the update_dr state is decoded in the TAP controller. If the OnCE is driven from the AP side, the debug request is sent when detecting a write access to the ONCE_CMD register. When the SDMA is already in debug mode, this command is simply ignored.
- `rbuffer` command execution—The `rbuffer` command exports the content of the real time buffer (RTB). If the JTAG is used, the content of the real time buffer (RTB) is captured in the SDMA data register during the capture_dr state. The register is completely shifted out after maintaining the shift_dr state during 32 TCK clock cycles. If the OnCE is driven from the AP side, the content of the RTB is captured in the ONCE_DATA register after detecting a write access to the ONCE_CMD register.
- `bypass` command execution—This command is only available from the JTAG interface. It enables daisy-chaining of the SDMA JTAG TAP controller with other JTAG TAP controllers. This command does not change the SDMA state and can be executed in any mode (run, debug, or sleep). It selects the bypass register of the TAP controller.

40.17.4 Registers Descriptions

See Section 40.12.3, “SDMA Core Register Descriptions” and Section 40.10, “AP Memory Map and Control Register Definitions” for detailed information on each register.

40.17.4.1 Event Cell Counter Register (ECOUNT)

The event cell counter register is a 16-bit register that contains the number of times minus one that an event detection occurs before generating a debug request. This register should be written before attempting to use the event detection counter during an event detection process. The event cell counter register is cleared on a JTAG reset.

40.17.4.2 Event Cell Address Registers (EAA or EAB)

The event cell contains two address registers—the event cell address register (a), called EAA, and the event cell address register (b), called EAB. Every address register is a 16-bit register that stores a user-defined address value. This value computes one of the following address conditions: addra_cond or addrb_cond. Every address register is cleared on a JTAG reset.

40.17.4.3 Event Cell Address Mask Register (EAM)

The event cell address mask register is a 16-bit register that contains a user-defined address mask value. This mask is applied to the address value latched from the memory address bus before comparing addresses.

NOTE

There is a common address mask value for the two address comparators. If bit *i* of this register is set, then bit *i* of the address value latched from the memory bus does not influence the result of the address comparison. The event cell address mask register is cleared on a JTAG reset.

40.17.4.4 Event Cell Data Register (ED)

The event cell data register is a 32-bit register that contains a user-defined data value. This data value is an input for the data comparator, which generates the data_cond condition. The event cell data register is cleared on a JTAG reset.

40.17.4.5 Event Cell Data Mask Register (EDM)

The event cell data mask register is a 32-bit register that contains a user-defined data mask value. This mask is applied to the data value latched from the memory bus before comparing data. Setting bit *i* of the event cell data mask register means that bit *i* of the data value latched from the address bus does not influence the result of the data comparison. The event cell data mask register is cleared on a JTAG reset.

40.17.4.6 Real Time Buffer Register (RTB)

The real Time Buffer register is a 32-bit register that stores and retrieves run-time information without putting the SDMA in debug mode. Refer to Section 40.18.8.2, “Real Time Buffer” for more details.

40.17.4.7 Event Control Register (ECTL)

The event cell control register is a 16-bit register that defines cell event occurrence conditions. The event cell control register is cleared on a JTAG reset. See also Section 40.18.6, “OnCE Event Detection Unit” for more details.

40.17.4.8 Trace Buffer (TB)

The Trace Buffer register retrieves the information in the Trace Buffer. See Section 40.18.8.1, “Trace Buffer” for more details.

40.17.4.9 OnCE Status Register (OSTAT)

The OnCE status register is a 16-bit register that contains processor and event detection unit status. The OSTAT is a read-only register. Refer to Section 40.10.3.18, “OnCE Status Register (ONCE_STAT)” for detailed description of the individual fields in the OSTAT register.

Figure 40-67 shows the OSTAT structure.

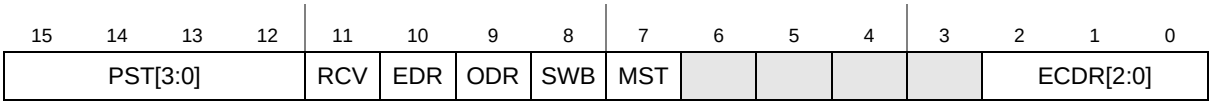


Figure 40-67. OnCE Status Register (OnCE)

Where PST[3:0] is the SDMA core state, RCV is set when the real-time buffer (RTB) is modified. EDR, ODR, and SWB are set, respectively, when the SDMA has entered debug mode because of an external debug request, a OnCE debug_rqst command, or a software breakpoint. MST is set when the OnCE is controlled from the AP control interface, and when ECDR is a three-flag set that shows the event cell condition(s) that put the core in debug mode. The OSTAT never provides more than one reason for entering debug mode.

There are two ways of accessing OSTAT content, as follows:

1. Send an `rstatus` command to the OnCE controller through the JTAG, or read the ONCE_STAT register through the AP access. Executing the `rstatus` command through the JTAG can be performed in both user and debug modes.
2. Perform an SDMA read access to the location in the SDMA core memory map (OSTAT register) debug mode using the `exec_once` command. With this method of access, the SDMA state reflected by the PST (processor status bit) is always DATA.

The register may also be accessed by a running application.

40.17.5 JTAG Interface Requirements

Because the signals received from the JTAG (running on TCK) are transferred to the OnCE controller (running on the SDMA clock), a synchronization mechanism is required.

40.17.5.1 TCK Speed Limitation

In the JTAG top-level layer, the TDO signal is always captured on a TCK falling edge. To guarantee a stable TDO signal from the SDMA during this operation, a falling edge detection is performed on TCK.

Before being latched in the *I* flip-flop (see Figure 40-68) on TCK falling edge, the TDO signal must be stable at the input of the flip-flop. This condition is verified if the TCK period is superior to the following delay:

worst-case edge detection delay + negative-edge signal propagation delay + JTAG top-level logic propagation delay

The frequency relationship, TCK < CLK/8, limitation guarantees that all operations are performed as expected.

40.17.5.2 Synchronization Implementation

Figure 40-68 shows the synchronization mechanism. Flip-flops tck0, tck1, and tck2 perform falling- and rising-edge detections on TCK. They generate the posedge_detected and negedge_detected nets that are used to sample the TDI and TMS inputs into the respective tdi and tms flip-flops, and update the tdo flip-flop to `yield` the TDO output. In the design, the only signal that might go metastable is the output of

the tck0 flip-flop. This signal is captured in the tck1 flip-flop and no logical operation is performed on it to minimize a metastability propagation risk.

The TDI and TMS flip-flops also cannot go metastable: The propagation time of the rising-edge detection signal through tck0, tck1, and tck2 guarantees that the TDI and TMS inputs are stable when captured in the TDI and TMS flip-flops.

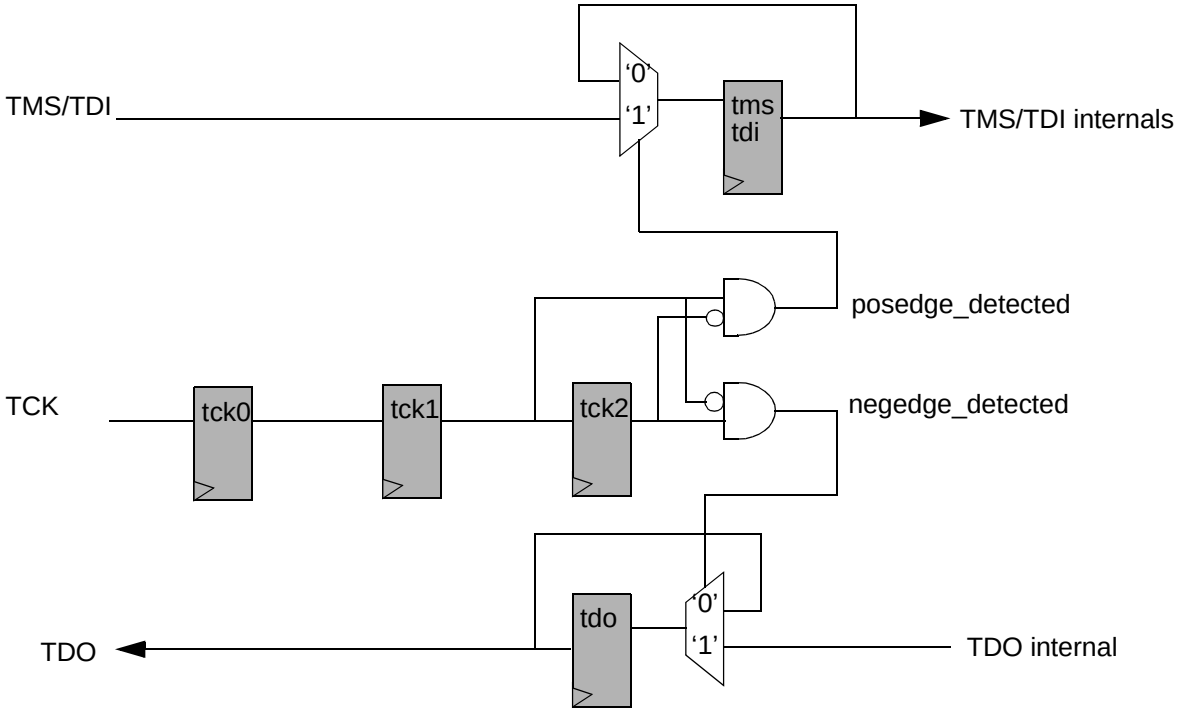


Figure 40-68. OnCE Synchronization Layer

Figure 40-69 shows synchronization timings. It takes three CLK clock cycles to synchronize TDI on the SDMA clock.

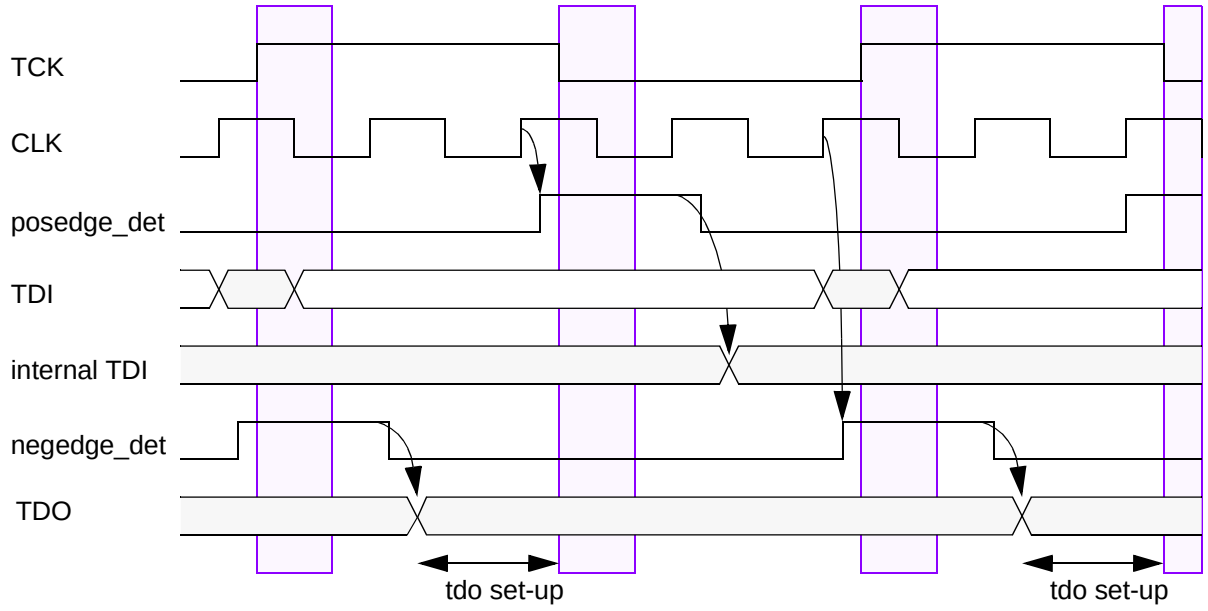


Figure 40-69. Synchronization Timings

40.17.5.3 JTAG Controller Start-Up Recommended Procedure

To ensure correct TAP controller initialization, it is recommended to use the following procedure:

1. Assert JTAG reset TRSTB (for example, set low).
2. Set TMS low.
3. Wait for 1 TCK clock.
4. Release JTAG reset TRSTB (for example, set high).
5. Wait for a minimum of five TCK cycles.

40.18 Using the OnCE

This section provides the elements necessary to run the OnCE module during a debug process. In addition to the basic set of commands described in Section 40.17.1, “OnCE Commands,” more complex commands can be built to meet users’ requirements.

40.18.1 Activating Clocks in Debug Mode

For power consumption issues, some clocks in the SDMA are disabled when not needed. This is the case for instance when the SDMA is in sleep mode. Clock gating management depends on the interface used to control the OnCE.

- For the JTAG access, the SDMA clock gating must be turned off via the clk_gating_off input.
- For the AP access, the SDMA clock gating is automatically turned off when the AP access is enabled (see Section 40.10.3.15, “OnCE Enable (ONCE_ENB)”).

40.18.2 Getting the Current Status

Most of the commands the OnCE supports have an impact on the status of the SDMA. It is not permissible to request the execution of an instruction on the SDMA from the OnCE while the SDMA is not in debug mode. Such a violation may cause unpredictable behavior, and it might be necessary to reset the SDMA. Therefore, the value of the PST bits provided in the OnCE status register should always be checked before sending any request to the SDMA.

40.18.3 Methods of Entering Debug Mode

A debug request may be asserted at any time, but it is not always taken into account immediately. Debug mode cannot be entered in the middle of an instruction, or during the save or restore states of a context switch. The request is ignored when the core is already in debug mode. Refer to Figure 40-4, which shows all possible transitions to the debug state, as there are several ways to enter debug mode.

40.18.3.1 External Debug Request During Reset

To enter debug mode after exiting reset, the external debug line has to be maintained high. This line is handled by the JTAG top-level block.

NOTE

The SDMA detects the debug requests only if the SDMA clock is running (see Section 40.18.1, “Activating Clocks in Debug Mode”). The debug request line should be not be maintained high when the SDMA is in debug mode.

The `debug_rqst` command (from the OnCE command set) is not supported during system reset.

40.18.3.2 Debug Request During Normal Activity

During normal activity, the SDMA enters debug mode when the following is true:

1. If the debug request line from the JTAG top-level is asserted, or
2. If the OnCE controller receives a debug_rqst command.

   The `debug_rqst` command can be sent by the JTAG access or by an access on the AP side (if the AP access is enabled).

40.18.3.3 Software Breakpoint Instruction

The SDMA enters debug mode at the end of the execution of a software breakpoint instruction. This instruction must be inserted in program flow executed by the core.

40.18.3.4 Event Detection Unit Matching Condition

If the event detection is enabled, a debug request is sent to the core after detecting a matching condition on the SDMA memory bus. See Section 40.18.6, “OnCE Event Detection Unit” for more details.

40.18.4 Executing Instructions in Debug Mode

The OnCE supports a mechanism to execute instructions in debug mode. If the SDMA is in debug mode, then the `exec_once` command can be used to execute an SDMA instruction from the OnCE controller. The SDMA returns to debug mode at the end of each execution. Some instructions are not supported by the `exec_once` command: `done/yield/yiedge` (except `done` 5), BF, BT, BSF, BDF, JMP, JSR, JMPR, JSRR, RET, and LOOP, as well as all the illegal instructions are not supported.

NOTE

While instructions are executed in debug mode from the OnCE, the program counter of the SDMA is not incremented.

40.18.5 Command Sequences Examples

This section provides examples of command sequences that run the SDMA in debug mode. These sequences are available for both the AP and JTAG accesses.

The following presents the syntax used in this section. The data field provided with each command is put in parenthesis with the command name. A '-' is used if the data field provided is a *Don't Care* value.

```
my_command(data_field);    // executing my_command with a data field
my_command(-);             // executing my_command with a don't care data field
```

The value returned by the command (if there is one) is referred by an assignment. In case the value returned by the command is not used, the assignment is omitted. For an AP access, the value returned (it is always a data value) is obtained by reading back into the SDMA data register.

```
data_out = my_command(data_in);// returning a data value
```

To clarify the syntax, the instructions' opcodes are referred to by their names. In practice, use the corresponding 16-bit encoding.

40.18.5.1 Getting the SDMA Status

NOTE

Before executing any command that affects the SDMA (like `dmov` or `exec_once`), check that the SDMA is in debug mode.

Use the following snippet:

```
rstatus(); // read SDMA status until the SDMA is in debug mode
...
rstatus();
```

If the SDMA is not in debug mode, then a debug request must be generated. In this case, the SDMA enters debug mode at the end of the execution of the current instruction. Use this snippet:

```
debug_rqst(-); // debug request
```

In the following sections, it is assumed that the SDMA was successfully put into debug mode.

40.18.5.2 Saving the Context

The first debug task is to save the SDMA context, which is the content of the eight general-purpose registers, the loop and PC-related registers, and the flags. Use the general register GReg[1] as an intermediate register to export the entire context of the SDMA.

Example 40-3 shows how to save GReg[0], GReg[1], GReg[2] and GReg[3]. The sequence of commands used to export additional general registers is very similar to this.

Example 40-3. Save GReg[0], GReg[1], GReg[2], and GReg[3]

```
GReg1_data = dmov(-);                // the value exported is the content of GReg[1]
exec_once("mov GReg1,GReg0");        // puts the content of GReg[0] into GReg[1]
GReg0_data = dmov(-);                // the value exported is the content of GReg[0]
exec_once("mov GReg1, GReg2");       // puts the content of GReg[2] into GReg[1]
GReg2_data = dmov(-);                // the value exported is the content of GReg[2]
exec_once("mov GReg1, GReg3");       // puts the content of GReg[3] into GReg[1]
GReg3_data = dmov(-);                // the value exported is the content of GReg[3]
```

Get the value of the internal flags (SF, DF, T, and LM), of the loop related registers (EPC and SPC), and of the PC-related registers (PC and RPC). Use a `done` 5, which is the formatting instruction dedicated to the debug. This instruction formats the flags and the values contained in the registers. It also writes the resulting values into the channel context memory. It should not be used when entering debug from the IDLE state (for example, with no active channel script running on the SDMA), because it will update a channel context that may belong to any channel.

```
exec_once("done 5");       // formatting the value of flags and registers
```

At this point, the channel context should be up-to-date in memory, and debug operations should now be possible. However, the context can be exported with the following instructions:

Example 40-4. Exporting the Context

```
dmov(ctx_base_addr);                 // loading GReg[1] with the channel context base address
exec_once("ld GReg0,(GReg1,0)");     // get RPC-PC into GReg0
exec_once("ld GReg1, (GReg1,1)");    // get SPC-EPC into GReg1
Loop_data = dmov(-);                 // read back the value of Loop registers
exec_once("mov GReg1, GReg0");       // puts the PC info into GReg1
PC_data = dmov(-);                   // reads back the content of the PC registers
```

After this sequence of operations, the entire SDMA context is exported via the OnCE.

40.18.5.3 Restoring the Context

At this point in the operation, restore the context of the SDMA. It can be different from the original context located in memory, and the content previously saved into the debugging application via the OnCE. Example 40-5 shows how it is possible to modify the current channel context:

Example 40-5. Modifying the Current Channel Context

```
dmov(Loop_data);                    // put Loop former value into GReg[1]
exec_once("mov GReg0, GReg1");      // copy to GReg[0]
dmov(PC_data);                      // put PC former value into GReg[1]
exec_once("mov GReg2, GReg1");      // copy to GReg[2]
dmov(ctx_base_addr);                // put channel context base address into GReg[1]
exec_once("st GReg0, (GReg1,1)");   // restore Loop context
exec_once("st GReg2, (GReg1,0)");   // restore PC context
```

Once the context in memory is the desired context (with or without applying the previous instruction sequence), it can be restored to the *real* PC and loop registers in the SDMA core:

```
exec_once("cpShReg");      // restore flags and PC & loop related registers
```

After this command, the SDMA core PC, RPC, SPC, EPC registers, as well as the flags contain the same data as what is stored in the context RAM for the current channel.

Example 40-6 shows how to restore the context of general registers GReg[0], GReg[1], GReg[2] and GReg[3].

Example 40-6. Restoring the General Register Context

```
dmov(GReg3_data);          // put GReg[3] restore value in GReg[1]
exec_once("mov GReg3, GReg1");// restore GReg[3]
dmov(GReg2_data);          // put GReg[2] restore value in GReg[1]
exec_once("mov GReg2, GReg1");// restore GReg[2]
dmov(GReg0_data);          // put GReg[0] restore value in GReg[1]
exec_once("mov GReg0, GReg1");// restore GReg[0]
dmov(GReg1_data);          // restore GReg[1]
```

At this point, it is possible to restart the normal program execution.

NOTE

Every SDMA core general register value can be modified by a `mov` instruction, which makes modification of these registers easy during debug. Unfortunately, there is no such instruction as a `mov` to directly modify the contents of either PCU register or flag (PC, RPC, SPC, EPC, T, LM, SF, or DF). The `cpShReg` instruction is meant to provide a means for changing these register contents via the context memory.

40.18.5.4 Accessing the Memory

In the following example, it is assumed that the SDMA context is entirely saved. If true, it is permissible to modify the general purpose registers during debugging activity.

To perform a memory read access, the target address is stored via the OnCE in GReg[1], then the load instruction is executed on the SDMA (the data loaded from the memory overwrites the address contained in GReg[1]), and then the result value is read back via the OnCE.

```
macro READ:       dmov(target_addr);                 // put the target address in GReg[1]
                  exec_once("ld GReg1,(GReg1,0)");   // execute the load instruction
                  res_data = dmov(-);                // exports the result data value
```

For a memory write access, the target address is written in GReg[0], and the value to store is written in GReg[1]. Then the store instruction is executed on the SDMA.

```
macro WRITE:      dmov(target_addr);                 // puts the target address in GReg[1]
                  exec_once("mov GReg0,GReg1");      // puts the target address in GReg[0]
                  dmov(target_data);                 // puts the target data in GReg[1]
                  exec_once("st GReg1,(GReg0,0)");   // performs the store operation
```

This sequence is shown as an example; however, many other sequences are possible.

NOTE

This sequence of commands can also be applied to memory-mapped registers.

40.18.5.5 Resuming Program Execution

Before resuming program execution, it is assumed that the SDMA context is properly restored. There are two ways to restart the SDMA. Start by executing the last instruction fetched before entering debug mode, as follows:

```
run_core(-);              // resume execution from where we stopped before
```

If necessary, restart the execution from a different address. In this case, use the `exec_core` command. The data field provided with this command must be the encoding of a `jump` instruction.

```
exec_core("jmp start_addr");// rerun the SDMA from another address
```

In these two examples, the SDMA exits debug mode and keeps executing the code fetched from the memory.

40.18.5.6 Single Stepping in RAM

To execute a program step-by-step from the RAM, insert software breakpoints in the program flow at appropriate places so that the SDMA only executes one instruction before returning to debug mode.

First, read the next instruction to execute in the RAM. Then, depending on the value of this instruction, compute the address where a software breakpoint instruction should be inserted. The instruction at the corresponding address must be saved, and, the software breakpoint instruction is inserted. After restarting the SDMA, there is only one instruction executed before meeting the software breakpoint.

Example 40-7 shows the macro functions READ and WRITE, which correspond to the sequence of commands (described above) used to access the memory.

NOTE

The data read from the memory are 32-bit values, while the instructions are 16-bit values only. This is why it is best to only use addresses divided by two when accessing the memory.

Example 40-7. READ and WRITE Macro Functions

```
next_instr = READ(run_addr/2);      // read the next instruction to execute
// the tool now has to compute the address where the breakpoint
// instruction should be inserted, this address is the "bkpt_addr"
instr_save = READ(bkpt_addr/2);     // save the instruction before overwriting
STORE("bkpt instruction",bkpt_addr/2);// store the bkpt instruction in memory
exec_core("jmp run_addr");          // rerun the SDMA
rstatus(-);                         // wait for the SDMA to enter debug mode
...
rstatus(-);
STORE(instr_save,bpkt_addr/2);      // restore the instruction overwritten
```

In case of branched conditional instructions, a breakpoint instruction should be written at the two possible target addresses.

40.18.5.7 Single Stepping in ROM

No single-step mechanism is supported in ROM. The program code can be loaded in the RAM, where the single-step mechanism can be executed.

40.18.6 OnCE Event Detection Unit

The event detection unit watches signals from the data memory bus (DMBUS), which the SDMA core uses to access its RAM, ROM, and memory mapped registers. A debug request is sent to the OnCE controller when user-defined conditions on address and/or data values are true. See Figure 40-70.

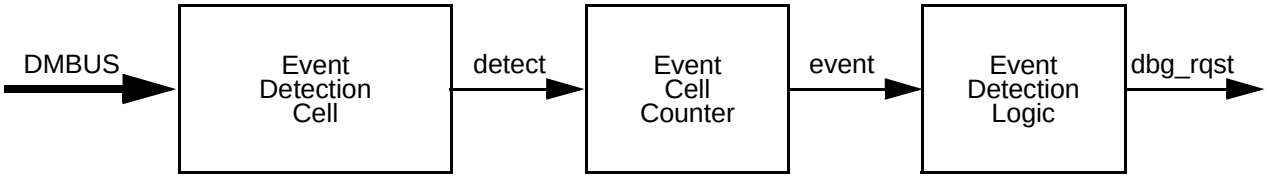


Figure 40-70. Event Detection Unit

A counter, provided with the detection cell, is decreased after an event detection. A debug request is sent to the core only when the counter reaches the value of 0. It is possible to disable the use of the counter if a debug request has to be generated after each event detection.

The event cell is the basic module that supports hardware breakpoints on an address value and/or data values coming from the SDMA memory bus. The trigger condition that generates the debug request is a mixed condition based on those values.

Figure 40-71 shows the event cell architecture. The event cell contains the address (stored in the memory address register) and the data (stored in the memory data register) used during the last memory access. There are some user-defined reference values located in memory mapped registers—the event cell addresses, the event cell address mask, the event cell data, and the event cell data mask. These registers are accessed by standard load/store instructions just like regular memory locations.

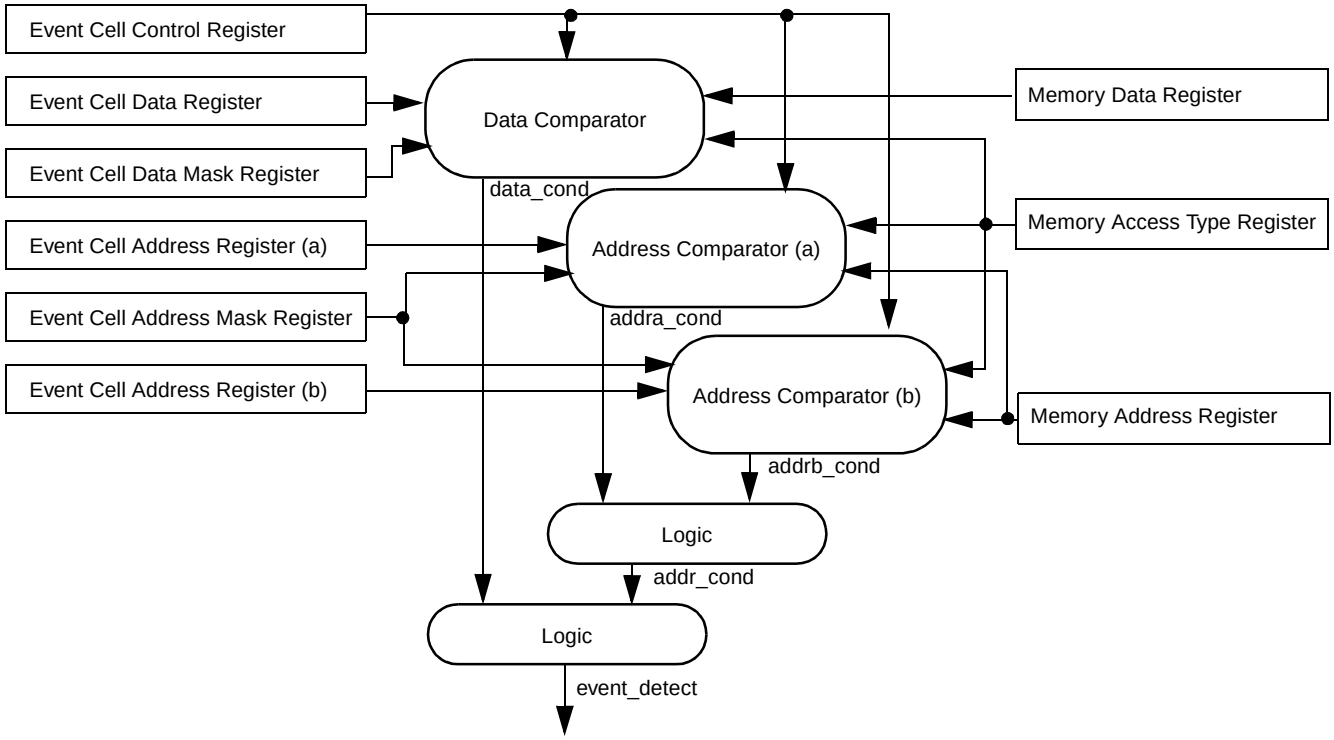


Figure 40-71. Event Cell Architecture

To define a memory breakpoint, three conditions are taken into account: The first two conditions are comparisons of the current memory address with user-defined reference addresses (these conditions are called addressA and addressB). The third condition consists of a comparison between the data received on the DMBUS and a user-defined reference data (this condition is called data). An intermediate address condition is set to express a dependency between addressA and addressB conditions.

40.18.7 Clock Gating and Reset

This section details how to use the clocks and handle the reset signals.

40.18.7.1 Clocks

Because the SDMA uses clock gating to save power, it is necessary to disable the clock gating and force the clocks to be enabled when using the OnCE. When the OnCE is accessed through its JTAG interface, clock gating must be disabled outside the SDMA via a dedicated SDMA input port clk_gating_off. The reason why detection is not performed automatically by the SDMA internal hardware is that it would cost power to monitor activity on the JTAG interface.

This clk_gating_off input is controlled by a control bit in the System JTAG Controller. Refer to the System JTAG Controller chapter for more information.

When the OnCE is accessed through the AP Control interface, clock gating is automatically turned off. This is done when bit 0 of the ONCE_ENB register (see Section 40.10.3.15, “OnCE Enable (ONCE_ENB)”) is set. A write access to this register is possible even when the OnCE clock is not running. If the AP access is used, the bit in the ONCE_ENB register must be set before any attempt to access any other OnCE register.

40.18.7.2 Resets

The OnCE reset is different from the SDMA main reset. The OnCE reset is connected to the TRST_B pin.

Normally, activating the SDMA reset while keeping the OnCE reset inactive (when possible) enables you to reset the core without having to reprogram the OnCE.

40.18.8 Real Time Features

To rebuild the skeleton of a program execution, it is necessary to store the addresses of the program instructions where jumps are taken: A trace buffer is therefore provided. A real time buffer has also been added to receive data values written during a program execution. The content of this register may be exported through JTAG ports without stopping the core.

40.18.8.1 Trace Buffer

The Trace Buffer is a 32-stage buffer that contains appropriate information to identify the 32 last changes of flow detected during a program execution. Figure 40-72 shows an overview of the Trace Buffer.

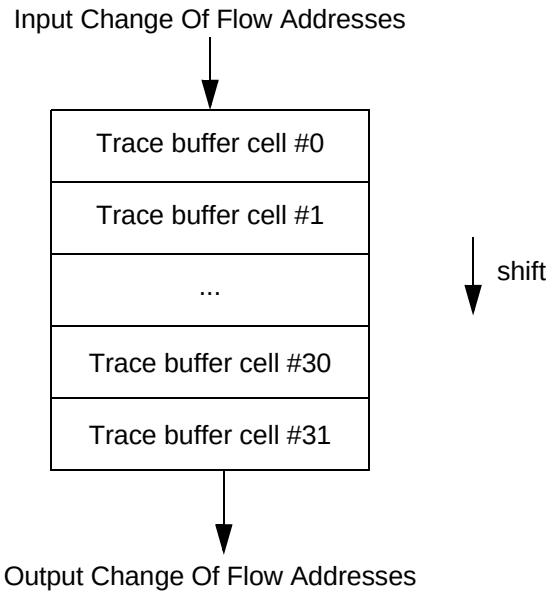


Figure 40-72. Trace Buffer

Each cell of the trace buffer contains two reference addresses and a flag. The flag is set when the addresses stored in the cell correspond to a valid change of flow; otherwise, the flag is cleared. The three most significant bits are unused.

After every change of flow detection, the address of current instruction and the address of the target instruction are stored at the top of the Trace Buffer (cell #0). The flag in the cell is set to indicate that a valid change of flow was detected. Former cell values are shifted one level down. The Trace Buffer contains the 32 last changes of flow. All the flags are reset on a software or a hardware reset, and after each transition from debug mode to user mode.

A memory mapped register, the Trace Buffer register (TB), is provided to read the content of the Trace Buffer. This operation should be done in debug mode. Performing a read access to the Trace Buffer register returns the content of the bottom of the Trace Buffer (cell #31). After every read access, the trace buffer is shifted one level down, and the flag at the top of the trace buffer is cleared.

A typical OnCE command sequence that retrieves the oldest change-of-flow information is a follows:

```
exec_once("mov r1, TB");          // stores the oldest change-of-flow in GReg1
dmov(-);                          // retrieves GReg1 contents
```

This sequence requires the SDMA to be put in debug mode.

40.18.8.2 Real Time Buffer

The Real Time Buffer register (RTB) is a memory mapped register that can be accessed as a regular memory location by the SDMA core during program execution. This register is located in the OnCE. Executing ar `rbuffer` command (see Section 40.17, "The OnCE Controller" for further details) exports the content of this register through JTAG ports.

When a write access is performed at the memory location corresponding to the RTB, the receive flag (for example, the RCV bit) is set in the OnCE Status Register (OSR). This flag is cleared at the end of the execution of a `rbuffer` command.

NOTE

Every write access to the RTB memory location updates the RTB register even if the RCV flag is set. The RTB is cleared on a JTAG reset.

40.18.8.3 Emulation Pin

The debug_matched_event emulation pin reflects the matching condition status detected by the Event Detection Unit. Since it can be necessary to detect conditions without triggering debug requests, it is possible to disable the generation of debug requests by the Event Detection Unit and still have the matching condition available on the emulation pin. This can be done by clearing the EN flag in the ECTL register.

40.18.8.4 Real-Time Debug Outputs

Table 40-86 shows the debug signals that are available at the SDMA boundaries. Their availability at chip boundaries depends on the project.

Table 40-86. Real-Time Debug Output Pins

Pin	Description
debug_core_state[4:0]	The core_state bits reflect the state of the SDMA core. • The “Program” state is the usual instruction execution cycle. • The “Data” state is inserted when there are wait-states during a load or a store on the data bus (ld or st). • The “Change of Flow” state is the second cycle of any instruction that breaks the sequence of instructions (jumps and channel switching instructions). • The “Change of Flow in Loop” state is used when an error causes a hardware loop exit. • The “Debug” state means the SDMA is in debug mode. • The “Functional Unit” state is inserted when there are wait-states during a load or a store on the functional units bus (ldf or stf). • In “Sleep” modes, no script is running (this is the core idle state); the “after Reset” is slightly different because no context restoring phase will happen when a channel is triggered: The script located at address 0 is executed (boot operation). • The “in Sleep” states are the same as above except they do not have any corresponding channel: they are used when entering debug mode after reset; the reason is that it is necessary to return to the “Sleep after Reset” state when leaving debug mode. 0 Program 1 Data 2 Change of Flow 3 Change of Flow in Loop 4 Debug 5 Functional Unit 6 Sleep 7 Context Switch Saving Channel 8 Program in Sleep 9 Data in Sleep 10 Change of Flow in Sleep 11 Change of Flow in Loop in Sleep 12 Debug in Sleep 13 Functional Unit in Sleep 14 Sleep after Reset 15 Context Switch Restoring Channel
debug_yield	Pulse that is active when a `yield` (`done` 0) or a `yieldge` (`done` 1) instruction is executed. 0 — 1 `yield`/`yieldge` executed
debug_core_run	Active when the SDMA core is executing instructions. 0 Debug or sleep mode 1 Run mode
debug_event_channel[5:0]	Gives the number of any DMA request as soon as it is received or the number of the current channel. When bit 5 value is 0 and the core is in run mode, bits 4-0 give the number of the current channel. When bit 5 value is 1, bits 4-0 give the number of the last received request. A DMA request is visible during one single cycle only. If several DMA requests are simultaneously received, they are made visible during successive cycles. 0CCCC: CCCC gives the number of the current channel 1EEEE: EEEE gives the number of the last received event
debug_pc[13:0]	Program Counter value; it has a meaning when the core is in run mode.

Table 40-86. Real-Time Debug Output Pins (continued)

Pin	Description
debug_mode	Set when the core is in debug. 0 — 1 Core is in debug
debug_bus_error	Set when an error was received during a load or a store (`ld`, `st`, `ldf`, or `stf` instruction) and registered in SF or DF flag. 0 No error during last load/store 1 Error during last load/store
debug_bus_device[4:0]	Indicates the device or functional unit that is accessed by the current instruction. The debug_bus_device output is always valid when in sleep mode, debug mode, or executing any instruction that does not access the functional units or the memory mapped devices, "no access" is output. 0 No access 1 MSA 2 MDA 3 MD 4 MS 5 PSA 6 PDA 7 PD 8 PS 9 DSA 10 DDA 11 DD 12 DS 13 CA 14 CS 15 Reserved 16 Memory (RAM or ROM) 17 Memory mapped register 18 Peripheral #1 19 Peripheral #2 20 Peripheral #3 21 Peripheral #4 22 Peripheral #5 23 Peripheral #6 24 Peripheral #7 25 Peripheral #8 26 Peripheral #9 27 Peripheral #10 28 Peripheral #11 29 Peripheral #12 30 Peripheral #13 31 Peripheral #14
debug_bus_rwb	Indicates the direction of the access given by debug_bus_device 0 Write access (st or stf) 1 Read access (ld or ldf)
debug_matched_dmbus	Pulse indicating the OnCE event detection unit has detected a match on the data bus during an access to memory (RAM or ROM), a memory mapped register or a peripheral that is hooked to the SDMA. 0 — 1 data bus match detected

Table 40-86. Real-Time Debug Output Pins (continued)

Pin	Description
debug_rtbuffer_write	Pulse indicating when the real-time buffer is written by the core. 0 — 1 RTB was modified
debug_evt_chn_lines[7:0]	Eight lines that generate short pulses when DMA requests are received or channels are (re)started. Every line is controlled through two parameters defined in registers Cross-Trigger Events Configuration Register (1) and (2) (XTRIG_CONF1 and XTRIG_CONF2) (as described in Section 40.10, "AP Memory Map and Control Register Definitions"). The following two parameters are available for every line: • CNF—Indicates what is monitored on the line: 0 for a channel start, 1 for a DMA request reception • NUM[4:0]—Gives the number of the DMA request or channel to monitor

The matched_event emulation pin reflects the matching condition status detected by the Event Detection Unit. Because it can be necessary to detect conditions without triggering debug requests, it is possible to disable the generation of debug requests by the Event Detection Unit and still have the matching condition available on the emulation pin. This can be done by clearing the EN flag in the ECTL register.

All real-time debug outputs are disabled by default (for example, they are stuck to 0) to avoid power consumption when they are not used. They are enabled when bit 11 (RTDOBS) of the Configuration Register (CONFIG) is set.

40.19 Instruction Set

40.19.1 Instruction Encoding

This section presents a short summary of the instruction codes. All context switch instructions are listed for information only; they cannot function properly out of the context switch routine.

```
x...x             - don't care
rrr               - destination/source general register
sss               - additional source general register
bbb               - general register used as address base register
ddddd             - address displacement
nnnnn             - bit number
uuuuuuuu          - function unit command bits
pppppppp          - branch displacement (signed)
iiiiiiii          - 8-bit immediate
jjj               - control bit to clear
ff                - flag to clear

00000jjj00000000  - done (done,yield,wait)
00000jjj00000001  - notify
00000xxx00000010  - reserved
00000xxx00000011  - reserved
00000xxx00000100  - reserved
00000xxx00000101  - reserved
0000000000000101  - softBkpt
0000000100000101  - reserved
0000001000000101  - reserved
```

```
0000001100000101  - reserved
0000010000000101  - reserved
0000010100000101  - reserved
0000011000000101  - reserved
0000011100000101  - reserved
0000000000000110  - ret
0000000100000110  - reserved
0000001000000110  - reserved
0000001100000110  - reserved
0000010000000110  - reserved
0000010100000110  - reserved
0000011000000110  - reserved
0000011100000110  - reserved
000000ff00000111  - clrf ff
0000010000000111  - reserved
0000010100000111  - reserved
0000011000000111  - reserved
0000011100000111  - illegal
00000rrr00001000  - jmpr r
00000rrr00001001  - jsr
00000rrr00001010  - ldrpc r
00000rrr00001011  - reserved
00000rrr000011xx  - reserved
00000rrr00010000  - revb
00000rrr00010001  - revblo
00000rrr00010010  - rorb
00000rrr00010011  - reserved
00000rrr00010100  - ror1
00000rrr00010101  - lsr1
00000rrr00010110  - asr1
00000rrr00010111  - lsl1
00000rrr001nnnnn  - bclri r,n
00000rrr010nnnnn  - bseti r,n
00000rrr011nnnnn  - btsti r,n
00000xxx10000xxx  - reserved
00000rrr10001sss  - mov
00000rrr10010sss  - xor
00000rrr10011sss  - add
00000rrr10100sss  - sub
00000rrr10101sss  - or
00000rrr10110sss  - andn
00000rrr10111sss  - and
00000rrr11000sss  - tst
00000rrr11001sss  - cmpeq
00000rrr11010sss  - cmplt
00000rrr11011sss  - cmphs
0000011011100000  - reserved
0000011011100001  - reserved
0000011011100010  - cpShReg
0000011011100011  - reserved
0000011011100100  - reserved
0000011011100101  - reserved
0000011011100110  - reserved
0000011011100111  - reserved
00000xxx11101xxx  - reserved
00000xxx11110xxx  - reserved
00000xxx11111xxx  - reserved
```

```
00001rrriiiiiiiii  - ldi r,i
00010rrriiiiiiiii  - xori r,i
00011rrriiiiiiiii  - addi r,i
00100rrriiiiiiiii  - subi r,i
00101rrriiiiiiiii  - ori r,i
00110rrriiiiiiiii  - andni r,i
00111rrriiiiiiiii  - andi r,i
01000rrriiiiiiiii  - tsti r,i
01001rrriiiiiiiii  - cmpeqi r,i
01010rrrddddddbbb  - ld r,(d,b)
01011rrrddddddbbb  - st r,u
01100rrruuuuuuuuu  - ldf r,u
01101rrruuuuuuuuu  - stf r,u
011100ffnnnnnnnnn  - Loop ff flags are reseted
011101xxxxxxxxxx  - reserved
011110xxxxxxxxxx  - reserved
01111100pppppppp  - bf pc=pc+signed(pppppppp)+1
01111101pppppppp  - bt pc=pc+signed(pppppppp)+1
01111110pppppppp  - bsf pc=pc+signed(pppppppp)+1
01111111pppppppp  - bdf pc=pc+signed(pppppppp)+1
10aaaaaaaaaaaaaa  - jmp absolute
11aaaaaaaaaaaaaa  - jsr absolute
```

40.19.2 SDMA Instruction Set

This section describes all the useful instructions from the SDMA set. Table 40-87 summarizes the instructions.

Table 40-87. SDMA Instruction List

Instruction	Description	Page
ADD	Addition	on page 40-154
ADDI	Add with Immediate Value	on page 40-155
AND	Logical AND	on page 40-156
ANDI	Logical AND with Immediate Value	on page 40-157
ANDN	Logical AND NOT	on page 40-158
ANDNI	Logical AND with Negated Immediate Value	on page 40-159
ASR1	Arithmetic Shift Right by 1 Bit	on page 40-160
BCLRI	Bit Clear Immediate	on page 40-161
BDF	Conditional Branch if Destination Fault	on page 40-162
BF	Conditional Branch if False	on page 40-163
BSETI	Bit Set Immediate	on page 40-164
BSF	Conditional Branch if Source Fault	on page 40-165
BT	Conditional Branch if True	on page 40-166
BTSTI	Bit Test immediate	on page 40-167
CLRF	Clear CPU flags	on page 40-168

Table 40-87. SDMA Instruction List (continued)

Instruction	Description	Page
CMPEQ	Compare for Equal	on page 40-169
CMPEQI	Compare with Immediate for Equal	on page 40-170
CMPHS	Compare for Higher or Same	on page 40-171
CMPLT	Compare for Less Than	on page 40-172
cpShReg	Update Context of PCU Registers and Flags	on page 40-173
DONE	DONE, Yield	on page 40-174
ILLEGAL	ILLEGAL Instruction	on page 40-176
JMP	Unconditional Jump Immediate	on page 40-177
JMPR	Unconditional Jump	on page 40-178
JSR	Unconditional Jump to Subroutine Immediate	on page 40-179
JSRR	Unconditional Jump to Subroutine	on page 40-180
LD	Load Register	on page 40-181
LDF	Load Register from Functional Unit	on page 40-183
LDI	Load Register with Immediate Value	on page 40-185
LDRPC	Load from RPC to Register	on page 40-186
LOOP	Hardware Loop	on page 40-187
LSL1	Logical Shift Left by 1 Bit	on page 40-190
LSR1	Logical Shift Right by 1 Bit	on page 40-191
MOV	Logical Move	on page 40-192
NOTIFY	Notify to MCU	on page 40-193
OR	Logical OR	on page 40-194
ORI	Logical OR with Immediate Value	on page 40-195
RET	Return from Subroutine	on page 40-196
REVB	Reverse Byte Order	on page 40-197
REVBLO	Reverse Low Order Bytes	on page 40-198
ROR1	Rotate Right by 1 Bit	on page 40-199
RORB	Rotate Right by 1 Byte	on page 40-200
SOFTBKPT	Software Breakpoint	on page 40-201
ST	Store Register	on page 40-202
STF	Store Register in Functional Unit	on page 40-203
SUB	Subtract	on page 40-206
SUBI	Subtract with Immediate	on page 40-207
TST	Test with Zero	on page 40-208

Table 40-87. SDMA Instruction List (continued)

Instruction	Description	Page
TSTI	Test Immediate	on page 40-209
XOR	Logical Exclusive OR	on page 40-210
XORI	Exclusive OR with Immediate	on page 40-211

ADD

Addition

Operation:

```
GReg[r] ← GReg[s] + GReg[r]
T ← (GReg[r] == 0)
```

Assembler:

Syntax: `add r,s`

Example: `add 0,3`
to ADD GReg[3] and GReg[0] and store the result in GReg[0]

CPU Flags: T

Cycles: 1

Description: Performs the ADDition of the source general register *s* and the destination general register *r*, and stores the result in the destination general register *r*. The T flag is set if the result of the operation is 0. It is cleared if the result is not 0.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	0	0	1	1	s	s	s

Instruction Fields:

`rrr` - destination register field:

```
000 - GReg[0]
001 - GReg[1]
010 - GReg[2]
011 - GReg[3]
100 - GReg[4]
101 - GReg[5]
110 - GReg[6]
111 - GReg[7]
```

`sss` - source register field:

```
000 - GReg[0]
001 - GReg[1]
010 - GReg[2]
011 - GReg[3]
100 - GReg[4]
101 - GReg[5]
110 - GReg[6]
111 - GReg[7]
```

ADDI

Add with Immediate Value

Operation:

```
GReg[r] ← GReg[r] + immediate
T ← (GReg[r] == 0)
```

Assembler:

Syntax: `addi r,immediate`

Example: `addi 6,112`
to ADD GReg[6] and decimal value 112 and store the result in GReg[6]

CPU Flags: T

Cycles: 1

Description: Add a 0-extended immediate value to a general register; stores the result in the general register. The flag T is set when the result of the operation is 0; otherwise, it is cleared. The immediate value is the low-order byte of the instruction.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	r	r	r	i	i	i	i	i	i	i	i

Instruction Fields:

`rrr` - register field:
`000` - `GReg[0]`
`001` - `GReg[1]`
`010` - `GReg[2]`
`011` - `GReg[3]`
`100` - `GReg[4]`
`101` - `GReg[5]`
`110` - `GReg[6]`
`111` - `GReg[7]`

`iiiiiiii` - immediate value:
`00000000` - 0
`00000001` - 1
...
`11111110` - 254
`11111111` - 255

AND

Logical AND

Operation:

GReg[r] ← GReg[s] & GReg[r]

Assembler:

Syntax: and r,s

Example: and 1,2
to AND GReg[1] and GReg[2] and store the result in GReg[1]

CPU Flags: Unaffected

Cycles: 1

Description: Performs the AND of the source general register *s* and the destination general register *r*, and stores the result in the destination general register *r*.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	0	1	1	1	s	s	s

Instruction Fields:

rrr - destination register field:
000 - GReg[0]
001 - GReg[1]
010 - GReg[2]
011 - GReg[3]
100 - GReg[4]
101 - GReg[5]
110 - GReg[6]
111 - GReg[7]

sss - source register field:
000 - GReg[0]
001 - GReg[1]
010 - GReg[2]
011 - GReg[3]
100 - GReg[4]
101 - GReg[5]
110 - GReg[6]
111 - GReg[7]

ANDI

Logical AND with Immediate Value

Operation:

GReg[r] ← GReg[r] & immediate

Assembler:

Syntax: andi r,immediate

Example: andi 7,45
to AND GReg[7] and decimal value 45 and store the result in GReg[7]

CPU Flags: unaffected

Cycles: 1

Description: Performs an AND between a 0-extended immediate value and a general register; stores the result in the general register. The immediate value is the low-order byte of the instruction.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	1	r	r	r	i	i	i	i	i	i	i	i

Instruction Fields:

rrr - register field:
000 - GReg[0]
001 - GReg[1]
010 - GReg[2]
011 - GReg[3]
100 - GReg[4]
101 - GReg[5]
110 - GReg[6]
111 - GReg[7]

iiiiiiii - immediate value:
00000000 - 0
00000001 - 1
...
11111110 - 254
11111111 - 255

ANDN

Logical AND NOT

Operation:

GReg[r] ← ~GReg[s] & GReg[r]

Assembler:

Syntax: andn r,s

Example: andn 3,4
to AND GReg[3] and NOT GReg[4] (bit inverted) and store the result in GReg[3]

CPU Flags: Unaffected

Cycles: 1

Description: Performs the AND of the negation of the source general register *s* and the destination general register *r*, and stores the result in the destination general register *r*.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	0	1	1	0	s	s	s

Instruction Fields:

rrr - destination register field:
000 - GReg[0]
001 - GReg[1]
010 - GReg[2]
011 - GReg[3]
100 - GReg[4]
101 - GReg[5]
110 - GReg[6]
111 - GReg[7]

sss - source register field:
000 - GReg[0]
001 - GReg[1]
010 - GReg[2]
011 - GReg[3]
100 - GReg[4]
101 - GReg[5]
110 - GReg[6]
111 - GReg[7]

ANDNI Logical AND with Negated Immediate Value

Operation:

GReg[r] ← GReg[r] & ~immediate

Assembler:

Syntax: andni r,immediate

Example: andni 0,2
to AND GReg[0] and decimal value -3 (inverted 32-bit value 2) and store the result in GReg[0]

CPU Flags: unaffected

Cycles: 1

Description: Performs an AND between the negation of a 0-extended immediate value and a general register; stores the result in the general register. The immediate value is the low-order byte of the instruction.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	r	r	r	i	i	i	i	i	i	i	i

Instruction Fields:

rrr - register field:
000 - GReg[0]
001 - GReg[1]
010 - GReg[2]
011 - GReg[3]
100 - GReg[4]
101 - GReg[5]
110 - GReg[6]
111 - GReg[7]

iiiiiiii - immediate value:
00000000 - 0
00000001 - 1
...
11111110 - 254
11111111 - 255

ASR1 Arithmetic Shift Right by 1 Bit

Operation:

GReg[r]:{b31,b30,...,b1,b0} ← GReg[r]:{b31,b31,b30,...,b1}

Assembler:

Syntax: asr1 r

Example: asr1 3
to divide by 2 the signed value of GReg[3] and store the result in GReg[3]

CPU Flags: Unaffected

Cycles: 1

Description: Shift the bits of any general register to the right and keep the same sign: The left bit (bit 31) is kept untouched.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	1	0	1	1	0

Instruction Fields:

rrr - register field:
000 - GReg[0]
001 - GReg[1]
010 - GReg[2]
011 - GReg[3]
100 - GReg[4]
101 - GReg[5]
110 - GReg[6]
111 - GReg[7]

BCLRI

Bit Clear Immediate

Operation:

GReg[r]:{b31,...,b(i+1),0,b(i-1),...,b0} ←
GReg[r]:{b31,...,b(i+1),b(i),b(i-1),...,b0}

Assembler:

Syntax: bclri r,i

Example: bclri 1,12
to clear bit 12 in GReg[1]

CPU Flags: Unaffected

Cycles: 1

Description: Clear the bit of register r specified by the immediate field

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	1	i	i	i	i	i

Instruction Fields:

rrr - register field:
000 - GReg[0]
001 - GReg[1]
010 - GReg[2]
011 - GReg[3]
100 - GReg[4]
101 - GReg[5]
110 - GReg[6]
111 - GReg[7]

BDF

Conditional Branch if Destination Fault

Operation:

```
if (DF == 1) PC ← PC + 1 + displacement else PC ← PC + 1
```

Assembler:

Syntax: `bdf label`

Example: bdf LLL
To jump to LLL if DF is set, or go to the next instruction if DF is cleared; the displacement value is calculated by the assembler.

CPU Flags: Unaffected

Cycles: 2 when the branch is done, 1 otherwise

Description: Conditional branch: If flag DF is set, jump to the new address that is calculated by adding the sign-extended 8-bit displacement to the next PC address. If flag DF is cleared, no jump is performed: The next instruction is located at the next PC address.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	1	p	p	p	p	p	p	p	p

Instruction Fields:

pppppppp - signed displacement field:
00000000 - 0
00000001 - 1
...
01111110 - 126
01111111 - 127
10000000 - -128
10000001 - -127
...
11111110 - -2
11111111 - -1

BF

Conditional Branch if False

Operation:

```
if (T == 0) PC ← PC + 1 + displacement
else PC ← PC + 1
```

Assembler:

```
Syntax:  bf label
```

Example: `bf LLL`
To jump to LLL if T is cleared, or go to the next instruction if T is set. The displacement value is calculated by the assembler.

CPU Flags: Unaffected

Cycles: 2 when the branch is done, 1 otherwise

Description: Conditional branch: If flag T is cleared, jump to the new address that is calculated by adding the sign-extended 8-bit displacement to the next PC address. If flag T is set, no jump is performed: The next instruction is located at the next PC address.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	0	0	p	p	p	p	p	p	p	p

Instruction Fields:

`pppppppp` - signed displacement field:
`00000000` - 0
`00000001` - 1
...
`01111110` - 126
`01111111` - 127
`10000000` - -128
`10000001` - -127
...
`11111110` - -2
`11111111` - -1

BSETI

Bit Set Immediate

Operation:

```
GReg[r]:{b31,...,b(i+1),1,b(i-1),...,b0} ←
GReg[r]:{b31,...,b(i+1),b(i),b(i-1),...,b0}
```

Assembler:

Syntax: `bseti r,i`

Example: bseti 6,5
to set bit 5 in GReg[6]

CPU Flags: Unaffected

Cycles: 1

Description: Sets bit number i in the selected General Register.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	1	0	i	i	i	i	i

Instruction Fields:

`rrr` - register field:
`000` - `GReg[0]`
`001` - `GReg[1]`
`010` - `GReg[2]`
`011` - `GReg[3]`
`100` - `GReg[4]`
`101` - `GReg[5]`
`110` - `GReg[6]`
`111` - `GReg[7]`

`iii` - bit number field:
`00000` - bit 0
`00001` - bit 1
...
`11110` - bit 30
`11111` - bit 31

BSF

Conditional Branch if Source Fault

Operation:

```
if (SF == 1) PC ← PC + 1 + displacement
else PC ← PC + 1
```

Assembler:

Syntax: `bsf label`

Example: `bsf LLL`
To jump to LLL if SF is set, or go to the next instruction if SF is cleared. The displacement value is calculated by the assembler.

CPU Flags: Unaffected

Cycles: 2 when the branch is done, 1 otherwise

Description: Conditional branch: If flag SF is set, jump to the new address that is calculated by adding the sign-extended 8-bit displacement to the next PC address. If flag SF is cleared, no jump is performed: The next instruction is located at the next PC address.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	0	p	p	p	p	p	p	p	p

Instruction Fields:

`pppppppp` - signed displacement field:
`00000000` - 0
`00000001` - 1
...
`01111110` - 126
`01111111` - 127
`10000000` - -128
`10000001` - -127
...
`11111110` - -2
`11111111` - -1

BT

Conditional Branch if True

Operation:

```
if (T == 1) PC ← PC + 1 + displacement
else PC ← PC + 1
```

Assembler:

Syntax: `bt label`

Example: bt LLL
To jump to LLL if T is set, or go to the next instruction if T is cleared. The displacement value is calculated by the assembler.

CPU Flags: Unaffected

Cycles: 2 when the branch is done, 1 otherwise

Description: Conditional branch: If flag T is set, jump to the new address that is calculated by adding the sign-extended 8-bit displacement to the next PC address. If flag T is cleared, no jump is performed: The next instruction is located at the next PC address.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	0	1	p	p	p	p	p	p	p	p

Instruction Fields:

pppppppp - signed displacement field:
00000000 - 0
00000001 - 1
...
01111110 - 126
01111111 - 127
10000000 - -128
10000001 - -127
...
11111110 - -2
11111111 - -1

BTSTI

Bit Test immediate

Operation:

`T ← GReg[r]:b(i)`

Assembler:

`Syntax: btsti r,i`

Example: `btsti 2,29`
to test bit 29 in GReg[2] and copy its value in flag T

CPU Flags: T

Cycles: 1

Description: T is loaded with the value of bit number i from the selected general register.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	1	1	i	i	i	i	i

Instruction Fields:

`rrr` - register field:
`000` - `GReg[0]`
`001` - `GReg[1]`
`010` - `GReg[2]`
`011` - `GReg[3]`
`100` - `GReg[4]`
`101` - `GReg[5]`
`110` - `GReg[6]`
`111` - `GReg[7]`

`iii` - bit number field:
`00000` - bit 0
`00001` - bit 1
...
`11110` - bit 30
`11111` - bit 31

CLRF

Clear CPU flags

Operation:

```
if (ff%2 == 0) SF ← 0
if (ff/2 == 0) DF ← 0
```

Assembler:

Syntax: `clrf ff`

Example: clrf 2
to clear flag SF and keep flag DF unchanged

CPU Flags: SF, DF

Cycles: 1

Description: Clears a selection of the CPU fault flags: SF, DF, both SF and DF or none can be cleared.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	f	f	0	0	0	0	0	1	1	1

Instruction Fields:

`ff` - flags field:
`00` - clear `SF` and clear `DF`
`01` - clear `DF`
`10` - clear `SF`
`11` - no clear

CMPEQ

Compare for Equal

Operation:

T ← (GReg[s] == GReg[r])

Assembler:

Syntax: `cmpeq r,s`

Example: `cmpeq 7,5`
to compare GReg[7] and GReg[5] and set flag T if they are equal

CPU Flags: T

Cycles: 1

Description: Subtracts the destination general register *r* from the source general register *s*, and sets T if the result is 0, clears T if the result is not 0.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	1	0	0	1	s	s	s

Instruction Fields:

`rrr` - destination register field:
`000` - GReg[0]
`001` - GReg[1]
`010` - GReg[2]
`011` - GReg[3]
`100` - GReg[4]
`101` - GReg[5]
`110` - GReg[6]
`111` - GReg[7]

`sss` - source register field:
`000` - GReg[0]
`001` - GReg[1]
`010` - GReg[2]
`011` - GReg[3]
`100` - GReg[4]
`101` - GReg[5]
`110` - GReg[6]
`111` - GReg[7]

CMPEQI Compare with Immediate for Equal

Operation:

```
T ← (GReg[r] == immediate)
```

Assembler:

Syntax: `cmpeqi r,immediate`

Example: `cmpeqi 2,13`
To compare GReg[2] and decimal value 13 and set flag T if they are equal

CPU Flags: T

Cycles: 1

Description: Subtracts the 0-extended immediate value from the general register, and sets T if the result is 0, clears T if the result is not 0. The immediate value is the low-order byte of the instruction.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	r	r	r	i	i	i	i	i	i	i	i

Instruction Fields:

`rrr` - register field:
`000` - `GReg[0]`
`001` - `GReg[1]`
`010` - `GReg[2]`
`011` - `GReg[3]`
`100` - `GReg[4]`
`101` - `GReg[5]`
`110` - `GReg[6]`
`111` - `GReg[7]`

`iiiiiiii` - immediate value:
`00000000` - 0
`00000001` - 1
...
`11111110` - 254
`11111111` - 255

CMPHS

Compare for Higher or Same

Operation:

T ← (GReg[r] ≥ GReg[s])

Assembler:

Syntax: `cmphs r,s`

Example: `cmphs 0,1`
To compare GReg[0] and GReg[1] and set flag T if GReg[0] is higher than or equal to GReg[1]

CPU Flags: T

Cycles: 1

Description: Compares the destination general register *r* and the source general register *s*, and sets T if the destination general register *r* is higher than or equal to the source general register *s*, clears T otherwise. The comparison is unsigned.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	1	0	1	1	s	s	s

Instruction Fields:

`rrr` - destination register field:
`000` - GReg[0]
`001` - GReg[1]
`010` - GReg[2]
`011` - GReg[3]
`100` - GReg[4]
`101` - GReg[5]
`110` - GReg[6]
`111` - GReg[7]

`sss` - source register field:
`000` - GReg[0]
`001` - GReg[1]
`010` - GReg[2]
`011` - GReg[3]
`100` - GReg[4]
`101` - GReg[5]
`110` - GReg[6]
`111` - GReg[7]

CMPLT

Compare for Less Than

Operation:

T ← (GReg[r] < GReg[s])

Assembler:

Syntax: `cmplt r,s`

Example: `cmplt 7,4`
To compare GReg[7] and GReg[4] and set flag T if GReg[7] is lower than GReg[4]

CPU Flags: T

Cycles: 1

Description: Compares the destination general register *r* and the source general register *s*, and sets T if the destination general register *r* is lower than the source general register *s*, clears T otherwise. The comparison is signed.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	1	0	1	0	s	s	s

Instruction Fields:

`rrr` - destination register field:
`000` - GReg[0]
`001` - GReg[1]
`010` - GReg[2]
`011` - GReg[3]
`100` - GReg[4]
`101` - GReg[5]
`110` - GReg[6]
`111` - GReg[7]

`sss` - source register field:
`000` - GReg[0]
`001` - GReg[1]
`010` - GReg[2]
`011` - GReg[3]
`100` - GReg[4]
`101` - GReg[5]
`110` - GReg[6]
`111` - GReg[7]

cpShReg Update Context of PCU Registers and Flags

Assembler:

Syntax: `cpShReg`

CPU Flags: none

Cycles: 1

Description: SF, RPC, T, PC,LM, EPC, DF, and SPC registers are updated according to the value of their corresponding bits in the context memory. This instruction must only be used in debug mode via the OnCE. It reverses the `done` 5 operation.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	1	0	1	1	1	0	0	0	1	0

Instruction Fields:

DONE

DONE, Yield

Operation:

```
if (jjj&6 == 2) HE[CCR] ← 0
if (jjj == 3) HI[CCR] ← 1
if (jjj == 4) EP[CCR] ← 0
if ((jjj == 0) && (NCP > CCP)) CCR ← NCR
else if ((jjj == 1) && (NCP >= CCP)) CCR ← NCR
else CCR ← NCR
```

(CCR stands for Current Channel Register; NCR stands for Next Channel Register)

Assembler:

Syntax: `done jjj`

Example: `done 3`
To clear HE bit for the current channel, send an interrupt to the AP for the current channel and reschedule.

CPU Flags: Unaffected

Cycles: Variable if a context switch is done, 1 otherwise

Description: Clears one of the channel enabling bits (HE or EP for the corresponding channel number) if required. Sends an interrupt to the corresponding CPU (AP) by setting the appropriate flag, if required (HI for the corresponding channel number). Reschedules according to the mode and the NCP (Next Channel Priority) and CCP (Current Channel Priority) values. According to the scheduling decision, the NCR (Next Channel Register) is copied to the CCR (Current Channel Register) and channel contexts are switched.
If several channels with the same highest priority are pending, they are ordered by their number from 31 down to 0. The higher number is selected (for example, channel 26 is selected if channels 3, 12, 14, and 26 with the same highest priority are pending).
If no flag is modified, the reschedule can allow the replacement of the current channel by another channel with a priority strictly greater than the current channel priority (`yield`). Or, it can allow the replacement of the current channel by another channel with a priority greater than or equal to the current channel priority (`yieldge`). In the latter case, the selected channel will always be the first one with the same priority, starting from channel number 31 down to channel 0 (the current channel does not belong to the set of selectable channels).

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	j	j	j	0	0	0	0	0	0	0	0

Instruction Fields:

jjj - Channel Flags field:
000 - No channel flags affected: Reschedule only if the next channel priority is greater than current channel priority (yield)
001 - No channel flags affected: Reschedule only if the next channel priority is greater than or equal to the current channel priority (yieldge)
010 - Clear HE for the current channel and reschedule
011 - Clear HE, set HI for the current channel and reschedule
100 - Clear EP for the current channel and reschedule
101 - Reserved for debug to copy relevant registers into context memory
110 - RESERVED
111 - RESERVED

For the scheduling rules, refer to Section 40.5.3, "Scheduler Functional Description." Every possible done instruction is further described as follows:

- done 0/yield is executed by a channel script when it accepts preemption by a higher priority channel;
- done 1/yieldge is executed by a channel script when it accepts preemption by a higher priority channel and it also accepts a roll-up with other channels that have the same priority;
- done 2 is executed by a channel script that was triggered by a AP start via the Channel Start (HSTART) register, when its task is completed and it requires termination;
- done 3 is executed by a channel script that was triggered by a AP start via the Channel Start (HSTART) register, when its task is completed, it requires termination and it needs to trigger an interrupt to the AP upon closure;
- done 4 is executed by a channel script that was triggered by a DMA request, when its task is completed and it requires termination;
- done 5 is used in debug mode only; it copies the PCU registers and flags to the context memory of the current channel;

ILLEGAL ILLEGAL Instruction

Operation:

PC ← 0001

Assembler:

Syntax: illegal

CPU Flags: Unaffected

Cycles: 2

Description: Jumps to the Illegal instruction routine located at address 0001. All unauthorized instructions result in an Illegal instruction behavior; however, the ILLEGAL instruction must be used to guarantee software compatibility with future versions of the SDMA.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1

Instruction Fields:

JMP

Unconditional Jump Immediate

Operation:

PC ← absolute_address

Assembler:

Syntax: jmp label

Example: jmp LLL
the assembler translates the label to the exact address

CPU Flags: Unaffected

Cycles: 2

Description: Jumps to the absolute address contained the lower 14 bits of the instruction (the PC is a 14-bit register).

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	a	a	a	a	a	a	a	a	a	a	a	a	a	a

Instruction Fields:

aaaaaaaaaaaaaa - address field:
00000000000000 - 0
00000000000001 - 1
...
11111111111110 - 16382
11111111111111 - 16383

JMPR

Unconditional Jump

Operation:

PC ← GReg[r]

Assembler:

Syntax: `jmpr r`

Example: `jmpr 0`
To jump to address stored in GReg[0]

CPU Flags: Unaffected

Cycles: 2

Description: Jumps to the absolute address contained in a General Register.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	0	1	0	0	0

Instruction Fields:

`rrr` - register field:
`000` - `GReg[0]`
`001` - `GReg[1]`
`010` - `GReg[2]`
`011` - `GReg[3]`
`100` - `GReg[4]`
`101` - `GReg[5]`
`110` - `GReg[6]`
`111` - `GReg[7]`

JSR Unconditional Jump to Subroutine Immediate

Operation:

```
RPC ← PC + 1
PC ← absolute_address
```

Assembler:

Syntax: `jsr r`

Example: `jsr LLL`
Jumps to subroutine starting at LLL; the assembler translates the label to exact address

CPU Flags: Unaffected

Cycles: 2

Description: Jumps to the subroutine located at the absolute address contained the lower 14 bits of the instruction (the PC is a 14-bit register).

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	a	a	a	a	a	a	a	a	a	a	a	a	a	a

Instruction Fields:

`aaaaaaaaaaaaaa` - address field:
`00000000000000` - 0
`00000000000001` - 1
...
`11111111111110` - 16382
`11111111111111` - 16383

JSRR

Unconditional Jump to Subroutine

Operation:

```
RPC ← PC + 1
PC ← GReg[r]
```

Assembler:

Syntax: `jsrr r`

Example: `jsrr 5`
Jumps to subroutine located at address stored in GReg[5]

CPU Flags: Unaffected

Cycles: 2

Description: Jumps to the subroutine at address contained in a General Register

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	0	1	0	0	1

Instruction Fields:

`rrr` - register field:
`000` - `GReg[0]`
`001` - `GReg[1]`
`010` - `GReg[2]`
`011` - `GReg[3]`
`100` - `GReg[4]`
`101` - `GReg[5]`
`110` - `GReg[6]`
`111` - `GReg[7]`

LD

Load Register

Operation:

```
GReg[r] ← [GReg[b] + displacement]
if (transfer_error) SF ← 1
else SF ← 0
```

Assembler:

Syntax: `ld r,(b,displacement)`

Example: `ld 1,(2,23)`
Loads data into GReg[1]; the data is located at address obtained by adding decimal value 23 to GReg[2]

CPU Flags: SF

Cycles: 2+n where n is 0 for ROM, RAM or memory mapped registers, and n is the number of wait-states of the peripheral for a peripheral access

Description: Adds a 5-bit 0-extended displacement to a base address in General Register b; the result is the address of the data to fetch on the DM bus. The data received from the bus is stored in the destination General Register r. If an error occurs during the transfer, the flag SF is set, else it is cleared.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	0	r	r	r	d	d	d	d	d	b	b	b

Instruction Fields:

`rrr` - destination register field:
`000` - `GReg[0]`
`001` - `GReg[1]`
...
`111` - `GReg[7]`

`bbb` - base address register field:
`000` - `GReg[0]`
`001` - `GReg[1]`
...
`111` - `GReg[7]`

`ddddd` - displacement value:
`00000` - 0
`00001` - 1

...
11111 - 31

LDF

Load Register from Functional Unit

Operation:

```
GReg[r] ← [fu_address]
if (transfer_error) SF ← 1
else SF ← 0
```

fu_address is an 8-bit field and depends on addressed functional unit

Assembler:

Syntax: `ldf r,fu_address`

Example: `ldf 0,13`
Loads data coming from the M3IF DMA register MD into GReg[0]; it is a 32-bit access with no prefetch

CPU Flags: SF

Cycles: 1+n where n is the number of wait-states that may be inserted by the functional unit

Description: Sends an 8-bit address on the Functional Unit Bus (FU bus) and stores the data received from the bus in the destination General Register r. If an error occurs during the transfer, the flag SF is set, else it is cleared.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	0	r	r	r	f	f	f	f	f	f	f	f

See the following sections for more details of the LDF instruction usage with each functional unit:

- Section 40.16.1.6, “Burst DMA Read (ldf)” for Burst DMA
- Section 40.16.2.6, “Peripheral DMA Read (ldf)—Read Mode” for Peripheral DMA
- Section 40.16.4.4, “Read Instruction (ldf)” for CRC unit

Instruction Fields:

`rrr` - register field:
`000` - `GReg[0]`
`001` - `GReg[1]`
`010` - `GReg[2]`
`011` - `GReg[3]`
`100` - `GReg[4]`
`101` - `GReg[5]`
`110` - `GReg[6]`
`111` - `GReg[7]`

ffffffff - functional unit source register and action (unspecified values are reserved):
00000000 - MSA
00000100 - MDA
00001001 - MD byte
00001010 - MD halfword
00001011 - MD word
00001100 - MS
00101001 - MD byte - prefetch
00101010 - MD halfword - prefetch
00101011 - MD word - prefetch
01000000 - DSA
01000100 - DDA
01001001 - DD byte
01001010 - DD halfword
01001011 - DD word
01001100 - DS
01101001 - DD byte - prefetch
01101010 - DD halfword - prefetch
01101011 - DD word - prefetch
10000000 - CA
10000100 - CS (CRC checksum)
11000000 - PSA
11001000 - PD
11010000 - PDA
11011000 - PD in copy mode (rrr contents are lost)
11101000 - PD - prefetch next data
11111111 - PS

LDI

Load Register with Immediate Value

Operation:

GReg[r] ← immediate

Assembler:

Syntax: ldi r,immediate

Example: ldi 6,1
loads decimal value 1 into GReg[6]

CPU Flags: Unaffected

Cycles: 1

Description: Stores a 0-extended immediate value in a General Register. The immediate value is the low-order byte of the instruction.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	r	r	r	i	i	i	i	i	i	i	i

Instruction Fields:

rrr - register field:
000 - GReg[0]
001 - GReg[1]
010 - GReg[2]
011 - GReg[3]
100 - GReg[4]
101 - GReg[5]
110 - GReg[6]
111 - GReg[7]

iiiiiiii - immediate value:
00000000 - 0
00000001 - 1
...
11111110 - 254
11111111 - 255

LDRPC — Load from RPC to Register

Operation:

GReg[r] ← RPC

Assembler:

Syntax: `ldrpc r`

Example: `ldrpc 3`
copies RPC to GReg[3]

CPU Flags: Unaffected

Cycles: 1

Description: Stores the contents of the RPC in a General Register. That instruction may be used to have more than one level of subroutines.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	0	1	0	1	0

Instruction Fields:

`rrr` - register field:
`000` - GReg[0]
`001` - GReg[1]
`010` - GReg[2]
`011` - GReg[3]
`100` - GReg[4]
`101` - GReg[5]
`110` - GReg[6]
`111` - GReg[7]

LOOP

Hardware Loop

Operation:

```
if (ff%2 == 0) SF ← 0
if (ff/2 == 0) DF ← 0
if ((GReg[0] == 0) || (SF == 1) || (DF == 1))
            PC ← PC + loop_size + 1
else
        {
            SPC ← PC + 1
            EPC ← PC + loop_size + 1
            LM ← 1
            PC ← PC + 1
        }
```

during every instruction execution in the loop:

```
if ((SF == 1) || (DF == 1))
        {
            LM ← 0
            PC ← EPC
        }
else if ((PC + 1) == EPC)
        {
            GReg[0] ← GReg[0] - 1
            if (GReg[0] == 0)
            {
                LM ← 0
                PC ← EPC
            }
            else PC ← SPC
        }
else PC ← nextPC(instruction)
```

after the execution of the last instruction of the loop body:

```
if (GReg[0] == 0)
            T ← 1
else
            T ← 0
```

Assembler:

Syntax: `loop n{,ff}`

Example: `loop 3,0`
Executes GReg[0] times the instructions comprised between PC+1 and PC+3 (included); both SF and DF flags are cleared before starting the loop. When omitted, the `ff` field is set to 0 (clearing both SF and DF).

CPU Flags: LM[1:0], T

Cycles: 2 when the loop count (GReg[0]) is 0 or SF or DF is set at loop start, 1+1 when the loop starts but exits abnormally (SF or DF set inside the loop which adds 1 cycle to the offending load or store to jump to EPC), 1 when the loop is executed normally

Description: The `loop` instruction executes a sequence of instructions several times. The number of times is given by the contents of GReg[0], the loop counter. SDMA will jump to the first instruction after the end of the loop if the value in GReg[0] is 0. Otherwise the SDMA enters loop mode. It sets the most significant bit of the LM flag that will only be reset once the last instruction of the last loop is executed. The instructions in the loop are executed GReg[0] times.
The management of fault flags (SF and DF) is as follows. When entering the hardware loop, SF and DF can be cleared according to the ff field of the instruction. After that operation, if any flag is still set the loop will not be executed. The SDMA will jump to the first instruction after the end of the loop without entering loop mode. During the execution of the loop, if any fault flag is set by a LD, LDF, ST, or STF instruction, the SDMA will immediately exit loop mode and jump to the first instruction after the end of the loop. In that case, GReg0 is not decremented for that last piece of the loop body execution (even if the SF or DF flag is set at the last instruction of the loop body).
The T flag reflects the state of GReg[0] after the end of the loop, which is an indicator of the complete execution of the loop. If the loop exited because of an error (SF or DF set), GReg[0] will not be 0 at the end of the loop, hence T will be cleared. If the loop executes without fault, GReg[0] will be 0 at the end of the loop, hence T will be set. The boundary case when a source or destination fault occurs at the last instruction of the last loop is considered as an anticipated exit of the loop, which causes the T flag to be cleared. If the last instruction executed before leaving the hardware loop also tries to modify the T flag, the flag is updated according to the value of GReg[0], NOT according to the result of the last executed instruction.

Limitations: 1. Jump instructions (JMP, JMPR, JSR, JSRR, BF, BT, BSF, BDF) are not allowed inside the hardware loop.
2. GReg[0] cannot be written to inside the hardware loop (it can be read).
3. The empty loop (0 instruction in the body) is forbidden.
4. If GReg[0] == 0 at the start of the loop, which causes a jump to EPC, the T flag is not updated.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	0	f	f	n	n	n	n	n	n	n	n

Instruction Fields:

`ff` - flags field:
`00` - clear `SF` and clear `DF`
`01` - clear `DF`
`10` - clear `SF`
`11` - no clear

nnnnnnnn - loop size
00000000 - empty loop: forbidden value
00000001 - 1 instruction in the loop
00000010 - 2 instructions in the loop
...
11111111 - 255 instructions in the loop

LSL1 Logical Shift Left by 1 Bit

Operation:

GReg[r]:{b30,...,b1,b0,0} ← GReg[r]:{b31,b30,...,b1,b0}

Assembler:

Syntax: `lsl1 r`

Example: `lsl1 2`
multiplies by 2 the value in GReg[2]

CPU Flags: Unaffected

Cycles: 1

Description: Shift the bits of any General Register to the left. The right bit (bit 0) is set to 0. No overflow is detected by the hardware.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	1	0	1	1	1

Instruction Fields:

`rrr` - register field:
`000` - GReg[0]
`001` - GReg[1]
`010` - GReg[2]
`011` - GReg[3]
`100` - GReg[4]
`101` - GReg[5]
`110` - GReg[6]
`111` - GReg[7]

LSR1

Logical Shift Right by 1 Bit

Operation:

GReg[r]:{0,b31,b30,...,b1} ← GReg[r]:{b31,b30,...,b1,b0}

Syntax: `lsr1 r`

Example: `lsr1 4`
divides by 2 the unsigned value contained in GReg[4]

CPU Flags: Unaffected

Cycles: 1

Description: Shift the bits of any General Register to the right. The left bit (bit 31) is set to 0.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	1	0	1	0	1

Instruction Fields:

`rrr` - register field:
`000` - `GReg[0]`
`001` - `GReg[1]`
`010` - `GReg[2]`
`011` - `GReg[3]`
`100` - `GReg[4]`
`101` - `GReg[5]`
`110` - `GReg[6]`
`111` - `GReg[7]`

MOV

Logical Move

Operation:

GReg[r] ← GReg[s]

Assembler:

Syntax: mov r,s

Example: mov 4,0
copies GReg[0] to GReg[4]

CPU Flags: Unaffected

Cycles: 1

Description: Move the contents of the source General Register s to the destination General Register r.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	0	0	0	1	s	s	s

Instruction Fields:

rrr - destination register field:
000 - GReg[0]
001 - GReg[1]
010 - GReg[2]
011 - GReg[3]
100 - GReg[4]
101 - GReg[5]
110 - GReg[6]
111 - GReg[7]

sss - source register field:
000 - GReg[0]
001 - GReg[1]
010 - GReg[2]
011 - GReg[3]
100 - GReg[4]
101 - GReg[5]
110 - GReg[6]
111 - GReg[7]

NOTIFY

Notify to MCU

Operation:

```
if (jjj&4 == 0) {
             if (jjj&2 == 2) HE[CCR] ← 0
             if (jjj&1 == 1) HI[CCR] ← 1
}
else if (jjj == 4) EP[CCR] ← 0
```

(CCR stands for Current Channel Register)

Assembler:

Syntax: `notify jjj`

Example: `notify 3`
clears the HE bit for the current channel and sends an interrupt to the Host for the current channel

CPU Flags: Unaffected

Cycles: 1

Description: Clears one of the channel enabling bits (HE or EP for the corresponding channel number) if required, sends an interrupt to the corresponding CPU by setting the appropriate flag if required (HI for the corresponding channel number).

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	j	j	j	0	0	0	0	0	0	0	1

Instruction Fields:

`jjj` - Channel Flags field:
`000` - unused
`001` - set HI for the current channel
`010` - clear HE for the current channel
`011` - clear HE, set HI for the current channel
`100` - clear EP for the current channel
`101` - RESERVED
`110` - RESERVED
`111` - RESERVED

OR

Logical OR

Operation:

GReg[r] ← GReg[s] | GReg[r]

Assembler:

Syntax: `or r,s`

Example: `or 3,6`
ORs GReg[3] and GReg[6] and stores the result in GReg[3]

CPU Flags: Unaffected

Cycles: 1

Description: Performs the OR of the source General Register s and the destination General Register r, and stores the result in the destination General Register r.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	0	1	0	1	s	s	s

Instruction Fields:

`rrr` - destination register field:
`000` - GReg[0]
`001` - GReg[1]
`010` - GReg[2]
`011` - GReg[3]
`100` - GReg[4]
`101` - GReg[5]
`110` - GReg[6]
`111` - GReg[7]

`sss` - source register field:
`000` - GReg[0]
`001` - GReg[1]
`010` - GReg[2]
`011` - GReg[3]
`100` - GReg[4]
`101` - GReg[5]
`110` - GReg[6]
`111` - GReg[7]

ORI

Logical OR with Immediate Value

Operation:

GReg[r] ← GReg[r] | immediate

Assembler:

Syntax: ori r,immediate

Example: ori 1,56
ORs GReg[1] and the decimal value 56 and stores the result in GReg[1]

CPU Flags: unaffected

Cycles: 1

Description: Performs an OR between a 0-extended immediate value and a General Register; stores the result in the General Register. The immediate value is the low-order byte of the instruction.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	r	r	r	i	i	i	i	i	i	i	i

Instruction Fields:

rrr - register field:
000 - GReg[0]
001 - GReg[1]
010 - GReg[2]
011 - GReg[3]
100 - GReg[4]
101 - GReg[5]
110 - GReg[6]
111 - GReg[7]

iiiiiiii - immediate value:
00000000 - 0
00000001 - 1
...
11111110 - 254
11111111 - 255

RET

Return from Subroutine

Operation:

PC ← RPC

Assembler:

Syntax: ret

CPU Flags: Unaffected

Cycles: 2

Description: Return from subroutine.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

Instruction Fields: None

REVB

Reverse Byte Order

Operation:

GReg[r]:{B3,B2,B1,B0} ← GReg[r]:{B0,B1,B2,B3}

Assembler:

Syntax: `revb r`

Example: `revb 5`
reverses bytes order in GReg[5]

CPU Flags: Unaffected

Cycles: 1

Description: Reverse the byte order of any General Register.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	1	0	0	0	0

Instruction Fields:

`rrr` - register field:
`000` - GReg[0]
`001` - GReg[1]
`010` - GReg[2]
`011` - GReg[3]
`100` - GReg[4]
`101` - GReg[5]
`110` - GReg[6]
`111` - GReg[7]

REVBLO Reverse Low Order Bytes

Operation:

GReg[r]:{B3,B2,B0,B1} ← GReg[r]:{B3,B2,B1,B0}

Assembler:

Syntax: revblo r

Example: revblo 0
reverses low order bytes in GReg[0]

CPU Flags: Unaffected

Cycles: 1

Description: Reverse both low order bytes of any General Register.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	1	0	0	0	1

Instruction Fields:

rrr - register field:
000 - GReg[0]
001 - GReg[1]
010 - GReg[2]
011 - GReg[3]
100 - GReg[4]
101 - GReg[5]
110 - GReg[6]
111 - GReg[7]

ROR1

Rotate Right by 1 Bit

Operation:

GReg[r]:{b0,b31,b30,...,b1} ← GReg[r]:{b31,b30,...,b1,b0}

Assembler:

Syntax: ror1 r

Example: ror1 3
rotates bits to the right in GReg[3]

CPU Flags: Unaffected

Cycles: 1

Description: Rotate the bits of any General Register to the right.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	1	0	1	0	0

Instruction Fields:

rrr - register field:
000 - GReg[0]
001 - GReg[1]
010 - GReg[2]
011 - GReg[3]
100 - GReg[4]
101 - GReg[5]
110 - GReg[6]
111 - GReg[7]

RORB

Rotate Right by 1 Byte

Operation:

GReg[r]:{B0,B3,B2,B1} ← GReg[r]:{B3,B2,B1,B0}

Assembler:

Syntax: rorb r

Example: rorb 2
rotates bytes to the right in GReg[2]

CPU Flags: Unaffected

Cycles: 1

Description: Rotate the bytes of any General Register to the right.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	1	0	0	1	0

Instruction Fields:

rrr - register field:
000 - GReg[0]
001 - GReg[1]
010 - GReg[2]
011 - GReg[3]
100 - GReg[4]
101 - GReg[5]
110 - GReg[6]
111 - GReg[7]

SOFTBKPT Software Breakpoint

Operation:

Stops the current script and enters debug mode

Assembler:

Syntax: softbkpt

CPU Flags: Unaffected

Cycles:

Description: When the core executes this instruction, it has the same effect as receiving a debug request from the OnCE or via the external debug request input: the script execution halts, the PCU enters its debug state and waits for the OnCE commands that are described in Section 40.16.5, "OnCE and Real-Time Debug."

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

Instruction Fields: None

ST

Store Register

Operation:

```
[GReg[b] + displacement] ← GReg[r]
if (transfer_error) DF ← 1
else DF ← 0
```

Assembler:

Syntax: `st r,(b,displacement)`

Example: `st 7,(0,9)`
stores the value from GReg[7] into memory at address obtained by adding decimal value 9 to GReg[0]

CPU Flags: DF

Cycles: 2+n where n is 0 for ROM, RAM or memory mapped registers, and n is the number of wait-states of the peripheral for a peripheral access

Description: Adds a 5-bit 0-extended displacement to a base address in General Register b; the result is the address of the data to store on the DM bus. The data sent on the bus comes from the source General Register r. If an error occurs during the transfer, the flag DF is set, else it is cleared.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	1	r	r	r	d	d	d	d	d	b	b	b

Instruction Fields:

`rrr` - source register field:
`000` - `GReg[0]`
`001` - `GReg[1]`
...
`111` - `GReg[7]`

`bbb` - base address register field:
`000` - `GReg[0]`
...
`110` - `GReg[6]`
`111` - `GReg[7]`

`ddddd` - displacement value:
`00000` - 0
`00001` - 1
...
`11111` - 31

STF

Store Register in Functional Unit

Operation:

```
[fu_address] ← GReg[r]
if (transfer_error) DF ← 1
else DF ← 0
```

fu_address is an 8-bit field

Assembler:

Syntax: `stf r,fu_address`

Example: `stf 3,0x2B`
stores the 32-bit contents of GReg[3] to the M3IF DMA register MF; waits until the flush to external M3IF memory is completed

CPU Flags: DF

Cycles: 1+n where n is the number of wait-states that may be inserted by the functional unit

Description: Sends an 8-bit address on the Functional Unit Bus (FU bus) and sends the contents of the source General Register r on the bus. If an error occurs during the transfer, the flag DF is set, else it is cleared.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	1	r	r	r	f	f	f	f	f	f	f	f

See the following sections for more details of the STF instruction usage with each functional unit:

- Section 40.16.1.5, “Burst DMA Write (stf)” for Burst DMA
- Section 40.16.2.5, “Peripheral DMA Write (stf)—Write Mode” for Peripheral DMA
- Section 40.16.4.3, “Write Instruction (stf)” for CRC

Instruction Fields:

`rrr` - register field:
`000` - `GReg[0]`
`001` - `GReg[1]`
`010` - `GReg[2]`
`011` - `GReg[3]`
`100` - `GReg[4]`
`101` - `GReg[5]`
`110` - `GReg[6]`
`111` - `GReg[7]`

ffffffff - functional unit destination register and action (unspecified values are reserved):
00000000 - MSA in incremented mode
00000100 - MDA in incremented mode
00001001 - MD byte
00001010 - MD halfword
00001011 - MD word
00001100 - clear MS error flag
00001111 - MS
00010000 - MSA in frozen mode
00010100 - MDA in frozen mode
00011000 - MD in copy mode - number of words in rrr
00100000 - MSA in incremented mode - start prefetch
00101000 - MD no data - flush
00101001 - MD byte - flush
00101010 - MD halfword - flush
00101011 - MD word - flush
00110000 - MSA in frozen mode - start prefetch
01000000 - DSA in incremented mode
01000100 - DDA in incremented mode
01001001 - DD byte
01001010 - DD halfword
01001011 - DD word
01001100 - clear DS error flag
01001111 - DS
01010001 - DSA in frozen mode mode - 8-bit data width
01010010 - DSA in frozen mode mode - 16-bit data width
01010011 - DSA in frozen mode mode - 32-bit data width
01010101 - DDA in frozen mode mode - 8-bit data width
01010110 - DDA in frozen mode mode - 16-bit data width
01010111 - DDA in frozen mode mode - 32-bit data width
01011000 - DD in copy mode - number of data in rrr
01100000 - DSA in incremented mode - prefetch data
01101000 - DD no data - flush
01101001 - DD byte - flush
01101010 - DD halfword - flush
01101011 - DD word - flush
01110001 - DSA in frozen mode mode - 8-bit data width - prefetch data
01110010 - DSA in frozen mode mode - 16-bit data width - prefetch data
01110011 - DSA in frozen mode mode - 32-bit data width - prefetch data
10000000 - CA
10000100 - init CRC accumulator
10010100 - CS byte - compute CRC
10010101 - CS byte - compute CRC
10010110 - CS halfword - compute CRC
10010111 - CS word - compute CRC

11000001 - PSA in frozen mode - 8-bit data width
11000010 - PSA in frozen mode - 16-bit data width
11000011 - PSA in frozen mode - 32-bit data width
11000101 - PSA in incremented mode - 8-bit data width
11000110 - PSA in incremented mode - 16-bit data width
11000111 - PSA in incremented mode - 32-bit data width
11001000 - PD
11001001 - PSA in decremented mode - 8-bit data width
11001010 - PSA in decremented mode - 16-bit data width
11001011 - PSA in decremented mode - 32-bit data width
11001100 - clear PS error flag
11001101 - PSA data width becomes 8-bit
11001110 - PSA data width becomes 16-bit
11001111 - PSA data width becomes 32-bit
11010001 - PDA in frozen mode - 8-bit data width
11010010 - PDA in frozen mode - 16-bit data width
11010011 - PDA in frozen mode - 32-bit data width
11010101 - PDA in incremented mode - 8-bit data width
11010110 - PDA in incremented mode - 16-bit data width
11010111 - PDA in incremented mode - 32-bit data width
11011001 - PDA in decremented mode - 8-bit data width
11011010 - PDA in decremented mode - 16-bit data width
11011011 - PDA in decremented mode - 32-bit data width
11011101 - PDA data width becomes 8-bit
11011110 - PDA data width becomes 16-bit
11011111 - PDA data width becomes 32-bit
11100001 - PSA in frozen mode - 8-bit data width - prefetch data
11100010 - PSA in frozen mode - 16-bit data width - prefetch data
11100011 - PSA in frozen mode - 32-bit data width - prefetch data
11100101 - PSA in incremented mode - 8-bit data width - prefetch data
11100110 - PSA in incremented mode - 16-bit data width - prefetch data
11100111 - PSA in incremented mode - 32-bit data width - prefetch data
11101001 - PSA in decremented mode - 8-bit data width - prefetch data
11101010 - PSA in decremented mode - 16-bit data width - prefetch data
11101011 - PSA in decremented mode - 32-bit data width - prefetch data
11101101 - PSA data width becomes 8-bit - prefetch data
11101110 - PSA data width becomes 16-bit - prefetch data
11101111 - PSA data width becomes 32-bit - prefetch data
11111111 - PS

SUB

Subtract

Operation:

```
GReg[r] ← GReg[r] - GReg[s]
T ← (GReg[r] == 0)
```

Assembler:

Syntax: `sub r,s`

Example: `sub 4,7`
SUBtracts GReg[7] from GReg[4] and stores the result in GReg[4]

CPU Flags: T

Cycles: 1

Description: Subtracts the source General Register s from the destination General Register r, and stores the result in the destination General Register r. The T flag is set if the result of the operation is 0; it is cleared if the result is not 0.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	0	1	0	0	s	s	s

Instruction Fields:

`rrr` - destination register field:
`000` - `GReg[0]`
`001` - `GReg[1]`
`010` - `GReg[2]`
`011` - `GReg[3]`
`100` - `GReg[4]`
`101` - `GReg[5]`
`110` - `GReg[6]`
`111` - `GReg[7]`

`sss` - source register field:
`000` - `GReg[0]`
`001` - `GReg[1]`
`010` - `GReg[2]`
`011` - `GReg[3]`
`100` - `GReg[4]`
`101` - `GReg[5]`
`110` - `GReg[6]`
`111` - `GReg[7]`

SUBI

Subtract with Immediate

Operation:

`GReg[r] ← GReg[r] - immediate T ← (GReg[r] == 0)`

Assembler:

Syntax: `sub r,immediate`

Example: `sub 1,255`
SUBtracts decimal value 255 from GReg[1] and stores the result in GReg[1]

CPU Flags: T

Cycles: 1

Description: Subtracts a 0-extended immediate value from a General Register; stores the result in the General Register. The flag T is set when the result of the operation is 0; otherwise, it is cleared. The immediate value is the low-order byte of the instruction.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	r	r	r	i	i	i	i	i	i	i	i

Instruction Fields:

`rrr` - register field:
`000` - `GReg[0]`
`001` - `GReg[1]`
`010` - `GReg[2]`
`011` - `GReg[3]`
`100` - `GReg[4]`
`101` - `GReg[5]`
`110` - `GReg[6]`
`111` - `GReg[7]`

`iiiiiiii` - immediate value:
`00000000` - 0
`00000001` - 1
...
`11111110` - 254
`11111111` - 255

TST

Test with Zero

Operation:

```
T ← ((GReg[s] & GReg[r]) != 0)
```

Assembler:

Syntax: `tst r,s`

Example: `tst 2,3`
ANDs GReg[2] and GReg[3] and sets T if the result is non-null

CPU Flags: T

Cycles: 1

Description: Performs the AND of the source General Register s and the destination General Register r, and sets T if the result is not 0, clears T if the result is 0.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	1	0	0	0	s	s	s

Instruction Fields:

`rrr` - destination register field:
`000` - GReg[0]
`001` - GReg[1]
`010` - GReg[2]
`011` - GReg[3]
`100` - GReg[4]
`101` - GReg[5]
`110` - GReg[6]
`111` - GReg[7]

`sss` - source register field:
`000` - GReg[0]
`001` - GReg[1]
`010` - GReg[2]
`011` - GReg[3]
`100` - GReg[4]
`101` - GReg[5]
`110` - GReg[6]
`111` - GReg[7]

TSTI

Test Immediate

Operation:

```
T ← ((GReg[r] & immediate) != 0)
```

Assembler:

Syntax: `tsti r,immediate`

Example: `tsti 5,13`
ANDs GReg[5] and decimal value 13 and sets T if the result is non-null

CPU Flags: T

Cycles: 1

Description: Performs the AND of a 0-extended immediate value and the destination General Register r, and sets T if the result is not 0, clears T if the result is 0. The immediate value is the low-order byte of the instruction.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	r	r	r	i	i	i	i	i	i	i	i

Instruction Fields:

`rrr` - register field:
`000` - `GReg[0]`
`001` - `GReg[1]`
`010` - `GReg[2]`
`011` - `GReg[3]`
`100` - `GReg[4]`
`101` - `GReg[5]`
`110` - `GReg[6]`
`111` - `GReg[7]`

`iiiiiiii` - immediate value:
`00000000` - 0
`00000001` - 1
...
`11111110` - 254
`11111111` - 255

XOR

Logical Exclusive OR

Operation:

GReg[r] ← GReg[s] ^ GReg[r]

Assembler:

Syntax: `xor r,s`

Example: `xor 0,3`
XORs GReg[0] and GReg[3] and stores the result in GReg[0]

CPU Flags: Unaffected

Cycles: 1

Description: Performs the eXclusive OR of the source General Register s and the destination General Register r, and stores the result in the destination General Register r.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	0	0	1	0	s	s	s

Instruction Fields:

`rrr` - destination register field:
`000` - GReg[0]
`001` - GReg[1]
`010` - GReg[2]
`011` - GReg[3]
`100` - GReg[4]
`101` - GReg[5]
`110` - GReg[6]
`111` - GReg[7]

`sss` - source register field:
`000` - GReg[0]
`001` - GReg[1]
`010` - GReg[2]
`011` - GReg[3]
`100` - GReg[4]
`101` - GReg[5]
`110` - GReg[6]
`111` - GReg[7]

XORI

Exclusive OR with Immediate

Operation:

GReg[r] ← GReg[r] ^ immediate

Assembler:

Syntax: xori r,immediate

Example: xor 7,5
XORs GReg[5] and decimal value 5 and stores the result in GReg[7]

CPU Flags: Unaffected

Cycles: 1

Description: Performs an eXclusive OR between a 0-extended immediate value and a General Register; stores the result in the General Register. The immediate value is the low-order byte of the instruction.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	r	r	r	i	i	i	i	i	i	i	i

Instruction Fields:

rrr - register field:
000 - GReg[0]
001 - GReg[1]
010 - GReg[2]
011 - GReg[3]
100 - GReg[4]
101 - GReg[5]
110 - GReg[6]
111 - GReg[7]

iiiiiiii - immediate value:
00000000 - 0
00000001 - 1
...
11111110 - 254
11111111 - 255

YIELD, YIELDGE DONE, Yield

Limitations: By default, unsupported assembler syntax. Can be aliased to the corresponding `done` instructions (`yield` = `done` 0; `yieldge` = `done` 1). Refer to the `done` instruction description on page 40-174.

40.20 Reference Clocks

There are numerous clock sources that are used in the SDMA. They belong to one of three possible groups, and have frequency constraints within every group and between the groups:

- AP/SDMA clock domain:
  - sdma_133_clk
  - sdma_66_clk
  - ips_hostctrl_clk
- JTAG clock domain:
  - tck

Within the AP/SDMA clock domain, all clocks must come from the same PLL and be edge-aligned (for example, balanced). The AHB interfaces receive their clock from the sdma_133_clk source whose frequency can be once or twice the frequency of the main clock source (sdma_66_clk). The interfaces are designed to work at 133 MHz, but the rest of the SDMA can only run at 104 MHz. Therefore, two configurations are available (see Table 40-88). The ips_hostctrl_clk clock source must be an exact sub-frequency of the sdma_66_clk clock source (any integer value greater or equal to 1), but it will not run at frequencies higher than 66 MHz.

Table 40-88. AP/SDMA Clocks Configuration[1]

Clock	Configuration #1		Configuration #2	
	Ratio	Max Frequency	Ratio	Max Frequency
sdma_66_clk	1 (reference)	66 MHz	1 (reference)	104 MHz
sdma_133_clk	2	133 MHz	1	104 MHz
ips_hostctrl_clk (Peripheral IPG clock)	1/n (n ≥ 1)	66 MHz (n = 1) 33 MHz (n = 2) 22 MHz (n = 3) and so on.	1/n (n ≥ 2)	52 MHz (n = 2) 35 MHz (n = 3) 26 MHz (n = 4) and so on.

1 n is always an integer

The JTAG clock is not used as a clock inside the SDMA. It is sampled by the SDMA main clock to determine its rising edge. This simplifies design and clock management, but it also adds a ratio constraint between those two clocks. It is guaranteed the JTAG interface works properly when the frequency of TCK is lower than 1/8th of the frequency of the SDMA main clock (which is about 8 MHz when the SDMA main clock frequency is 66 MHz).

40.21 Software Restrictions

40.21.1 Unsupported Burst DMA Access Sequence

The SDMA does not support triggering a pre-fetch followed by a flush of the Burst DMA without reading or writing any data. If the flush occurs while the background pre-fetch DMA operation is still in progress, it could result in un-defined behavior. An example of the sequence which could result in undefined results is in **Example 40-8.**

Example 40-8. Instruction sequence not supported

```
stf r1, MSA|PF    ; Update source address, triggers data pre-fetch in the background
mov R0,R0         ; Execute multiple assembly instructions, none of which read
mov R0,R0         ; or write data to/from MD
stf MD|SZ0|FL     ; Flush FIFO without writing data. If the pre-fetch is still in
                  ; progress when this instruction is executed, there could be
                  ; undefined operation.
```

A work-around to avoid any undesirable results is to first read MD to ensure the pre-fetch is complete before the flush is attempted.

Example 40-9. Work-Around to Example 40-8

```
stf r1, MSA|PF    ; Update source address, triggers data pre-fetch
mov R0,R0         ; Execute multiple assembly instructions, none of which read
mov R0,R0         ; or write data to/from MD
ldf r2, MD        ; dummy read of MD to ensure pre-fetch is complete before the
                  ; next instruction
stf MD|SZ0|FL     ; Flush FIFO without writing data.
```

40.22 Application Notes

40.22.1 Typical Data Transfer Supported by SDMA DMA Units

This section presents a library of SDMA scripts that perform data transfers through the peripheral DMA and the burst DMA units. The AP memory and peripherals are devices that either the peripheral DMA or the burst DMA can access. The scripts are given for a peripheral DMA whose address registers are programmed in incremented mode when internal memory is involved. See Table 40-89 for the summary.

Table 40-89. Typical Data Transfers Summary

Data Transfer	Peripheral DMA	Burst DMA	Comments
AP External Memory ↔ AP External Memory		3	Copy mode Script example, see Section 40.16.1.9, "Copy Mode" and Section 40.22.1.1, "External Memory to External Memory."
AP Peripheral ↔ AP Peripheral	3		Copy mode if same data path width Script example, see Section 40.22.1.2, "Peripheral to Peripheral Transfer."

Table 40-89. Typical Data Transfers Summary (continued)

Data Transfer	Peripheral DMA	Burst DMA	Comments
AP External Memory ↔ AP Peripheral	3	3	Data transit through SDMA Script example, see Section 40.22.1.3, "Transfer Between Peripheral and External Memory."
AP External Memory ↔ AP Internal Memory		3	Copy mode Script example, see Section 40.22.1.4, "Transfer Between External Memory and Internal Memory."
AP Internal Memory ↔ AP Internal Memory		3	Copy mode Script example, see Section 40.22.1.4.1, "Internal Memory to Internal Memory."
AP Internal memory ↔ AP Peripheral	3		Data transit through SDMA Script example, see Section 40.22.1.4.2, "Transfer Between Peripheral and Internal Memory."

NOTE

These scripts are provided as examples of how to use DMA modules to perform required data transfers: They are not "official" programs.

40.22.1.1 External Memory to External Memory

This section describes the SDMA script that performs data moves in external memory. For this particular data transfer, only the burst DMA is used. It is programmed in copy mode, so no data transmits through an SDMA general register. The SDMA core only monitors data transfer status. It is assumed source and destination address values are already present in two SDMA general registers (r1 and r2). For this example, it is also assumed that a 32-bit word-to-move for source-to-destination address is present in r0 and equals 64.

Example 40-10. Data Moves in External Memory

```
1         stf r1,MSA                  // Source address setup
2         stf r2,MDA                  // Destination address setup
3         ldi r0,0x64                 // 64 words must be transferred from MSA to MDA
4         ldi r1,0x8
MAIN_XFER:
5         cmphs r0,r1                 // Is r0 >= 0x8
6         bf LAST_XFER                // If not, jump to last transfer label
7         stf r1,MD|CPY               // Copy 8 words from MSA to MDA address.
8         subi r0,0x8                 // Decrement counter
9         jmp MAIN_XFER               // return to main transfer loop
LAST_XFER:
10        stf r0,MD|CPY               // perform last transfer
```

All instructions are performed in one cycle (jumps excepted). Instruction 7 triggers a copy transfer: A read burst access of 8-word starts, data is staged in MD and then a write burst of 8 words is executed. Instruction

8, 9, 5, and 6 are executed while the burst access is in progress. If this access is not complete when instruction 7 is executed a second time, SDMA stalls on this instruction as long as the previous copy transfer is not over. In this case, the instruction is no longer a one-cycle instruction.

During the main loop (MAIN_XFER), `r1` always equals 8, so burst lengths are 8 words. On the last `ldf |CPY` instruction (10), `r1` equals the reminder of `r0` divided by 8; therefore, the length of bursts triggered in copy mode equal `r1` value, which is between 1 and 7.

40.22.1.2 Peripheral to Peripheral Transfer

For this data transfer, only the peripheral DMA is used. It is programmed in copy mode, so no data will transmit through the SDMA general register used in the `ldf` instruction, but the contents of the general register are lost. The SDMA core only monitors the transfer.

40.22.1.2.1 Source and Destination Target Have the Same Data Path Width

When the source and destination target have the same data path width, the following is true:

- Source target is a *halfword* (16-bit) peripheral located at address 0x1002.
- Destination is a *halfword* (16-bit) peripheral located at address 0x2006.

It is assumed the address values are already present in two SDMA general registers (r1, r2). The script for a transfer of 10 halfword is as follows:

Example 40-11. Same Data Path Width for Source and Destination

```
//SETUP SECTION
1        stf r1, PSA|SZ16|F         //r1=0x1002 Source address register setup
2        stf r2, PDA|SZ16|F         //r2=0x2006 Destination address register setup
3        bdf ERROR_ADDR_SETUP
4        ldi r0,0xa                 //loop counter is 10
//MAIN LOOP TRANFER
copy_loop:
5        loop 2,0
6          ldf r7,PD|CPY            //Reads 1 half-word from src and writes to dest.
7          yield
8        bdf ERROR_DURING_XFER

ERROR_ADDR_SETUP:
//correction of PSA/PDA setup and jumps to main loop transfer
ERROR_DURING_XFER:
//flag error is set,
//PS can be read to know if error occurs during read or write access.
```

If a data transfer must occur between two word peripherals, only the setup section should be updated. The transfer itself is always performed by the hardware `loop` instruction.

All instructions are executed in one cycle (change of flow excepted). On instruction 6, a single read access is triggered, read data is staged in PD, and a write-to-destination is executed. When the transfers are in progress, the SDMA can execute he next instructions in parallel. If instruction 6, which performs the copy transfer, is executed while the previous access is not over, SDMA is stalled and instruction `ldf` is a multi-cycle instruction.

40.22.1.2.2 Source and Destination Target Have a Different Data Path Width

When the source and destination target have a different data path width, copy mode cannot be used, and any attempt to initiate a copy transfer immediately raises an error, which is stored in the SF flag. Example 40-12 shows the SDMA code that could transfer 10 words from a *word* (32-bit) peripheral to a *halfword* peripheral whose addresses are preliminary and stored in r1 and r2.

Example 40-12. Different Data Path Width for Source and Destination

```
//SETUP SECTION
1        stf r1, PSA|SZ32|F|PF      //r1=0x1000 and prefetch data
2        stf r2, PDA|SZ16|F         //r2=0x2006
3        bdf ERROR_ADDR_SETUP
4        ldi r0,0xa                 //loop counter is 10
//MAIN LOOP TRANFER
main_loop_xfer_16_16:
5        loop 6,0
6          ldf r7,PD                //copy 32-bit of PD in r7
7          stf r7,PD                //store 16 LSB of r7 in PD and a flush is executed
8          rorb r7
9          rorb r7                  //16 MSB --> 16 LSB
10         stf r7,PD                //store 16 LSB of r6 in PD and a flush.
11         yield
```

On instruction 1, when the source address register is programmed and a data prefetch is required, a read access is executed. In parallel, the SDMA executes instructions 2 to 5. On instruction 6, the SDMA tries to read data that was fetched by instruction 1. If data is ready, the `ldf` will be a one cycle instruction; otherwise, the SDMA is stalled as long as the read access is not finished. Then, the 16 LSB of the read data is stored in PD and automatically flushed to the destination peripheral. In parallel, the SDMA executes the rotation instructions (8, 9), and stores the 16 MSB of the read data into PD. If a previous write access is finished, instruction 10 will be a one-cycle instruction.

The main loop transfer may appear inefficient, but due to wait states imposed to the peripheral DMA each time an external access is performed, a software pipeline is in place. During the time needed to flush PD, the SDMA executes the move and rotation operations. SDMA executes instructions in parallel with DMA accesses.

40.22.1.3 Transfer Between Peripheral and External Memory

40.22.1.3.1 Peripheral to External Memory Transfer

A transfer from a peripheral to the external memory controller involves the peripheral DMA and the burst DMA. The code for transferring 100 word from word peripheral to the external memory would be as follows:

Example 40-13. Peripheral to External Memory Transfer

```
//SETUP SECTION source and destination addresses are already in r1 and r2
1        stf r1, PSA|SZ16|F|PF      //r1=0x1000 and prefetch 32-bit data
2        stf r2, MDA                //r2=0x2000, setup burst DMA destination address
3        bdf ERROR_ADDR_SETUP
4        ldi r0,0x64                //loop counter is 100
```

```
5
//MAIN LOOP TRANFER
6         loop 3,0
7           ldf r1,PD|PF              // read 32 bits of PD and initiate a new read access.
8           stf r1,MD|32              // store 32 bits of r1 in the MD fifo.
9           yield
10        ldf r1,PD                   // last word data is read
11        stf r1,MD|32|FL             // to flush all remaining bytes of MD
```

On instruction 1, the source address register of the peripheral DMA is programmed and data is fetched. This data is stored in PD and the SDMA reads PD during instruction 7, which is a one-cycle instruction that is read-access finished. On the same instruction (7), a data prefetch is required and a read access to the source peripheral is executed. In parallel, the SDMA stored the previous read data into the data register of MD. When MD (which is an eight-word FIFO) is full, a burst write access is executed to empty the FIFO. As long as the next SDMA instructions do not access the burst DMA, they will be one-cycle instructions. Figure 40-73 and Figure 40-74 show how the peripheral DMA and burst DMA work in parallel.

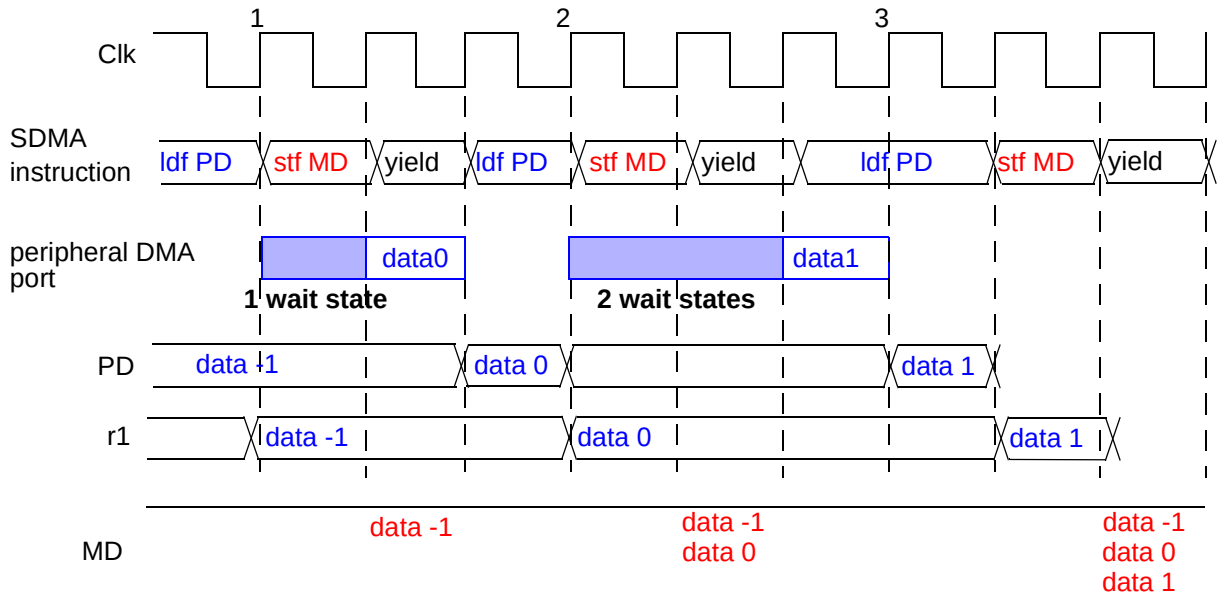


Figure 40-73. Peripheral to External Memory Example (1)

As seen in Figure 40-73, the read access triggered by the `ldf PD` instruction is symbolized by the blue bar when in progress. After wait states, the read data (data 0, data 1) is stored in PD on the clk rising edge. On edge 2, data 0 is available in PD so it can be transferred to the SDMA general register r1, and then stored in MD FIFO. On edge 3, data 1 is not in PD; therefore, SDMA is stalled on the `ldf` instruction, which lasts two cycles. Figure 40-74 shows an example of when MD FIFO is full with data.

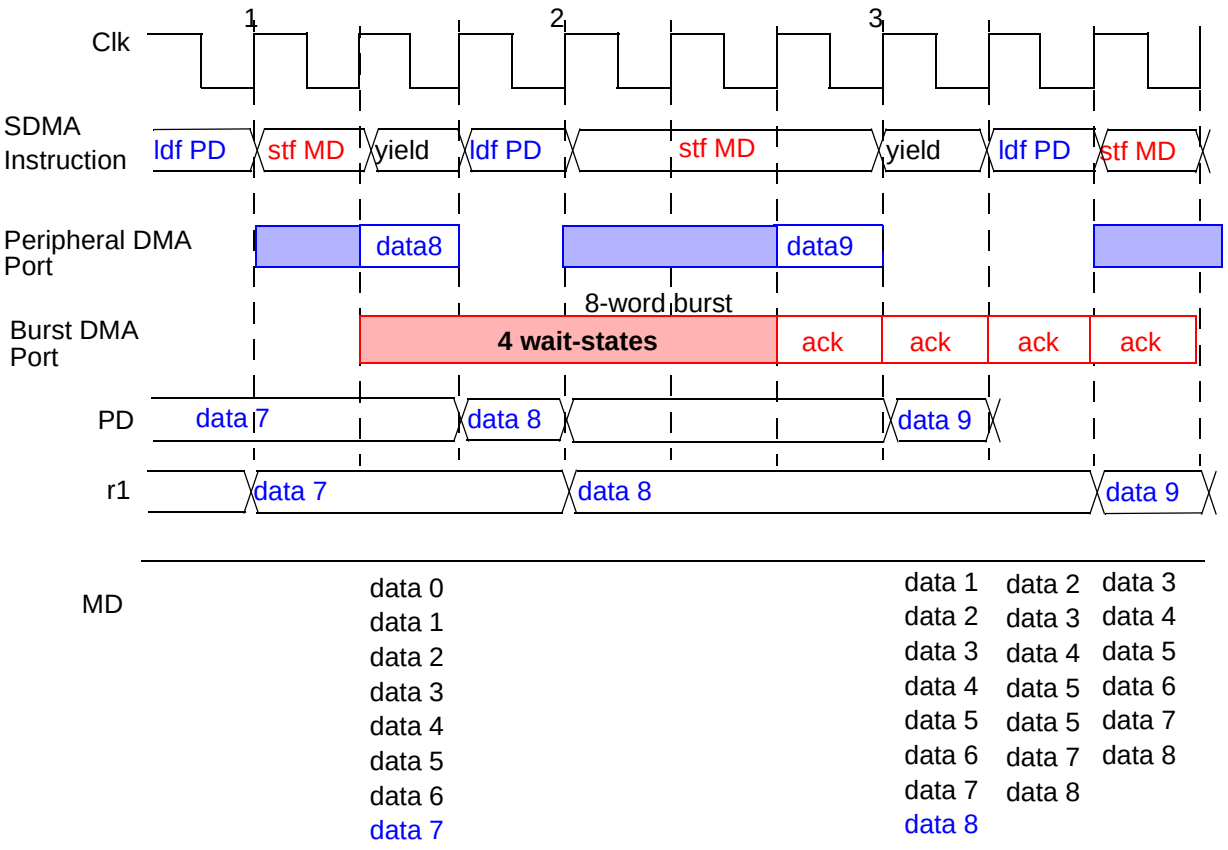


Figure 40-74. Peripheral to External Memory Example (2)

In Figure 40-74, the write bar means the burst DMA is performing a write burst access. The latency to have the first write acknowledge is four cycles. SDMA is stalled on instruction `stf` because no acknowledge was received, MD FIFO is full, and there is no empty slot to store data 9. When an acknowledge is sampled by the burst DMA, FIFO is shifted and data 8 is written. As long as there is at least one empty slot in MD FIFO, the `stf MD` instruction lasts one cycle.

40.22.1.3.2 External Memory to Peripheral Transfer

A transfer from the external memory to a peripheral involves the peripheral DMA and the burst DMA. The code for transferring 100 word from external memory to a word peripheral would be as follows:

Example 40-14. External Memory to Peripheral Transfer

```
//SETUP SECTION source and destination addresses are already in r1 and r2
1        stf r1, MSA|PF    //r1=0x1000 and starts a 8-word read burst
2        stf r2, PDA|SZ32|P//r2=0x2010, setup peripheral DMA destination address
3        bdf ERROR_ADDR_SETUP
4        ldi r0,0x64       //loop counter is 100

//MAIN LOOP TRANFER
6        loop 3,0
7          ldf r1,MD|32|PF // read 32 bits of MD and initiate a new read access
                           // if MD is empty after this reading.
```

```
8          stf r1,PD       // store 32 bits of r1 in the PD.
9          yield

10       ldf r1,MD|32      // last word data is read
11       stf r1,PD         // last write access
```

On instruction 1, a read burst of 8 words begins. Read data is staged into MD. On instruction 7 (and if data is available in MD), 32 bits are copied into r1. Then instruction 8 writes them into PD and an automatic flush is executed. The SDMA core, peripheral DMA, and burst DMA can work in parallel as long as no SDMA instruction tries to start a new write access on the peripheral DMA while the previous access is still in progress, or as long as there is data in MD when the SDMA tries to read it.

40.22.1.4 Transfer Between External Memory and Internal Memory

Since the internal memory (AP RAM) is accessed via the peripheral DMA and the external memory is accessed via the burst DMA, the SDMA scripts that are described in Section 40.22.1.3, “Transfer Between Peripheral and External Memory” can be reused. The exception is that the peripheral DMA address registers (PSA or PDA, depending on the script) should be programmed in incremented mode rather than frozen mode.

40.22.1.4.1 Internal Memory to Internal Memory

The internal memory can only be accessed via the peripheral DMA, so the script described in Section 40.22.1.2, “Peripheral to Peripheral Transfer” can be reused with a different programming of the peripheral DMA address registers.

40.22.1.4.2 Transfer Between Peripheral and Internal Memory

For this transfer, the peripheral DMA is also used in copy mode. The SDMA script is very similar to the one described in Section 40.22.1.2, “Peripheral to Peripheral Transfer,” except for the peripheral DMA address registers programming.

Chapter 41
Shared Peripheral Bus Arbiter (SPBA)

The Shared Peripheral Bus Arbiter (SPBA) is a three-to-one IP SkyBlue line interface (IP-Bus) arbiter with a resource locking mechanism. The masters can access up to thirty-one shared peripherals through the SPBA.

Figure 41-1 shows the SPBA implementation with the three masters and the shared peripherals.

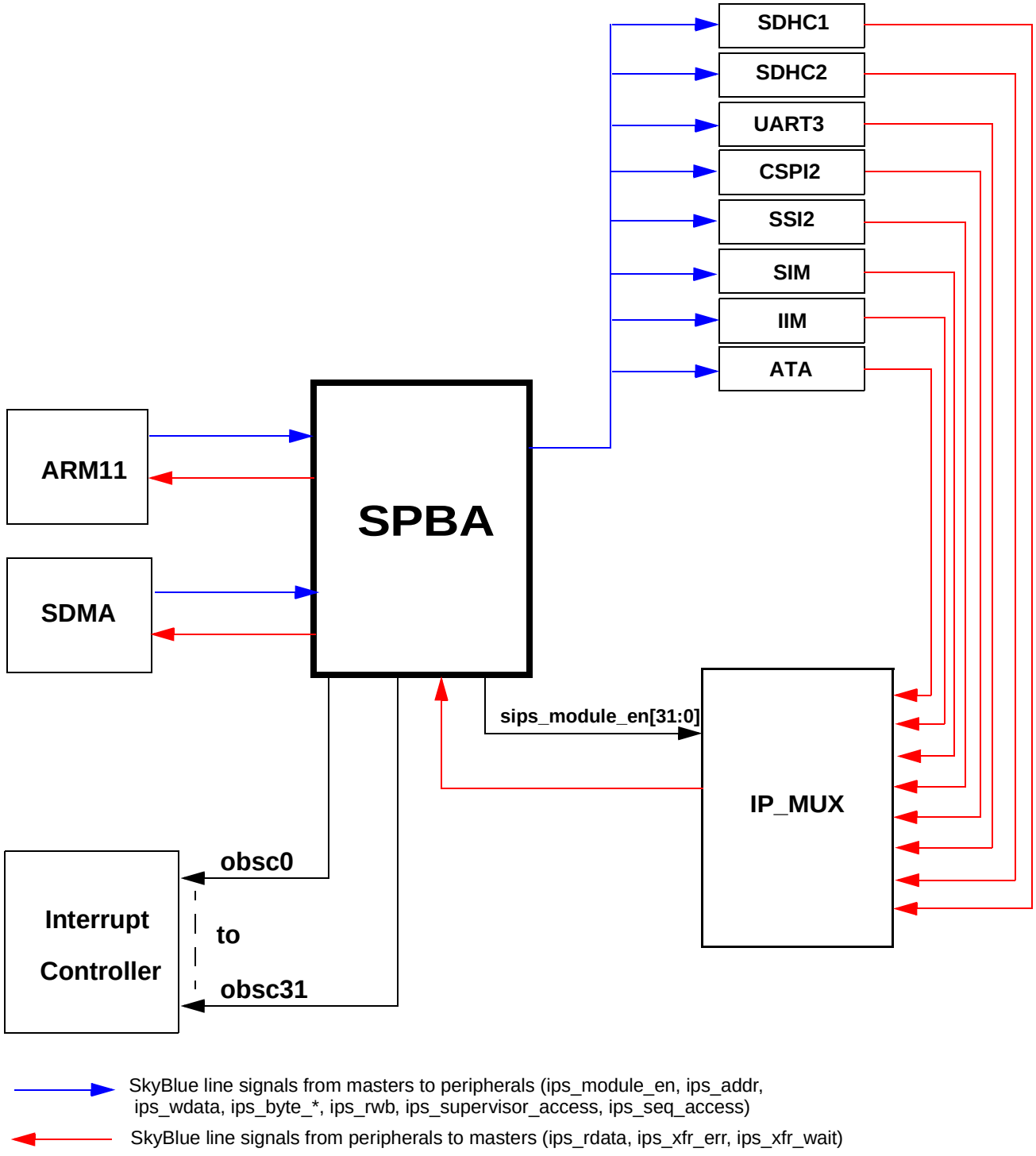


Figure 41-1. i.MX31 and i.MX31L SPBA Connectivity

Figure 41-2 shows the SPBA block diagram and details the Out-of-Band signals routing.

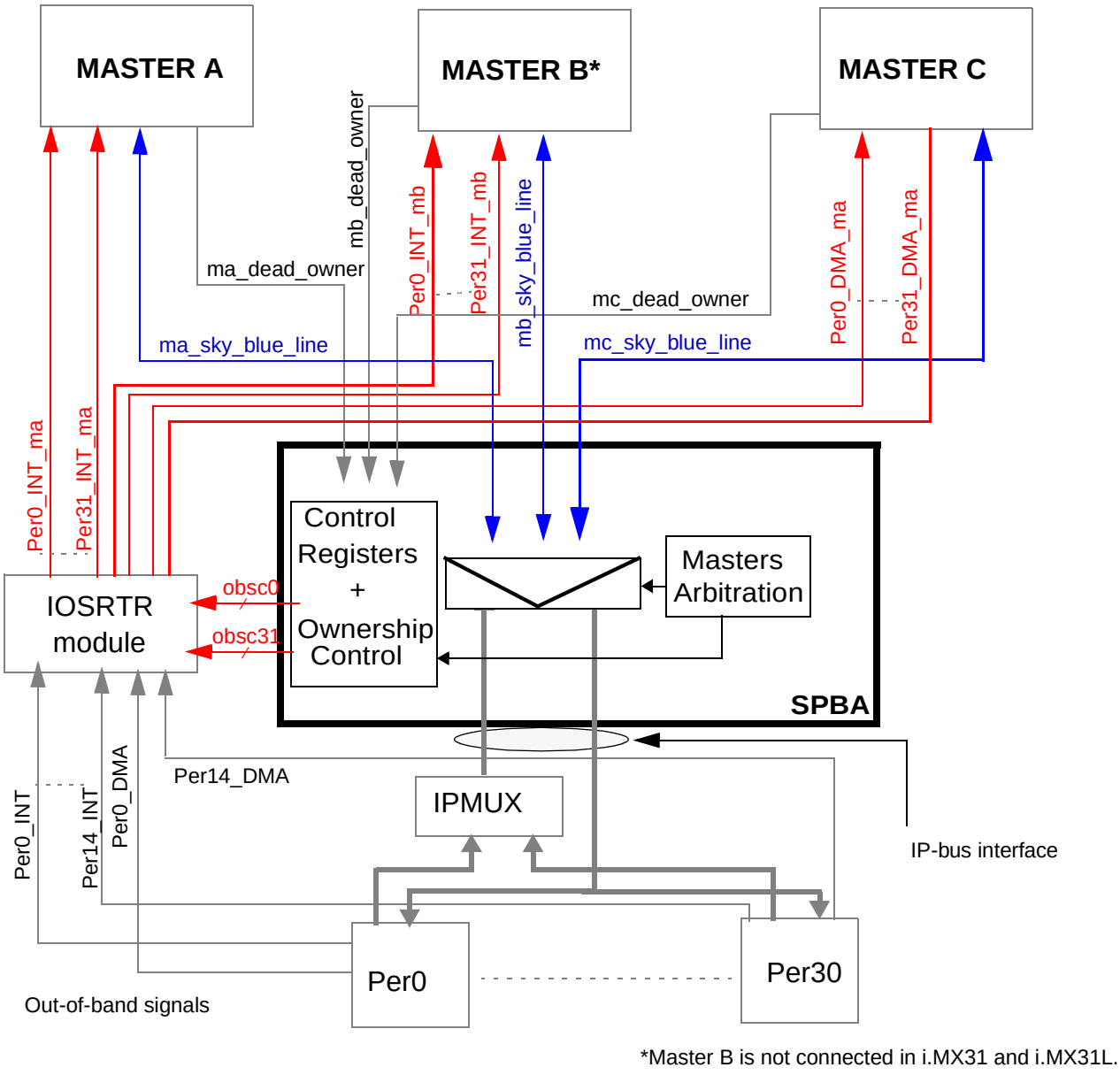


Figure 41-2. SPBA Block Diagram

41.1 Overview

The SPBA is a three-to-one IP SkyBlue interfaces (IP-Bus) arbiter. Three masters arbitration for shared peripherals access through the MODULE.

Three main parts are involved in SPBA, as follows:

- The SkyBlue Line switches a master to one peripheral.

- The Masters Arbitration arbitrates between the three masters to solve concurrent access or restricted access to peripherals.
- The Control Registers and Ownership Control including a set of registers, which are reachable through software, permits the access scheme definition to each peripheral (Resource Ownership and Access Control). It generates signals usable for the external steering logic of interrupts and DMA signals.

41.1.1 Features

The MODULE includes the following features:

- Three IP-Bus masters arbitration, master A (ARM11 Platform), master B (not connected in i.MX31 and i.MX31L) and master C (SDMA)
- 32-bits data
- Supports up to 31 shared peripherals, each consuming 16 Kbytes of address space
- SPBA can be considered as the 32th peripheral, used for resource ownership and access control mechanism to the 31 peripherals
- Provides 31 sets of Out of Band Steering Control (OBSC) signals to the off-module steering logic
- IP SkyBlue v3.0 compliant
- Operating frequency up to 67 MHz
- Clocks: ipg_clk, ipg_clk_s (AP clock domain)

41.1.2 Modes of Operation

Thus the MODULE behavior is transparent when accessing a peripheral, it can distinguish different modes of operation:

- Reset/Abort:

  The MODULE has a hardware reset which initialize all registers, arbitration and PRR (peripherals rights registers).

  Additionally, an abort signal input is provided allowing each master to abort their current access and release their ownership (in case of master reset sequence, and so on).
- Functional:

  Once a master request is granted, its IP-Bus signals are steering to the requested peripheral.
- Standby:

  No clock needed. The SPBA needs clocks only during: access to the PRRs, arbitration and abort phases. It generates two clock enable signals indicating when the clocks must be provided.
- Configuration:

  This is the phase where a master is accessing the SPBA PRRs. Indeed, SPBA memory mapped registers are seen as a shared peripheral.

41.2 Signal Description

41.2.1 Out-of-Band Steering Control

41.2.1.1 obsc(n)[4:0]

As the steering logic of out-of-band signals (for example, DMA requests and interrupts signals) is project and peripheral dependent, it is not part of the MODULE.

MODULE provides up-to 31 sets of Out-of-Band Steering Control signals to the In band Out band Steering logic module, these signals reflect the current ownership of the peripheral and the masters which have access to the peripheral. These values are respectively stored in the Resource Owner ID (ROI(n)[1:0]) and in the Register Access Right (RAR(n)[2:0]) in the Peripheral Right Register (Section 41.5.1, “Peripheral Right Register (PRRn)” for ROI and RAR values detailed).

Table 41-1 shows how ROI and RAR bits are available in the obsc(n)[4:0] bus.

Table 41-1. Out-Of Band Steering Control Signals -obsc(n)[4:0]

bit 4	bit 3	bit 2	bit 1	bit 0
ROI[1]	ROI[0]	RAR[2]	RAR[1]	RAR[0]

These signals will be handled at SoC level in the in band out band steering control module in order to drive the interrupt back to the appropriate(s) master(s) (value stored in RAR and/or ROI).

41.3 Memory Map and Register Definition

The following section describes the memory maps and detailed descriptions of all registers that are accessible within and through the SPBA.

41.3.1 Memory Map

Table 41-2 shows the SPBA memory map.

Table 41-2. SPBA Memory Map

Address	Register	Access	Reset Value	Section/Page
0x5003_Cn*4 (PRRn)	Peripheral Right Register (PRRn)	R/W	0x0000_0007	41.5.1/41-10

From a master side, the absolute address of one peripheral memory map registers is accessible by setting the <master>_ips_addr[24:0]:

- <master>_ips_addr[24:19] to the SPBA master base address (defined by top level parameters SPBA_<MASTER>_BASE_ADDR)
- <master>_ips_addr[18:14] to one of the 32 IPS peripherals (including SPBA). Refer to table Table 41-3.
- and <master>_ips_addr[13:0] to one of the 16 kbytes memory mapped registers (16 Kbytes only if accessing to a shared peripheral, not the same for SPBA PRR access)

Table 41-3 describes for each IPS peripheral, its absolute memory mapped address range accessible trough the SPBA for one master (does not include the <master>_ips_addr[24:19] bits).

Table 41-3. Peripherals Absolute Address

Peripheral	<master>_ips_addr[18:0] Start Address	<master>_ips_addr[18:0] End Address [1]
Not used	{5'b0_0000, 14'b00_0000_0000_0000}	{5'b0_0000, 14'b11_1111_1111_1111}
SDHC1	{5'b0_0001, 14'b00_0000_0000_0000}	{5'b0_0001, 14'b11_1111_1111_1111}
SDHC2	{5'b0_0010, 14'b00_0000_0000_0000}	{5'b0_0010, 14'b11_1111_1111_1111}
UART3	{5'b0_0011, 14'b00_0000_0000_0000}	{5'b0_0011, 14'b11_1111_1111_1111}
CSPI2	{5'b0_0100, 14'b00_0000_0000_0000}	{5'b0_0100, 14'b11_1111_1111_1111}
SSI2	{5'b0_0101, 14'b00_0000_0000_0000}	{5'b0_0101, 14'b11_1111_1111_1111}
SIM	{5'b0_0110, 14'b00_0000_0000_0000}	{5'b0_0110, 14'b11_1111_1111_1111}
IIM	{5'b0_0111, 14'b00_0000_0000_0000}	{5'b0_0111, 14'b11_1111_1111_1111}
ATA	{5'b0_1000, 14'b00_0000_0000_0000}	{5'b0_1000, 14'b11_1111_1111_1111}
Not used	{5'b0_1001, 14'b00_0000_0000_0000}	{5'b0_1001, 14'b11_1111_1111_1111}
Not used	{5'b0_1010, 14'b00_0000_0000_0000}	{5'b0_1010, 14'b11_1111_1111_1111}
Not used	{5'b0_1011, 14'b00_0000_0000_0000}	{5'b0_1011, 14'b11_1111_1111_1111}
Not used	{5'b0_1100, 14'b00_0000_0000_0000}	{5'b0_1100, 14'b11_1111_1111_1111}
Not used	{5'b0_1101, 14'b00_0000_0000_0000}	{5'b0_1101, 14'b11_1111_1111_1111}
Not used	{5'b0_1110, 14'b00_0000_0000_0000}	{5'b0_1110, 14'b11_1111_1111_1111}
SPBA	{5'b0_1111, 14'b00_0000_0000_0000}	{5'b0_1111, PRR addresses Table 41-5}
Not used	{5'b1_0000, 14'b00_0000_0000_0000}	{5'b1_0000, 14'b11_1111_1111_1111}
Not used	{5'b1_0001, 14'b00_0000_0000_0000}	{5'b1_0001, 14'b11_1111_1111_1111}
Not used	{5'b1_0010, 14'b00_0000_0000_0000}	{5'b1_0010, 14'b11_1111_1111_1111}
Not used	{5'b1_0011, 14'b00_0000_0000_0000}	{5'b1_0011, 14'b11_1111_1111_1111}
Not used	{5'b1_0100, 14'b00_0000_0000_0000}	{5'b1_0100, 14'b11_1111_1111_1111}
Not used	{5'b1_0101, 14'b00_0000_0000_0000}	{5'b1_0101, 14'b11_1111_1111_1111}
Not used	{5'b1_0110, 14'b00_0000_0000_0000}	{5'b1_0110, 14'b11_1111_1111_1111}
Not used	{5'b1_0111, 14'b00_0000_0000_0000}	{5'b1_0111, 14'b11_1111_1111_1111}
Not used	{5'b1_1000, 14'b00_0000_0000_0000}	{5'b1_1000, 14'b11_1111_1111_1111}

Table 41-3. Peripherals Absolute Address (continued)

Peripheral	<master>_ips_addr[18:0] Start Address	<master>_ips_addr[18:0] End Address [1]
Not used	{5'b1_1001, 14'b00_0000_0000_0000}	{5'b1_1001, 14'b11_1111_1111_1111}
Not used	{5'b1_1010, 14'b00_0000_0000_0000}	{5'b1_1010, 14'b11_1111_1111_1111}
Not used	{5'b1_1011, 14'b00_0000_0000_0000}	{5'b1_1011, 14'b11_1111_1111_1111}
Not used	{5'b1_1100, 14'b00_0000_0000_0000}	{5'b1_1100, 14'b11_1111_1111_1111}
Not used	{5'b1_1101, 14'b00_0000_0000_0000}	{5'b1_1101, 14'b11_1111_1111_1111}
Not used	{5'b1_1110, 14'b00_0000_0000_0000}	{5'b1_1110, 14'b11_1111_1111_1111}
Not used	{5'b1_1111, 14'b00_0000_0000_0000}	{5'b1_1111, 14'b11_1111_1111_1111}

1

41.3.2 Register Summary

See Figure 41-3 and Table 41-4 for guide to register figures.

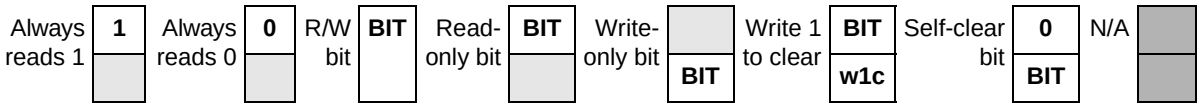


Figure 41-3. Key to Register Fields

Table 41-4. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.

Table 41-4. Register Figure Conventions (continued)

Convention	Description
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

41.4 SPBA Register Definition

The MODULE control registers (that is, Peripheral Right Registers) are mapped as a virtual shared peripheral using by default the sips_module_en[15] (or another one using Verilog parameter during integration phase) which must not be used for shared IPs.

MODULE can support up to 31 shared peripherals. Each of them has its own Peripheral Right Register (PRR) accessible within the SPBA memory mapped registers, and consists of the Requesting Master Owner, the Resource Owner ID and the Resource Access Right fields (using verilog parameters it can be defined which peripheral is present or not, and so if the corresponding PRR exists or not). Table 41-4 shows the summary of SPBA register.

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRRn[1] address = (0x5003_C[2] + n*4)	R	RMOn		0	0	0	0	0	0	0	0	0	0	0	0	ROIn	
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	RARn		
	W																

Figure 41-4. SPBA PRR Register Summary

1 for n=0 to 31

2 $BASE = {6'bSPBA_<MASTER>_BASE_ADDR, 5'bx_xxxx, 14'b00_0000_0000_0000}, where 5'bx_xxxx are <master>_ips_addr[18:14] reserved for SPBA registers access

Table 41-5 details the value of <master>_ips_addr[13:0] for master access any of the shared PRR (as explained above, <master>_ips_addr[24:19] must be SPBA_<MASTER>_BASE_ADDR and <master>_ips_addr[18:14] is the SPBA address value in Figure 41-4).

Table 41-5. PRR Addresses

Peripheral Rights Registers (SPBA Control Registers)	<master>_ips_addr[13:0] [1]	Comments
Not used	14'b00_0000_0000_0000	MODULE0_PRESENT = 1'b0 & MODULE0_IS_SPBA = 1'b0
SDHC1 Rights Registers	14'b00_0000_0000_0100	MODULE1_PRESENT = 1'b1 & MODULE1_IS_SPBA = 1'b0

Table 41-5. PRR Addresses (continued)

Peripheral Rights Registers (SPBA Control Registers)	<master>_ips_addr[13:0] [1]	Comments
SDHC2 Rights Registers	14'b00_0000_0000_1000	MODULE2_PRESENT = 1'b1 & MODULE2_IS_SPBA = 1'b0
UART3 Rights Registers	14'b00_0000_0000_1100	MODULE3_PRESENT = 1'b1 & MODULE3_IS_SPBA = 1'b0
CSPI2 Rights Registers	14'b00_0000_0001_0000	MODULE4_PRESENT = 1'b1 & MODULE4_IS_SPBA = 1'b0
SSI2 Rights Registers	14'b00_0000_0001_0100	MODULE5_PRESENT = 1'b1 & MODULE5_IS_SPBA = 1'b0
SIM Rights Registers	14'b00_0000_0001_1000	MODULE6_PRESENT = 1'b1 & MODULE6_IS_SPBA = 1'b0
IIM Rights Registers	14'b00_0000_0001_1100	MODULE7_PRESENT = 1'b1 & MODULE7_IS_SPBA = 1'b0
ATA Rights Registers	14'b00_0000_0010_0000	MODULE8_PRESENT = 1'b1 & MODULE8_IS_SPBA = 1'b0
Not used	14'b00_0000_0010_0100	MODULE9_PRESENT = 1'b0 & MODULE9_IS_SPBA = 1'b0
Not used	14'b00_0000_0010_1000	MODULE10_PRESENT = 1'b0 & MODULE10_IS_SPBA = 1'b0
Not used	14'b00_0000_0010_1100	MODULE11_PRESENT = 1'b0 & MODULE11_IS_SPBA = 1'b0
Not used	14'b00_0000_0011_0000	MODULE12_PRESENT = 1'b0 & MODULE12_IS_SPBA = 1'b0
Not used	14'b00_0000_0011_0100	MODULE13_PRESENT = 1'b0 & MODULE13_IS_SPBA = 1'b0
Not used	14'b00_0000_0011_1000	MODULE14_PRESENT = 1'b0 & MODULE14_IS_SPBA = 1'b0
Not used (module 15 is the SPBA)	14'b00_0000_0011_1100	MODULE15_PRESENT = 1'b0 & MODULE15_IS_SPBA = 1'b1
Not used	14'b00_0000_0100_0000	MODULE16_PRESENT = 1'b0 & MODULE16_IS_SPBA = 1'b0
Not used	14'b00_0000_0100_0100	MODULE17_PRESENT = 1'b0 & MODULE17_IS_SPBA = 1'b0
Not used	14'b00_0000_0100_1000	MODULE18_PRESENT = 1'b0 & MODULE18_IS_SPBA = 1'b0
Not used	14'b00_0000_0100_1100	MODULE19_PRESENT = 1'b0 & MODULE19_IS_SPBA = 1'b0

Table 41-5. PRR Addresses (continued)

Peripheral Rights Registers (SPBA Control Registers)	<master>_ips_addr[13:0] [1]	Comments
Not used	14'b00_0000_0101_0000	MODULE20_PRESENT = 1'b0 & MODULE20_IS_SPBA = 1'b0
Not used	14'b00_0000_0101_0100	MODULE21_PRESENT = 1'b0 & MODULE21_IS_SPBA = 1'b0
Not used	14'b00_0000_0101_1000	MODULE22_PRESENT = 1'b0 & MODULE22_IS_SPBA = 1'b0
Not used	14'b00_0000_0101_1100	MODULE23_PRESENT = 1'b0 & MODULE23_IS_SPBA = 1'b0
Not used	14'b00_0000_0110_0000	MODULE24_PRESENT = 1'b0 & MODULE23_IS_SPBA = 1'b0
Not used	14'b00_0000_0110_0100	MODULE25_PRESENT = 1'b0 & MODULE24_IS_SPBA = 1'b0
Not used	14'b00_0000_0110_1000	MODULE26_PRESENT = 1'b0 & MODULE26_IS_SPBA = 1'b0
Not used	14'b00_0000_0110_1100	MODULE27_PRESENT = 1'b0 & MODULE27_IS_SPBA = 1'b0
Not used	14'b00_0000_0111_0000	MODULE28_PRESENT = 1'b0 & MODULE28_IS_SPBA = 1'b0
Not used	14'b00_0000_0111_0100	MODULE29_PRESENT = 1'b0 & MODULE29_IS_SPBA = 1'b0
Not used	14'b00_0000_0111_1000	MODULE30_PRESENT = 1'b0 & MODULE30_IS_SPBA = 1'b0
Not used	14'b00_0000_0111_1100	MODULE31_PRESENT = 1'b0 & MODULE31_IS_SPBA = 1'b0

[1] Any accesses to SPBA inexistent location will generate a <master>_ips_xfr_err

41.5 Register Descriptions

41.5.1 Peripheral Right Register (PRRn)

Figure 41-5 shows the PRRn register and Table 41-3 shows its field descriptions.

0x5003_C[1]n*4 (PRRn)[2] Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RMOn		0	0	0	0	0	0	0	0	0	0	0	0	0	0
W															ROIn	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	RARn		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

1 0x5003_C = {6'bSPBA_MX_BASE_ADDR, 5'bx_xxxx, 14'b00_0000_0000_0000}, where 5'bx_xxxx are the bit [18:14] of master address bus reserved for SPBA access

2 For n=0 to 31

Figure 41-5. Peripheral Right Register Diagram

Table 41-6. Peripheral Right Register Field Descriptions

Field	Description
31–30 RMOn	Requesting Master Owner. This 2-bit register field indicates if the corresponding resource is owned by the requesting master or not. This register is reset to 2'b0 if ROI[1:0] = 2'b0. 00 resource un-owned 10 resource owned by another master 11 resource owned by requesting master
29–18	Reserved
17–16 ROIn	Resource Owner ID. This 2-bits register field indicates which master (one at a time) can access to the PRR for rights modification. This is a Read-Only register. 00 resource un-owned 01 resource owned by master A port 10 resource owned by master B port 11 resource owned by master C port After reset, ROI bits are cleared ("00" -> un-owned resource). A master performing a write access to the an un-owned PRR will get its ID automatically written into ROI, while modifying RAR bits. It can then read back the RMO, RAR, ROI bits to make sure RMO returns the right value, ROI bits contain its ID and RAR bits are correctly asserted. Then no other master (whom ID is different from the one stored in ROI) will be able to modify RAR fields. Owner master of a peripheral can assert its dead_owner signal, or write 3'b0 in the RAR to release the ownership (ROI[1:0] reset to 2'b0).

Table 41-6. Peripheral Right Register Field Descriptions (continued)

Field	Description
15–3	Reserved
2–0 RARn	Resource Access Right. This 3-bit register field indicates which master can access to the peripheral. From 0 up to 3 masters can have permission to access a resource (all the master can be granted on a peripheral, but only one access at a time will be granted by SPBA). RARn[0] — Control and Status bit for master A which is connected to port MA, 0 Access to peripheral is not allowed. 1 Access to peripheral is granted. RARn[1] — Control and Status bit for master B which is connected to port MB, 0 Access to peripheral is not allowed. 1 Access to peripheral is granted. RARn[2] — Control and Status bit for master C which is connected to port MC, 0 Access to peripheral is not allowed. 1 Access to peripheral is granted. After reset, all RAR registers are set to value 1. So by default all masters have access granted to all the shared peripherals. This is verified until one master modify this default value.

41.6 Functional Description

The following section intends to describe the main features of the MODULE module.

41.7 Masters Arbitration

The arbitration mechanism determines which port will control the master port, based on a simple round-robin arbitration scheme.

We can distinguish between different cases:

- Only one master request per different access. So the master is switched to the shared peripheral bus, without arbitration. Figure 41-7 shows the MB request on the global module enable signal, served without wait state.
- If two masters simultaneously access to SPBA, then the last granted master is held-off, using <master>_ips_xfr_wait output signal (default value is high). When the master is granted sips_xfr_wait from shared IPS peripheral is connected to <master>_ips_xfr_wait outputs.
- If three masters simultaneously access to SPBA, then the last two granted masters are held-off, using their <master>_ips_xfr_wait signal. Figure 41-8 shows the case when the last two accesses granted are MA then MB, and how the requests are used even if they are in the same cycle.
- After reset, at the first multiple access, no master has been granted, thus, the priority is static: Master A (MA), Master B (MB) and last Master C (MC) port.
- No master request. No more master switch to shared peripherals.

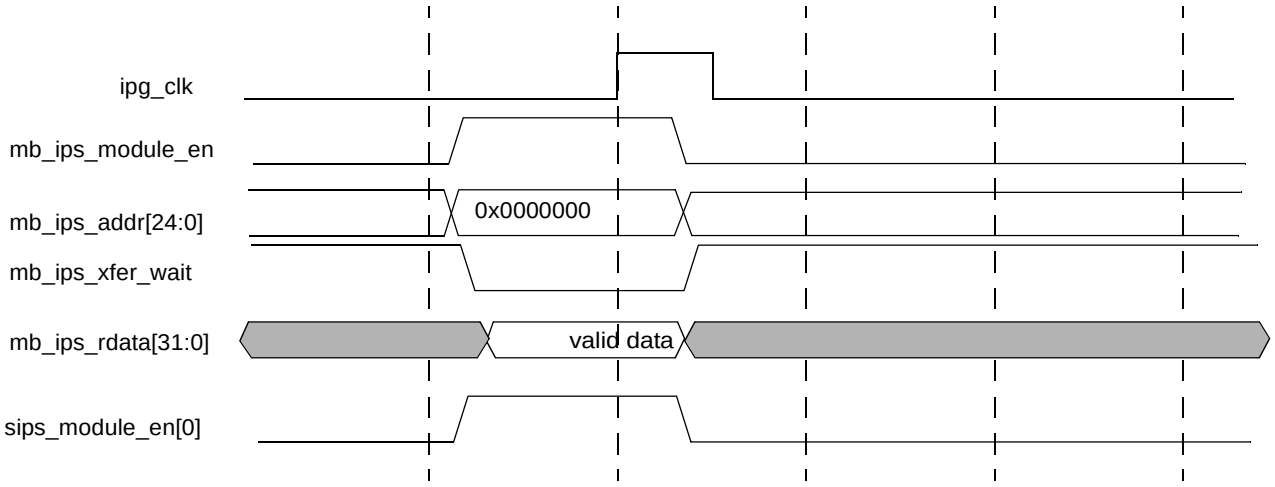


Figure 41-7. Example of One Master Request: no MODULE Arbitration

Figure 41-8 assumes MA and MB have been the last two masters already granted in the previous transfers (MA then MB).

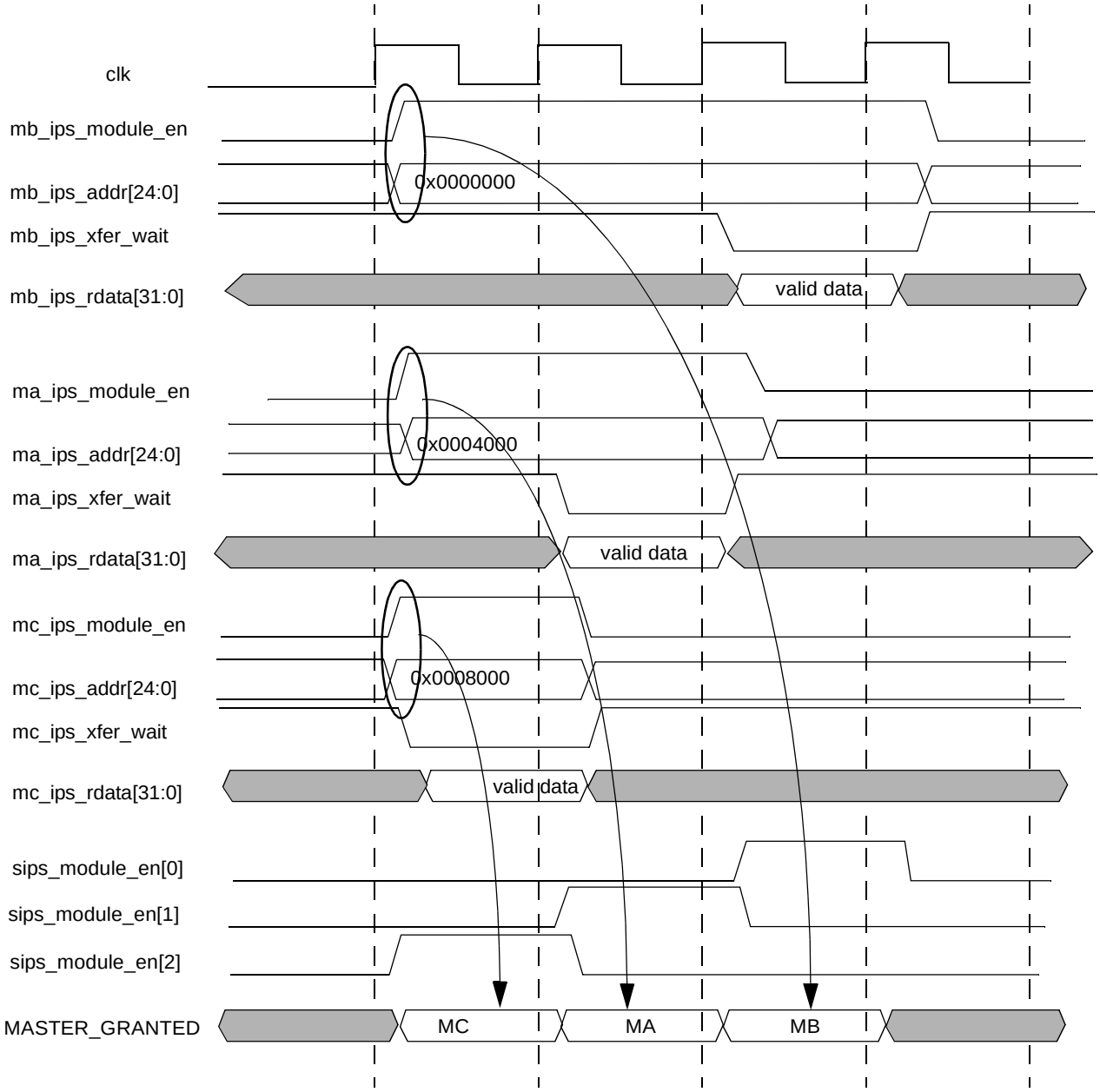


Figure 41-8. ExamplE Of Three Master Requests: Masters Already Granted Are “Waited”

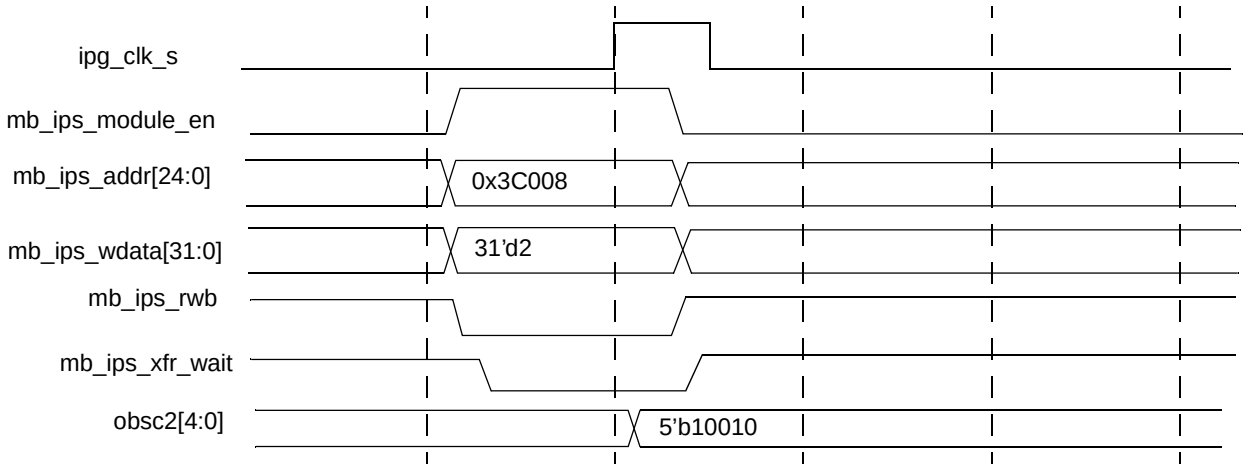
41.8 Resource Ownership Control

The Resource Ownership Control directs accesses to the shared peripherals and determines steering of out-of-band signals.

41.8.1 Access Control

41.8.1.1 Peripheral Access

The peripheral access (a.k.a resource access) of the requesting master is given by the corresponding RAR bit of the Peripheral Right Register. It determines if the master has access privilege to the resource. Any attempted access to a resource by a requesting master whose access privilege bit is not set (in the PRR), is terminated with a bus error (<master>_ips_xfr_err is asserted by SPBA logic). The master which owns the resource can lock the peripheral for itself and/or grant other masters access to the peripheral by setting the appropriate bit(s) in the RAR field. See Figure 41-9.



Master B is taking ownership of peripheral 2 by writing 3'b010 in the SPBA peripheral 2 right register (rar field)
This ownership can be checked on obsc2 output as roi2[1:0] = 2'b10 and rar2[2:0] = 3'b010
(obsc2[4:0] = {roi2[1], roi2[0], rar2[2], rar2[1], rar2[0]})

Figure 41-9. Example of One Master B Gaining Ownership Of Peripheral 2

41.8.1.2 Peripheral Right Register Access

The ROI bits of the Peripheral Right Register (PRR) determine which master is allowed to make write access to PRR. The identification of the requesting master is compared to the ROI bits of the Peripheral Right Register to determine if the master has ownership of the corresponding register.

Any attempted write access to a Peripheral Right Register (PRR) already owned by another master will be ignored.

41.8.2 Owner Election

When the peripheral is not owned by any master (for example, ROI= "00", such as after coming out of reset), the first master to perform successfully a write to the RAR bits of the PRR is granted ownership of the peripheral and its associated PRR.

After writing to the PRR (for example, RAR bit(s)), the master must read it back to make sure that it was actually granted ownership.

If the RMO field is 2’b11, then the ownership claim is successful. If RMO is 2’b10, then another master claimed ownership before this master was able to complete its write.
This resolves the case in which two or more masters attempt to write the PRR at the same time; only the first master will be granted ownership. However all masters must read the PRR to determine if this case occurred, and if so, whether they were the first master which was actually granted ownership. Refer to Section 41.4, “SPBA Register Definition” for more detail on the fields of the PRR.

NOTE

A master which has been granted ownership of the PRR does not automatically have the right access to the peripheral; it must still set its own RAR bits in the PRR to access the peripheral.

41.8.3 Ending Ownership

Ownership may be voluntarily ended by the owning master, or automatically upon assertion of a master-specific dead_owner signal. The former is appropriate for software-controlled yielding of ownership. The latter is appropriate for automatic yielding of ownership when the owner has gone into reset.

41.8.3.1 Hardware Controlled Ownership Ending

When a <master>_dead_owner signal is asserted, it clears the ROI bits of the PRRs owned by the corresponding master.

When the owner is dead (for example, owner is in reset), all peripherals previously owned by that master must be transitioned to the un-owned state.

NOTE

It is the programmer’s responsibility to make sure the peripherals are placed in an appropriate state before ending ownership.

41.8.3.2 Software Controlled Ownership Ending

The ROI bits will be automatically cleared when the master which owns the PRR access right clears (write) the RAR bits. (Figure 41-5). It will then end the ownership of the PRR.

41.8.4 The Un-Owned State

During the time when the peripheral is un-owned (for example, the ROI field contains all 0’s), all masters have full access to it (RAR bits can then be modified by a master if ROI[1:0] = 2’b0). In such cases it is necessary for software to ensure any necessary coherency in the resource, there is no hardware protection.

41.9 IP-Multiplexing

Since there are multiple sips_xfr_rdata (sips_xfr_wait and sips_xfr_error) signal coming from the shared peripherals and only one sips_xfr_rdata (sips_xfr_wait and sips_xfr_error) input to the SPBA, the SoC

integrator must externally multiplex all shared peripheral sips_xfr_rdata (sips_xfr_wait and sips_xfr_error) signals based on the sips_module_en outputs of the SPBA.

41.10 Clock Usage

41.10.1 SPBA Clocks

Figure 41-10 provides the high-level block diagram of the SPBA clocking. Two clocks are used within the SPBA:

- ipg_clk used for non RW registers.This clock is gated using the ipg_clk_enable signal, ipg_clk_enable is activated when a master access (IP-Bus and abort signals) is detected by SPBA internal logic.
- ipg_clk_s used for SPBA RW registers. This clock is already gated in the LPG module using the AIPI interface. Moreover, SPBA provides the sips_module_en[15] signal to gate the ipg_clk_s. This ensures that the clock will not toggle during periods of SPBA registers (PRR) inactivity.

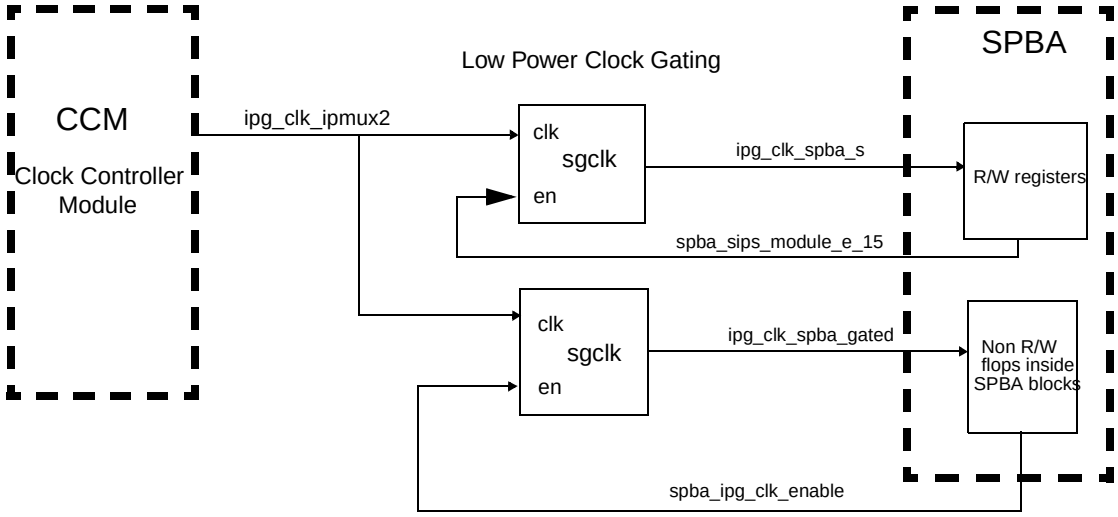


Figure 41-10. High-Level Block Diagram of the SPBA Clocking

NOTE

All shared peripherals and SPBA are running on the same AP clock domain, however all the shared peripherals have their own clocks and their own clock gating circuitry.

41.10.2 SPBA and Synchronization

The SPBA operates off of a single clock domain, and as such doesn't handle any synchronization of data, address and control signals. Such synchronization, where required is done by a dedicated synchronization module, IPSYNC in i.MX31.

41.11 Reset Usage

The hard reset (ipg_hard_async_reset) is directly connected to all SPBA registers:

- It clears to value 0 the registers used for round robin arbitration storing the two last masters granted, so that after HW reset we have the default priority master A (AP), master B (DSP, not connected in i.MX31 and i.MX31L), and then master C (SDMA).
- It clears to value 0 all the ROI memory mapped registers, so that after HW reset all the shared peripherals are unowned.
- It sets to value 1 all the RAR memory mapped registers, so that after HW reset all the shared peripherals are accessible by default to the three masters.

Book II, Part 8: Multimedia Peripherals

Introduction

Digital Audio MUX (AUDMUX)

The Digital Audio MUX (AUDMUX) provides a programmable interconnect fabric for voice, audio, and synchronous data routing between host serial interfaces (for example, SSI or SAP) and peripheral serial interfaces (for example, audio and voice CODECs). The AUDMUX allows the audio system connectivity to be modified through programming (as opposed to altering the PCB schematics of the system). The Digital Audio MUX is configured by software.

With the AUDMUX, resources do not need to be hard-wired and can be effectively shared in different configurations. The AUDMUX interconnections allow multiple, simultaneous audio/voice/data flows between the ports in point-to-point or point-to-multipoint configurations.

The AUDMUX includes two types of interfaces. Internal ports connect to the processor serial interfaces and external ports connect to off-chip audio devices and serial interfaces of other processors. A desired connectivity is achieved by configuring the appropriate internal and external ports.

MPEG-4 Encoder

The MPEG-4 encoder in the i.MX31 and i.MX31L accelerates video compression, in compliance with the MPEG-4 standard. The encoder provides several levels of compression formats including MPEG-4 simple profile (all levels) and H.263 baseline. The encoder can encode at a pixel rate up to VGA @ 30 fps and compressed bit-rates up to 4 Mbps. The MPEG-4 encoder provides what is essentially the complete video processing chain, generating a Huffman-coded stream with the exception of the formation of the final MPEG-4 stream which is the only burden put on the ARM11 processor. Additional processing provided by the MPEG-4 encoder includes picture smoothening (low-pass filter) and camera movement stabilization. Support for enhanced conference call format in the form of additional information inserted within the MPEG stream, used by a MPEG-4 decoder to improve performance.

Image Processing Unit (IPU)

The IPU is designed to support video and graphics processing functions in the i.MX31 and i.MX31L, and to interface to video/still image sensors and displays. The IPU can capture image data from a camera sensor or from a TV decoder. The captured image can be sent to pre-processing or stored in an external system memory for additional processing on the ARM11 Platform. Pre-processing of data can be programmed from the sensor or from the external system memory. There are two pre-processing channels determined by the data destination—an encoder or a display (viewfinder mode). Pre-processing includes downsizing with independent integer horizontal and vertical ratios, resizing with independent fractional horizontal and vertical ratios, color space conversion, combining a video plane with a graphics plane (blending on graphics on top of video plane).

Synchronous Serial Interface (SSI)

The SSI is a full-duplex, serial port that allows the chip to communicate with a variety of serial devices. These serial devices can be standard CODECs, Digital Signal Processors (DSPs), microprocessors, peripherals, and popular industry audio CODECs that implement the inter-IC sound bus standard (I2S) and Intel AC97 standard.

SSI is typically used to transfer samples in a periodic manner. The SSI consists of independent transmitter and receiver sections with independent clock generation and frame synchronization.

The SSI contains independent (asynchronous) or shared (synchronous) transmit and receive sections with separate or shared internal/external clocks and frame syncs, operating in Master or Slave mode. The SSI can work in normal mode operation using frame sync and in Network mode operation allowing multiple devices to share the port with as many as thirty-two time slots. The SSI provides 2 sets of Transmit and Receive FIFOs. Each of the four FIFOs is 8x24 bits. The two sets of Tx/Rx FIFOs can be used in Network mode to provide 2 independent channels for transmission and reception. It also has programmable data interface modes such like I2S, LSB, MSB aligned and programmable word lengths. Other program options include frame sync and clock generation and programmable I2S modes (Master, Slave or Normal). Oversampling clock, ccm_ssi_clk available as output from SRCK in I2S Master mode.

In addition to AC97 support the SSI has completely separate clock and frame sync selections for the receive and transmit sections. In AC97 standard, the clock is taken from an external source and frame sync is generated internally. the SSI also has a programmable internal clock divider and Time Slot Mask.

Graphics Accelerator (MBX R-S)

NOTE

The MBX R-S graphics accelerator is not available in the i.MX31L.

Chapter 46, "Graphics Accelerator (MBX R-S)," contains third-party content only, and is provided by—and used with permission from—ARM Ltd. and Imagination Technologies Ltd.

The MBX R-S 3D Graphics Core is an *Advanced Microcontroller Bus Architecture* (AMBA) compliant *System-on-a-Chip* (SoC) component.

Chapter 42
Digital Audio Multiplexer (AUDMUX)

The Digital Audio Multiplexer (AUDMUX) provides a programmable interconnect device for voice, audio, and synchronous data routing between host serial interfaces (SSI) and peripheral serial interfaces (that is, audio and voice coder-decoders, also known as CODECs).

42.1 Overview

With the AUDMUX, resources do not need to be hard-wired and can be effectively shared in different configurations. The AUDMUX interconnections enable multiple, simultaneous audio/voice/data flows between the ports in point-to-point or point-to-multipoint configurations.

The AUDMUX includes two types of interfaces. Internal ports connect to the processor serial interfaces and external ports connect to off-chip audio devices and serial interfaces of other processors. A desired connectivity is achieved by configuring the appropriate internal and external ports.

The AUDMUX allows the audio system connectivity to be modified through programming (as opposed to altering the PCB schematics of the system). Figure 42-1 and Figure 42-2 show the block diagram of the AUDMUX (Figure 42-2 is an extension of Figure 42-1).

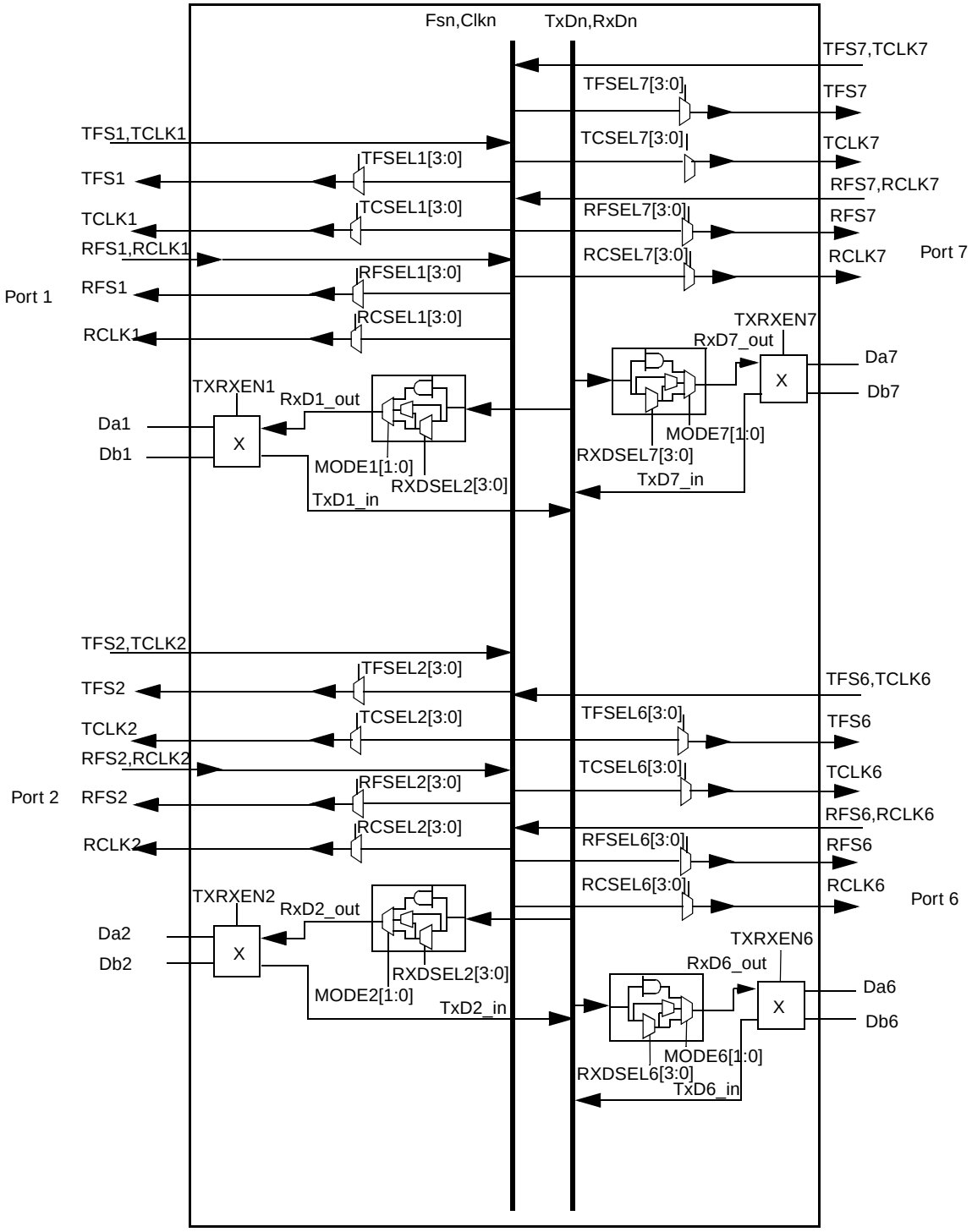


Figure 42-1. AUDMUX Block Diagram A

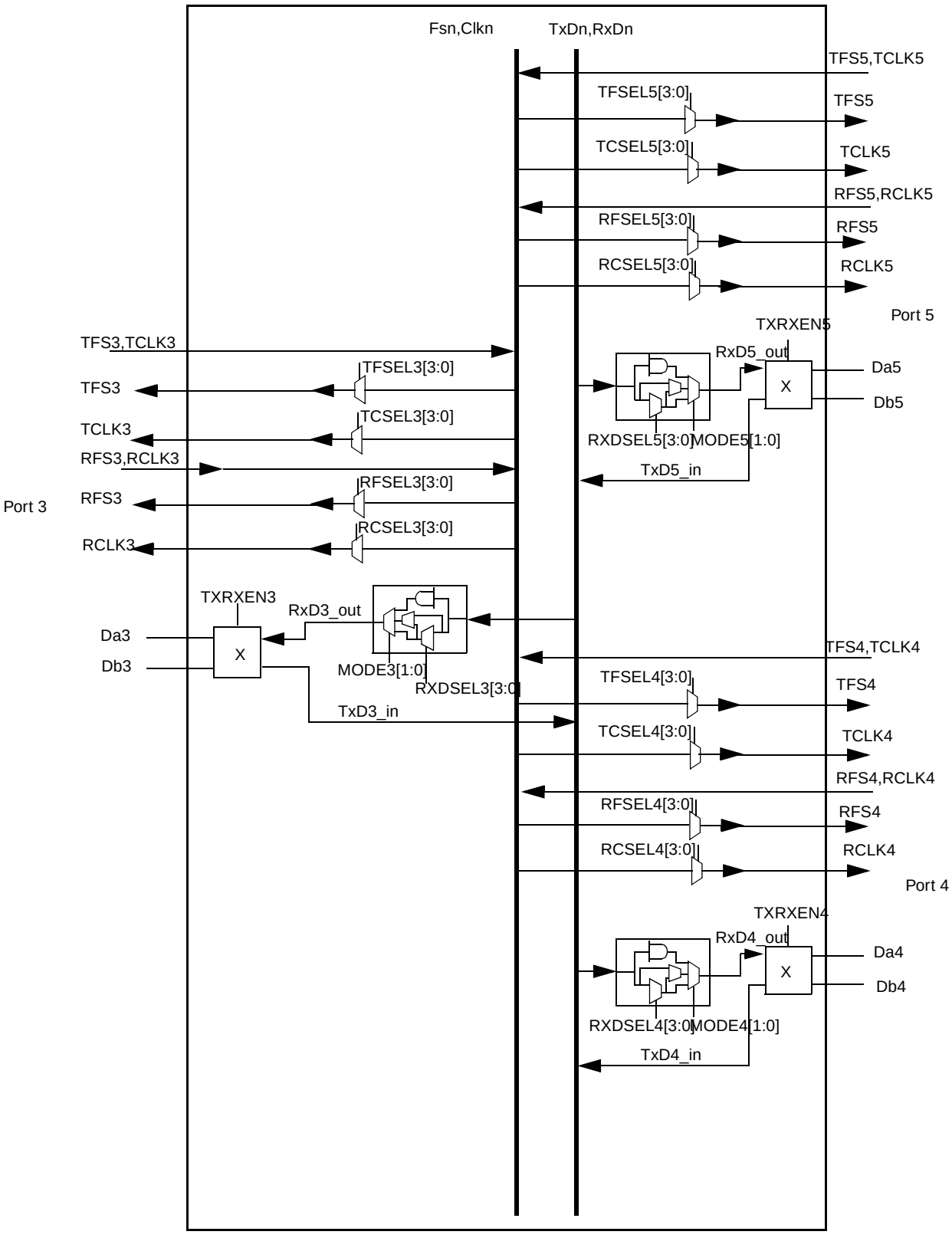


Figure 42-2. AUDMUX Block Diagram B

42.1.1 Features

The following are the AUDMUX features:

- Two internal ports (ports 1 and 2)
- Five external ports (port 3 through 7 inclusive)
- Full 6-wire SSI interfaces for asynchronous receive and transmit
- Configurable 4-wire (synchronous) or 6-wire (asynchronous) peripheral interfaces
- Independent Tx/Rx Frame sync and clock direction selection for host or peripheral
- Each host interface's capability to connect to any other host or peripheral interface in a point-to-point or point-to-multipoint (network mode)
- Transmit and Receive Data switching to support external network mode

42.1.2 Port Descriptions

All the ports are basically identical. The major difference is whether a port is connected to an on-chip serial interface (for example, SSI) or connected through the external contacts allowing connection to off-chip serial devices (that is, any 4-wire or 6-wire external SSI, voice, I2S, or AC97 CODEC).

Port 7 can be physically connected to any of the peripherals previously mentioned. Port 7 pins provide additional multiplexing to allow a set of signals (JTAG, UART, or audio) to be connected to a single connector. It also provides additional functionality that is explained in the description of the bottom connector mode in the next section.

There is no functional difference between Ports 1 through 7. Internal ports 1 and 2 are connected internally to SSI-1 and SSI-2, respectively.

Each of the ports can be configured as either four- or six-wire interfaces. When configured as a six-wire interface, the additional RFS and RCLK signals of the interface enable a serial interface to be used in asynchronous mode with separate receive and transmit clocks.

All ports have a Tx/Rx switch to provide flexibility for supporting network mode configurations. The Tx/Rx switch enables the transmit and receive data lines to be swapped so mastership of the serial bus can be passed among multiple external devices connected to a single port.

42.1.3 Network Modes

In addition to supporting the default external network mode, each port supports two special types of network modes—internal network mode and bottom connector network mode. Internal network mode is a point-to-multipoint network configuration that supports an arbitrary number of slaves. The external slaves must be put into the high-impedance state (as defined in the SSI network mode protocol) and have pull-up resistors on their TxD pins. (Alternatively, this can be viewed as requiring a pull-up resistor on the corresponding AUDMUX RxD pin.) The bottom connector mode is a point-to-multipoint network that can support two slaves. The slaves do not have to put the TxD line into high-impedance state, and they do not require any pull-up resistors.

Bit clock direction selection enables each port to be configured as a master or slave in the flow.

Possible scenarios include the following:

- SSI1 (internal port) drives a voice CODEC and a BT CODEC (both on external Port 6) and the Bottom Connector (on external Port 7) simultaneously using network mode. SSI1 is the master.
- An external processor (external Port 5) drives a voice CODEC and a BT CODEC (both on external Port 6) and the Bottom Connector (on Port 7) simultaneously using network mode. SAP is the master.

NOTE

SSI1 (internal port), which is the master, drives a voice CODEC and a BT CODEC (both on external Port 6) and the bottom connector (on external Port 7) simultaneously using network mode.

An external processor (external Port 5), which is the master, drives a voice CODEC and a BT CODEC (both on external Port 6) and the bottom connector (on Port 7) simultaneously using network mode.

42.1.3.1 Port Receive Data Modes

Each port has logic to select which data lines are used to create the RxD line for the corresponding host interface. Figure 42-3 shows the logic used to create the RxD line for Port 1.

This logic has the following modes of operation (as determined by MODE[1:0]):

- Normal
- Bottom Connector network mode
- Internal network mode

The subsequent sections describe the various modes of the port receive data logic.

The following terms are used to define the operation of the AUDMUX:

Network mode—Time-Division Multiplexed protocol for sending unique data to multiple devices on a serial bus.

Internal network mode—Physical bus configuration where multiple serial buses are effectively connected within the AUDMUX via digital logic to create point-to-multipoint connectivity. An arbitrary number of devices are supported. Devices must be put into the high-impedance state as specified by the network mode protocol. TxDATA lines of devices must be pulled high.

External network mode—Physical bus configuration where multiple serial buses are electrically connected together on a printed circuit board (that is, external to the AUDMUX). Devices must put their TxDATA lines into the high-impedance state as specified by the network mode protocol.

Bottom Connector network mode—Physical bus configuration where multiple serial buses are effectively connected within the AUDMUX via digital logic to create point-to-multipoint connectivity. This mode can only utilize three AUDMUX ports simultaneously. One of the ports must be Port 7. Devices do not have to be put into the high-impedance state, nor do they require pull-up resistors on their TxDATA pins.

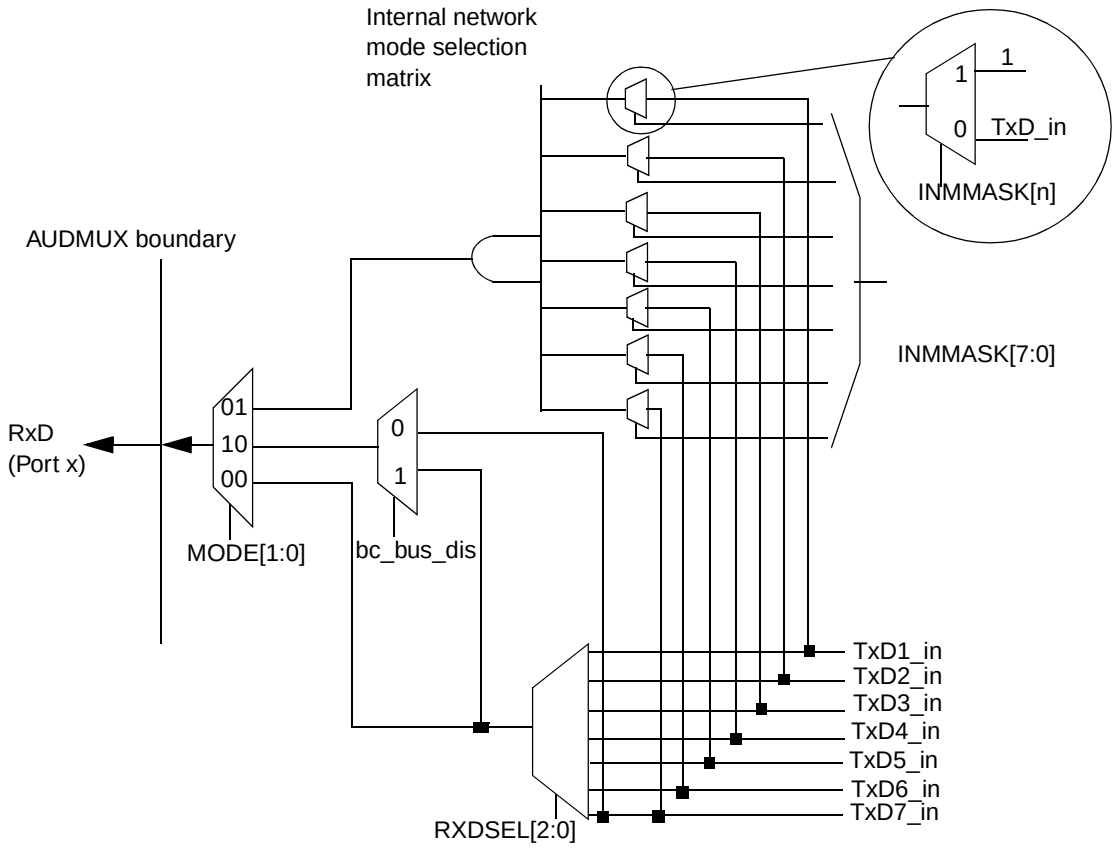


Figure 42-3. Receive Data Logic for Port x

42.1.3.1.1 Normal Mode

In normal mode (MODE[1:0]=00), the port is connected in a point-to-point configuration (as a master or a slave) and the RXDSEL[2:0] setting selects the transmit signal from any port. In normal mode, any data format can be used (that is, SSI normal mode, SSI network mode, AC-97, and others.

42.1.3.1.2 Internal Network Mode

In internal network mode (MODE[1:0] = 01), the output of the AND gate is routed (via the output of the port) to the RxD signal of the corresponding host interface. The INMMASK bit vector selects the transmit signals of the ports that are to be connected in network mode. The transmit signals received at the AUDMUX ports (TxDn_in) are AND'ed together to form the output. In internal network mode, only one device can be transmitting in its predesignated timeslot and all other transmit signals must remain high (be in high-impedance state and pulled-up). Therefore, non-active signals in the selection will be high and do not influence the output of the AND gate.

Network mode is a protocol where a master SSI is connected to more than one slave SSI device and communication occurs on a time-slotted frame. Though network mode can allow master-slave and slave-slave communication, internal network mode supports only master-slave communication.

There are two scenarios where internal network mode can be used with external network mode:

1. Slave-only devices are attached to an external port.
2. A master device is attached to an external port and all slave devices connected to the same external port are disabled.

NOTE

When internal network mode is enabled at an external port, RXDSEL[3:0] selection is ignored and the RXD output buffer is enabled for all the time slots. All slave devices connected to the same port must be disabled.

Internal Network Mode Example 1

SSI1 and SSI2 are used with Port 4 in internal network mode as shown in Figure 42-4. No pull-up resistors are required, since all the interfaces combined in internal network mode are on-chip interfaces. The on-chip interfaces drive a logic ‘1’ when their output enables are logic ‘0’.

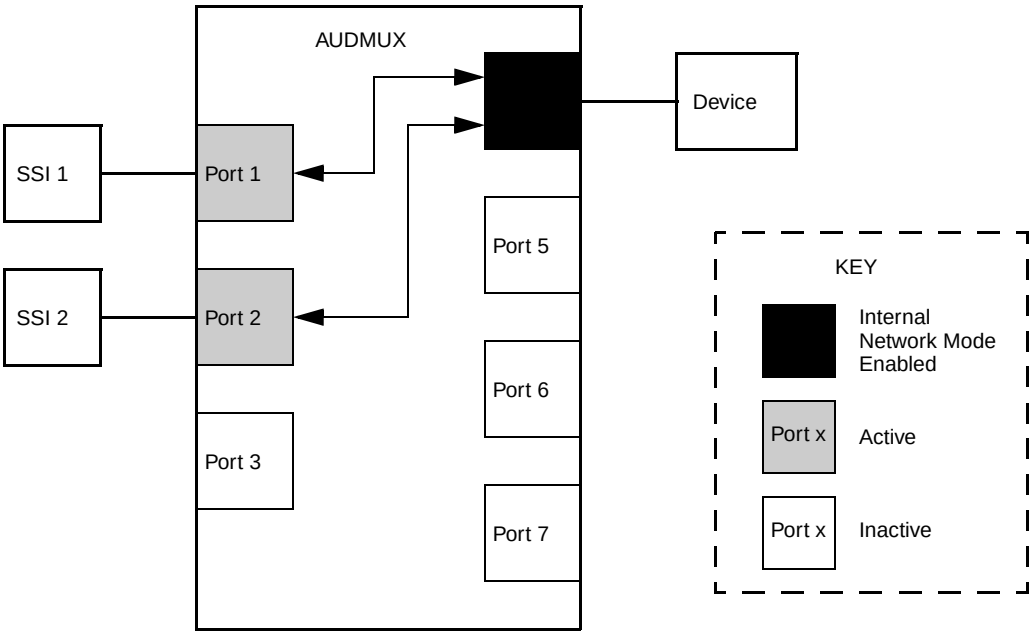


Figure 42-4. Block Diagram For Example 1

See Figure 42-5 for the timing diagram of Example 1. The clock and frame sync signals show the bit and frame timing for the serial bus. The vertical dashed lines divide the frame into four timeslots.

The data lines for SSI1 and SSI2 (as well as their output enables) are shown. Note that the SSI transmits a logic ‘1’ when its corresponding output enable is a logic ‘0’. The combined TxDATA line, which is the logical AND of SSI1 and SSI2’s TxDATA lines, is used for Port 4’s TxDATA line.

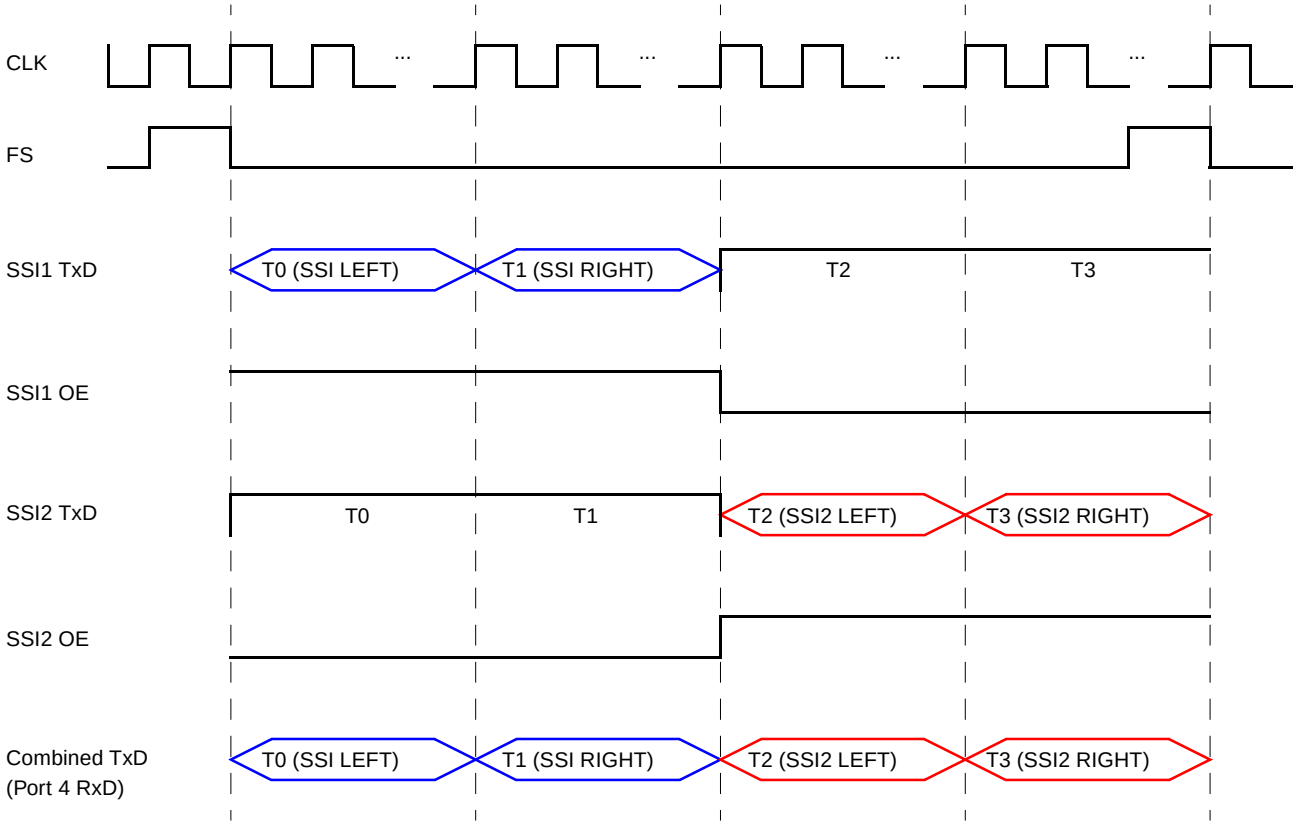


Figure 42-5. Example Using All Internal Ports For Transmit Data

Internal Network Mode Example 2

The SSI, Port 4, and Port 5 are used with Port 6 in internal network mode, as shown in Figure 42-6. Note that Port 4 and Port 5 are external ports. Therefore, pull-up resistors are required on the Port 4 RxDATA and Port 5 RxDATA pins. This example shows the timing associated with using adjacent timeslots for the SSI, Port 4, and Port 5.

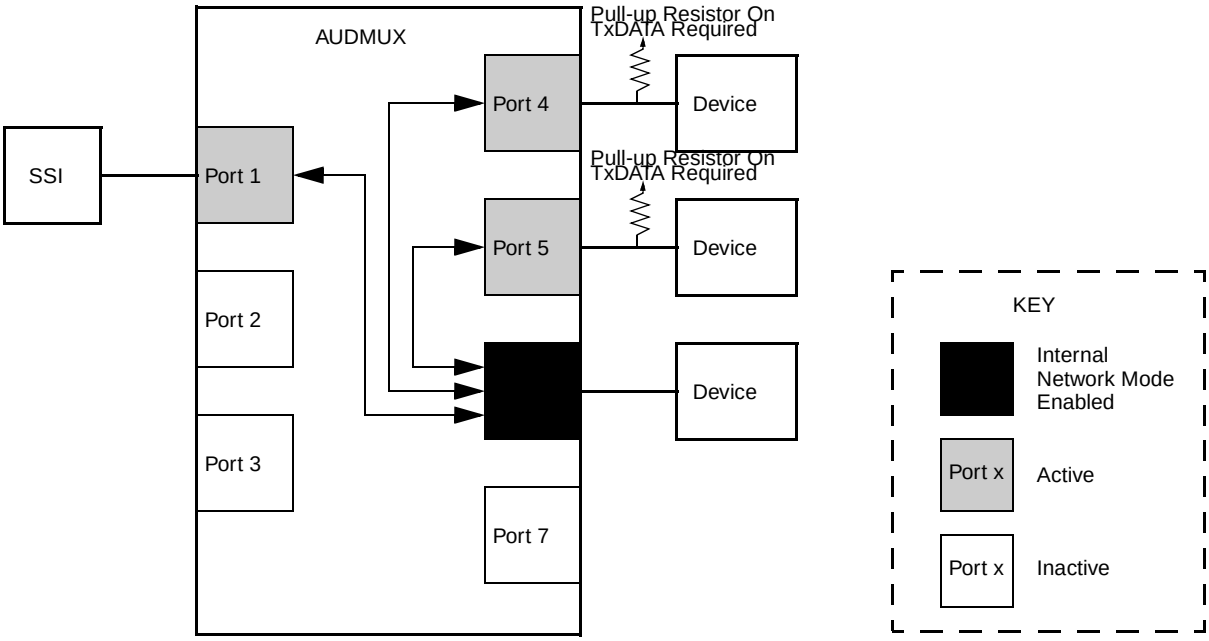


Figure 42-6. Block Diagram For Example 2

The resistance value of the pull-up resistors must be sufficiently high such that a value of ‘0’ can be pulled up to logic ‘1’ within half of a period of the bitclock. The required resistance must be no larger than:

Rmax = 1 / (2 * f_{bc} * C) where:

f_{bc} is the frequency of the bitclock

C is the total system capacitance (ICs, board traces, and so on)

Figure 42-7 shows the timing diagram for this example. The clock and frame sync signals show the bit and frame timing for the serial bus. The vertical dashed lines divide the frame into four timeslots.

The data lines for the SSI, Port 4, and Port 5 are shown. Note that the SSI transmits a logic ‘1’ when its corresponding output enable is a logic ‘0’. The data lines from Port 4 and Port 5 *at the pad* are pulled high by pull-up resistors when they are in the high-impedance state. The data lines from Port 4 and Port 5 *at the AUDMUX* are pure digital signals and are constantly driven. The combined TxDATA line, which is the logical AND of the SSI, Port 4, and Port 5’s TxDATA lines, is used for Port 6’s TxDATA line.

Note the highlighted areas in Figure 42-7. This shows the transition time that occurs while a TxDATA line is being pulled high. In this example, this transition time is a maximum of 1/2 the period of the serial bit clock. This prevents corruption of the first data bit of the next timeslot. It is critical that the pull-up resistance is sufficient for the given bit clock frequency and system capacitance.

Note that hysteresis should be enabled at Port 4’s RxDATA pad and Port 5’s RxDATA pad to prevent the digital signals created by the pad from toggling rapidly during the pull-up period. The pads typically require a transition within 25ns unless hysteresis is enabled. Instead of using hysteresis, one could select a pull-up resistor sufficiently high to pull-up the signal at the pad within 25 ns; however, that would result in a higher resistance value and higher current drain.

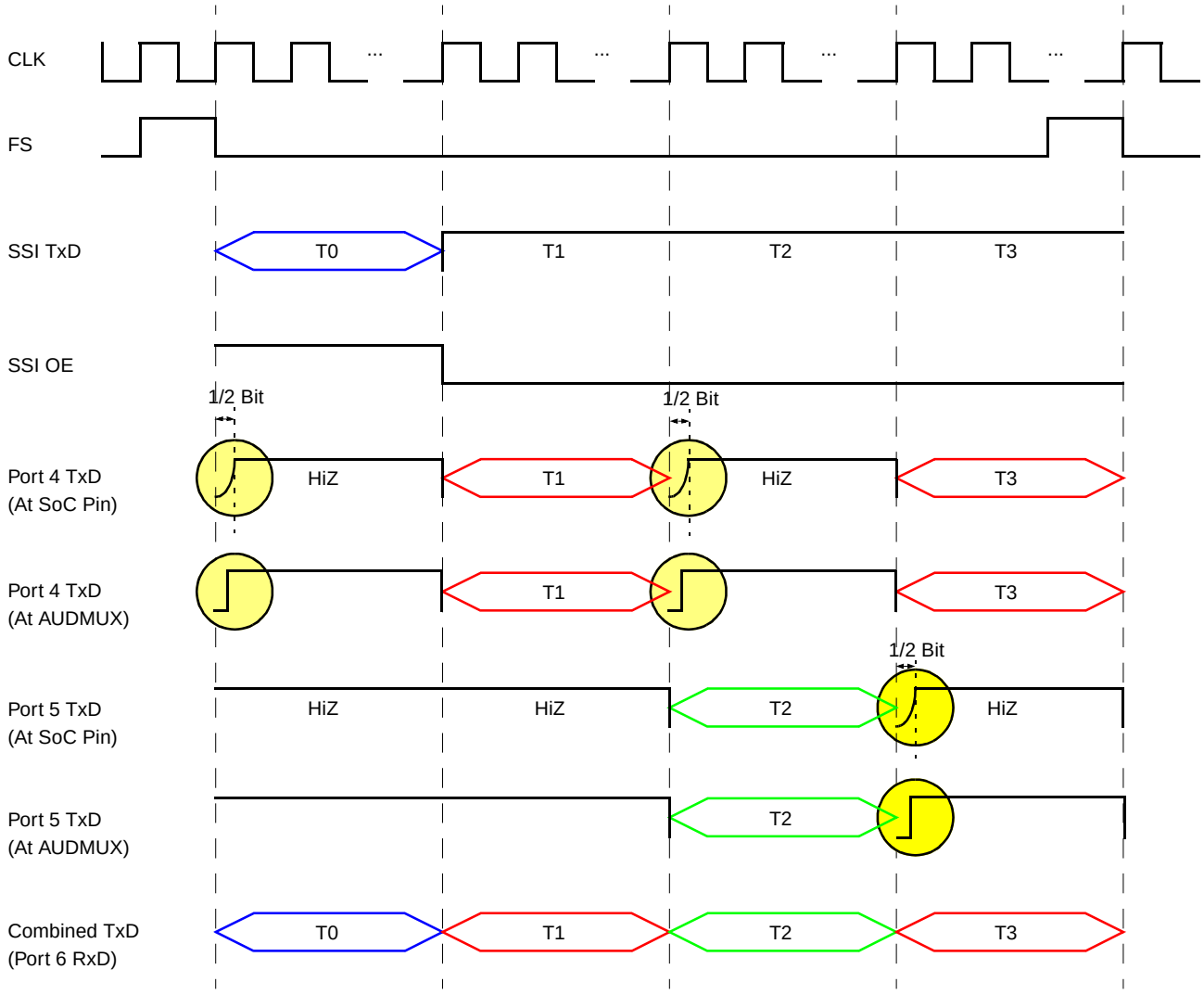


Figure 42-7. Example Using External Ports for Transmit Data in Consecutive Timeslots

Internal Network Mode Example 3

The SSI and Port 4 are used with Port 6 in internal network mode as shown in Figure 42-8. Note that Port 4 is an external port. Therefore, a pull-up resistor is required on the Port 4 TxDATA pin. This example shows the timing associated with inserting empty timeslots after the timeslots have been used by external ports.

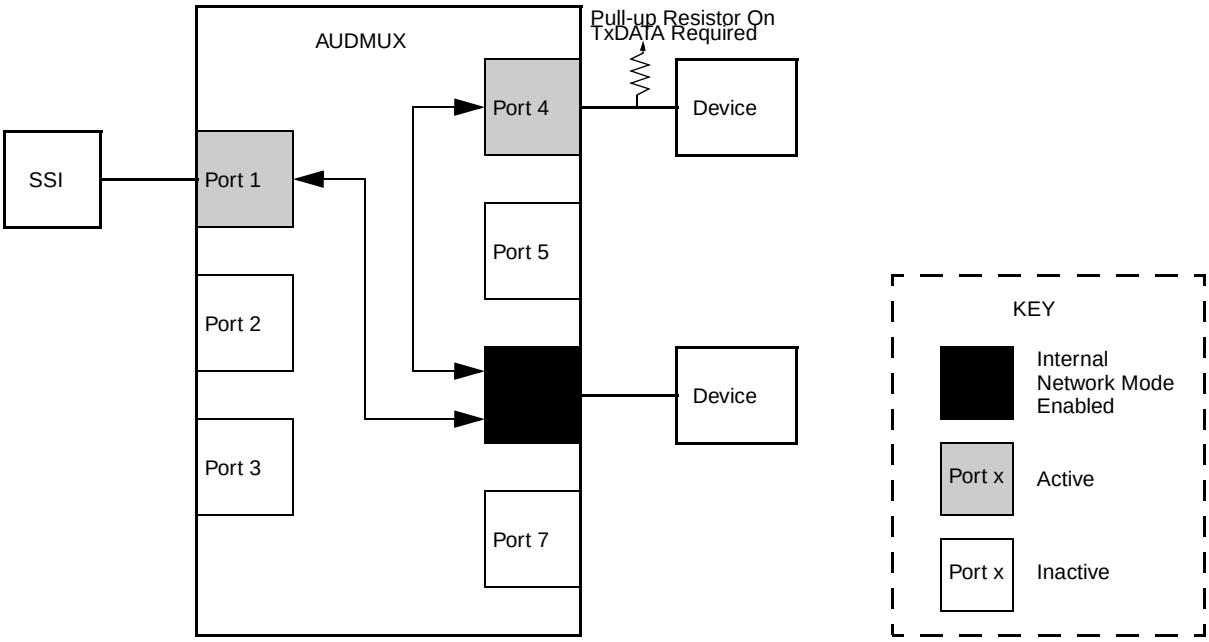


Figure 42-8. Block Diagram For Example 3

The resistance value of the pull-up resistors must be sufficiently high such that a value of ‘0’ can be pulled up to logic ‘1’ by the time that the next occupied timeslot occurs. This allows a much weaker pull-up to be used as compared to Example 2. The required resistance must be no larger than:

Rmax = (4 * n + 1) / (2* f_{bc} * C) where:

n is the number of bits per timeslot

f_{bc} is the frequency of the bitclock

C is the total system capacitance (ICs, board traces, and so on)

Figure 42-9 shows the timing diagram for this example. The clock and frame sync signals show the bit and frame timing for the serial bus. The vertical dashed lines divide the frame into four timeslots.

The data lines for the SSI and Port 4 are shown. Note that the SSI transmits a logic ‘1’ when its corresponding output enable is a logic ‘0’. The data line from Port 4 *at the pad* is pulled high by a pull-up resistor when they are in the high-impedance state. The data line from Port 4 *at the AUDMUX* is a pure digital signal and is constantly driven. The combined TxDATA line, which is the logical AND of the SSI and Port 4’s TxDATA lines, is used for Port 6’s RxDATA line.

Note the highlighted area in Figure 42-9. This shows the transition time that occurs while Port 4’s TxDATA line is being pulled high. In this example, this transition time is a maximum of two timeslots plus 1/2 the period of the serial bit clock. This prevents corruption of the first data bit of the next timeslot. It is critical that the pull-up resistance is sufficient for the given bit clock frequency and system capacitance.

Note that hysteresis must be enabled at Port 4’s RxDATA pad to prevent the digital signal created by the pad from toggling rapidly during the extended pull-up period. The pads typically require a transition within 25 ns unless hysteresis is enabled.

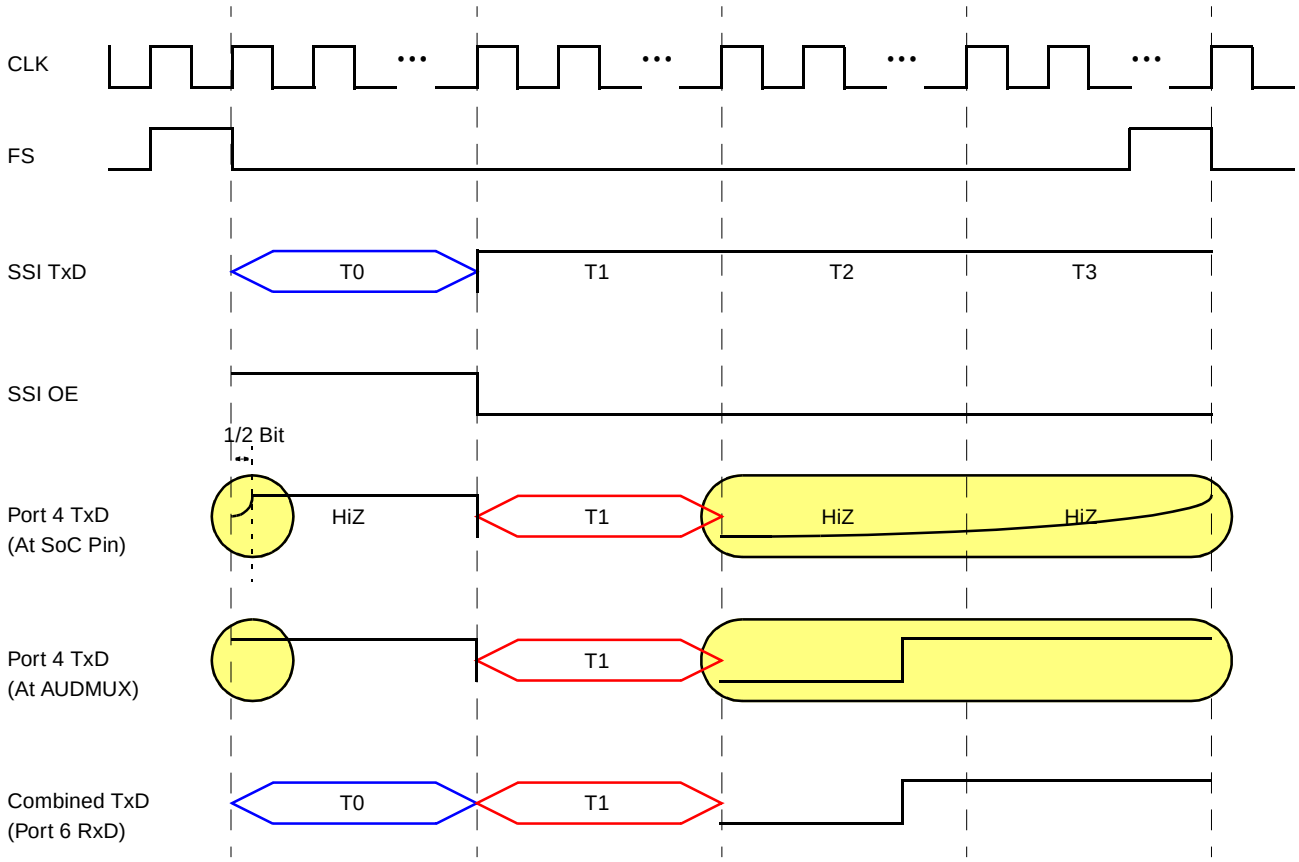


Figure 42-9. Example Using External Ports For Transmit Data In Nonconsecutive Timeslots

42.1.3.1.3 Bottom Connector Network Mode

To support network mode with devices connected to the bottom connector, a special network mode has been added. In Bottom Connector network mode (MODE[1:0] = 10), a 2 x 1 MUX is used to synchronously switch between two ports (that is, network slaves) as determined by a signal called bc_bus_dis, which is generated by the AUDMUX. Bottom Connector network mode supports switching between two ports. One of the ports is the Bottom Connector port (Port 7) and the other port is selected by RXDSEL[2:0]. In contrast to internal network mode, external ports do not require pull-up resistors, as the Bottom Connector network mode logic switches data lines according to the timeslot timing.

Generating the bc_bus_dis Signal

The AUDMUX uses clock and frame sync timing to generate bc_bus_dis (and thus drive the synchronous MUX used in Bottom Connector network mode). The Bottom Connector controls a contiguous block of timeslots. In addition, each timeslot is controlled by one of the two ports. For example, Port 7 (Bottom Connector port) could transmit in timeslots 1-2 and Port 6 could transmit in timeslots 0 and 3.

If the BEN bit is set, the bc_bus_dis signal generation logic is enabled. Otherwise, the signal generation logic is disabled and the bc_bus_dis signal remains low.

The CNTLOW[7:0] field controls the number of bit clock periods for which the bc_bus_dis signal is held low. This establishes the length of the Bottom Connector port's timeslot(s).

The number of bit clock periods for which the bc_bus_dis signal is held high after frame sync detection is controlled by the CNTHI[7:0] field. This establishes the number of bits between the frame sync detection and the start of the Bottom Connector port's timeslot(s). The internal counter used to determine the deassertion of the bc_bus_dis signal is reset on every frame sync. This assures that the number of bit clock periods during which the bc_bus_dis signal is held high after the assertion of frame sync is always correct.

The frame sync used to generate the bc_bus_dis signal is selected by the Port 7 control register PTCR7. If the SYN bit is set, then frame sync is determined by the TFSDIR and TFSEL[3:0] fields of PTCR7. If the SYN bit is clear (that is, Port 7 is 6-wire), then TxFS is determined by the TFSDIR and TFSEL[3:0] and the RxFS is determined by the RFSDIR and RFSEL[3:0] fields.

Similarly, the bit clock used for counting the low and high periods of the bc_bus_dis signal is determined by the Port 7 control register PTCR7. If the SYN bit is set, then the bit clock is determined by the TCLKDIR and TCSEL[3:0] fields of PTCR7. If the SYN bit is clear (that is, Port 7 is 6-wire), then TxCLK is determined by the TCLKDIR and TCSEL[3:0] and the RxCLK is determined by RCLKDIR and RCSEL[3:0] fields of PTCR7.

The FSPOL and CLKPOL fields of the CNMCR determine the frame sync and bit clock polarities used for frame sync detection. Different scenarios with combinations of frame sync and bit clock polarities are shown in Figure 42-10, Figure 42-11, Figure 42-12, Figure 42-13, and Figure 42-14. For all these scenarios, BEN is set as '1'. RxDn_in could be RxD1_in, RxD2_in, RxD3_in, RxD4_in, RxD5_in, or RxD6_in.

The Bottom Connector Network Mode control register CNMCR is used to control the generation of the bc_bus_dis signal. Refer to Section 42.3.2.15, "Bottom Connector Network Mode Control Register (CNMCR)," for complete details on the CNMCR.

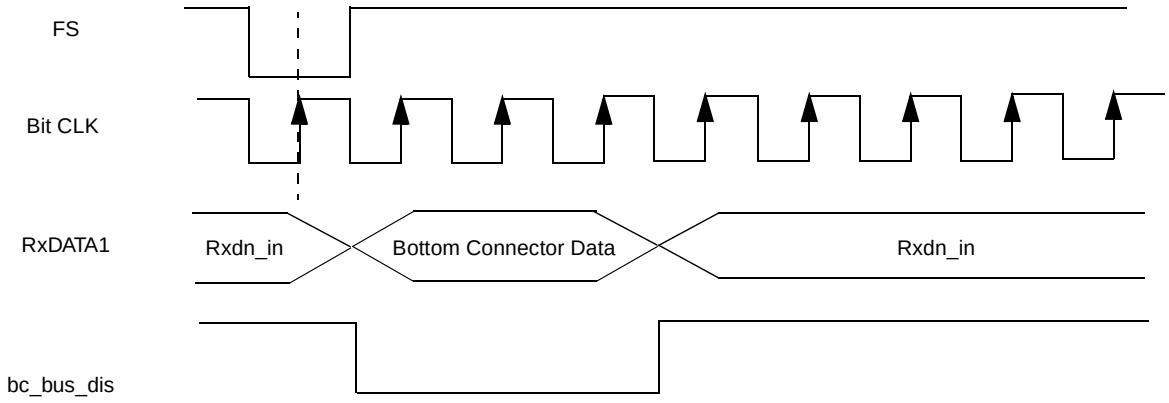


Figure 42-10. FSPOL-0, CLKPOL-0, CNTLOW-3, CNTHI-0

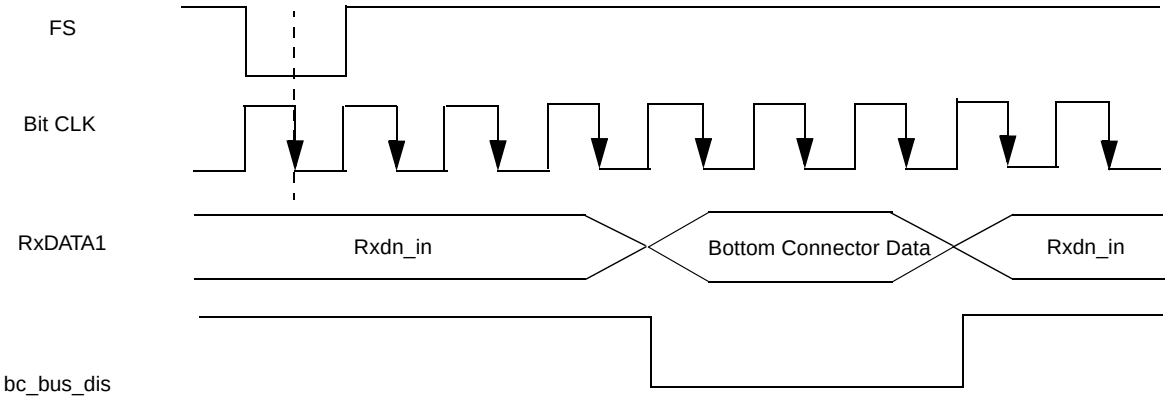


Figure 42-11. FSPOL-0, CLKPOL-1, CNTLOW-3, CNTHI-3

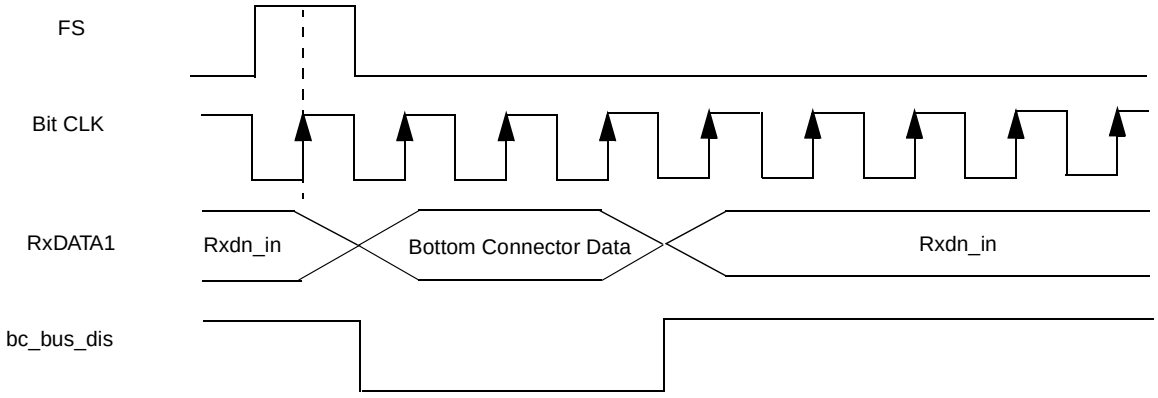


Figure 42-12. FSPOL-1, CLKPOL-0, CNTLOW-3, CNTHI-0

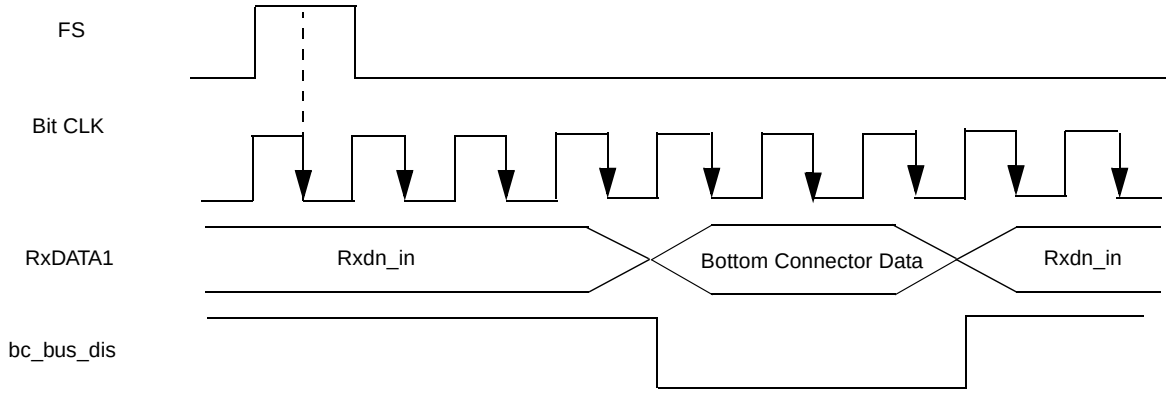


Figure 42-13. FSPOL-1, CLKPOL- 1, CNTLOW-3, CNTHI-3

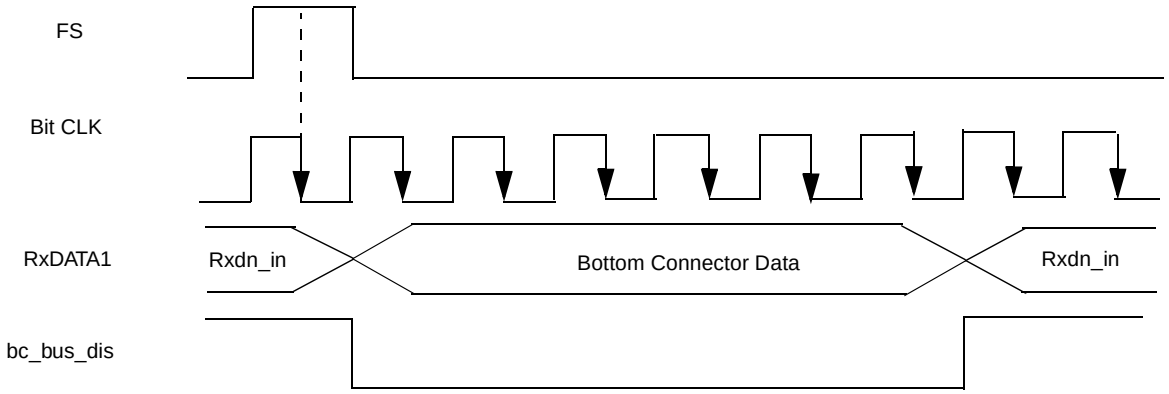


Figure 42-14. FSPOL-1, CLKPOL-1, CNTLOW-6, CNTHI-0

42.1.3.1.4 Transmit Data Output Enable Assertion

In normal mode or network mode from an internal master (not internal network mode) the TxDATA output is active during the active timeslot and high impedance during inactive timeslots. In internal network mode theTxDATA output is always active (always driver, never in high impedance mode, independently on the state of the timeslot - active or not).

42.1.3.2 Tx/Rx Switch and External Network Mode

External network mode is the traditional network mode connection. It is called external network mode to differentiate from the internal network mode. In external network mode, devices are connected to a single external port in a star or multi-drop configuration.

In network mode, there can be only one master (driving the frame sync and clock source) with the other devices configured in normal slave mode or network slave mode. Unlike internal network mode, both master-slave and slave-slave communication can take place in external network mode. CODEC devices transmit on a single timeslot while processor serial interfaces (that is, SSI, SAP) can process more than one timeslot of data while in network master or slave mode.

Figure 42-15 shows the Tx/Rx data switch. This function is controlled by TxRxEN bit of Port Data Control Register (PDCR). It permits the roles of the TXD pin and RXD pin of an audio port to be swapped. This is typically used when switching from a configuration where an internal port is connected to an external port to a configuration in which two external ports are connected together.

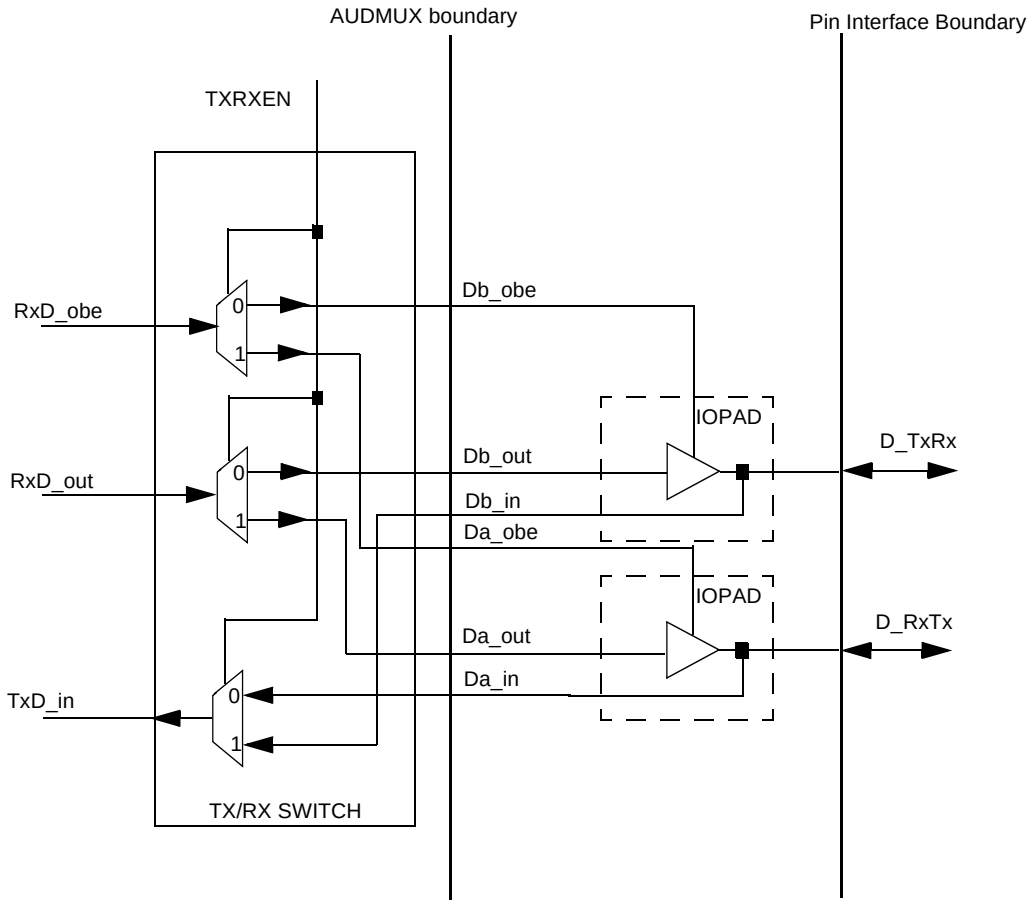


Figure 42-15. Tx/Rx Switch

42.1.3.3 Timing Modes

The AUDMUX ports are constructed as 6-wire interfaces. However, they can be used either in synchronous or asynchronous modes as determined by the SYN bit.

42.1.3.3.1 Synchronous Mode (4-wire Interface)

In Synchronous mode, the port has a 4-wire interface (that is, RxD, TxD, TxCLK, TxFS). The receive data timing is determined by TxCLK and TxFS. As shown in Figure 42-16, Port x signals can be routed to Port y, producing 6-wire to 4-wire port connectivity. TFS_in, RFS_in, TCLK_in, and RCLK_in are the input frame sync and bit clocks from the serial interface (Port x). TFS_out, RFS_out, TCLK_out, and RCLK_out are the frame sync and bit clocks that are transmitted to the serial interface from the other ports. The TFS_out and TCLK_out are selected at Port x by the TFSEL and TCSEL MUX settings, respectively. RFS_out and RCLK_out are selected at Port x by the RFSEL and RCSEL MUX settings, respectively. Similarly, in the external direction, Port y is configured as a 4-wire port; TFSEL selects the FS_out signal. In this mode, the configuration of RFSEL and RCSEL is not used, since the RFS_out and RCLK_out pins at Port y are not available. TFDIR and TCLKDIR are used to configure the direction of the frame sync and clock, RFDIR and RCLKSEL are ignored in Synchronous mode.

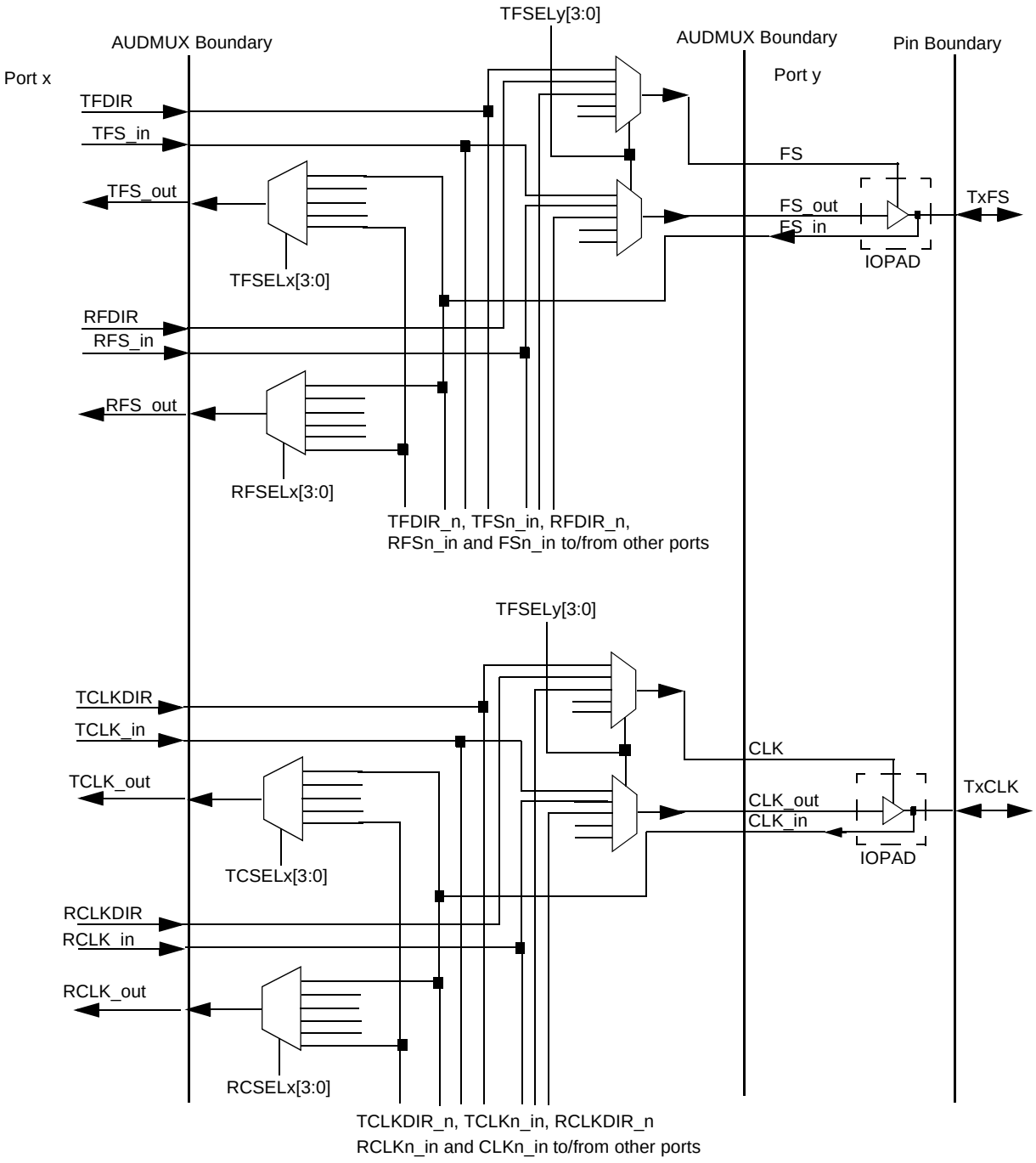


Figure 42-16. Frame Sync and Clock Routing When External Port Is 4-Wire

42.1.3.3.2 Asynchronous Mode (6-Wire Interface)

In Asynchronous mode, the port has a 6-wire interface (meaning RxD, TxD, TxCLK, TxFS, RxCLK, RxFS). This mode has additional receive clock (RxCLK) and frame sync (RxFS) signals as compared to the synchronous or 4-wire interface.

As shown in Figure 42-17 and Figure 42-18, the port x signals can be routed to Port y, producing 6-wire to 6-wire port connectivity. The TFS_in, RFS_in, TCLK_in, and RCLK_in signals are input frame sync and bit clocks from the serial interface (Port x). The signals TFS_out, RFS_out, TCLK_out, and RCLK_out are the frame sync and bit clocks that are transmitted to the serial interface from the other ports.

Both TFS_out and TCLK_out are selected by the TFSEL and TCSEL MUX settings, respectively. RFS_out and RCLK_out are selected by the RFSEL and RCSEL MUX settings, respectively. The direction of TFS and TCLK is configured by TFSDIR and TCLKDIR respectively. The direction of RFS and RCLK is configured by RFSDIR and RCLKDIR, respectively.

Similarly, in the external direction, the TFSEL selects from which port TxFS_out signal comes, and TFDIR configures its direction. TCSEL selects from which port TxClk_out signal comes, TCLKDIR configures its direction. The RFSEL selects RxFS_out signal source and RFDIR configures its direction. The RCSEL selects RxCLK_out signal and RXCLKDIR configures its direction.

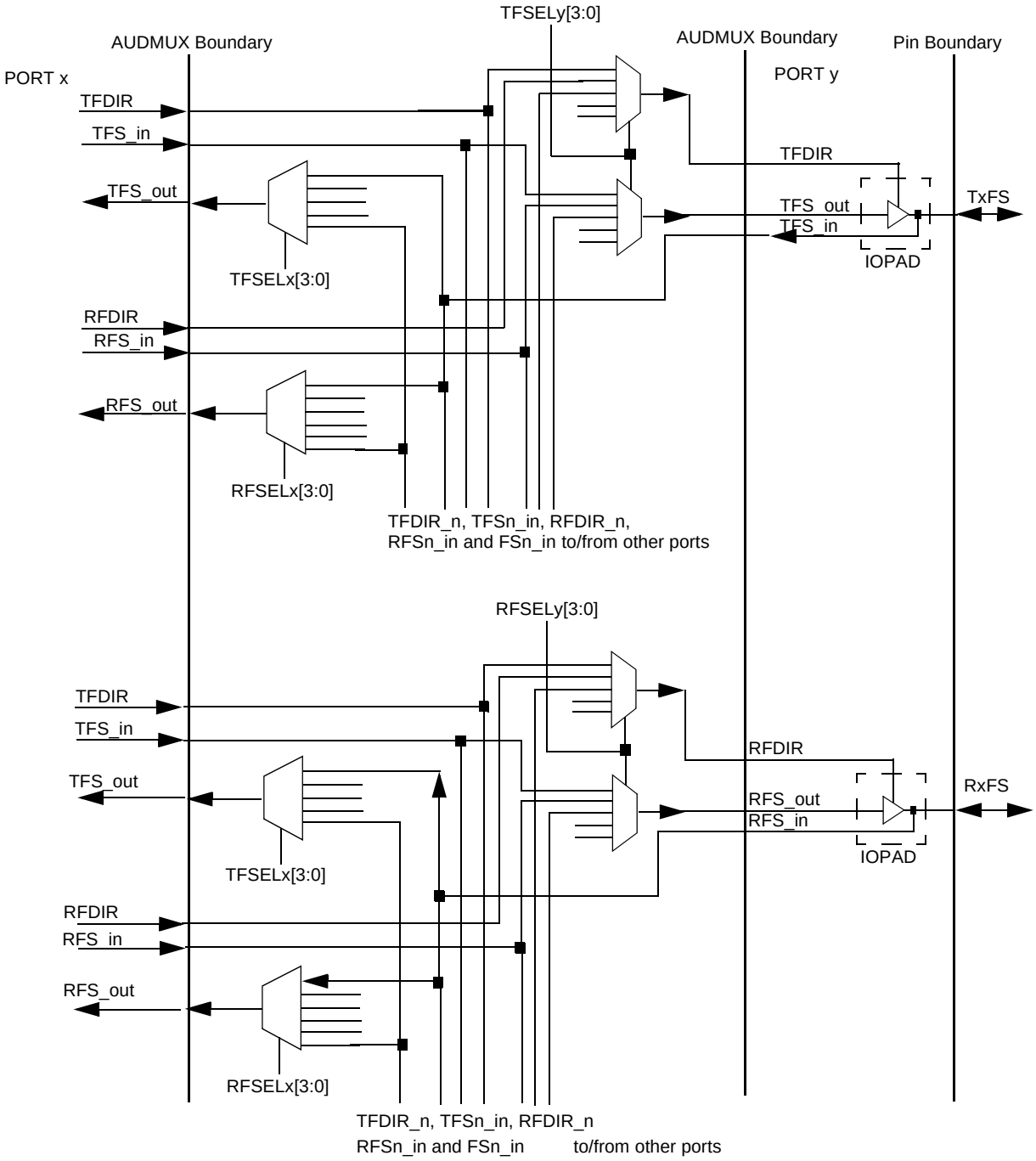


Figure 42-17. Frame Sync Routing When External Port Is 6-Wire

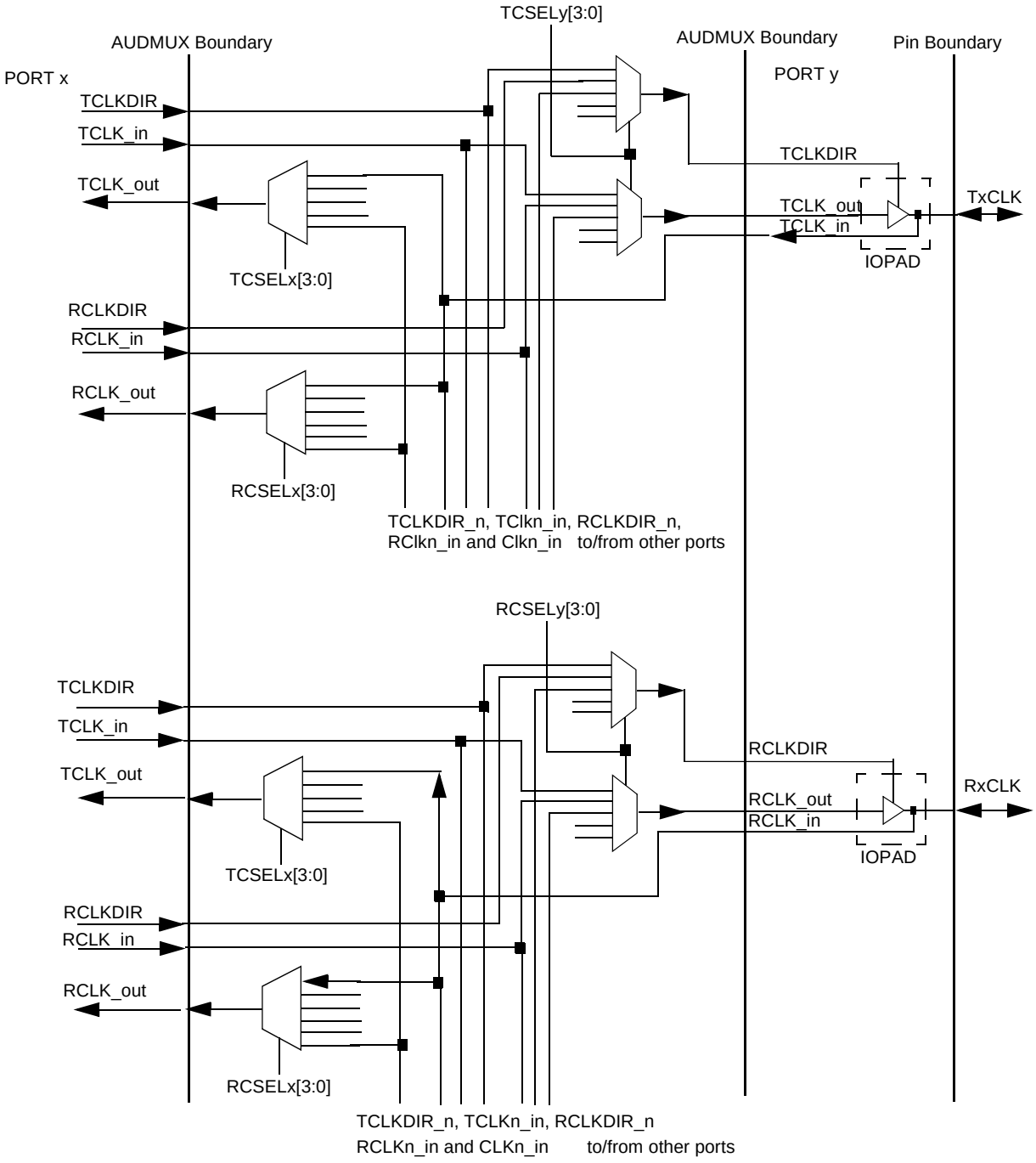


Figure 42-18. Clock Routing When External Port Is 6-Wire

42.1.4 Connectivity Between Ports

Four basic types of connections are provided by the AUDMUX:

- Internal port to external port
- External port to external port

- Internal port to internal port
- Loopback

The corresponding data connections are described in the following sections.

42.1.4.1 Internal Port to External Port Connectivity

Figure 42-19 shows the data path connections between an internal port and an external port. The internal port is connected to a processor's serial interface. TxD_in is the input transmit data from the serial interface to the AUDMUX, and RxD_out is the receive data output from the AUDMUX to the serial interface.

RXDSEL[2:0] controls the transmit data output (TxD_out) signal from the RxD_in signals. RXDSEL[2:0] is a common signal to both selection muxes.

NOTE

Since buffer TxD_in signals from external interfaces do not have corresponding buffer enable signals, their buffer enable signals into the selection MUX are tied high. This ensures that selection of TxD_in, as RxD_out will also drive the RxD output high.

Transmit Data from the serial interface goes into the RXDSEL data MUX and comes out as RxD_out. RxD_out is routed to Da_TxRx when TXRXEN is disabled and to D_RxTx when TXRXEN is enabled. Similarly, D_RxTx is routed to TxD_in when TXRXEN is disabled and D_TxRx is routed to TxD_in when TXRXEN is enabled. The routing of frame syncs is shown in Figure 42-17 and the routing of interface clocks is shown in Figure 42-18.

If internal network mode is disabled, then RXDSEL selects the TxD_in, which is sent from the AUDMUX to the serial interface connected at Port x. When the internal network mode is selected, RxD_out is constructed by AND'ing selected TxD_in signals from the ports (as determined by INMMASK).

If there is more than one device attached to the external port at D_TxRx and D_RxTx and one of the devices is a network master, then two conditions must be noted:

1. When the external master is enabled in network mode, then the serial interface at Port x must be configured as a slave (normal or network mode). No Tx/Rx switching is required.
2. When the external master is disabled and the serial interface at Port x and other slave devices must communicate, then the serial interface at Port x must be configured as a network mode master and the Tx/Rx switch at Port y must be enabled (TXRXEN=1). This will ensure that the transmit and receive paths are connected appropriately.

To communicate with more than one port, internal network mode can be enabled at Port x. In internal network mode, it is possible to communicate with any device attached to the other ports. Internal network mode shall be enabled at the port that is the SSI network mode master.

It is also possible for a port to communicate with two (and only two) ports by enabling Bottom Connector network mode at Port x. It is then possible to communicate with any device attached to two enabled ports; one of those ports is Port 7 and the other is selected by RXDSELx[2:0]. Bottom Connector network mode is enabled at the port that is the SSI network mode master.

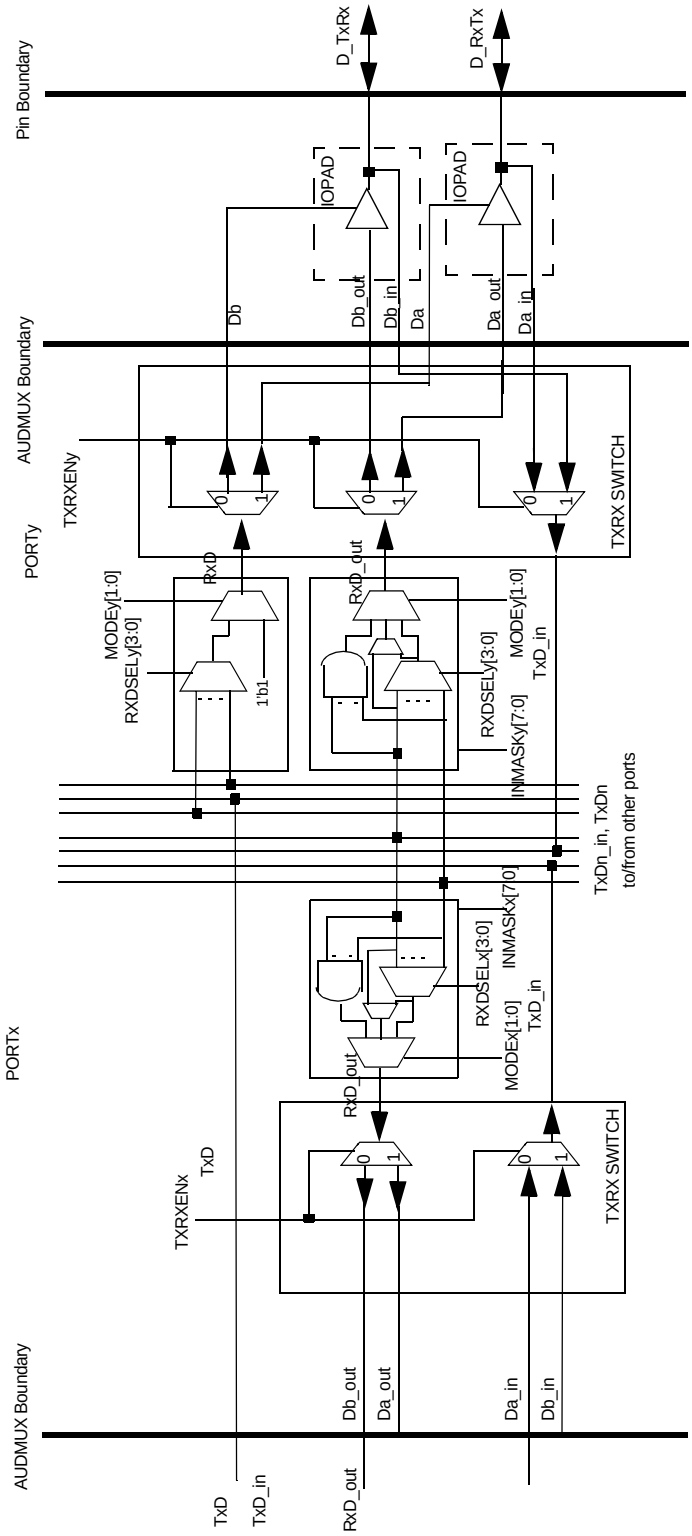


Figure 42-19. Internal to External Port Interconnection

42.1.4.2 External Port to External Port Connectivity

External ports can communicate with external ports directly. External ports can communicate together in three ways:

1. Each port's receive logic is configured in normal mode (MODE[0:1] = 00). Each port's RXDSEL[2:0] field is configured to select the other port's transmit data. Bit fields associated with clock/frame sync selection and direction are configured for each port. Either port can be the master.
2. One port is configured in internal network mode (MODE[0:1] = 01). All desired data lines are combined by the AND gate as determined by INMMASK[7:0]. Since an external port is being used as the internal network mode master, all other devices on the same AUDMUX port as the internal network mode master must be disabled. This configuration can be used with a combination of internal and external ports. All external ports must have a pull-up resistor on its RxDATA pin. Bit fields associated with clock/frame sync selection and direction are configured for each port. Any port can be the master.
3. One port is configured in Bottom Connector network mode (MODE[0:1] = 10) to receive and/or transmit data from and to the Bottom Connector connected to Port 7 and one other port. The bc_bus_dis signal is generated by the AUDMUX depending upon the CNMCR register configuration. Bit fields associated with clock/frame sync selection and direction are configured for each port. Any of the three ports can be the master.

42.1.4.3 Internal Port to Internal Port Connectivity

Internal ports can communicate with other internal ports directly. Internal ports can communicate together in three ways:

1. Each port's receive logic is configured in normal mode (MODE[0:1] = 00). Each port's RXDSEL[2:0] field is configured to select the other port's transmit data. Bit fields associated with clock/frame sync selection and direction are configured for each port. Either port can be the master.
2. One port is configured in internal network mode (MODE[0:1] = 01). All desired data lines are combined by the AND gate as determined by INMMASK[7:0]. This configuration can be used with a combination of internal and external ports. All external ports must have a pull-up resistor on its RxDATA pin. Bit fields associated with clock/frame sync selection and direction are configured for each port. Any port can be the master.
3. One port is configured in Bottom Connector network mode (MODE[0:1] = 10) to receive and/or transmit data from and to the Bottom Connector connected at Port 7 and one other port. The bc_bus_dis signal is generated by the AUDMUX depending upon the CNMCR register configuration. Bit fields associated with clock/frame sync selection and direction are configured for each port. Any of the three ports can be the master.

42.1.4.4 Loopback Connectivity

AUDMUX ports can communicate with themselves to provide loopback functionality. Port x can route its TxDATA signal to its own RxD_out signal by setting RXDSELx[2:0] to its own port number. This is supported by all ports in the AUDMUX.

In addition, ports can provide loopback support in internal network mode as well as Bottom Connector network mode. With internal network mode, the internal network mode master can loop its TxDATA signal (combined with those of other ports, if desired) back into its RxD_out signal. Port x's INMMASK should be set such that bit (x-1) is clear to enable the loopback.

With Bottom Connector network mode, the Bottom Connector network mode master can loop its TxDATA signal (combined with Bottom Connector's TxDATA) back into its RxD_in signal. RXDSELx[2:0] should be set to select Port x.

42.2 External Signal Description

The following section provides the external signal descriptions for AUDMUX.

The two internal ports are connected to the 2 SSI modules in the i.MX31. The five external ports are connected to the external contacts of the i.MX31, as described in the following table. Table 42-1 provides general guidance about signals of the AUDMUX and their association with the external ports. Refer to I/O Multiplexing chapter for the details about how to set up these MUX configurations.

Table 42-1. AUDMUX External Signals and Port Numbers

AUDMUX Port Number	i.MX31 pin group	MX31 pins	
		Synchronous mode	Asynchronous mode
1	Internal port, connected to SSI1	N/A	N/A
2	Internal port, connected to SSI2	N/A	N/A
3	External port, connected to pin group Audio port 3-BB(HP3)	SRXD3, STXD3, SCK3, SFS3	SRXD3, STXD3, SCK3, SFS3, SRXD4, STXD4,
4	External port, connected to pin group Audio port 4-PM-NB(PP1)	SRXD4, STXD4, SCK4, SFS4	N/A
5	External port, connected to pin group Audio port 5-PM-WB(PP2)	SRXD5, STXD5, SCK5, SFS5	SRXD5, STXD5, SCK5, SFS5, SCK4, SFS4
6	External port, connected to Audio port 6-BT(PP3)	SRXD6, STXD6, SCK6, SFS6	N/A
7	External port, connected to Bottom Connector bus(PP4) (muxed with UART1 signals)	RXD1, TXD1, RTS1, DTR_DCE1	N/A

42.3 Memory Map and Register Definition

The AUDMUX memory map is shown in Table 42-2.

Table 42-2. AUDMUX Memory Map

Address	Register	Access	Reset Value	Section/Page
0x53FC_4000 (PTCR1)	Port Timing Control Register 1 (PTCR1)	R/W	0xAD40_0800	42.3.2.1/42-28
0x53FC_4004 (PDCR1)	Port Data Control Register 1 (PDCR1)	R/W	0x0000_A000	42.3.2.2/42-31

Table 42-2. AUDMUX Memory Map (continued)

Address	Register	Access	Reset Value	Section/Page
0x53FC_4008 (PTCR2)	Port Timing Control Register 2 (PTCR2)	R/W	0xA500_0800	42.3.2.3/42-32
0x53FC_400C (PDCR2)	Port Data Control Register 2 (PDCR2)	R/W	0x0000_8000	42.3.2.4/42-34
0x53FC_4010 (PTCR3)	Port Timing Control Register 3 (PTCR3)	R/W	0x9CC0_0800	42.3.2.5/42-36
0x53FC_4014 (PDCR3)	Port Data Control Register 3 (PDCR3)	R/W	0x0000_6000	42.3.2.6/42-38
0x53FC_4018 (PTCR4)	Port Timing Control Register 4 (PTCR4)	R/W	0x0000_0800	42.3.2.7/42-39
0x53FC_401C (PDCR4)	Port Data Control Register 4 (PDCR4)	R/W	0x0000_4000	42.3.2.8/42-41
0x53FC_4020 (PTCR5)	Port Timing Control Register 5 (PTCR5)	R/W	0x0000_0800	42.3.2.9/42-43
0x53FC_4024 (PDCR5)	Port Data Control Register 5 (PDCR5)	R/W	0x0000_2000	42.3.2.10/42-45
0x53FC_4028 (PTCR6)	Port Timing Control Register 6 (PTCR6)	R/W	0x0000_0800	42.3.2.11/42-46
0x53FC_402C (PDCR6)	Port Data Control Register 6 (PDCR6)	R/W	0x0000_0000	42.3.2.12/42-48
0x53FC_4030 (PTCR7)	Port Timing Control Register 7 (PTCR7)	R/W	0x0000_0800	42.3.2.13/42-50
0x53FC_4034 (PDCR7)	Port Data Control Register 7 (PDCR7)	R/W	0x0000_C000	42.3.2.14/42-52
0x53FC_4038 (CNMCR)	Bottom Connector Network Mode Control Register (CNMCR)	R/W	0x0003_1010	42.3.2.15/42-53

42.3.1 Register Summary

Table 42-4 shows the control register and address mapping for the AUDMUX.

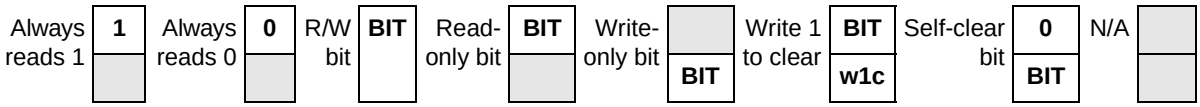


Figure 42-20. Key to Register Fields

Table 42-3. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writeable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.

Table 42-3. Register Figure Conventions (continued)

Convention	Description
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 42-4. AUDMUX Register Summary

Address		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x53FC_4000 (PTCR1)	R	TFSDIR	TFSEL[3:0]				TCLKDIR	TCSEL[3:0]				RFSDIR	RFSEL[3:0]				RCLKDIR
	W																
	R	RCSEL[3:0]				SYN	0	0	0	0	0	0	0	0	0	0	0
	W																
0x53FC_4004 (PDCR1)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	RXDSEL[2:0]			TXRXEN	0	0	MODE[1:0]		INMMASK[7:0]							
	W																
0x53FC_4008 (PTCR2)	R	TFSDIR	TFSEL[3:0]				TCLKDIR	TCSEL[3:0]				RFSDIR	RFSEL[3:0]				RCLKDIR
	W																
	R	RCSEL[3:0]				SYN	0	0	0	0	0	0	0	0	0	0	0
	W																
0x53FC_400C (PDCR2)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	RXDSEL[2:0]			TXRXEN	0	0	MODE[1:0]		INMMASK[7:0]							
	W																

Table 42-4. AUDMUX Register Summary (continued)

Address		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x53FC_4010 (PTCR3)	R	TFSDIR	TFSEL[3:0]				TCLKDIR	TCSEL[3:0]				RFSDIR	RFSEL[3:0]				RCLKDIR
	W																
	R	RCSEL[3:0]				SYN	0	0	0	0	0	0	0	0	0	0	0
	W																
0x53FC_4014 (PDCR3)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	RXDSEL[2:0]			TXRXEN	0	0	MODE[1:0]		INMMASK[7:0]							
	W																
0x53FC_4018 (PTCR4)	R	TFSDIR	TFSEL[3:0]				TCLKDIR	TCSEL[3:0]				RFSDIR	RFSEL[3:0]				RCLKDIR
	W																
	R	RCSEL[3:0]				SYN	0	0	0	0	0	0	0	0	0	0	0
	W																
0x53FC_401C (PDCR4)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	RXDSEL[2:0]			TXRXEN	0	0	MODE[1:0]		INMMASK[7:0]							
	W																
0x53FC_4020 (PTCR5)	R	TFSDIR	TFSEL[3:0]				TCLKDIR	TCSEL[3:0]				RFSDIR	RFSEL[3:0]				RCLKDIR
	W																
	R	RCSEL[3:0]				SYN	0	0	0	0	0	0	0	0	0	0	0
	W																
0x53FC_4024 (PDCR5)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	RXDSEL[2:0]			TXRXEN	0	0	MODE[1:0]		INMMASK[7:0]							
	W																
0x53FC_4028 (PTCR6)	R	TFSDIR	TFSEL[3:0]				TCLKDIR	TCSEL[3:0]				RFSDIR	RFSEL[3:0]				RCLKDIR
	W																
	R	RCSEL[3:0]				SYN	0	0	0	0	0	0	0	0	0	0	0
	W																

Table 42-4. AUDMUX Register Summary (continued)

Address		31 15	30 14	29 13	28 12	27 11	26 10	25 9	24 8	23 7	22 6	21 5	20 4	19 3	18 2	17 1	16 0
0x53FC_402C (PDCR6)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	RXDSEL[2:0]			TXRXEN	0	0	MODE[1:0]		INMMASK[7:0]							
	W																
0x53FC_4030 (PTCR7)	R	TFSDIR	TFSEL[3:0]				TCLKDIR	TCSEL[3:0]				RFSDIR	RFSEL[3:0]				RCLKDIR
	W																
	R	RCSEL[3:0]				SYN	0	0	0	0	0	0	0	0	0	0	0
	W																
0x53FC_4034 (PDCR7)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	RXDSEL[2:0]			TXRXEN	0	0	MODE[1:0]		INMMASK[7:0]							
	W																
0x53FC_4038 (CNMCR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	BEN	FSPOL	CLKPOL
	W																
	R	CNTHI[7:0]								CNTLOW[7:0]							
	W																

42.3.2 Register Descriptions

There are two configuration registers for each port. There is also a register for Bottom Connector Network mode control. There are a total of 15 configuration registers.

42.3.2.1 Port Timing Control Register 1 (PTCR1)

PTCR1 is the Port Timing Control Register for Port 1. Figure 42-21 shows the register; Table 42-5 provides its field descriptions.

0x53FC_4000 (PTCR1) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	TFS DIR	TFSEL[3:0]				TCLK DIR	TCSEL[3:0]				RFS DIR	RFSEL[3:0]				RCL KDIR
Reset	1	0	1	0	1	1	0	1	0	1	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RCSEL[3:0]				SYN	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

Figure 42-21. Port Timing Control Register for Port 1 (PTCR1)

Table 42-5. PTCR1 Field Descriptions

Field	Description
31 TFS DIR	Transmit Frame Sync Direction Control. This bit sets the direction of the TxFS pin of the interface as an output or input. When set as an input, the TFSEL settings are ignored. When set as an output, the TFSEL settings determine the source port of the frame sync. 0 TxFS is an input. 1 TxFS is an output.
30–27 TFSEL[3:0]	Transmit Frame Sync Select. Selects the source port from which TxFS is sourced. 0xxx Selects TxFS from port. 1xxx Selects RxFS from port. x000 Port 1 —— x110 Port 7 x111 Reserved
26 TCLKDIR	Transmit Clock Direction Control. This bit sets the direction of the TxClk pin of the interface as an output or input. When set as an input, the TCSEL settings are ignored. When set as an output, the TCSEL settings determine the source port of the clock. 0 TxClk is an input. 1 TxClk is an output.
25–22 TCSEL[3:0]	Transmit Clock Select. Selects the source port from which TxClk is sourced. 0xxx Selects TxClk from port. 1xxx Selects RxClk from port. x000 Port 1 —— x110 Port 7 x111 Reserved

Table 42-5. PTCR1 Field Descriptions (continued)

Field	Description
21 RFS DIR	Receive Frame Sync Direction Control. This bit sets the direction of the RxFS pin of the interface as an output or input. When set as an input, the RFSEL settings are ignored. When set as an output, the RFSEL settings determine the source port of the frame sync. 0 RxFS is an input. 1 RxFS is an output.
20–17 RFSEL[3:0]	Receive Frame Sync Select. Selects the source port from which RxFS is sourced. RxFS can be sourced from TxFS and RxFS from other ports. 0xxx Selects TxFS from port. 1xxx Selects RxFS from port. x000 Port 1 —— x110 Port 7 x111 Reserved
16 RCLKDIR	Receive Clock Direction Control. This bit sets the direction of the RxClk pin of the interface as an output or input. When set as an input, the RCSEL settings are ignored. When set as an output, the RCSEL settings determine the source port of the clock. 0 RxClk is an input. 1 RxClk is an output.
15–12 RCSEL[3:0]	Receive Clock Select. Selects the source port from which RxClk is sourced. RxClk can be sourced from TxClk and RxClk from other ports. 0xxx Selects TxClk from port. 1xxx Selects RxClk from port. x000 Port 1 —— x110 Port 7 x111 Reserved
11 SYN	Synchronous/Asynchronous Select. When SYN is set, synchronous mode is chosen and the transmit and receive sections use common clock and frame sync signals (that is, the port is a 4-wire interface). When SYN is cleared, asynchronous mode is chosen and separate clock and frame sync signals are used for the transmit and receive sections (that is, the port is a 6-wire interface). 0 Asynchronous mode 1 Synchronous mode (default)
10–0	Reserved

42.3.2.2 Port Data Control Register 1 (PDCR1)

PDCR1 is the Port Data Control Register for Port 1. Figure 42-22 shows the register; Table 42-6 provides its field descriptions.

0x53FC_4004 (PDCR1) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RXDSEL[2:0]			TXR XEN	0	0	MODE[1:0]		INMMASK[7:0]							
W																
Reset	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 42-22. Port Data Control Register for Port 1 (PDCR1)

Table 42-6. PDCR (Port 1) Field Descriptions

Field	Description
31–16	Reserved
15–13 RXDSEL[2:0]	Receive Data Select. Selects the source port for the RxD data. RXDSEL is ignored if MODE[1:0] is 2'b01 (that is, Internal Network Mode is enabled). xxx Port number for RxD 000 Port 1 —— 110 Port 7 111 Reserved
12 TXRXEN	Transmit/Receive Switch Enable. Swaps the transmit and receive signals. 0 No switch (Transmit Pin = Transmit, Receive Pin = Receive) 1 Switch (Transmit Pin = Receive, Receive Pin = Transmit)
11–10	Reserved

Table 42-6. PDCR (Port 1) Field Descriptions (continued)

Field	Description
9–8 MODE[1:0]	Mode Select. This field selects the mode in which the port is to operate. The modes of operation include the following: • Normal mode, in which the RxD from the port selected by RXDSEL is routed to the port. • Internal Network mode in which RxD from other ports are AND'ed together. RXDSEL is ignored. INMMASK determines which RxD signals are AND'ed together. • Bottom Connector Network mode, in which the port receives data from Bottom Connector port (Port 7) and any other port as selected by RXDSEL[3:0] in different time slots within a frame. The bc_bus_dis signal routes the data from Bottom Connector port if low otherwise data from the other port (as selected by RXDSEL) is routed to the port. 00 Normal mode 01 Internal Network mode 10 Bottom Connector Network mode 11 Reserved
7–0 INMMASK[7:0]	Internal Network Mode Mask. Bit mask that selects the ports from which the RxD signals are to be AND'ed together for internal network mode. Bit 6 represents RxD from Port 7 and bit0 represents RxD from Port 1. 0 Includes RxDn for AND'ing 1 Excludes RxDn from AND'ing

42.3.2.3 Port Timing Control Register 2 (PTCR2)

PTCR2 is the Port Timing Control Register for Port 2. Figure 42-23 shows the register; Table 42-7 provides its field descriptions.

0x53FC_4008 (PTCR2) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TFS DIR	TFSEL[3:0]				TCLK DIR	TCSEL[3:0]				RFS DIR	RFSEL[3:0]				RCL KDIR
W																
Reset	1	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RCSEL[3:0]				SYN	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

Figure 42-23. Port Timing Control Register 2 (PTCR2)

Table 42-7. PTCR2 Field Descriptions

Field	Description
31 TFS DIR	Transmit Frame Sync Direction Control. This bit sets the direction of the TxFS pin of the interface as an output or input. When set as an input, the TFSEL settings are ignored. When set as an output, the TFSEL settings determine the source port of the frame sync. 0 TxFS is an input. 1 TxFS is an output.
30–27 TFSEL[3:0]	Transmit Frame Sync Select. Selects the source port from which TxFS is sourced. 0xxx Selects TxFS from port. 1xxx Selects RxFS from port. x000 Port 1 —— x110 Port 7 x111 Reserved
26 TCLKDIR	Transmit Clock Direction Control. This bit sets the direction of the TxClk pin of the interface as an output or input. When set as an input, the TCSEL settings are ignored. When set as an output, the TCSEL settings determine the source port of the clock. 0 TxClk is an input. 1 TxClk is an output.
25–22 TCSEL[3:0]	Transmit Clock Select. Selects the source port from which TxClk is sourced. 0xxx Selects TxClk from port. 1xxx Selects RxClk from port. x000 Port 1 —— x110 Port 7 x111 Reserved
21 RFS DIR	Receive Frame Sync Direction Control. This bit sets the direction of the RxFS pin of the interface as an output or input. When set as an input, the RFSEL settings are ignored. When set as an output, the RFSEL settings determine the source port of the frame sync. 0 RxFS is an input. 1 RxFS is an output.
20–17 RFSEL	Receive Frame Sync Select. Selects the source port from which RxFS is sourced. RxFS can be sourced from TxFS and RxFS from other ports. 0xxx Selects TxFS from port. 1xxx Selects RxFS from port. x000 Port 1 —— x110 Port 7 x111 Reserved
16 RCLKDIR	Receive Clock Direction Control. This bit sets the direction of the RxClk pin of the interface as an output or input. When set as an input, the RCSEL settings are ignored. When set as an output, the RCSEL settings determine the source port of the clock. 0 RxClk is an input. 1 RxClk is an output.

Table 42-7. PTCR2 Field Descriptions (continued)

Field	Description
15–12 RCSEL[3:0]	Receive Clock Select. Selects the source port from which RxClk is sourced. RxClk can be sourced from TxClk and RxClk from other ports. 0xxx Selects TxClk from port. 1xxx Selects RxClk from port. x000 Port 1 —— x110 Port 7 x111 Reserved
11 SYN	SYN—Synchronous/Asynchronous Select. When SYN is set, synchronous mode is chosen and the transmit and receive sections use common clock and frame sync signals (that is, the port is a 4-wire interface). When SYN is cleared, asynchronous mode is chosen and separate clock and frame sync signals are used for the transmit and receive sections (that is, the port is a 6-wire interface). 0 Asynchronous mode 1 Synchronous mode (default)
10–0	Reserved

42.3.2.4 Port Data Control Register 2 (PDCR2)

PDCR2 is the Port Data Control Register for Port 2. Figure 42-24 shows the register; Table 42-8 provides its field descriptions.

0x53FC_400C (PDCR2) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RXDSEL[2:0]			TXR XEN	0	0	MODE[1:0]		INMMASK[7:0]							
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 42-24. Port Data Control Register 2 (PDCR2)

Table 42-8. PDCR2 Field Descriptions

Field	Description
31–16	Reserved
15–13 RXDSEL[2:0]	Receive Data Select. Selects the source port for the RxD data. RXDSEL is ignored if MODE[1:0] is 01 (that is, Internal Network mode is enabled). xxx Port number for RxD 000 Port 1 —— 110 Port 7 111 Reserved
12 TXRXEN	Transmit/Receive Switch Enable. Swaps the transmit and receive signals. 0 No switch (Transmit Pin = Transmit, Receive Pin = Receive) 1 Switch (Transmit Pin = Receive, Receive Pin = Transmit)
11–10	Reserved
9–8 MODE[1:0]	Mode Select. This field selects the mode in which the port is to operate.The modes of operation include the following: • Normal mode, in which the RxD from the port selected by RXDSEL is routed to the port. • Internal Network mode in which RxD from other ports are ANDED together. RXDSEL is ignored. INMMASK determines which RxD signals are AND'ed together. • Bottom Connector Network mode, in which the port receives data from Bottom Connector port (Port 7) and any other port as selected by RXDSEL[3:0] in different time slots within a frame. The bc_bus_dis signal routes the data from Bottom Connector port if low otherwise data from the other port (as selected by RXDSEL) is routed to the port. 00 Normal mode 01 Internal Network mode 10 Bottom Connector Network mode 11 Reserved
7–0 INMMASK[7:0]	Internal Network Mode Mask. Bit mask that selects the ports from which of the RxD signals are to be AND'ed together for internal network mode. Bit 6 represents RxD from Port 7 and bit0 represents RxD from Port 1. 0 Includes RxDn for AND'ing 1 Excludes RxDn from AND'ing

42.3.2.5 Port Timing Control Register 3 (PTCR3)

PTCR3 is the Port Timing Control Register for Port 3. Figure 42-25 shows the register; Table 42-9 provides its field descriptions.

0x53FC_4010 (PTCR3) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	TFS DIR	TFSEL[3:0]				TCLK DIR	TCSEL[3:0]				RFS DIR	RFSEL[3:0]				RCL KDIR
Reset	1	0	0	1	1	1	0	0	1	1	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RCSEL[3:0]				SYN	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

Figure 42-25. Port Timing Control Register 3 (PTCR3)

Table 42-9. PTCR3 Field Descriptions

Field	Description
31 TFS DIR	Transmit Frame Sync Direction Control. This bit sets the direction of the TxFS pin of the interface as an output or input. When set as an input, the TFSEL settings are ignored. When set as an output, the TFSEL settings determine the source port of the frame sync. 0 TxFS is an input. 1 TxFS is an output.
30–27 TFSEL[3:0]	Transmit Frame Sync Select. Selects the source port from which TxFS is sourced. 0xxx Selects TxFS from port. 1xxx Selects RxFS from port. x000 Port 1 —— x110 Port 7 x111 Reserved
26 TCLKDIR	Transmit Clock Direction Control. This bit sets the direction of the TxClk pin of the interface as an output or input. When set as an input, the TCSEL settings are ignored. When set as an output, the TCSEL settings determine the source port of the clock. 0 TxClk is an input. 1 TxClk is an output.
25–22 TCSEL[3:0]	Transmit Clock Select. Selects the source port from which TxClk is sourced. 0xxx Selects RxClk from port. 1xxx Selects RxClk from port. x000 Port 1 —— x110 Port 7 x111 Reserved

Table 42-9. PTCR3 Field Descriptions (continued)

Field	Description
21 RFS DIR	Receive Frame Sync Direction Control. This bit sets the direction of the RxFS pin of the interface as an output or input. When set as an input, the RFSEL settings are ignored. When set as an output, the RFSEL settings determine the source port of the frame sync. 0 RxFS is an input. 1 RxFS is an output.
20–17 RFSEL[3:0]	Receive Frame Sync Select. Selects the source port from which RxFS is sourced. RxFS can be sourced from TxFS and RxFS from other ports. 0xxx Selects TxFS from port. 1xxx Selects RxFS from port. x000 Port 1 —— x110 Port 7 x111 Reserved
16 RCLKDIR	Receive Clock Direction Control. This bit sets the direction of the RxClk pin of the interface as an output or input. When set as an input, the RCSEL settings are ignored. When set as an output, the RCSEL settings determine the source port of the clock. 0 RxClk is an input. 1 RxClk is an output.
15–12 RCSEL[3:0]	Receive Clock Select. Selects the source port from which RxClk is sourced. RxClk can be sourced from TxClk and RxClk from other ports. 0xxx Selects TxClk from port. 1xxx Selects RxClk from port. x000 Port 1 —— x110 Port 7 x111 Reserved
11 SYN	SYN—Synchronous/Asynchronous Select. When SYN is set, synchronous mode is chosen and the transmit and receive sections use common clock and frame sync signals (that is, the port is a 4-wire interface). When SYN is cleared, asynchronous mode is chosen and separate clock and frame sync signals are used for the transmit and receive sections (that is, the port is a 6-wire interface). 0 Asynchronous mode 1 Synchronous mode (default)
10–0	Reserved

42.3.2.6 Port Data Control Register 3 (PDCR3)

PDCR3 is the Port Data Control Register for Port 3. Figure 42-26 shows the register; Table 42-10 provides its field descriptions.

0x53FC_4014 (PDCR3) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RXDSEL[2:0]			TXR XEN	0	0	MODE[1:0]		INMMASK[7:0]							
W																
Reset	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 42-26. Port Data Control Register 3 (PDCR3)

Table 42-10. PDCR3 Field Descriptions

Field	Description
31–16	Reserved
15–13 RXDSEL[2:0]	Receive Data Select. Selects the source port for the RxD data. RXDSEL is ignored if MODE[1:0] is 01 (that is, Internal Network Mode is enabled). xxx Port number for RxD 000 Port 1 —— 110 Port 7 111 Reserved
12 TXRXEN	Transmit/Receive Switch Enable. Swaps the transmit and receive signals. 0 No switch (Transmit Pin = Transmit, Receive Pin = Receive) 1 Switch (Transmit Pin = Receive, Receive Pin = Transmit)
11–10	Reserved

Table 42-10. PDCR3 Field Descriptions (continued)

Field	Description
9–8 MODE[1:0]	Mode Select. This field selects the mode in which the port is to operate.The modes of operation include the following: • Normal mode, in which the RxD from the port selected by RXDSEL is routed to the port. • Internal Network mode in which RxD from other ports are AND'ed together. RXDSEL is ignored. INMMASK determines which RxD signals are AND'ed together. • Bottom Connector Network mode, in which the port receives data from Bottom Connector port (Port7) and any other port as selected by RXDSEL[3:0] in different time slots within a frame. The bc_bus_dis signal routes the data from Bottom Connector port if low otherwise data from the other port (as selected by RXDSEL) is routed to the port. 00 Normal mode 01 Internal Network mode 10 Bottom Connector Network mode 11 Reserved
7–0 INMMASK[7:0]	Internal Network Mode Mask. Bit mask that selects the ports from which of the RxD signals are to be AND'ed together for internal network mode. Bit 6 represents RxD from Port 7 and bit0 represents RxD from Port 1. 0 Includes RxDn for AND'ing 1 Excludes RxDn from AND'ing

42.3.2.7 Port Timing Control Register 4 (PTCR4)

PTCR4 is the Port Timing Control Register for Port 4. Figure 42-27 shows the register; Table 42-11 provides its field descriptions.

0x53FC_4018 (PTCR4) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	TFS DIR	TFSEL[3:0]				TCLK DIR	TCSEL[3:0]				RFS DIR		RFSEL[3:0]			RCL KDIR
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RCSEL[3:0]				SYN	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

Figure 42-27. Port Timing Control Register 4 (PTCR4)

Table 42-11. PTCR4 Field Descriptions

Field	Description
31 TFS DIR	Transmit Frame Sync Direction Control. This bit sets the direction of the TxFS pin of the interface as an output or input. When set as an input, the TFSEL settings are ignored. When set as an output, the TFSEL settings determine the source port of the frame sync. 0 TxFS is an input. 1 TxFS is an output.
30–27 TFSEL[3:0]	Transmit Frame Sync Select. Selects the source port from which TxFS is sourced. 0xxx Selects TxFS from port. 1xxx Selects RxFS from port. x000 Port 1 —— x110 Port 7 x111 Reserved
26 TCLKDIR	Transmit Clock Direction Control. This bit sets the direction of the TxClk pin of the interface as an output or input. When set as an input, the TCSEL settings are ignored. When set as an output, the TCSEL settings determine the source port of the clock. 0 TxClk is an input. 1 TxClk is an output.
25–22 TCSEL[3:0]	Transmit Clock Select. Selects the source port from which TxClk is sourced. 0xxx Selects RxClk from port. 1xxx Selects RxClk from port. x000 Port 1 —— x110 Port 7 x111 Reserved
21 RFS DIR	Receive Frame Sync Direction Control. This bit sets the direction of the RxFS pin of the interface as an output or input. When set as an input, the RFSEL settings are ignored. When set as an output, the RFSEL settings determine the source port of the frame sync. 0 RxFS is an input. 1 RxFS is an output.
20–17 RFSEL[3:0]	Receive Frame Sync Select. Selects the source port from which RxFS is sourced. RxFS can be sourced from TxFS and RxFS from other ports. 0xxx Selects TxFS from port. 1xxx Selects RxFS from port. x000 Port 1 —— x110 Port 7 x111 Reserved
16 RCLKDIR	Receive Clock Direction Control. This bit sets the direction of the RxClk pin of the interface as an output or input. When set as an input, the RCSEL settings are ignored. When set as an output, the RCSEL settings determine the source port of the clock. 0 RxClk is an input. 1 RxClk is an output.

Table 42-11. PTCR4 Field Descriptions (continued)

Field	Description
15–12 RCSEL[3:0]	Receive Clock Select. Selects the source port from which RxClk is sourced. RxClk can be sourced from TxClk and RxClk from other ports. 0xxx Selects TxClk from port. 1xxx Selects RxClk from port. x000 Port 1 —— x110 Port 7 x111 Reserved
11 SYN	AYN—Synchronous/Asynchronous Select. When SYN is set, synchronous mode is chosen and the transmit and receive sections use common clock and frame sync signals (that is, the port is a 4-wire interface). When SYN is cleared, asynchronous mode is chosen and separate clock and frame sync signals are used for the transmit and receive sections (that is, the port is a 6-wire interface). 0 Asynchronous mode 1 Synchronous mode (default)
10–0	Reserved

42.3.2.8 Port Data Control Register 4 (PDCR4)

PDCR4 is the Port Data Control Register for Port 4. Figure 42-28 shows the register; Table 42-12 provides its field descriptions.

0x53FC_401C (PDCR4) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RXDSEL[2:0]			TXR XEN	0	0	MODE[1:0]		INMMASK[7:0]							
W																
Reset	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 42-28. Port Data Control Register 4 (PDCR4)

Table 42-12. PDCR4 Field Descriptions

Field	Description
31–16	Reserved
15–13 RXDSEL[2:0]	Receive Data Select. Selects the source port for the RxD data. RXDSEL is ignored if MODE[1:0] is 01 (that is, Internal Network Mode is enabled). xxx Port number for RxD 000 Port 1 —— 110 Port 7 111 Reserved
12 TXRXEN	Transmit/Receive Switch Enable. Swaps the transmit and receive signals 0 No switch (Transmit Pin = Transmit, Receive Pin = Receive) 1 Switch (Transmit Pin = Receive, Receive Pin = Transmit)
11–10	Reserved
9–8 MODE[1:0]	Mode Select. This field selects the mode in which the port is to operate.The modes of operation include the following: • Normal mode, in which the RxD from the port selected by RXDSEL is routed to the port. • Internal Network mode in which RxD from other ports are ANDED together. RXDSEL is ignored. INMMASK determines which RxD signals are AND'ed together. • Bottom Connector Network mode, in which the port receives data from Bottom Connector port (Port 7) and any other port as selected by RXDSEL[3:0] in different time slots within a frame. The bc_bus_dis signal routes the data from Bottom Connector port if low otherwise data from the other port (as selected by RXDSEL) is routed to the port. 00 Normal mode 01 Internal Network mode 10 Bottom Connector Network mode 11 Reserved
7–0 INMMASK[7:0]	Internal Network Mode Mask. Bit mask that selects the ports from which of the RxD signals are to be AND'ed together for internal network mode. Bit 6 represents RxD from Port 7 and bit0 represents RxD from Port 1. 0 Includes RxDn for AND'ing 1 Excludes RxDn from AND'ing

42.3.2.9 Port Timing Control Register 5 (PTCR5)

PTCR5 is the Port Timing Control Register for Port 5. Figure 42-29 shows the register; Table 42-13 provides its field descriptions.

0x53FC_4020 (PTCR5) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TFS DIR	TFSEL[3:0]				TCLK DIR	TCSEL[3:0]				RFS DIR	RFSEL[3:0]				RCL KDIR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RCSEL[3:0]				SYN	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

Figure 42-29. Port Timing Control Register 5 (PTCR5)

Table 42-13. PTCR5 Field Descriptions

Field	Description
31 TFS DIR	Transmit Frame Sync Direction Control. This bit sets the direction of the TxFS pin of the interface as an output or input. When set as an input, the TFSEL settings are ignored. When set as an output, the TFSEL settings determine the source port of the frame sync. 0 TxFS is an input. 1 TxFS is an output.
30–27 TFSEL[3:0]	Transmit Frame Sync Select. Selects the source port from which TxFS is sourced. 0xxx Selects TxFS from port. 1xxx Selects RxFS from port. x000 Port 1 —— x110 Port 7 x111 Reserved
26 TCLKDIR	Transmit Clock Direction Control. This bit sets the direction of the TxClk pin of the interface as an output or input. When set as an input, the TCSEL settings are ignored. When set as an output, the TCSEL settings determine the source port of the clock. 0 TxClk is an input. 1 TxClk is an output.
25–22 TCSEL[3:0]	Transmit Clock Select. Selects the source port from which TxClk is sourced. 0xxx Selects RxClk from port. 1xxx Selects RxClk from port. x000 Port 1 —— x110 Port 7 x111 Reserved

Table 42-13. PTCR5 Field Descriptions (continued)

Field	Description
21 RFS DIR	Receive Frame Sync Direction Control. This bit sets the direction of the RxFS pin of the interface as an output or input. When set as an input, the RFSEL settings are ignored. When set as an output, the RFSEL settings determine the source port of the frame sync. 0 RxFS is an input. 1 RxFS is an output.
20–17 RFSEL[3:0]	Receive Frame Sync Select. Selects the source port from which RxFS is sourced. RxFS can be sourced from TxFS and RxFS from other ports. 0xxx Selects TxFS from port. 1xxx Selects RxFS from port. x000 Port 1 —— x110 Port 7 x111 Reserved
16 RCLKDIR	Receive Clock Direction Control. This bit sets the direction of the RxClk pin of the interface as an output or input. When set as an input, the RCSEL settings are ignored. When set as an output, the RCSEL settings determine the source port of the clock. 0 RxClk is an input. 1 RxClk is an output.
15–12 RCSEL[3:0]	Receive Clock Select. Selects the source port from which RxClk is sourced. RxClk can be sourced from TxClk and RxClk from other ports. 0xxx Selects TxClk from port. 1xxx Selects RxClk from port. x000 Port 1 —— x110 Port 7 x111 Reserved
11 SYN	SYN—Synchronous/Asynchronous Select. When SYN is set, synchronous mode is chosen and the transmit and receive sections use common clock and frame sync signals (that is, the port is a 4-wire interface). When SYN is cleared, asynchronous mode is chosen and separate clock and frame sync signals are used for the transmit and receive sections (that is, the port is a 6-wire interface). 0 Asynchronous mode 1 Synchronous mode (default)
10–0	Reserved

42.3.2.10 Port Data Control Register 5 (PDCR5)

Figure 42-30 shows the Port Data Control Register 5 (PDCR5) register; Table 42-14 provides its field descriptions.

0x53FC_4024 (PDCR5) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RXDSEL[2:0]			TXR XEN	0	0	MODE[1:0]		INMMASK[7:0]							
W																
Reset	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 42-30. Port Data Control Register 5 (PDCR5)

Table 42-14. PDCR5 Field Descriptions

Field	Description
31–16	Reserved
15–13 RXDSEL[2:0]	Receive Data Select. Selects the source port for the RxD data. RXDSEL is ignored if MODE[1:0] is 01 (that is, Internal Network Mode is enabled). xxx Port number for RxD 000 Port 1 ——— 110 Port 7 111 Reserved
12 TXRXEN	Transmit/Receive Switch Enable. Swaps the transmit and receive signals 0 No switch (Transmit Pin = Transmit, Receive Pin = Receive) 1 Switch (Transmit Pin = Receive, Receive Pin = Transmit)
11–10	Reserved

Table 42-14. PDCR5 Field Descriptions (continued)

Field	Description
9–8 MODE[1:0]	Mode Select. This field selects the mode in which the port is to operate.The modes of operation include the following: • Normal mode, in which the RxD from the port selected by RXDSEL is routed to the port. • Internal Network mode in which RxD from other ports are ANDED together. RXDSEL is ignored. INMMASK determines which RxD signals are AND'ed together. • Bottom Connector Network mode, in which the port receives data from Bottom Connector port (Port 7) and any other port as selected by RXDSEL[3:0] in different time slots within a frame. The bc_bus_dis signal routes the data from Bottom Connector port if low otherwise data from the other port (as selected by RXDSEL) is routed to the port. 00 Normal mode 01 Internal Network mode 10 Bottom Connector Network mode 11 Reserved
7–0 INMMASK[7:0]	Internal Network Mode Mask. Bit mask that selects the ports from which of the RxD signals are to be AND'ed together for internal network mode. Bit 6 represents RxD from Port 7 and bit0 represents RxD from Port 1. 0 Includes RxDn for AND'ing 1 Excludes RxDn from AND'ing

42.3.2.11 Port Timing Control Register 6 (PTCR6)

PTCR6 is the Port Timing Control Register for Port 6. Figure 42-31 shows the register; Table 42-15 provides its field descriptions.

0x53FC_4028 (PTCR6) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	TFS DIR	TFSEL[3:0]				TCLK DIR	TCSEL[3:0]				RFS DIR		RFSEL[3:0]			RCL KDIR
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RCSEL[3:0]				SYN	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

Figure 42-31. Port Timing Control Register 6 (PTCR6)

Table 42-15. PTCR6 Field Descriptions

Field	Description
31 TFS DIR	Transmit Frame Sync Direction Control. This bit sets the direction of the TxFS pin of the interface as an output or input. When set as an input, the TFSEL settings are ignored. When set as an output, the TFSEL settings determine the source port of the frame sync. 0 TxFS is an input. 1 TxFS is an output.
30–27 TFSEL[3:0]	Transmit Frame Sync Select. Selects the source port from which TxFS is sourced. 0xxx Selects TxFS from port. 1xxx Selects RxFS from port. x000 Port 1 —— x110 Port 7 x111 Reserved
26 TCLKDIR	Transmit Clock Direction Control. This bit sets the direction of the TxClk pin of the interface as an output or input. When set as an input, the TCSEL settings are ignored. When set as an output, the TCSEL settings determine the source port of the clock. 0 TxClk is an input. 1 TxClk is an output.
25–22 TCSEL[3:0]	Transmit Clock Select. Selects the source port from which TxClk is sourced. 0xxx Selects RxClk from port. 1xxx Selects RxClk from port. x000 Port 1 —— x110 Port 7 x111 Reserved
21 RFS DIR	Receive Frame Sync Direction Control. This bit sets the direction of the RxFS pin of the interface as an output or input. When set as an input, the RFSEL settings are ignored. When set as an output, the RFSEL settings determine the source port of the frame sync. 0 RxFS is an input. 1 RxFS is an output.
20–17 RFSEL[3:0]	Receive Frame Sync Select. Selects the source port from which RxFS is sourced. RxFS can be sourced from TxFS and RxFS from other ports. 0xxx Selects TxFS from port. 1xxx Selects RxFS from port. x000 Port 1 —— x110 Port 7 x111 Reserved
16 RCLKDIR	Receive Clock Direction Control. This bit sets the direction of the RxClk pin of the interface as an output or input. When set as an input, the RCSEL settings are ignored. When set as an output, the RCSEL settings determine the source port of the clock. 0 RxClk is an input. 1 RxClk is an output.

Table 42-15. PTCR6 Field Descriptions (continued)

Field	Description
15–12 RCSEL[3:0]	Receive Clock Select. Selects the source port from which RxClk is sourced. RxClk can be sourced from TxClk and RxClk from other ports. 0xxx Selects TxClk from port. 1xxx Selects RxClk from port. x000 Port 1 —— x110 Port 7 x111 Reserved
11 SYN	SYN—Synchronous/Asynchronous Select. When SYN is set, synchronous mode is chosen and the transmit and receive sections use common clock and frame sync signals (that is, the port is a 4-wire interface). When SYN is cleared, asynchronous mode is chosen and separate clock and frame sync signals are used for the transmit and receive sections (that is, the port is a 6-wire interface). 0 Asynchronous mode 1 Synchronous mode (default)
10–0	Reserved

42.3.2.12 Port Data Control Register 6 (PDCR6)

PDCR6 is the Port Data Control Register for Port 6. Figure 42-32 shows the register; Table 42-16 provides its field descriptions.

0x53FC_402C (PDCR6) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RXDSEL[2:0]			TXR XEN	0	0	MODE[1:0]		INMMASK[7:0]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 42-32. Port Data Control Register 6 (PDCR6)

Table 42-16. PDCR6 Field Descriptions

Field	Description
31–16	Reserved
15–13 RXDSEL[2:0]	Receive Data Select. Selects the source port for the RxD data. xxx Port number for RxD 000 Port 1 —— 110 Port 7 111 Reserved
12 TXRXEN	Transmit/Receive Switch Enable. Swaps the transmit and receive signals 0 No switch (Transmit Pin = Transmit, Receive Pin = Receive) 1 Switch (Transmit Pin = Receive, Receive Pin = Transmit)
11–10	Reserved
9–8 MODE[1:0]	Mode Select. This field selects the mode in which the port is to operate.The modes of operation include the following: • Normal mode, in which the RxD from the port selected by RXDSEL is routed to the port. • Internal Network mode in which RxD from other ports are AND'ed together. RXDSEL is ignored. INMMASK determines which RxD signals are AND'ed together. • Bottom Connector Network mode, in which the port receives data from Bottom Connector port (Port 7) and any other port as selected by RXDSEL[3:0] in different time slots within a frame. The bc_bus_dis signal routes the data from Bottom Connector port if low otherwise data from the other port (as selected by RXDSEL) is routed to the port. 00 Normal mode 01 Internal Network mode 10 Bottom Connector Network mode 11 Reserved
7–0 INMMASK[7:0]	Internal Network Mode Mask. Bit mask that selects the ports from which of the RxD signals are to be AND'ed together for internal network mode. Bit 6 represents RxD from Port 7 and bit0 represents RxD from Port 1. 0 Includes RxDn for AND'ing 1 Excludes RxDn from AND'ing

42.3.2.13 Port Timing Control Register 7 (PTCR7)

PTCR7 is the Port Timing Control Register for Port 7. Figure 42-33 shows the register; Table 42-17 provides its field descriptions.

0x53FC_4030 (PTCR7) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	TFS DIR	TFSEL[3:0]				TCLK DIR	TCSEL[3:0]				RFS DIR	RFSEL[3:0]				RCL KDIR
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RCSEL[3:0]				SYN	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

Figure 42-33. Port Timing Control Register 7 (PTCR7)

Table 42-17. PTCR7 Field Descriptions

Field	Description
31 TFS DIR	Transmit Frame Sync Direction Control. This bit sets the direction of the TxFS pin of the interface as an output or input. When set as an input, the TFSEL settings are ignored. When set as an output, the TFSEL settings determine the source port of the frame sync. 0 TxFS is an input. 1 TxFS is an output.
30–27 TFSEL[3:0]	Transmit Frame Sync Select. Selects the source port from which TxFS is sourced. 0xxx Selects TxFS from port. 1xxx Selects RxFS from port. x000 Port 1 —— x110 Port 7 x111 Reserved
26 TCLKDIR	Transmit Clock Direction Control. This bit sets the direction of the TxClk pin of the interface as an output or input. When set as an input, the TCSEL settings are ignored. When set as an output, the TCSEL settings determine the source port of the clock. 0 TxClk is an input. 1 TxClk is an output.
25–22 TCSEL[3:0]	Transmit Clock Select. Selects the source port from which TxClk is sourced. 0xxx Selects RxClk from port. 1xxx Selects RxClk from port. x000 Port 1 —— x110 Port 7 x111 Reserved

Table 42-17. PTCR7 Field Descriptions (continued)

Field	Description
21 RFS DIR	Receive Frame Sync Direction Control. This bit sets the direction of the RxFS pin of the interface as an output or input. When set as an input, the RFSEL settings are ignored. When set as an output, the RFSEL settings determine the source port of the frame sync. 0 RxFS is an input. 1 RxFS is an output.
20–17 RFSEL[3:0]	Receive Frame Sync Select. Selects the source port from which RxFS is sourced. RxFS can be sourced from TxFS and RxFS from other ports. 0xxx Selects TxFS from port. 1xxx Selects RxFS from port. x000 Port 1 —— x110 Port 7 x111 Reserved
16 RCLKDIR	Receive Clock Direction Control. This bit sets the direction of the RxClk pin of the interface as an output or input. When set as an input, the RCSEL settings are ignored. When set as an output, the RCSEL settings determine the source port of the clock. 0 RxClk is an input. 1 RxClk is an output.
15–12 RCSEL[3:0]	Receive Clock Select. Selects the source port from which RxClk is sourced. RxClk can be sourced from TxClk and RxClk from other ports. 0xxx Selects TxClk from port. 1xxx Selects RxClk from port. x000 Port 1 — x110 Port 7 x111 Reserved
11 SYN	Synchronous/Asynchronous Select. When SYN is set, synchronous mode is chosen and the transmit and receive sections use common clock and frame sync signals (that is, the port is a 4-wire interface). When SYN is cleared, asynchronous mode is chosen and separate clock and frame sync signals are used for the transmit and receive sections (that is, the port is a 6-wire interface). 0 Asynchronous mode 1 Synchronous mode (default)
10–0	Reserved

42.3.2.14 Port Data Control Register 7 (PDCR7)

PDCR7 is the Port Data Control Register for Port7. Figure 42-34 shows the register; Table 42-18 provides its field descriptions.

0x53FC_4034 (PDCR7) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RXDSEL[2:0]			TXR XEN	0	0	MODE[1:0]		INMMASK[7:0]							
W																
Reset	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 42-34. Port Data Control Register 7 (PDCR7)

Table 42-18. PDCR7 Field Descriptions

Field	Description
31–16	Reserved
15–13 RXDSEL[2:0]	Receive Data Select. Selects the source port for the RxD data. xxx Port number for RxD 000 Port 1 — 110 Port 7 111 Reserved
12 TXRXEN	Transmit/Receive Switch Enable. Swaps the transmit and receive signals 0 No switch (Transmit Pin = Transmit, Receive Pin = Receive) 1 Switch (Transmit Pin = Receive, Receive Pin = Transmit)
11–10	Reserved

Table 42-18. PDCR7 Field Descriptions (continued)

Field	Description
9–8 MODE[1:0]	Mode Select. This field selects the mode in which the port is to operate.The modes of operation include the following: • Normal mode, in which the RxD from the port selected by RXDSEL is routed to the port. • Internal Network mode in which RxD from other ports are AND'ed together. RXDSEL is ignored. INMMASK determines which RxD signals are AND'ed together. • Bottom Connector Network mode, in which the port receives data from Bottom Connector port (Port7) and any other port as selected by RXDSEL[3:0] in different time slots within a frame. The bc_bus_dis signal routes the data from Bottom Connector port if low otherwise data from the other port (as selected by RXDSEL) is routed to the port. 00 Normal mode 01 Internal Network mode 10 Bottom Connector Network mode 11 Reserved
7–0 INMMASK[7:0]	Internal Network Mode Mask. Bit mask that selects the ports from which of the RxD signals are to be AND'ed together for internal network mode. Bit 6 represents RxD from Port 7 and bit0 represents RxD from Port 1. 0 Includes RxDn for AND'ing 1 Excludes RxDn from AND'ing

42.3.2.15 Bottom Connector Network Mode Control Register (CNMCR)

CNMCR is the Bottom Connector Network Mode control register. Figure 42-35 shows the register; Table 42-19 provides its field descriptions.

0x53FC_4038 (CNMCR) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	BEN	FSP OL	CLK POL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CNTHI[7:0]								CNTLOW[7:0]							
W																
Reset	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0

Figure 42-35. Bottom Connector Network Mode Control Register (CNMCR)

Table 42-19. CNMCR Field Descriptions

Field	Description
31–19	Reserved
18 BEN	Bottom Connector enable. This signal controls the generation of bc_bus_dis signal. If set, the bc_bus_dis signal is generated as per the CNTHI and CNTLOW settings. 0 Bottom Connector disable signal is held low. 1 Bottom Connector disable signal is generated.
17 FSPOL	Frame Sync Polarity Select. This field selects the frame sync polarity of the Bottom Connector port (Port 7) used for the generation of bc_bus_dis signal. 0 Polarity 0 1 Polarity 1
16 CLKPOL	Frame Sync Polarity Select. This field selects the bit clock polarity of the Bottom Connector port (Port 7) used for generation of bc_bus_dis signal. 0 Polarity 0 1 Polarity 1

Table 42-19. CNMCR Field Descriptions (continued)

Field	Description
15–8 CNTHI[7:0]	Bottom Connector disable signal high period count. This field selects the number of bit clocks for which the Bottom Connector disable signal bc_bus_dis is to remain high following the detection of the frame sync, that is, when Bottom Connector is not transferring data. This allows the user to specify the time until the Bottom Connector starts transmitting (with respect to the beginning of timeslot 0). 00000000 0 bit clock period 00000001 1 bit clock period 00000010 2 bit clock period —— 11111111= 255 bit clock period • If operating in 4-wire mode (SYN is set), the clock used for bc_bus_dis generation is selected by the TCSEL[3:0] field of Port 7's control register PTCR7. The frame sync used for bc_bus_dis generation is selected by the TFSEL[3:0] field of Port 7's control register PTCR7. If operating in 6-wire mode (SYN is clear), the clock used for bc_bus_dis generation is selected by the RCSEL[3:0] field of Port 7's control register PTCR7. The frame sync used for bc_bus_dis generation is selected by the RFSEL[3:0] field of Port 7's control register PTCR7.
7–0 CNTLOW[7:0]	Bottom Connector disable signal low period count. This field selects the number of bit clocks for which the Bottom Connector disable signal bc_bus_dis is to remain low, that is, when Bottom Connector is transferring data. 00000000 = 0 bit clock period 00000001 = 1 bit clock period 00000010 = 2 bit clock period —— 11111111 = 255 bit clock period • If operating in 4-wire mode (SYN is set), the clock used for bc_bus_dis generation is selected by the TCSEL[3:0] field of Port 7's control register PTCR7. The frame sync used for bc_bus_dis generation is selected by the TFSEL[3:0] field of Port 7's control register PTCR7. If operating in 6-wire mode (SYN is clear), the clock used for bc_bus_dis generation is selected by the RCSEL[3:0] field of Port 7's control register PTCR7. The frame sync used for bc_bus_dis generation is selected by the RFSEL[3:0] field of Port 7's control register PTCR7.

42.3.2.15.1 Limitations of bc_bus_dis Signal Generation

Certain restrictions are in place for the utilization of Bottom Connector network mode. They are as follows:

- Only early frame syncs are supported.
- Bottom Connector data transfers only take place in contiguous time slots.
- Only one port can be connected to the Bottom Connector port at a time.
- Bottom Connector Network mode can be used for only two ports (Port 7 and one other port)

Assumptions:

- Transmission of Data by Bottom Connector and frame sync generation by the master (Bottom Connector/Port) occurs on the same clock edge. This assures a 1/2-bit delay between the moment

the RxD lines are switched inside the AUDMUX (as driven by bc_bus_dis) and the moment when the receive data is sampled by the serial interface.

42.3.3 AUDMUX Default Configuration

The AUDMUX reverts back to its default settings following a reset. Section 42.3.3.1, “Default Port Configuration” and Section 42.3.3.2, “Default Bottom Connector Configuration” describe the default configuration of the ports and Bottom Connector, respectively.

42.3.3.1 Default Port Configuration

The AUDMUX default port configuration after reset is as follows:

- Port 1 connected to Port 6
  - Port 6 provides the clock and frame sync.
  - Synchronous mode is enabled.
  - Normal mode is selected.
- Port 2 connected to Port 5
  - Port 5 provides the clock and frame sync.
  - Synchronous mode is enabled.
  - Normal mode is selected.
- Port 3 connected to Port 4
  - Port 4 provides the clock and frame sync.
  - Synchronous mode is enabled.
  - Normal mode is selected.
- Port 7 in data loopback mode
  - Clock and frame syncs are inputs.
  - Synchronous mode is enabled.
  - Normal mode is selected.

42.3.3.2 Default Bottom Connector Configuration

The default configuration of all the ports is to set the MODE field to normal mode (that is, no port selects Bottom Connector network mode). Correspondingly, the default configuration of the BEN field in the CNMCR is clear. To minimize AUDMUX setup for audio testing, the rest of the CNMCR fields default to the following configuration:

- FSPOL is set.
- CLKPOL is set.
- CNTHI is set to 16.
- CNTLOW is set to 16.

This configuration uses a frame sync that is logic high when asserted.

The clock is driven by the transmitter on the rising edge and is sampled by the receiver on the falling edge. The AUDMUX switches between devices on the rising edge; this provides a buffer of one-half clock period between the moment data is sampled and when the AUDMUX switches between devices. Refer to Figure 42-13 for a timing diagram where FSPOL and CLKPOL are both set.

CNTHI and CNTLOW are set to 16 to allow Bottom Connector to drive data during timeslot 1 and the other device to drive data during timeslots 0, 2, . . . , N-1 where N is the number of timeslots used and each timeslot has 16 bits.

42.4 AUDMUX Clocking

This section provides information about AUDMUX clocking, including clock inputs and the clock diagram.

42.4.1 AUDMUX Clock Inputs

The IP Bus read/write clock—ipg_clk_s—is an input to the AUDMUX. It is used for all AUDMUX register accesses. It is driven only when there is an AUDMUX access on the IP Bus.

The AUDMUX uses the selected Tx/Rx bit clock to drive the synchronous switching of the Bottom Connector Network mode. Specifically, this clock is used to drive the bc_bus_dis signal generation logic. The bit clock used for this function can come from either internal serial interfaces (that is, SSI1 or SSI2) or external CODECs. The clock used for Bottom Connector Network mode is determined by PTCR7. Refer to “Bottom Connector Network Mode Control Register (CNMCR)” for more details on the clock selection for Bottom Connector Network mode.

42.4.2 AUDMUX Clock Diagram

Figure 42-36 shows the clocking used in the AUDMUX.

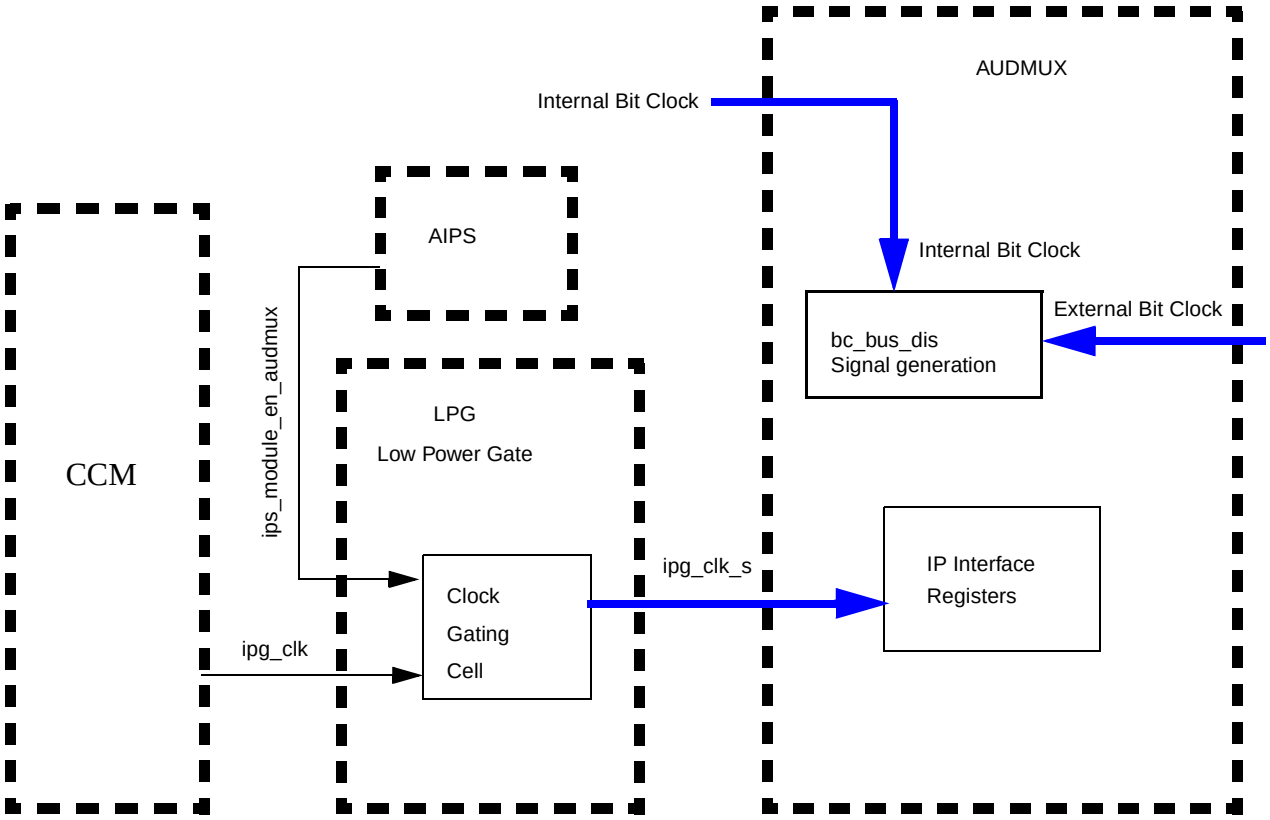


Figure 42-36. AUDMUX Clocking Scheme

42.4.3 Clocking Restrictions

The following are the clocking restrictions:

- Since the AUDMUX requires only ipg_clk_s, the AUDMUX places no restrictions on the bus frequency.
- All registers in the AUDMUX are control registers so their values will not change frequently. Their values will be programmed when changing between use cases (not during use cases).

Chapter 43
Moving Pictures Experts Group-4 (MPEG-4) Encoder

The MPEG-4 Encoder in the i.MX31 supports MPEG-4 SP and the H.263 baseline formats that are fully hardware accelerated, supporting resolutions up to VGA at 30 fps. As a result, a high degree of power efficiency is achieved, and the CPU is freed to perform other tasks. Encoding for other video standards are achieved using software implementation.

43.1 Overview

The video encoding hardware accelerator of the i.MX31 and i.MX31L processors support MPEG-4 Simple Profile (all levels) and H.263 Baseline. They enable pixel rates up to VGA at 30 fps and compressed bit rate up to 4 Mbps.

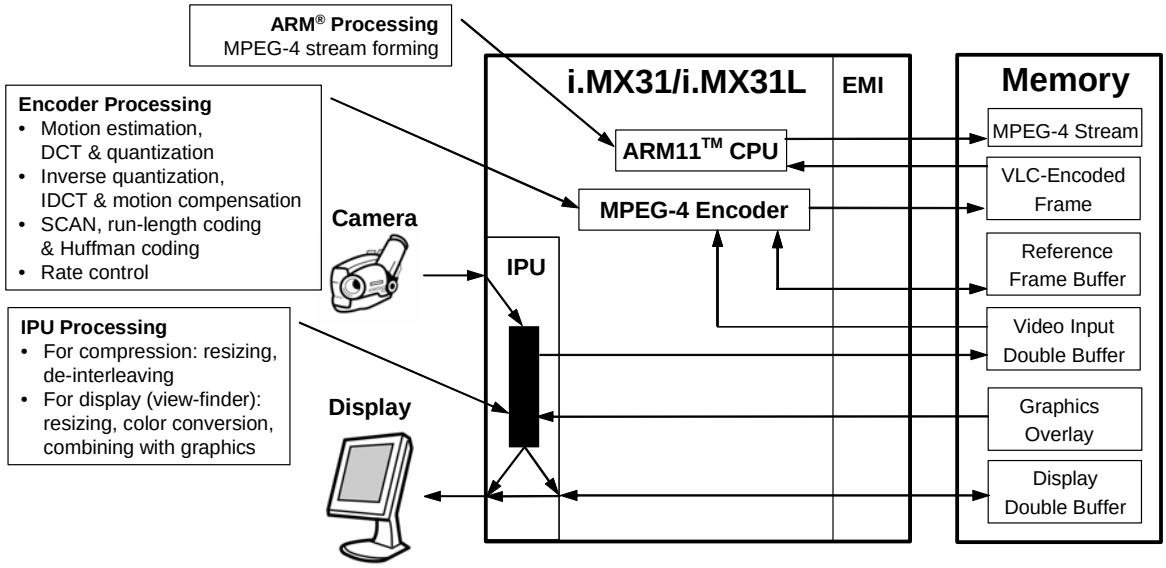


Figure 43-1. MPEG-4 SP Video Capturing—Data Flow

43.2 MPEG-4 Video Encoder

The MPEG-4 encoder accelerates video compression, following the MPEG-4 standard.

The encoder has the following main features:

- Compression formats: MPEG-4 simple profile (all levels), H.263 baseline
- Pixel rate: a maximum of VGA at 30 fps
- Compressed bit-rate: a maximum of 4 Mbps

- Essentially performs the complete video processing chain, generating a Huffman-coded stream. Only the formation of the final MPEG-4 stream is performed by the processor.
- Additional processing:
  - — Picture smoothening (low-pass filter)
  - — Camera movement stabilization
  - — Enhanced conference call format, which inserts additional information in the MPEG stream. An MPEG-4 decoder uses the additional information to improve p

43.3 Functional Description

Generally speaking the module encodes the MPEG-4 simple profile video bitstream with a proprietary VGA (640 x 280 pixels) extension, if the module is used in conjunction with Application Programming Interface 2.3 for MPEG-4 and H.263 Encoder. This API explains the different encoding modes of the block. It is important to be aware that the HW module does not create MPEG-4 compliant bitstream because the final bit-stream packing is done in the SW-API.

Chapter 44
Image Processing Unit (IPU)

44.1 Introduction

The Image Processing Unit (IPU) is designed to support video and graphics processing functions and to interface with video/still image sensors and displays.

Figure 44-1 is a block diagram of the IPU.

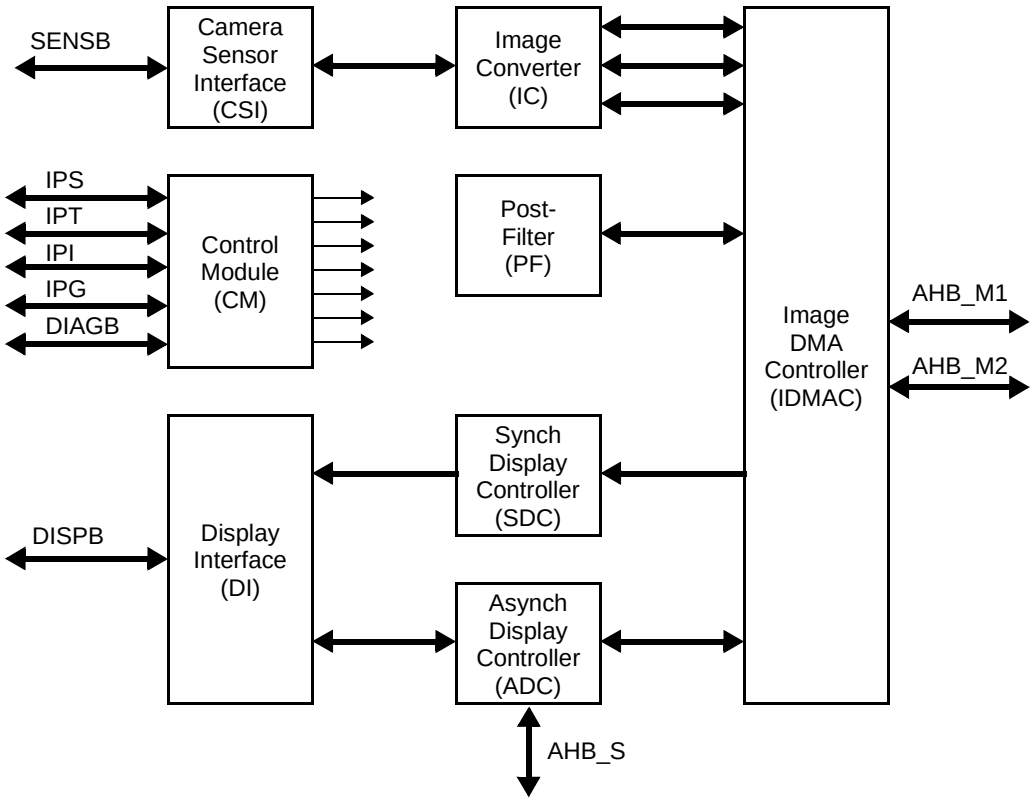


Figure 44-1. IPU Block Diagram

44.1.1 Overview

44.1.1.1 Performed Functions

The IPU performs the following main functions:

- Capturing image data from a camera sensor or from a TV decoder. The captured image can be sent to preprocessing or stored in an external system memory for additional processing by the MCU.
- Preprocessing of data from the sensor or from the external system memory. There are two preprocessing channels according to the data destination - an encoder or a display (viewfinder mode). Preprocessing includes the following operations:
  - Downsizing with independent integer horizontal and vertical ratios
  - Resizing with independent fractional horizontal and vertical ratios
  - Color space conversion (YUV to RGB, RGB to YUV, YUV to different YUV)
  - Combining a video plane with a graphics plane (blending on graphics on top of video plane)
  - 90 degree rotation, up/down and left/right flipping of the image.
- Postprocessing of data from the external system memory. The MCU can invoke a number of postprocessing channels sequentially by re-programming the IPU after finish of previous channel frame processing. Postprocessing includes the following operations:
  - Downsizing with independent integer horizontal and vertical ratios
  - Resizing with independent fractional horizontal and vertical ratios
  - Color space conversion (YUV to RGB, RGB to YUV, YUV to different YUV)
  - Combining a video plane with a graphics plane (blending on graphics on top of video plane)
  - 90 degree rotation, up/down and left/right flipping of the image.
- Post-filtering of data from the system memory with support of the MPEG-4 (both deblocking and deringing) and H.264 postfiltering algorithms.
- Displaying video and graphics on a synchronous (dump or memory-less) display with the following options:
  - On the fly combining video and graphics planes
  - Generation of the hardware cursor
  - Horizontal and vertical scrolling
- Displaying video and graphics on an asynchronous (smart) display. There are two mechanisms to support smart display or graphic accelerator functionality: interleaving data and commands from a command buffer prepared by the MCU or automatic commands generation according to a prepared template. The data can be sent to the smart display from the system memory, internal IPU processing modules or directly from the MCU or the system DMA controller.
- Transferring data between IPU submodules and to/from the system memory with flexible pixels re-formatting.

44.1.1.2 External Interfaces

The IPU is an autonomous module controlled by the MCU via the IP Skyblue Bus (IPS). Data from the sensor arrives to the IPU via the Sensor Bus (SENSB). Writing results to the external system memory and reading a previously stored data or a data received from communication channel can be performed via Master AHB Port(s). There are two possible operating modes corresponding to single and dual Master AHB Port. For dual mode, the IPU uses the ports in turn. This allows to improve data throughput in the Memory Interface by means of latency hiding. One of modes is selected by an external pin and a control bit in the peripheral bus register. If the pin sets single port mode, the control bit has no effect.

Additionally, the MCU can communicate directly with the IPU through the Slave AHB Port (AHB_S). The IPU output is sent to synchronous or asynchronous displays or to a TV encoder through the Display Bus (DISPB).

The IPU generates interrupts for the MCU via the IP Indigo Bus (IPI). The IP Green Bus provides system clock and reset signals. The IP Tan Bus is used to support scan and BIST functionality of the module. For debugging purposes, the IPU has a special bus - the Diagnostic Bus (DIAGB). This bus allows real-time monitoring internal signals. It is output via chip GPIO pins in debug mode.

Detailed description of all signals of the interface buses is found in Section 44.2.2, “Detailed Signal Descriptions.” Specific connection of the buses to the MCU platform, the Memory Controller, the Clock Controller etc. is chip dependent. It should be described in the chip specification integration part.

44.1.1.3 Data Flows and Formats

The IPU consists of the Camera Sensor Interface (CSI), the Image Converter (IC), the Post-Filter (PF), the Synchronous Display Controller (SDC), the Asynchronous Display Controller (ADC), the Display Interface (DI) and the Image DMA Controller (IDMAC).

The sensor data is fed to the CSI. The CSI output is a 64-bit word that includes two pixels (8- or 10-bits per color component) or two/eight words of generic data (15 most significant bits per 16-bit words or 8-bits per word). The IPU does not control the sensor. This function has to be performed by the MCU via the I^2C interface and GPIO pins connected to the sensor.

The sensor data arrives to the IC from the CSI. The CSI converts the data to one of the following formats:

- YUV 4:4:4 or RGB—8 bits per color component
- YUV 4:4:4 or RGB—10 bits per color component
- Generic data (from sensor to the system memory only)—8 or 16 bits per word (in the case of 16-bit words, only the 15 most significant bits are used)

The IC executes pre- and postprocessing tasks. There are two preprocessing and one postprocessing tasks performed by the IC in time multiplexed mode. The MCU programs the tasks parameters. Task context switching is transparent for the MCU.

The IC communicates with the IDMAC via three buses. The first bus is used for storing the sensor data in the system memory and for loading video data from the memory for pre- and postprocessing. The following data formats are supported for this bus:

- YUV 4:4:4 or RGB—8 bits per color component

- YUV 4:4:4 or RGB—10 bits per color component
- Generic data (from sensor to the system memory only)—8 or 16 bits per word (in the case of 16-bit words, only 15 most significant bits are used)

The data bus width is 64 bits for write and 48 bits for read.

The second bus is intended for loading graphics data and output processing results to the system memory or the ADC. The following data formats are supported for this bus:

- YUV 4:4:4—8 bits per color component
- RGB—8 bits per color component
- RGBA—8 bits per color component (only for graphics)

The data bus width is 48 bits for write and 64 bits for read.

The third bus transfers image data to be rotated from the system memory to the IC and returns the rotated image to the memory. The width of each of the buses corresponds to two pixels in the following formats

- YUV 4:4:4—8 bits per color component
- RGB—8 bits per color component
- RGBA—8 bits per color component (only for graphics)

The bus data width is 64 bits both for read and write.

The PF implements the MPEG-4 and H.264 post-filtering algorithms. It gets the input data from the system memory and returns it to the memory via the 64-bit bus connected to the IDMAC. For MPEG-4 mode, there are two filtering steps: deblocking and deringing. For H.264, only deblocking is performed. The bus transfers simultaneously eight components of one color (Y or U or V) related to eight pixels.

The SDC is designed to support memory-less synchronous displays and synchronous interfaces of smart displays. The supported display types are TFT and TV. For TV mode, the pixels are presented in the YUV 4:2:2 format, for TFT modes—in the RGB or monochrome format. The SDC receives a data from the IDMAC via a 64-bits bus. The IDMAC is responsible to convert the external memory data to one of the following bus formats:

- Two pixels in the RGB format, 8 bits per color component—for color video
- Two pixels in the RGBA format, 8 bits per color component—for color graphics
- Two pixels in the YUV 4:4:4 format, 8 bits per color component—for color video and a TV display
- Eight pixels in the monochrome format, 8 bits per pixel—for monochrome video
- Four pixels in the monochrome format, 16 bits per pixel (8 bits pixel data and 8 bits a-parameter)—for monochrome graphics
- Sixty four control bits—for the windowing function

The SDC combines video and graphics planes before sending data to a display. Combining is performed with α-blending. An image is combined also with a hardware cursor of programmable color and other attributes. The SDC supports the windowing function by means of display enable control. In addition to data, the SDC generates display controls with programmable timing.

The ADC controls asynchronous (smart) displays and external graphics accelerators. It supports both write and read access modes to the display memory. Write/read data can be transferred from/to the system

memory or from the IC (write only) via the IDMAC. Another option is direct writing/reading by the MCU or the system DMA controller via the AHB_S bus. The bus connected to the IDMAC is of the 64-bits width. It can transfer data in the following formats:

- Pixel data—Two pixels in RGB format, 24 bits/pixel (located in 32-bit word each one), each pixel is aligned to the MSB bits of a 32-bit word.
- Generic data—Two data words of 24 bits/word (located in 32-bit word each one) or four data words of 16 bits/word, each data word is aligned to the LSB bits of a 32-bit word (the MCU software is responsible for the proper format of generic data).
- Command—Two data words of 24 bits/word (located in 32-bit word each one) or four data words of 16 bits/word, each command word is aligned to the LSB bits of a 32-bit word.

Because smart displays are represented by two interface registers—one index register and one data register, the ADC generates a sequence of commands and combines them with data for both write and read operations. There are two modes of display access: streaming commands from a memory buffer prepared by the MCU and automatic emulation of transparent writing/reading using access templates. The ADC is able to work with two types of displays: with full linear addressing and X/Y addressing.

The ADC includes a special mechanism to avoid image tearing. This mechanism is operating while transferring data from the system memory or directly from postprocessing and in a special case—directly from preprocessing.

The DI combines outputs of the SDC and the ADC. The DI output bus (DISPB) is shared to support simultaneously one synchronous display and up to three devices like smart display in time multiplexed mode. The DI maps the SDC and ADC outputs to the DISPB and synchronizes the SDC and the ADC. If both types of displays are used, access to the smart displays is performed during vertical blanking intervals of the memory-less display. Typically, these intervals occupy up to 15% of refresh cycle. This is the limitation for throughput of the asynchronous interface in such mode. For a dual-port smart display, the display memory can be accessed both through the synchronous and asynchronous interfaces.

The DI provides programmable display access timings both for fast and slow displays. Control signals and data polarity is also programmable. The DI performs very flexible packing and unpacking display data which allows to support different display interfaces. For example, the DI can pack the data to the RGB565 format.

The IDMAC provides data transfer of two types:

- Between internal IPU modules
- Between IPU modules and the system memory.

The IDMAC main features are:

- General features
  - 32 full programmable DMA channels with two groups of channel priorities and random choice in a group
  - Programmable AHB burst length—from 1 to 9 words
  - End-of-frame interrupt generation for each channel
  - Error generation for each channel

  - Big and Little Endian AHB interface mode
- Addressing features
  - Transfer of two-dimensional image frames line-by-line or by two-dimensional blocks with programmable vertical overlap
  - Interleaved and non-interleaved data addressing modes
  - Panning with data access resolution of single pixel,
  - Frame rotation and flipping left/right option when transferring two-dimensional blocks
  - frame up/down flipping option
  - Line interlacing
  - Frame scrolling
  - Frame double buffering within a DMA channel when accessing the system memory
  - End of frame indication
- Data conversion features
  - Conversion of the data format from the system memory to the internal IPU formats (YUV 4:4:4, 8 bits/color or RGBA, 8 bits/color or RGB, 8 bits/color) including interleaving non-interleaved YUV pixels, programmable unpacking RGBA or RGB pixels, programmable look-up-table decoding color pixels from a palette. Conversion of the external formats YUV 4:2:0 and YUV 4:2:2 to the internal format YUV 4:4:4 is performed by repetition of the existing U and V color components.
  - Conversion of the internal data formats to system memory formats including de-interleaving YUV pixels, programmable packing RGBA or RGB pixels. Conversion of the internal format YUV 4:4:4 to the external formats YUV 4:2:0 and YUV 4:2:2 is performed by decimation of the U and V color components.
  - Transfer of a generic data without conversion

The IDMAC accesses the AHB_M1(2) buses with bursts of 32-bit words transferred every clock cycle. Internal data transfers can be performed every second clock cycle but, because the width of the internal buses is 48 or 64 bit, throughput is the same in the case of non-packed and non-coded AHB data.

The CM performs common IPU service functions: interfacing to the peripheral SkyBLue IP bus, interrupt generation, debugging, clock and reset generation, access to internal memories. Additionally, the CM synchronizes IPU tasks triggering and transferring image frames between internal IPU modules and the system memory.

44.1.1.4 Clocking Scheme

44.1.1.4.1 General

Table 44-1 describes the IPU clocks.

Table 44-1. IPU Clocks

Name	Symbol	Source	Destination	Rate	Description
AHB clock	HCLK	Clock Controller	Master and Slave AHB IPU interfaces	Up to 133 MHz	Input clock derived by integer division of high frequency MCU clock (~500 MHz)
Green Line IP clock	IPG_CLK	Clock Controller	IP Skyblue IPU interface	Up to 66 MHz	Input clock derived by division of HCLK clock by 1 or 2
High-speed processing clock	HSP_CLK	Clock Controller	all IPU submodules	Up to 178 MHz @ 1.45 V, up to 133 MHz @ 1.1 V, equal or higher than HCLK rate	Input clock derived by integer division of high frequency MCU clock (~500 MHz), rising edges are aligned with HCLK and IPG_CLK rising edges
Sensor clock	SENSB_SENS_CLK	Clock Controller	CSI	Max rate at least 73 MHz (see Section 44.4.1.5, “Sensor Interface Control”)	Clock Controller should provide the required clock frequency precision (about +/-3%)
Master clock to sensor	SENSB_MCLK	CSI	Sensor	Max rate at least 73 MHz (see Section 44.4.1.5, “Sensor Interface Control”)	Output clock derived by repeating the SENSB_SENS_CLK clock or by integer division of the HSP_CLK clock
Pixel clock from sensor	SENSB_PIX_CLK	Sensor	CSI	Max rate at least 73 MHz (see Section 44.4.1.5, “Sensor Interface Control”)	Input clock used as sensor data strobe

Table 44-1. IPU Clocks (continued)

Name	Symbol	Source	Destination	Rate	Description
Display interface clock	DISP0_IF_CLK, DISP1_IF_CLK, DISP2_IF_CLK, DISP3_IF_CLK	DI	Displays	Up to HSP_CLK rate	Output clocks used as display interface clock for driving corresponding display chip select or read/write strobe pins (for asynchronous displays 0, 1, 2) or display clock input pin (for synchronous display 3). DISP0_IF_CLK, DISP1_IF_CLK, DISP2_IF_CLK are generated internally by integer division of HSP_CLK clock, DISP3_IF_CLK is generated internally by fractional division of HSP_CLK clock.
Display pixel clock	DISP0_PIX_CLK, DISP1_PIX_CLK, DISP2_PIX_CLK,	DI	SDC, ADC	Up to HSP_CLK rate	Virtual clocks generated internally by fractional division of HSP_CLK clock, used internally in SDC and ADC for tracking display refresh. Display 3 pixel clock is generated by division of DISP3_IF_CLK, the division factor is a number of cycles required for one pixel output.

Figure 44-2 shows the high level block diagram of the IPU clocking.

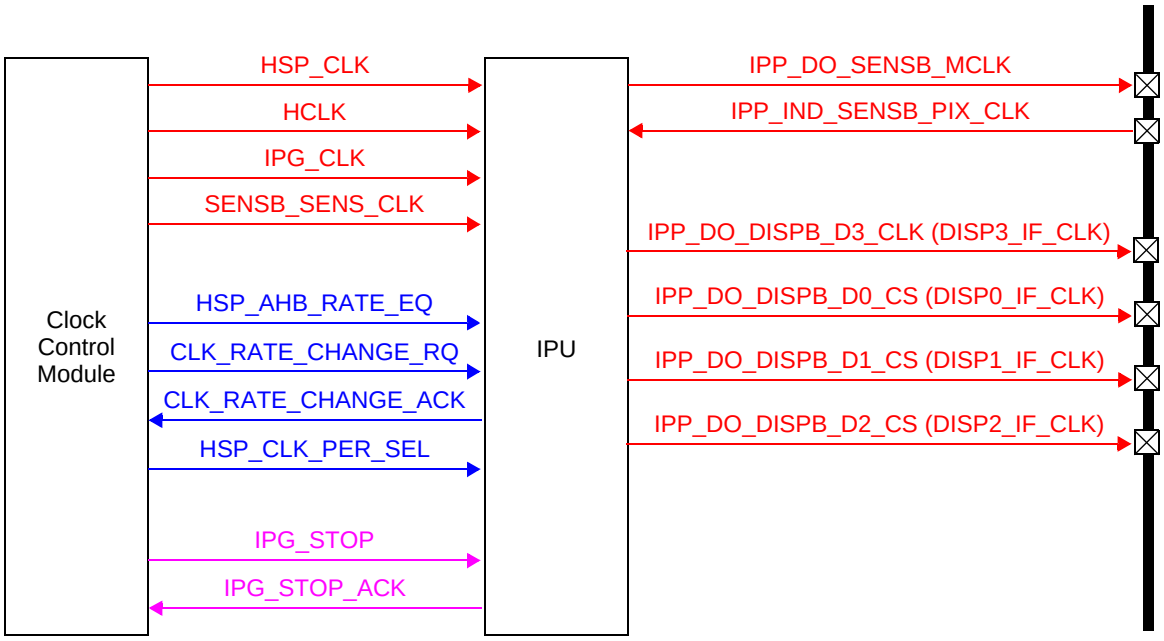


Figure 44-2. High Level Block Diagram of the IPU Clocking

44.1.1.4.2 Changing Clock Rates and Disabling Clocks

Figure 44-3 shows the relationship between the HSP_CLK, HCLK and IPG_CLK clocks.

The HSP_CLK rate can be equal or higher than the HCLK rate. The IPG_CLK rate is equal or lower than the HCLK rate. The IPU input pin HSP_AHB_RATE_EQ when high indicates equality of the HSP_CLK and HCLK rates. Transition between two HSP_AHB_RATE_EQ states (between equal and different HSP_CLK and HCLK rates and vice versa) allowed only when rising edges of all the clocks are aligned. At this time, the IDMAC and the ADC must be disabled and the MCU must not access the IPS and AHB_S buses.

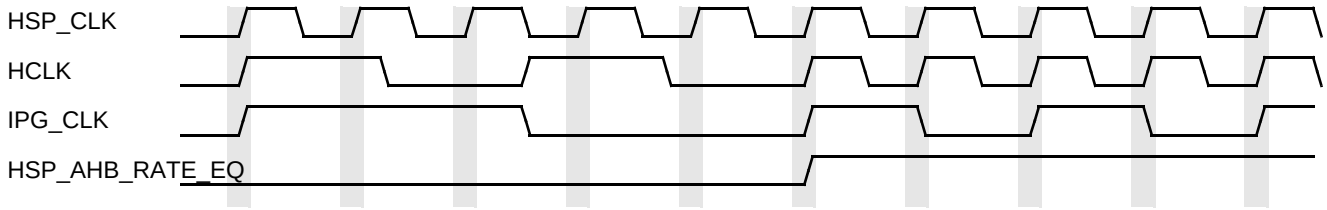


Figure 44-3. Relationship Between HSP_CLK, HCLK and IPG_CLK

The HSP_CLK and HCLK rates can be changed without stopping the IPU if the HSP_CLK rate is equal to the HCLK rate both before and after the change or the HSP_CLK rate is higher than the HCLK rate in both cases. Independently on the HSP_CLK rate change, the DISP#_IF_CLK and DISP#_PIX_CLK rates have to remain unaltered. This is achieved by means correction of the display clock rates according to the

IPU input pin HSP_CLK_PER_SEL which must be toggled every time when the HSP_CLK rate is changed. Before changing the HSP_CLK rate, the MCU should program the appropriate IPU register with a new value of the HSP_CLK period. The DI corrects the DISP#_IF_CLK and DISP#_PIX_CLK rates according to this data.

The operation sequence of clock rate change is as follows.

1. The MCU writes the new HSP_CLK period to the IPU control register.
2. The Clock Control Module asserts the CLK_RATE_CHANGE_RQ signal.
3. The IPU replays with the CLK_RATE_CHANGE_ACK pulse after it has finished current access to a display.
4. The Clock Control Module waits for alignment of positive edges of all clocks and changes clock rates simultaneously with toggling the HSP_CLK_PER_SEL signal. (See Figure 44-4.)

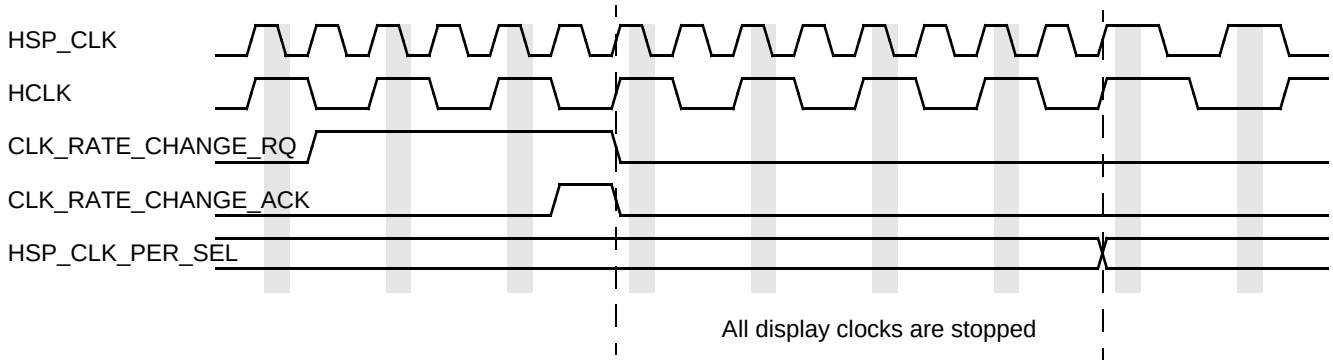


Figure 44-4. Operation Sequence of Clock Rate Change

In standby mode, all IPU clocks are gated off by the Clock Control Module. The hardware handshaking mechanism to enter standby mode is as follows (also see Figure 44-5):

1. An external standby mode controller asserts the IPG_STOP signal to the IPU.
2. The IPU stops running tasks synchronously at end of all active frames.
3. The IPU asserts the IPG_STOP_ACK signal
4. The IPU clocks are stopped by the Clock Control Module.
5. At resuming clock generation, the IPG_STOP and IPG_STOP_ACK signals are negated.

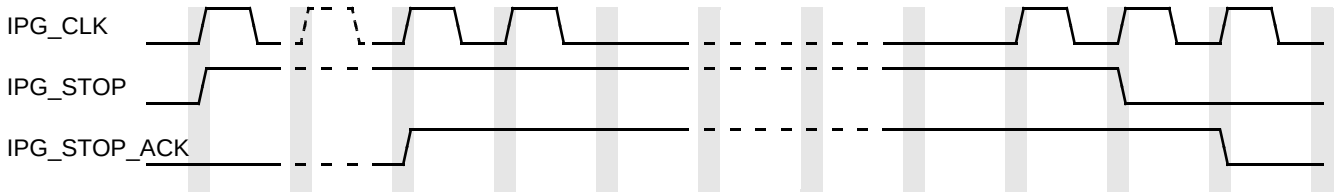


Figure 44-5. Entering and Exiting Standby Mode

44.1.1.4.3 Clocking Microarchitecture

Clocking microarchitecture in the IPU is explained in Figure 44-6.

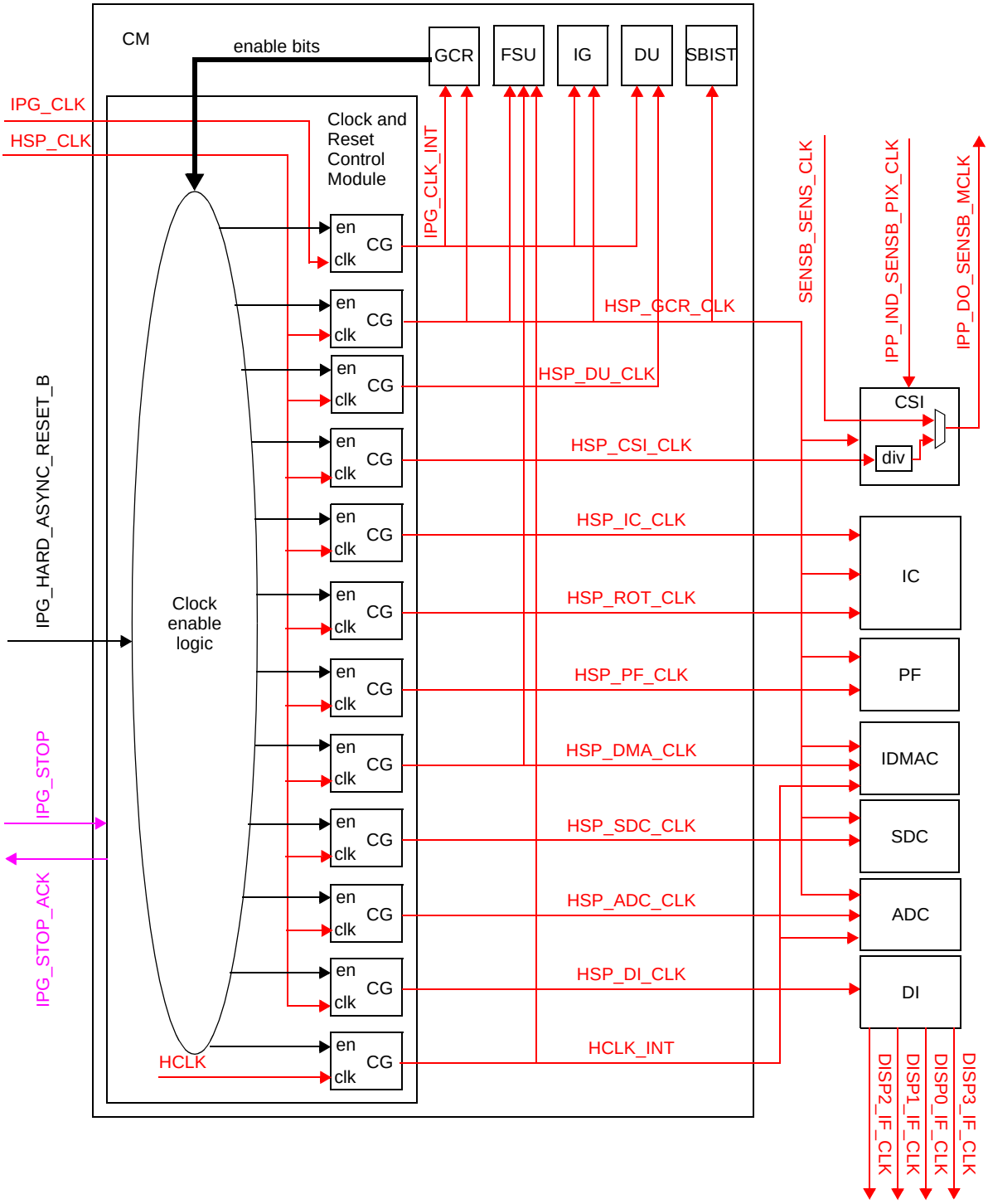


Figure 44-6. Clocking Microarchitecture in the IPU

There are two levels of clock gating. The first is done manually in the Clock and Reset Control Module (CRCU) which is included in the CM. The CRCU also generates reset signals for all IPU submodules. It guarantees exiting and entering reset free of contention with a submodule clock. Table 44-2 describes conditions for manual clock gating.

Table 44-2. Manual Clock Gating Conditions

Internal clock	Enabling condition
IPG_CLK_INT	IPU is out of hard reset
HSP_GCR_CLK	IPU is out of hard reset
HSP_DU_CLK	Debug Unit in the CM is enabled
HSP_CSI_CLK	CSI is enabled
HSP_IC_CLK	IC is enabled
HSP_ROT_CLK	IC Rotation Section is enabled
HSP_PF_CLK	PF is enabled
HSP_DMA_CLK	at least one of CSI, IC, PF, ADC, SDC submodules is enabled
HSP_SDC_CLK	SDC is enabled
HSP_ADC_CLK	ADC is enabled
HSP_DI_CLK	DI is enabled
HCLK_INT	IPU is out of hard reset

Clock gating at the second level is inserted automatically by a synthesis tool.

44.1.2 IPU Features

The following tables describe the IPU features.

Table 44-3 describes general IPU features.

Table 44-3. Basic IPU Features

Feature		Implementation	Comments
Interfaces	Master AHB	Configurable one or two shared master ports for all modules	Number of ports is defined by a VIA and by a configurations bit (if the VIA enables two ports)
	Slave AHB	Dedicated slave port for display interface	
	IP Skyblue	Single shared port for all modules	
	Bus modes	Bus Big or Little Endian, pixel Big or Little Endian according to configuration	
	Data burst length	1,2,..., 16 words for AHB slave, 1,2,..., 9 words for AHB master	

Table 44-3. Basic IPU Features (continued)

Feature		Implementation	Comments
Data processing flows	From pre- or post-processing to synchronous display	via external memory	
	From pre- or post-processing to smart display	via external memory or closed internally	
	Graphics and video combining	for all display types, performed at frame rate, additional combining for synchronous display, performed on-the-fly	
Synchronization of processing flows	Frame transfer between processing stages and double buffering	supported with MCU involvement or automatically	
	Automatic elimination of image tearing on display	supported both for synchronous displays (data source - system memory) and for smart displays (data source - system memory or postprocessing or preprocessing)	For preprocessing the following conditions must be fulfilled: 1. Display refresh rate is two times higher than sensor refresh rate. 2. Displays VSYNC is synchronized to sensor VSYNC by the IPU 3. Image occupies the whole screen
Clocking schemes	Internal high speed clock rate	equal or higher than the AHB clock rate (both clocks should be derived from the same high-frequency source)	
	Sensor clock	external clock, may be asynchronous to AHB clock and to internal clocks	
	Clock rate change	supported on-the-fly	
	Clock stop mode	supported	

Table 44-4 describes sensor interface features.

Table 44-4. Sensor Interface Features

Feature		Implementation	Comments
Input	Data bus width	4, 8, 10	10-bit color components packed into one 32-bit output word
		up to 15 bits (generic data)	15-bit generic data packed into 16-bit word (most significant bits are used)
	Data formats	Bayer, YUV 4:2:2 interleaved, RGB interleaved	
		YUV 4:4:4 interleaved, generic data	
	Maximal input data rate	73 Mbytes/s	For YUV 4:4:4–36.5 Mpixels/s
	Maximal frame size	4096 x 4096	
Synchronization and control	Master clock signal to sensor	external clock or clock generated internally by division of IPU high-speed clock, dividers: 1,2,3,...,256	
	Synchronization input signals	PIXCLK, HSYNC, VSYNC	
	Timing	gated clock (with HSYNC) non-gated clock (without HSYNC)	
	Pseudo-CCIR656 protocol decoding	interlaced and non-interlaced	Including TV decoder support
Output	Destination	memory or/and preprocessing	
	Conversion of interlaced format to progressive scan format	supported in hardware	
	Non-contiguous system memory buffer	supported	
	Skipping frames	no skip or skip 1/5, 2/5, 3/5, 4/5 according to programmable pattern	separate skipping pattern for encoding and viewfinder channels
External flash strobe generation	Synchronization	relative to the first HSYNC of sensor frame	
	Strobe start time	from 0 to 8191 sensor row	
	Strobe duration	from 1 to 16536 sensor rows	
	Strobe polarity	programmable	

Table 44-5 describes preprocessing features.

Table 44-5. Preprocessing Features

Feature		Implementation	Comments
Input	Source	sensor or memory	
	Pixel format	YUV 4:2:2, YUV 4:4:4 interleaved, YUV 4:2:0, YUV 4:2:2, YUV 4:4:4non-interleaved (from memory)	Row length is a multiple of 8 pixels
		RGB interleaved 24 bits/pixel (from sensor), RGB interleaved 16/24/32 bits/pixel, arbitrarily packed (from memory), RGB coded 4, 8 bits/pixel (from memory, palette look up table)	
	Maximal frame size	4096 x 4096	
Geometry conversion	Resizing options	both down- and upsizing	
	Downsizing algorithm	decimation with averaging followed by bilinear interpolation	
	Upsizing algorithm	bilinear interpolation	
	Resizing ratios	(K*N):M where K = 1, 2, 4 N = 1, 2... 16383 M = 8192	For direct capturing of sensor image upsizing ratio is limited by available MOPS
	Maximal frame size after downsizing	720x1024	
	Horizontal/vertical resizing	independent	
	Inversion	vertical and horizontal supported	
	Rotation	supported	Only via external memory
Combining with graphics	Algorithm	alpha blending with global or local (specified per pixel) alpha value or key color combining	
	Graphics format	RGBA interleaved 8/16/24/32 bits/pixel programmable arbitrarily packed, RGBA coded 4, 8 bits/pixel (palette look up table), YUVA interleaved 8 bits/pixel	
Color space conversion	YUV -> RGB	programmable coefficients	
	RGB -> YUV		
	YUV -> RGB -> YUV		For TV output
	YUV -> YUV		

Table 44-5. Preprocessing Features (continued)

Feature		Implementation	Comments
Output	Destination	memory or/and smart (asynchronous) display	
	Pixel format	YUV 4:2:0, YUV 4:2:2, YUV 4:4:4 non-interleaved, YUV 4:2:2, YUV 4:4:4 interleaved	
		arbitrarily packed interleaved RGB, 8/16/24/32 bits/pixel	
	Page-flip double buffering	supported	

Table 44-6 describes postprocessing features.

Table 44-6. Postprocessing Features

Feature		Implementation	Comments
Input	Source	memory	
	Pixel format	YUV 4:2:0, YUV 4:2:2, YUV 4:4:4 non-interleaved, YUV 4:2:2, YUV 4:4:4 interleaved	Row length is a multiple of 8 pixels
		RGB interleaved 8/16/24/32 bits/pixel arbitrarily packed, RGB coded 4/8 bits/pixel (palette look up table)	
	Maximal frame size	4096x4096	
Geometry conversion	Resizing options	both down- and upsizing	
	Downsizing algorithm	decimation with averaging followed by bilinear interpolation	
	Upsizing algorithm	bilinear interpolation	
	Resizing ratios	(K*N):M where K = 1, 2, 4 N = 1, 2... 16383 M = 8192	
	Maximal frame size after downsizing	720x1024	
	Horizontal/vertical resizing	independent	
	Inversion	vertical and horizontal supported	
	Rotation	supported	

Table 44-6. Postprocessing Features (continued)

Feature		Implementation	Comments
Combining with graphics	Algorithm	alpha blending with global or local (specified per pixel) alpha value or key color combining	
	Graphics format	RGBA interleaved 8/16/24/32 bits/pixel programmable arbitrarily packed, RGBA coded 4/8 bits/pixel (palette look up table), YUVA interleaved 8 bits/pixel	
Color space conversion	YUV -> RGB	programmable coefficients	
	RGB -> YUV		
	YUV -> RGB -> YUV		For TV output
	YUV -> YUV		
Output	Destination	memory or/and smart (asynchronous) display	
	Formats	YUV 4:2:0, YUV 4:2:2, YUV 4:4:4 non-interleaved or YUV 4:2:2, YUV 4:4:4 interleaved	
		programmable arbitrarily packed interleaved RGB 8/16/24/32 bits/pixel	
	Page-flip double buffering	supported	
	Processing several video streams	supported	

Table 44-7 describes post-filtering features.

Table 44-7. Post-filtering Features

Feature		Implementation	Comments
Input	Source	memory	
	Pixel format	YUV 4:2:0 non- interleaved	
Postfiltering algorithm	MPEG-4 (and WMV)	supported	Algorithm from Motorola Labs
	H.264	supported	Pause at desired row of the Y frame is supported
	Processing several video streams	supported	
Output	Destination	memory	
	Pixel format	YUV 4:2:0 non- interleaved	

Table 44-8 describes synchronous display interface features.

Table 44-8. Synchronous Display Interface Features

Feature		Implementation	Comments
Input	Source	memory	
	Pixel format	monochrome 8 bits/pixel	
		RGBA interleaved 8/16/24/32 bits/pixel arbitrarily packed, RGB coded 4, 8 bits/pixel (palette look up table)	
		YUV 4:2:2 interleaved	For TV support
		YUV 4:2:0, YUV 4:2:2 non-interleaved	For TV support, frame width must be a multiple of 8 pixels
	Maximal frame size	1024 x 1024	
Combining image planes	Number of planes	two	
	Transparency coding	alpha blending with global or local (specified per pixel) alpha value or key color combining	
	Windowing function	supported by display enable control according to pattern from system memory	
Cursor generation	Hardware cursor	uniform color—black/white/configurable reversed color OR/XOR/AND with graphics plane, blinking/steady, size up to 31 x 31 pixels	
	Automatically animated cursor	supported, maximal cursor size 32 x 32 pixels	Via graphics plane and scrolling
Scrolling and panning	Scrolling control	automatic with programmable step of 1, 2, 3, 4, 5, 6, 7, 8, 10, 12, 14, 16, 20, 24, 28, 32 pixels/frame	Only for interleaved input data
	Scrolling directions	both horizontal and vertical (programmable)	
	Panning	supported	
Display access synchronization	Automatic elimination of image tearing	supported	
	Automatic skipping output of unchanged frames for dual-port smart displays	supported	

Table 44-8. Synchronous Display Interface Features (continued)

Feature		Implementation	Comments
Display interface	Color TFT displays	12/16/18-bit I/F 4/8 bits/pixel out of a palette of 256k 12/16 bits/pixel uncoded	Data conversion is done by dropping and adding bits
		6/8/9-bit I/F 18/24 bits/pixel uncoded time-multiplexed	
	Sharp TFT display	supported	
	Contrast control	8-bit PWM	
	Programmable timing of display control signals	separate programmable read and write access periods with resolution of one high speed clock cycle, programmable strobe rise/fall time with resolution of a half of one high speed clock cycle	
	Maximal display clock rate	up to quarter of high speed processing clock rate	

Table 44-9 describes asynchronous display interface features.

Table 44-9. Asynchronous Display Interface Features

Feature		Implementation	Comments
Input for display data writing or output for display data reading	Source (writing) or destination (reading)	system memory (2 channels), preprocessing (1 channel), postprocessing (1 channel), direct MCU access via slave AHB bus	
	Data format	generic data 8/16/32 bits	
		pixel data in interleaved RGB format, 8/16/24/32 bits/pixel arbitrarily packed or 4, 8 bits/pixel coded (palette look up table)	
	Maximal frame size	1024 x 1024	

Table 44-9. Asynchronous Display Interface Features (continued)

Feature		Implementation	Comments
Display access features	Writing data to display	supported using command buffer (for system memory source) or using command template (for all sources)	
	Reading data from display	supported using command template (for system memory destination or MCU direct access)	
	MCU low level access to display registers via IP bus	supported	
	Burst transfer modes	single transfer or burst transfer with or without separate burst clock	
	Parallel transfer several windows from different sources to the same display	supported	
	Automatic display refresh	supported with programmable refresh rate and optional snooping condition	The snooping signal is received from the external memory interface
	Transfer synchronization with synchronous display	supported (transfer during blanking intervals of synchronous display)	
	Automatic elimination of image tearing	supported	For displays generated or accepted the VSYNC signal
Scrolling and panning	Scrolling	automatic with programmable step of 1, 2, 3, 4, 5, 6, 7, 8, 10, 12, 14, 16, 20, 24, 28, 32 pixels/frame in both horizontal and vertical directions	Only for interleaved input data
	Panning	supported	

Table 44-9. Asynchronous Display Interface Features (continued)

Feature		Implementation	Comments
Display interfaces	Number of displays supported	3	Display 0—only parallel interface, displays 1 and 2—parallel or serial interfaces
	Display addressing types	X/Y addressing or full linear address	
	Parallel interface bus type	System80 (Type 1 or Type 2) or System68 (Type 1 or Type 2)	For Type 1 interface, data is sampled by chip select signal. For Type 2 interface, data is sampled by read/write/enable strobes.
	Parallel interface bus width	6/8/9/12/16/18 bits	
	Programmable packing/unpacking data and commands to display	supported with separate packing/unpacking for each display and transfer type (data or command), output in 1, 2, 3 or 4 cycles	
	Serial interface	3/4/5 lines with programmable preamble	Two types of 5-line interface—with address sampling by the serial clock or by the chip enable signal
	Programmable timing of display control signals	separate programmable read and write access periods with resolution of one high speed clock cycle, programmable strobe rise/fall time with resolution of a half of one high speed clock cycle	Separate programming per display
	Maximal display access rate	up to high speed processing clock rate	
	Access with byte enable	supported for parallel interfaces	
	Read wait states for parallel interface	0/1/2 /3 programmable wait states	

44.1.3 Modes of Operation

The IPU is a very flexible module and can be programmed for different operating modes. The examples in the following tables are typical use cases supported by the IPU.

Table 44-10 shows IPU data flows for the use case ‘Video from the System Memory on a VGA Asynchronous Display.’

Table 44-11 shows IPU data flows for the use case ‘Video from the System Memory on a QVGA Synchronous Display.’

Table 44-10. Video from the System Memory on a VGA Synchronous Display

Task	Parameter		Flow	
			Postprocessing	
			Video	Graphics
Input data	Source		memory	memory
	Frame size (pixels)	h	640	640
		v	480	480
	Frame rate (fps)		30	30
	Pixel format		YUV420	RGB
	Pixel rate (Mpixels/s)		9.2	9.2
Output data	Destination		display	---
	Frame size (pixels)	h	640	---
		v	480	---
	Frame rate (fps)		30	---
	Pixel format		RGB	---
	Pixel rate (Mpixels/s)		9.2	---
Capturing image from sensor	Data unit size (bytes)		---	---
	Data rate (Mbytes/s)		---	---
Transferring image from memory	Data unit size (bytes)		1	---
	Burst size (data units)		32	---
	Data rate (Mbytes/s)		13.8	---
	Memory rate (Mcycles/s)		9.1	---
	DMA rate (Mcycles/s)		10.4	---
Deblocking (MPEG-4)	Proc. rate (Mcycles/s)		71.0	---
Deringing (MPEG-4)	Proc. rate (Mcycles/s)		64.5	---
Storing image in memory	Data rate (Mbytes/s)		13.8	---
	Memory rate (Mcycles/s)		6.7	---
	DMA rate (Mcycles/s)		8.0	---

Table 44-10. Video from the System Memory on a VGA Synchronous Display (continued)

Task	Parameter		Flow	
			Postprocessing	
			Video	Graphics
Restoring image from memory	Data unit size (bytes)		1	2
	Burst size (data units)		16	16
	Data rate (Mbytes/s)		13.8	18.4
	Memory rate (Mcycles/s)		25.9	12.1
	DMA rate (Mcycles/s)		31.1	13.8
Downsizing	Ratio	h	1	---
		v	1	---
	Proc. rate (Mcycles/s)		12.0	---
Resizing	Proc. rate (Mcycles/s)		21.2	---
Color conversion	Proc. rate (Mcycles/s)		31.8	---
Combining	Proc. rate (Mcycles/s)		10.6	---
Output to asynchronous display	Data unit size (bytes)		4	---
	Burst size (data units)		8	---
	Data rate (Mbytes/s)		36.9	---
	DMA rate (Mcycles/s)		10.4	---
Total load	IC rate (Mcycles/s)		80.9	
	PF rate (Mcycles/s)		136.9	
	IDMAC rate (Mcycles/s)		73.7	
	Memory rate (Mcycles/s)		53.8	

Table 44-11. Video from the System Memory on a QVGA Synchronous Display

Task	Parameter		Flow		
			Foreground	Background (postprocessing)	
			Graphics	Video	Graphics
Input data	Source		memory	memory	memory
	Frame size (pixels)	h	320	320	320
		v	240	240	240
	Frame rate (fps)		60	30	30
	Pixel format		RGB	YUV420	RGB
	Pixel rate (Mpixels/s)		4.6	2.3	2.3
Output data	Destination		---	display	---
	Frame size (pixels)	h	---	320	---
		v	---	240	---
	Frame rate (fps)		---	60	---
	Pixel format		---	RGB	---
	Pixel rate (Mpixels/s)		---	4.6	---
Transferring image from memory	Data unit size (bytes)		---	1	---
	Burst size (data units)		---	32	---
	Data rate (Mbytes/s)		---	3.5	---
	Memory rate (Mcycles/s)		---	2.3	---
	DMA rate (Mcycles/s)		---	2.6	---
Deblocking (MPEG-4)	Proc. rate (Mcycles/s)		---	17.7	---
Deringing (MPEG-4)	Proc. rate (Mcycles/s)		---	16.1	---
Storing image in memory	Data rate (Mbytes/s)		---	3.5	---
	Memory rate (Mcycles/s)		---	1.7	---
	DMA rate (Mcycles/s)		---	2.0	---
Restoring image from memory	Data unit size (bytes)		2	1	2
	Burst size (data units)		16	16	16
	Data rate (Mbytes/s)		9.2	3.5	4.6
	Memory rate (Mcycles/s)		6.0	6.5	3.0
	DMA rate (Mcycles/s)		6.9	7.8	3.5
Downsizing	Ratio	h	---	1	---
		v	---	1	---
	Proc. rate (Mcycles/s)		---	3.0	---

Table 44-11. Video from the System Memory on a QVGA Synchronous Display (continued)

Task	Parameter	Flow		
		Foreground	Background (postprocessing)	
		Graphics	Video	Graphics
Resizing	Proc. rate (Mcycles/s)	---	5.3	---
Color conversion	Proc. rate (Mcycles/s)	---	7.9	---
Combining	Proc. rate (Mcycles/s)	---	2.6	---
Storing image in memory	Data unit size (bytes)	---	2	---
	Burst size (data units)	---	16	---
	Data rate (Mbytes/s)	---	4.6	---
	Memory rate (Mcycles/s)	---	2.2	---
	DMA rate (Mcycles/s)	---	2.7	---
Output to synchronous display	Data unit size (bytes)	---	2	---
	Burst size (data units)	---	16	---
	Data rate (Mbytes/s)	---	9.2	---
	Memory rate (Mcycles/s)	---	6.0	---
	DMA rate (Mcycles/s)	---	6.9	---
Total load	IC rate (Mcycles/s)	25.3		
	PF rate (Mcycles/s)	34.2		
	IDMAC rate (Mcycles/s)	32.3		
	Memory rate (Mcycles/s)	27.8		

44.2 External Signal Description

44.2.1 Overview

Table 44-12 describes IPU external signals.

Table 44-12. Signal Properties

Name	Direction	Function	Reset State	Pull-Up
Sensor Interface				
IPP_IND_SENSB_DATA[15:1]	in	Sensor data (YUV, RGB)		
IPP_IND_SENSB_PIX_CLK	in	Data latch clock from sensor		
IPP_IND_SENSB_HSYNC	in	Horizontal synchronization pulse		

Table 44-12. Signal Properties (continued)

Name	Direction	Function	Reset State	Pull-Up
IPP_IND_SENSB_VSYNC	in	Vertical synchronization pulse		
SENSB_SENS_CLK	in	Sensor clock from CCM		
IPP_DO_SENSB_MCLK	out	Master clock to sensor	0	
Display Interface				
IPP_IND_DISPB_DATA[17:0]	in	Input data from display. In byte enable mode, bits 16 and 17 are unused		
IPP_DO_DISPB_DATA[15:0]	out	Output data to display.	0	
IPP_DO_DISPB_DATA[16]		Output data to display. In byte enable mode, WRITE_H (for sys80)	1	
IPP_DO_DISPB_DATA[17]		Output data to display. In byte enable mode, READ_H (for sys80) or ENABLE_H (for sys68k)	1	
IPP_OBE_DISPB_DATA	out	Display data direction select for data bits [15:0]	0	
IPP_OBE_DISPB_DATA16_17	out	Display data direction select for data bits [17:16]	0	
IPP_DO_DISPB_D3_VSYNC	out	Display 3 vertical synchronization pulse (FPFRAME/VSYNC/FLM/SPS/TV)	1	
IPP_DO_DISPB_D3_HSYNC	out	Display 3 horizontal synchronization pulse (FPLINE/HSYNC/LP)	1	
IPP_DO_DISPB_D3_CLK	out	Display 3 clock (FPSHIFT/DOTCLC/LSCLC/DCLK/CLOCK)	0	
IPP_IND_DISPB_D3_DRDY	out	Display 3 data enable or Sharp display PS signal (CONTROL1/DRDY/PS/VLD/BLANK)	1	
IPP_DO_DISPB_D3_SPL	out	SPL signal for Sharp display 3	0	
IPP_DO_DISPB_D3_CLS	out	CLS signal for Sharp display 3	0	
IPP_DO_DISPB_D3_REV	out	REV signal for Sharp display 3	0	
IPP_IND_DISPB_D0_VSYNC	in	Display 0 input vertical synchronization pulse		
IPP_DO_DISPB_D0_VSYNC	out	Display 0 output vertical synchronization pulse	1	
IPP_OBE_DISPB_D0_VSYNC	out	Display 0 vertical synchronization pulse direction select	0	
IPP_DO_DISPB_D0_CS	out	Display 1 chip select	1	
IPP_DO_DISPB_D1_CS	out	Display 2 chip select	1	
IPP_DO_DISPB_D2_CS	out	Display 3 chip select	1	
IPP_DO_DISPB_PAR_RS	out	Data/command select for parallel interface	0	
IPP_DO_DISPB_SER_RS	out	Data/command select for serial interface	0	

Table 44-12. Signal Properties (continued)

Name	Direction	Function	Reset State	Pull-Up
IPP_DO_DISPB_WR	out	System-80: WRITE System-68K: READ/WRITE In byte enable mode— System-80: WRITE_L System-68K: READ/WRITE	1	
IPP_DO_DISPB_RD	out	System-80: READ System 68K: ENABLE In byte enable mode— System-80: READ_L System-68K: ENABLE_L	1	
IPP_IND_DISPB_SD_D	in	Data input from serial interface		
IPP_DO_DISPB_SD_D	out	Data output to serial interface	0	
IPP_OBE_DISPB_SD_D	out	Serial interface data direction select	0	
IPP_DO_DISPB_SD_D_CLK	out	Serial interface clock	0	
IPP_IND_DISPB_D12_VSYNC	in	Displays 1,2 input vertical synchronization pulse		
IPP_DO_DISPB_D12_VSYNC	out	Displays 1,2 output vertical synchronization pulse	1	
IPP_OBE_DISPB_D12_VSYNC	out	Displays 1,2 vertical synchronization pulse direction select	0	
IPP_DO_DISPB_CONTRAST	out	Contrast control for the primary display	0	
IPP_DO_DISPB_BCLK	out	Burst clock for asynchronous parallel interfaces (displays 0, 1, 2)	0	

44.2.2 Detailed Signal Descriptions

44.2.2.1 Sensor Interface

44.2.2.1.1 IPP_IND_SENSB_DATA[15:1]

Input data from the sensor. For sensors providing less than 15-bit output, only the most significant bits of the bus are used.

44.2.2.1.2 IPP_IND_SENSB_PIX_CLK

Input strobe for the sensor data. A selected edge of this signal indicates that new data is valid on the IPP_IND_SENSB_D[15:1] bus.

44.2.2.1.3 IPP_IND_SENSB_HSYNC

Horizontal synchronization pulse. Valid only in “Gated Clock Mode” and only when high, it validates IPP_IND_SENSB_PIX_CLK which always toggles. Not valid in “Non-Gated Clock Mode” and in “Pseudo BT.656 Video Mode”.

44.2.2.1.4 IPP_IND_SENSB_VSYNC

Vertical synchronization pulse. Valid only in “Gated Clock Mode” and in “Non-Gated Clock Mode” and indicate start of frame. Not valid in “Pseudo BT.656 Video Mode”.

44.2.2.1.5 SENSB_SENS_CLK

Input signal from the Clock Controller (sensor clock).

44.2.2.1.6 IPP_DO_SENSB_MCLK

Output clock which used as master clock signal to the sensor. When IPP_DO_SENSB_MCLK is disabled, its output state level stays low.

44.2.2.2 Display Interface

44.2.2.2.1 IPP_IND_DISPB_DATA[17:0]

Display data input bus for the parallel interface. For the 16-bit display interface with byte enable support, the bits 16 and 17 are unused.

44.2.2.2.2 IPP_DO_DISPB_DATA[17:0]

Display data output bus for the parallel interface. For the 16-bit display interface with byte enable support, the bit 16 is the write high (WRITE_H) strobe (for the system 80 interface), the bit 17 is the read high (READ_H) strobe (for the system 80 interface) or the enable high (ENABLE_H) strobe (for the system 68k interface). The bit 16 is unused for the system 68k interface. Both bits are used for data output if the display does not support byte enable.

44.2.2.2.3 IPP_OBE_DISPB_DATA

Display data bus direction select signal. When it is low data input from the display is enabled, when it is high data output is enabled.

44.2.2.2.4 IPP_OBE_DISPB_DATA16_17

Display data bus direction select signal for the data bits 16 and 17. When it is low data input from the display is enabled, when it is high data output is enabled.

44.2.2.2.5 IPP_DO_DISPB_D3_VSYNC

Vertical synchronization signal to the synchronous display (display 3). It matches the FPFRAME/ VSYNC/FLM/SPS inputs of supported displays.

44.2.2.2.6 IPP_DO_DISPB_D3_HSYNC

Horizontal synchronization signal to the synchronous display (display 3). It matches the FPLINE/ HSYNC/LP inputs of supported displays.

44.2.2.2.7 IPP_DO_DISPB_D3_CLK

Display clocking signal to the synchronous display (display 3). It matches the FPSHIFT / DOTCLC/ LSCLC / DCLK / CLOCK inputs of supported displays.

44.2.2.2.8 IPP_IND_DISPB_D3_DRDY

Display control signal to the synchronous display (display 3). It matches the CONTROL1/DRDY/VLD/ BLANK inputs of supported displays. PS control signal to the synchronous Sharp HR-TFT displays.

44.2.2.2.9 IPP_DO_DISPB_D3_SPL

Display control signal to the synchronous Sharp HR-TFT display (display 3). It matches the SPL display input.

44.2.2.2.10 IPP_DO_DISPB_D3_CLS

Display control signal to the synchronous Sharp HR-TFT display (display 3). It matches the CLS display input.

44.2.2.2.11 IPP_DO_DISPB_D3_REV

Display control signal to the synchronous Sharp HR-TFT display (display 3). It matches the REV display input.

44.2.2.2.12 IPP_IND_DISPB_D0_VSYNC

Vertical synchronization input from the asynchronous (smart) primary display (display 0).

44.2.2.2.13 IPP_DO_DISPB_D0_VSYNC

Vertical synchronization output to the asynchronous (smart) primary display (display 0).

44.2.2.2.14 IPP_OBE_DISPB_D0_VSYNC

Display vertical synchronization signal direction select for the asynchronous (smart) primary display (display 0). When it is low VSYNC input from the display is enabled, when it is high VSYNC output to the display is enabled.

44.2.2.2.15 IPP_DO_DISPB_D0_CS

Chip select signal for the for the asynchronous (smart) primary display (display 0). Corresponds to the DISP0_IF_CLK (see Table 44-1).

44.2.2.2.16 IPP_DO_DISPB_D1_CS

Chip select signal for the for the asynchronous (smart) secondary display (display 1). Corresponds to the DISP1_IF_CLK (see Table 44-1).

44.2.2.2.17 IPP_DO_DISPB_D2_CS

Chip select signal for the for the asynchronous (smart) secondary display (display 2). Corresponds to the DISP2_IF_CLK (see Table 44-1).

44.2.2.2.18 IPP_DO_DISPB_PAR_RS

DATA/COMMAND (RS) control signal for the parallel interface of the asynchronous displays 1-2.

44.2.2.2.19 IPP_DO_DISPB_SER_RS

RS control signal for the serial interface of the asynchronous displays 1-2.

44.2.2.2.20 IPP_DO_DISPB_WR

Read/enable control signal for the asynchronous displays 0-2 with parallel interfaces. It matches the WRITE signal for the System-80 interface or the READ/WRITE signal for the System-68K interface.

44.2.2.2.21 IPP_DO_DISPB_RD

Write/read control signal for the asynchronous displays 0-2. It matches the READ signal for the System-80 interface or the ENABLE signal for the System-68K interface.

44.2.2.2.22 IPP_IND_DISPB_SD_D

Data input for the serial interface of the asynchronous displays 1 and 2.

44.2.2.2.23 IPP_DO_DISPB_SD_D

Data output for the serial interface of the asynchronous displays 1 and 2.

44.2.2.2.24 IPP_OBE_DISPB_SD_D

Data direction select signal for the serial interface of the asynchronous displays 1 and 2. When it is low serial input from the display is enabled, when it is high serial output to the display is enabled.

44.2.2.2.25 IPP_DO_DISPB_SD_D_CLK

Clocking signal for the serial interface of the asynchronous displays 1 and 2.

44.2.2.2.26 IPP_IND_DISPB_D12_VSYNC

Vertical synchronization input from the asynchronous (smart) secondary displays 1 and 2.

44.2.2.2.27 IPP_DO_DISPB_D12_VSYNC

Vertical synchronization output to the asynchronous (smart) secondary displays 1 and 2.

44.2.2.2.28 IPP_OBE_DISPB_D12_VSYNC

Direction select signal for vertical synchronization input/output from/to the asynchronous (smart) secondary displays 1 and 2.

44.2.2.2.29 IPP_DO_DISPB_CONTRAST

Contrast control for the primary displays 0 and 3.

44.2.2.2.30 IPP_DO_DISPB_BCLK

Burst clock for asynchronous parallel interfaces (displays 0, 1, 2).

44.2.2.2.31 Usage of the Display Interface Signals

Table 44-13 summarizes the display interface signals usage.

Table 44-13. Display Interface Signals Usage

Signal Name	Synchronous Display (Display 3)				Asynchronous Displays (Displays 0-2)	
	Generic	Sharp HR-TFT	Synch Interface Of Dual-port Display	TV Encoder	Primary Display 0	Secondary Displays 1, 2
					Parallel Interface	Serial Interface
IPP_DO_DISPB_DATA[17:0]	Output data				Output data In byte enable mode, bit 16 is WRITE_H (for sys80), bit 17 is READ_H (for sys80) or ENABLE_H (for sys68k)	
IPP_IND_DISPB_DATA[17:0]					Input data In byte enable mode, bits 16 and 17 are not used	
IPP_DO_DISPB_D3_VSYNC	FPFRAME / VSYNC/FLM (out)	SPS (out)	VSYNC (out)	VSYNC (out)		
IPP_DO_DISPB_D3_HSYNC	FPLINE/ HSYNC / LP	LP	HSYNC	HSYNC		
IPP_DO_DISPB_D3_CLK	FPSHIFT / DOTCLC/ LSCLC	DCLK	DOTCLC	CLOCK		
IPP_DO_DISPB_D3_DRDY	DRDY	PS	VLD	BLANK		
IPP_DO_DISPB_D3_SPL		SPL				
IPP_DO_DISPB_D3_CLS		CLS				
IPP_DO_DISPB_D3_REV		REV				
IPP_IND_DISPB_D0_VSYNC IPP_DO_DISPB_D0_VSYNC					VSYNC	

Table 44-13. Display Interface Signals Usage (continued)

Signal Name	Synchronous Display (Display 3)				Asynchronous Displays (Displays 0-2)		
	Generic	Sharp HR-TFT	Synch Interface Of Dual-port Display	TV Encoder	Primary Display 0	Secondary Displays 1, 2	
					Parallel Interface		Serial Interface
IPP_DO_DISPB_D0_CS					CS1		
IPP_DO_DISPB_D1_CS						CS2	CS2
IPP_DO_DISPB_D2_CS						CS3	CS3
IPP_DO_DISPB_PAR_RS					DATA/COMMAND		
IPP_DO_DISPB_SER_RS							RS
IPP_DO_DISPB_RD					System-80: READ System-68K: ENABLE_L byte enable mode— System-80: READ_L System-68K: ENABLE_L		
IPP_DO_DISPB_WR					System-80: WRITE System 68K: READ/WRITE, byte enable mode— System-80: WRITE_L System-68K: READ/WRITE		
IPP_IND_DISPB_SD_D IPP_DO_DISPB_SD_D							D
IPP_DO_DISPB_SD_D_CLK							CLK
IPP_IND_DISPB_D12_VSYNC IPP_DO_DISPB_D12_VSYNC						VSYNC	
IPP_DO_DISPB_BCLK					CPUCLK		

Burst clock for asynchronous parallel interfaces (displays 0, 1, 2)

44.3 Memory Map and Register Definition

The IPU programming model includes 112 registers. Most of them are placed physically in the GCR. Some of them will be placed at their submodules for convenience. All of the registers will be accessed by the MCU using the IP Bus. Section 44.3.3, “Register Descriptions” provides the detailed descriptions for all of the IPU registers.

44.3.1 Memory Map

Table 44-14 shows the IPU memory map.

Table 44-14. IPU Memory Map

Address	Register	Access	Reset Value	Section
0x53FC_0000 (IPU_CONF)	IPU Configuration Register	R/W	0x0000_0000	44.3.3.1.1/44-65
0x53FC_0004 (IPU_CHA_BUF0_RDY)	IPU Channel Buffer 0 Ready Register	R/W	0x0000_0000	44.3.3.1.2/44-67
0x53FC_0008 (IPU_CHA_BUF1_RDY)	IPU Channel Buffer 1 Ready Register	R/W	0x0000_0000	44.3.3.1.3/44-68
0x53FC_000C (IPU_CHA_DB_MODE_SEL)	IPU Channel Double Mode Select Register	R/W	0x0000_0000	44.3.3.1.4/44-69
0x53FC_0010 (IPU_CHA_CUR_BUF)	IPU Channel Current Buffer	RW	0x0000_0000	44.3.3.1.5/44-70
0x53FC_0014 (IPU_FS_PROC_FLOW)	IPU Frame Synchronization Processing Flow Register	R/W	0x0000_0000	44.3.3.1.6/44-71
0x53FC_0018 (IPU_FS_DISP_FLOW)	IPU Frame Synchronization Displaying Flow Register	R/W	0x0000_0000	44.3.3.1.7/44-74
0x53FC_001C (IPU_TASKS_STAT)	IPU Tasks Status Register	R	0x0000_0000	44.3.3.1.8/44-75
0x53FC_0020 (IPU_IMA_ADDR)	IPU Internal Memory Access Address Register	R/W	0x0000_0000	44.3.3.1.9/44-79
0x53FC_0024 (IPU_IMA_DATA)	IPU Internal Memory Access Data Register	R/W	0x0000_0000	44.3.3.1.9/44-79
0x53FC_0028 (IPU_INT_CTRL_1)	IPU Interrupt Control Register 1	R/W	0x0000_0000	44.3.3.1.10/44-102
0x53FC_002C (IPU_INT_CTRL_2)	IPU Interrupt Control Register 2	R/W	0x0000_0000	44.3.3.1.11/44-103
0x53FC_0030 (IPU_INT_CTRL_3)	IPU Interrupt Control Register 3	R/W	0x0000_0000	44.3.3.1.12/44-104
0x53FC_0034 (IPU_INT_CTRL_4)	IPU Interrupt Control Register 4	R/W	0x0000_0000	44.3.3.1.13/44-107
0x53FC_0038 (IPU_INT_CTRL_5)	IPU Interrupt Control Register 5	R/W	0x0000_0000	44.3.3.1.14/44-108
0x53FC_003C (IPU_INT_STAT_1)	IPU Interrupt Status Register 1	R/W1C	0x0000_0000	44.3.3.1.15/44-111
0x53FC_0040 (IPU_INT_STAT_2)	IPU Interrupt Status Register 2	R/W1C	0x0000_0000	44.3.3.1.16/44-112
0x53FC_0044 (IPU_INT_STAT_3)	IPU Interrupt Status Register 3	R/W1C	0x0000_0000	44.3.3.1.17/44-113
0x53FC_0048 (IPU_INT_STAT_4)	IPU Interrupt Status Register 4	R/W1C	0x0000_0000	44.3.3.1.18/44-117
0x53FC_004C (IPU_INT_STAT_5)	IPU Interrupt Status Register 5	R/W1C	0x0000_0000	44.3.3.1.19/44-118
0x53FC_0050 (IPU_BRK_CTRL_1)	IPU Break Control Register 1	R/W	0x0000_0000	44.3.3.1.20/44-120
0x53FC_0054 (IPU_BRK_CTRL_2)	IPU Break Control Register 2	R/W	0x0000_0000	44.3.3.1.21/44-122
0x53FC_0058 (IPU_BRK_STAT)	IPU Break Status Register	R	0x0000_0000	44.3.3.1.22/44-124
0x53FC_005C (IPU_DIAGB_CTRL)	IPU Diagnostic Bus Control Register	R/W	0x0000_0000	44.3.3.1.23/44-124
0x53FC_0060 (CSI_SENS_CONF)	CSI Sensor Configuration Register	R/W	0x0000_0000	44.3.3.2.1/44-137
0x53FC_0064 (CSI_SENS_FRM_SIZE)	CSI Sensor Frame Size Register	R/W	0x0000_0000	44.3.3.2.2/44-138
0x53FC_0068 (CSI_ACT_FRM_SIZE)	CSI Actual Frame Size Register	R/W	0x0000_0000	44.3.3.2.3/44-139

Table 44-14. IPU Memory Map (continued)

Address	Register	Access	Reset Value	Section
0x53FC_006C (CSI_OUT_FRM_CTRL)	CSI Output Frame Control Register	R/W	0x0000_0000	44.3.3.2.4/44-140
0x53FC_0070 (CSI_TST_CTRL)	CSI Test Control Register	R/W	0x0000_0000	44.3.3.2.5/44-141
0x53FC_0074 (CSI_CCIR_CODE_1)	CSI CCIR Code Register 1	R/W	0x0000_0000	44.3.3.2.6/44-142
0x53FC_0078 (CSI_CCIR_CODE_2)	CSI CCIR Code Register 2	R/W	0x0000_0000	44.3.3.2.7/44-143
0x53FC_007C (CSI_CCIR_CODE_3)	CSI CCIR Code Register 3	R/W	0x0000_0000	44.3.3.2.8/44-144
0x53FC_0080 (CSI_FLASH_STROBE_1)	CSI Flash Strobe Register 1	R/W	0x0000_0000	44.3.3.2.9/44-144
0x53FC_0084 (CSI_FLASH_STROBE_2)	CSI Flash Strobe Register 2	R/W	0x0000_0000	44.3.3.2.10/44-145
0x53FC_0088 (IC_CONF)	IC Configuration Register	R/W	0x0000_0000	44.3.3.3.1/44-146
0x53FC_008C (IC_PRP_ENC_RSC)	IC Preprocessing Encoder Resizing Coefficients Register	R/W	0x2000_2000	44.3.3.3.2/44-149
0x53FC_0090 (IC_PRP_VF_RSC)	IC Preprocessing View-Finder Resizing Coefficients Register	R/W	0x2000_2000	44.3.3.3.3/44-150
0x53FC_0094 (IC_PP_RSC)	IC Post-Processing Resizing Coefficients Register	R/W	0x2000_2000	44.3.3.3.4/44-151
0x53FC_0098 (IC_CMBP_1)	IC Combining Parameters Register 1	R/W	0x0000_0000	44.3.3.3.5/44-152
0x53FC_009C (IC_CMBP_2)	IC Combining Parameters Register 2	R/W	0x0000_0000	44.3.3.3.6/44-153
0x53FC_00A0 (PF_CONF)	Post Filter Configuration Register	R/W	0x0000_0000	44.3.3.4.1/44-153
0x53FC_00A4 (IDMAC_CONF)	IDMAC Configuration Register	R/W	0x0000_0000	44.3.3.5.1/44-155
0x53FC_00A8 (IDMAC_CHA_EN)	IDMAC Channel Enable Register	R/W	0x0000_0000	44.3.3.5.2/44-156
0x53FC_00Ac (IDMAC_CHA_PRI)	IDMAC Channel Priority Register	R/W	0x0000_0000	44.3.3.5.3/44-157
0x53FC_00B0 (IDMAC_CHA_BUSY)	IDMAC Channel Busy Register	Read only	0x0000_0000	44.3.3.5.4/44-158
0x53FC_00B4 (SDC_COM_CONF)	SDC Common Configuration Register	R/W	0x0000_0000	44.3.3.6.1/44-158
0x53FC_00B8 (SDC_GRAPH_WIND_CTRL)	SDC Graphic Window Control Register	R/W	0x0000_0000	44.3.3.6.2/44-160
0x53FC_00BC (SDC_FG_POS)	SDC Foreground Window Position Register	R/W	0x0000_0000	44.3.3.6.3/44-162
0x53FC_00C0 (SDC_BG_POS)	SDC Background Window Position Register	R/W	0x0000_0000	44.3.3.6.4/44-163
0x53FC_00C4 (SDC_CUR_POS)	SDC Cursor Position Register	R/W	0x0000_0000	44.3.3.6.5/44-163
0x53FC_00C8 (SDC_CUR_BLINK_PWM_CTRL)	SDC Cursor Blinking and PWM Contrast Control Register	R/W	0x0000_0000	44.3.3.6.6/44-164
0x53FC_00CC (SDC_CUR_MAP)	SDC Color Cursor Mapping Register	R/W	0x0000_0000	44.3.3.6.7/44-165
0x53FC_00D0 (SDC_HOR_CONF)	SDC Horizontal Configuration Register	R/W	0x0000_0000	44.3.3.6.8/44-166
0x53FC_00D4 (SDC_VER_CONF)	SDC Vertical Configuration Register	R/W	0x0000_0000	44.3.3.6.9/44-167

Table 44-14. IPU Memory Map (continued)

Address	Register	Access	Reset Value	Section
0x53FC_00D8 (SDC_SHARP_CONF_1)	SDC Sharp Configuration Register 1	R/W	0x0000_0000	44.3.3.6.10/44-169
0x53FC_00DC (SDC_SHARP_CONF_2)	SDC Sharp Configuration Register 2	R/W	0x0000_0000	44.3.3.6.11/44-170
0x53FC_00E0 (ADC_CONF)	ADC Configuration Register	R/W	0x2020_0420	44.3.3.7.1/44-170
0x53FC_00E4 (ADC_SYSCHA1_SA)	ADC System Channel 1 Start Address Register	R/W	0x0000_0000	44.3.3.7.2/44-174
0x53FC_00E8 (ADC_SYSCHA2_SA)	ADC System Channel 2 Start Address Register	R/W	0x0000_0000	44.3.3.7.3/44-175
0x53FC_00EC (ADC_PRPCHAN_SA)	ADC Pre-Processing Channel Start Address Register	R/W	0x0000_0000	44.3.3.7.4/44-175
0x53FC_00F0 (ADC_PPCHAN_SA)	ADC Post-Processing Channel Start Address Register	R/W	0x0000_0000	44.3.3.7.5/44-176
0x53FC_00F4 (ADC_DISP0_CONF)	ADC Display 0 Configuration Register	R/W	0x0000_0000	44.3.3.7.6/44-177
0x53FC_00F8 (ADC_DISP0_RD_AP)	ADC Display 0 Read Acknowledge Pattern Register	R/W	0x0000_0000	44.3.3.7.7/44-178
0x53FC_00FC (ADC_DISP0_RDM)	ADC Display 0 Read Mask Register	R/W	0x0000_0000	44.3.3.7.8/44-179
0x53FC_0100 (ADC_DISP0_SS)	ADC Display 0 Screen Size Register	R/W	0x0000_0000	44.3.3.7.9/44-179
0x53FC_0104 (ADC_DISP1_CONF)	ADC Display 1 Configuration Register	R/W	0x0000_0000	44.3.3.7.10/44-180
0x53FC_0108 (ADC_DISP1_RD_AP)	ADC Display 1 Read Acknowledge Pattern Register	R/W	0x0000_0000	44.3.3.7.11/44-182
0x53FC_010C (ADC_DISP1_RDM)	ADC Display 1 Read Mask Register	R/W	0x0000_0000	44.3.3.7.12/44-182
0x53FC_0110 (ADC_DISP12_SS)	ADC Display 1 Screen Size Register	R/W	0x0000_0000	44.3.3.7.13/44-183
0x53FC_0114 (ADC_DISP2_CONF)	ADC Display 2 Configuration Register	R/W	0x0000_0000	44.3.3.7.14/44-184
0x53FC_0118 (ADC_DISP2_RD_AP)	ADC Display 2 Read Acknowledge Pattern Register	R/W	0x0000_0000	44.3.3.7.15/44-185
0x53FC_011C (ADC_DISP2_RDM)	ADC Display 2 Read Mask Register	R/W	0x0000_0000	44.3.3.7.16/44-186
0x53FC_0120 (ADC_DISP_VSYNC)	ADC Displays Vertical Synchronization Register	R/W	0x0000_0000	44.3.3.7.17/44-187
0x53FC_0124 (DI_DISP_IF_CONF)	DI Display Interface Configuration Register	R/W	0x0000_0000	44.3.3.8.1/44-189
0x53FC_0128 (DI_DISP_SIG_POL)	DI Display Signals Polarity Register	R/W	0x0000_0000	44.3.3.8.2/44-192
0x53FC_012C (DI_SER_DISP1_CONF)	DI Serial Display 1 Configuration Register	R/W	0x0000_0000	44.3.3.8.3/44-195
0x53FC_0130 (DI_SER_DISP2_CONF)	DI Serial Display 2 Configuration Register	R/W	0x0000_0000	44.3.3.8.4/44-197
0x53FC_0134 (DI_HSP_CLK_PER)	DI HSP_CLK Period Register	R/W	0x0000_0000	44.3.3.8.5/44-199

Table 44-14. IPU Memory Map (continued)

Address	Register	Access	Reset Value	Section
0x53FC_0138 (DI_DISP0_TIME_CONF_1)	DI Display 0 Time Configuration Register 1	R/W	0x0000_0000	44.3.3.8.6/44-200
0x53FC_013C (DI_DISP0_TIME_CONF_2)	DI Display 0 Time Configuration Register 2	R/W	0x0000_0000	44.3.3.8.7/44-202
0x53FC_0140 (DI_DISP0_TIME_CONF_3)	DI Display 0 Time Configuration Register 3	R/W	0x0000_0000	44.3.3.8.8/44-203
0x53FC_0144 (DI_DISP1_TIME_CONF_1)	DI Display 1 Time Configuration Register 1	R/W	0x0000_0000	44.3.3.8.9/44-205
0x53FC_0148 (DI_DISP1_TIME_CONF_2)	DI Display 1 Time Configuration Register 2	R/W	0x0000_0000	44.3.3.8.10/44-206
0x53FC_014C (DI_DISP1_TIME_CONF_3)	DI Display 1 Time Configuration Register 3	R/W	0x0000_0000	44.3.3.8.11/44-208
0x53FC_0150 (DI_DISP2_TIME_CONF_1)	DI Display 2 Time Configuration Register 1	R/W	0x0000_0000	44.3.3.8.12/44-209
0x53FC_0154 (DI_DISP2_TIME_CONF_2)	DI Display 2 Time Configuration Register 2	R/W	0x0000_0000	44.3.3.8.13/44-210
0x53FC_0158 (DI_DISP2_TIME_CONF_3)	DI Display 2 Time Configuration Register 3	R/W	0x0000_0000	44.3.3.8.14/44-212
0x53FC_015C (DI_DISP3_TIME_CONF)	DI Display 3 Time Configuration Register	R/W	0x0000_0000	44.3.3.8.15/44-213
0x53FC_0160 (DI_DISP0_DB0_MAP)	DI Display 0 Data Byte 0 Mapping Register	R/W	0x0000_FFFF	44.3.3.8.16/44-215
0x53FC_0164 (DI_DISP0_DB1_MAP)	DI Display 0 Data Byte 1 Mapping Register	R/W	0x0000_FFFF	44.3.3.8.17/44-216
0x53FC_0168 (DI_DISP0_DB2_MAP)	DI Display 0 Data Byte 2 Mapping Register	R/W	0x0000_FFFF	44.3.3.8.18/44-217
0x53FC_016C (DI_DISP0_CB0_MAP)	DI Display 0 Command Byte 0 Mapping Register	R/W	0x0000_FFFF	44.3.3.8.19/44-219
0x53FC_0170 (DI_DISP0_CB1_MAP)	DI Display 0 Command Byte 1 Mapping Register	R/W	0x0000_FFFF	44.3.3.8.20/44-220
0x53FC_0174 (DI_DISP0_CB2_MAP)	DI Display 0 Command Byte 2 Mapping Register	R/W	0x0000_FFFF	44.3.3.8.21/44-221
0x53FC_0178 (DI_DISP1_DB0_MAP)	DI Display 0 Data Byte 0 Mapping Register	R/W	0x0000_FFFF	44.3.3.8.22/44-222
0x53FC_017C (DI_DISP1_DB1_MAP)	DI Display 0 Data Byte1 Mapping Register	R/W	0x0000_FFFF	44.3.3.8.23/44-224
0x53FC_0180 (DI_DISP1_DB2_MAP)	DI Display 0 Data Byte 2 Mapping Register	R/W	0x0000_FFFF	44.3.3.8.24/44-225
0x53FC_0184 (DI_DISP1_CB0_MAP)	DI Display 1 Command Byte 0 Mapping Register	R/W	0x0000_FFFF	44.3.3.8.25/44-226

Table 44-14. IPU Memory Map (continued)

Address	Register	Access	Reset Value	Section
0x53FC_0188 (DI_DISP1_CB1_MAP)	DI Display 1 Command Byte 1 Mapping Register	R/W	0x0000_FFFF	44.3.3.8.26/44-227
0x53FC_018C (DI_DISP1_CB2_MAP)	DI Display 1 Command Byte 2 Mapping Register	R/W	0x0000_FFFF	44.3.3.8.27/44-229
0x53FC_0190 (DI_DISP2_DB0_MAP)	DI Display 2 Data Byte 0 Mapping Register	R/W	0x0000_FFFF	44.3.3.8.28/44-230
0x53FC_0194 (DI_DISP2_DB1_MAP)	DI Display 2 Data Byte 1 Mapping Register	R/W	0x0000_FFFF	44.3.3.8.29/44-231
0x53FC_0198 (DI_DISP2_DB2_MAP)	DI Display 2 Data Byte 2 Mapping Register	R/W	0x0000_FFFF	44.3.3.8.30/44-232
0x53FC_019C (DI_DISP2_CB0_MAP)	DI Display 2 Command Byte 0 Mapping Register	R/W	0x0000_FFFF	44.3.3.8.31/44-234
0x53FC_01A0 (DI_DISP2_CB1_MAP)	DI Display 2 Command Byte 1 Mapping Register	R/W	0x0000_FFFF	44.3.3.8.32/44-235
0x53FC_01A4 (DI_DISP2_CB2_MAP)	DI Display 2 Command Byte 2 Mapping Register	R/W	0x0000_FFFF	44.3.3.8.33/44-236
0x53FC_01A8 (DI_DISP3_B0_MAP)	DI Display 3 Byte 0 Mapping Register	R/W	0x0000_FFFF	44.3.3.8.34/44-237
0x53FC_01AC (DI_DISP3_B1_MAP)	DI Display 3 Byte 1 Mapping Register	R/W	0x0000_FFFF	44.3.3.8.35/44-239
0x53FC_01B0 (DI_DISP3_B2_MAP)	DI Display 3 Byte 2 Mapping Register	R/W	0x0000_FFFF	44.3.3.8.36/44-240
0x53FC_01B4 (DI_DISP_ACC_CC)	DI Display Access Cycles Count Register	R/W	0x0000_0000	44.3.3.8.37/44-241
0x53FC_01B8 (DI_DISP_LLA_CONF)	DI Display Low Level Access Configuration Register	R/W	0x0000_0000	44.3.3.8.38/44-243
0x53FC_01BC (DI_DISP_LLA_DATA)	DI Display Low Level Access Data Register	R/W	0x0000_0000	44.3.3.8.39/44-245

44.3.2 Register Summary

Figure 44-7 shows the key to the register fields and Table 44-15 shows the register figure conventions.

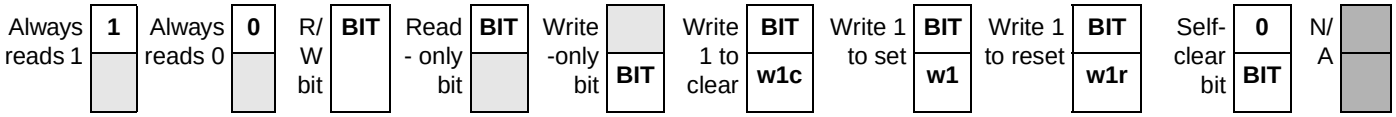


Figure 44-7. Key to Register Fields

Table 44-15. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writeable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.

Table 44-15. Register Figure Conventions (continued)

Convention	Description
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
w1	Write one to set. A status bit that can be read, and is set by writing a one.
w1r	Write one to reset. A status bit that can be read, and is got a reset value by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 44-16 shows the IPU register summary.

Table 44-16. IPU Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x53FC_0000 (IPU_CONF)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	PXL_ENDIAN	DU_EN	DI_EN	ADC_EN	SDC_EN	PF_EN	ROT_EN	IC_EN	CSI_EN
	W																

Table 44-16. IPU Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x53FC_0004 (IPU_CHA_BUF0_RDY)	R	DMAPF_7_BUF0_RDY	DMAPF_6_BUF0_RDY	DMAPF_5_BUF0_RDY	DMAPF_4_BUF0_RDY	DMAPF_3_BUF0_RDY	DMAPF_2_BUF0_RDY	DMAPF_1_BUF0_RDY	DMAPF_0_BUF0_RDY	DMAADC_7_BUF0_RDY	DMAADC_6_BUF0_RDY	DMAADC_5_BUF0_RDY	DMAADC_4_BUF0_RDY	DMAADC_3_BUF0_RDY	DMAADC_2_BUF0_RDY	DMASDC_3_BUF0_RDY	DMASDC_2_BUF0_RDY
	W	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s
	R	DMASDC_1_BUF0_RDY	DMASDC_0_BUF0_RDY	DMAIC_13_BUF0_RDY	DMAIC_12_BUF0_RDY	DMAIC_11_BUF0_RDY	DMAIC_10_BUF0_RDY	DMAIC_9_BUF0_RDY	DMAIC_8_BUF0_RDY	DMAIC_7_BUF0_RDY	DMAIC_6_BUF0_RDY	DMAIC_5_BUF0_RDY	DMAIC_4_BUF0_RDY	DMAIC_3_BUF0_RDY	DMAIC_2_BUF0_RDY	DMAIC_1_BUF0_RDY	DMAIC_0_BUF0_RDY
	W	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s
0x53FC_0008 (IPU_CHA_BUF1_RDY)	R	DMAPF_7_BUF1_RDY	DMAPF_6_BUF1_RDY	DMAPF_5_BUF1_RDY	DMAPF_4_BUF1_RDY	DMAPF_3_BUF1_RDY	DMAPF_2_BUF1_RDYY	DMAPF_1_BUF1_RDY	DMAPF_0_BUF1_RDY	DMAADC_7_BUF1_RDY	DMAADC_6_BUF1_RDY	DMAADC_5_BUF1_RDY	DMAADC_4_BUF1_RDY	DMAADC_3_BUF1_RDY	DMAADC_2_BUF1_RDY	DMASDC_3_BUF1_RDY	DMASDC_2_BUF1_RDY
	W	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s
	R	DMASDC_1_BUF1_RDY	DMASDC_0_BUF1_RDY	DMAIC_13_BUF1_RDY	DMAIC_12_BUF1_RDY	DMAIC_11_BUF1_RDY	DMAIC_10_BUF1_RDY	DMAIC_9_BUF1_RDY	DMAIC_8_BUF1_RDY	DMAIC_7_BUF1_RDY	DMAIC_6_BUF1_RDY	DMAIC_5_BUF1_RDY	DMAIC_4_BUF1_RDY	DMAIC_3_BUF1_RDY	DMAIC_2_BUF1_RDY	DMAIC_1_BUF1_RDY	DMAIC_0_BUF1_RDY
	W	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s

Table 44-16. IPU Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x53FC_000C (IPU_CHA_DB_MODE_SEL)	R	0	0	DMAPF_5_DBMS	0	0	DMAPF_2_DBMS	0	0	0	0	0	0	DMAADC_3_DBMS	DMAADC_2_DBMS	0	DMASDC_2_DBMS
	W																
	R	DMASDC_1_DBMS	DMASDC_0_DBMS	DMAIC_13_DBMS	DMAIC_12_DBMS	DMAIC_11_DBMS	DMAIC_10_DBMS	DMAIC_9_DBMS	DMAIC_8_DBMS	DMAIC_7_DBMS	DMAIC_6_DBMS	DMAIC_5_DBMS	DMAIC_4_DBMS	DMAIC_3_DBMS	DMAIC_2_DBMS	DMAIC_1_DBMS	DMAIC_0_DBMS
	W																
0x53FC_0010 (IPU_CHA_CUR_BUF)	R	w1r	w1r	w1r	w1r	w1r	w1r	w1r	w1r	w1r	w1r	w1r	w1r	w1r	w1r	w1r	w1r
	W	DMAPF_7_CUR_BUF	DMAPF_6_CUR_BUF	DMAPF_5_CUR_BUF	DMAPF_4_CUR_BUF	DMAPF_3_CUR_BUF	DMAPF_2_CUR_BUF	DMAPF_1_CUR_BUF	DMAPF_0_CUR_BUF	DMAADC_7_CUR_BUF	DMAADC_6_CUR_BUF	DMAADC_5_CUR_BUF	DMAADC_4_CUR_BUF	DMAADC_3_CUR_BUF	DMAADC_2_CUR_BUF	DMASDC_3_CUR_BUF	DMASDC_2_CUR_BUF
	R	w1r	w1r	w1r	w1r	w1r	w1r	w1r	w1r	w1r	w1r	w1r	w1r	w1r	w1r	w1r	w1r
	W	DMASDC_1_CUR_BUF	DMASDC_0_CUR_BUF	DMAIC_13_CUR_BUF	DMAIC_12_CUR_BUF	DMAIC_11_CUR_BUF	DMAIC_10_CUR_BUF	DMAIC_9_CUR_BUF	DMAIC_8_CUR_BUF	DMAIC_7_CUR_BUF	DMAIC_6_CUR_BUF	DMAIC_5_CUR_BUF	DMAIC_4_CUR_BUF	DMAIC_3_CUR_BUF	DMAIC_2_CUR_BUF	DMAIC_1_CUR_BUF	DMAIC_0_CUR_BUF

Table 44-16. IPU Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x53FC_0014 (IPU_FS_PROC_FLOW)	R	0	PP_ROT_DEST_SEL			0	PP_DEST_SEL			0	PRPVF_ROT_DEST_SEL			0	PRPVF_DEST_SEL		
	W																
	R	0	0	PF_DEST_SEL		PP_ROT_SRC_SEL		PP_SRC_SEL		0	PRPVF_ROT_SRC_SEL	PRPENC_ROT_SRC_SEL	PRPENC_DEST_SEL	0	0	VF_IN_VALID	ENC_IN_VALID
	W																
0x53FC_0018 (IPU_FS_DISP_FLOW)	R	0	0	0	0	0	0	AUTO_REF_PER									
	W																
	R	0	ADC3_SRC_SEL			0	ADC2_SRC_SEL			0	SDC1_SRC_SEL			0	SDC0_SRC_SEL		
	W																
0x53FC_001C (IPU_TASKS_STAT)	R	ADC_SYS2CHAN_LOCK	ADC_SYS1CHAN_LOCK	ADC_PPCHAN_LOCK	ADC_PRPCHAN_LOCK	ADCSYS2_TSTAT		ADCSYS1_TSTAT		PF_TSTAT		PP_ROT_TSTAT		VF_ROT_TSTAT		ENC_ROT_TSTAT	
	W																
	R	0	PF_H264_Y_PAUSE	SDC_PIX_SKIP	PP_TSTAT		VF_TSTAT		ENC_TSTAT		MEM2PRP_TSTAT			CSI2MEM_TSTAT		CSI_SKIP_TSTAT	
	W																

Table 44-16. IPU Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x53FC_0020 (IPU_IMA_ADDR)	R	0	0	0	0	0	0	0	0	0	0	0	0	MEM_NU			
	W																
	R	ROW_NU													WORD_NU		
	W																
0x53FC_0024 (IPU_IMA_DATA)	R	IMA_DATA[31:16]															
	W																
	R	IMA_DATA[15:0]															
	W																
0x53FC_0028 (IPU_INT_CTRL_1)	R	DMAPF_7_EOF_EN	DMAPF_6_EOF_EN	DMAPF_5_EOF_EN	DMAPF_4_EOF_EN	DMAPF_3_EOF_EN	DMAPF_2_EOF_EN	DMAPF_1_EOF_EN	DMAPF_0_EOF_EN	DMAADC_7_EOF_EN	DMAADC_6_EOF_EN	DMAADC_5_EOF_EN	DMAADC_4_EOF_EN	DMAADC_3_EOF_EN	DMAADC_2_EOF_EN	DMASDC_3_EOF_EN	DMASDC_2_EOF_EN
	W																
	R	DMASDC_1_EOF_EN	DMASDC_0_EOF_EN	DMAIC_13_EOF_EN	DMAIC_12_EOF_EN	DMAIC_11_EOF_EN	DMAIC_10_EOF_EN	DMAIC_9_EOF_EN	DMAIC_8_EOF_EN	DMAIC_7_EOF_EN	DMAIC_6_EOF_EN	DMAIC_5_EOF_EN	DMAIC_4_EOF_EN	DMAIC_3_EOF_EN	DMAIC_2_EOF_EN	DMAIC_1_EOF_EN	DMAIC_0_EOF_EN
	W																
0x53FC_002C (IPU_INT_CTRL_2)	R	DMAPF_7_NFACK_EN	DMAPF_6_NFACK_EN	DMAPF_5_NFACK_EN	DMAPF_4_NFACK_EN	DMAPF_3_NFACK_EN	DMAPF_2_NFACK_EN	DMAPF_1_NFACK_EN	DMAPF_0_NFACK_EN	DMAADC_7_NFACK_EN	DMAADC_6_NFACK_EN	DMAADC_5_NFACK_EN	DMAADC_4_NFACK_EN	DMAADC_3_NFACK_EN	DMAADC_2_NFACK_EN	DMASDC_3_NFACK_EN	DMASDC_2_NFACK_EN
	W																
	R	DMASDC_1_NFACK_EN	DMASDC_0_NFACK_EN	DMAIC_13_NFACK_EN	DMAIC_12_NFACK_EN	DMAIC_11_NFACK_EN	DMAIC_10_NFACK_EN	DMAIC_9_NFACK_EN	DMAIC_8_NFACK_EN	DMAIC_7_NFACK_EN	DMAIC_6_NFACK_EN	DMAIC_5_NFACK_EN	DMAIC_4_NFACK_EN	DMAIC_3_NFACK_EN	DMAIC_2_NFACK_EN	DMAIC_1_NFACK_EN	DMAIC_0_NFACK_EN
	W																

Table 44-16. IPU Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x53FC_0030 (IPU_INT_CTRL_3)	R	0	0	0	0	0	0	0	0	STOP_MODE_ACK_EN	ADC_SYS2_EOF_EN	ADC_SYS1_EOF_EN	ADC_PP_EOF_EN	ADC_PRP_EOF_EN	ADC_DISP12_VSYNC_EN	ADC_DISP0_VSYNC_EN	SDC_DISP3_VSYNC_EN
	W																
	R	DMAADC_3_SBUF_END_EN	DMAADC_2_SBUF_END_EN	DMASDC_2_SBUF_END_EN	DMASDC_1_SBUF_END_EN	DMASDC_0_SBUF_END_EN	DMAIC_6_SBUF_END_EN	DMAIC_5_SBUF_END_EN	DMAIC_4_SBUF_END_EN	DMAIC_3_SBUF_END_EN	CSI_EOF_EN	CSI_NF_EN	SERIAL_DATA_FINISH_EN	SDC_MSK_EOF_EN	SDC_FG_EOF_EN	SDC_BG_EOF_EN	BRK_RQ_STAT_EN
	W																
0x53FC_0034 (IPU_INT_CTRL_4)	R	DMAPF_7_NFB4EOF_ERR_EN	DMAPF_6_NFB4EOF_ERR_EN	DMAPF_5_NFB4EOF_ERR_EN	DMAPF_4_NFB4EOF_ERR_EN	DMAPF_3_NFB4EOF_ERR_EN	DMAPF_2_NFB4EOF_ERR_EN	DMAPF_1_NFB4EOF_ERR_EN	DMAPF_0_NFB4EOF_ERR_EN	DMAADC_7_NFB4EOF_ERR_EN	DMAADC_6_NFB4EOF_ERR_EN	DMAADC_5_NFB4EOF_ERR_EN	DMAADC_4_NFB4EOF_ERR_EN	DMAADC_3_NFB4EOF_ERR_EN	DMAADC_2_NFB4EOF_ERR_EN	DMASDC_3_NFB4EOF_ERR_EN	DMASDC_2_NFB4EOF_ERR_EN
	W																
	R	DMASDC_1_NFB4EOF_ERR_EN	DMASDC_0_NFB4EOF_ERR_EN	DMAIC_13_NFB4EOF_ERR_EN	DMAIC_12_NFB4EOF_ERR_EN	DMAIC_11_NFB4EOF_ERR_EN	DMAIC_10_NFB4EOF_ERR_EN	DMAIC_9_NFB4EOF_ERR_EN	DMAIC_8_NFB4EOF_ERR_EN	DMAIC_7_NFB4EOF_ERR_EN	DMAIC_6_NFB4EOF_ERR_EN	DMAIC_5_NFB4EOF_ERR_EN	DMAIC_4_NFB4EOF_ERR_EN	DMAIC_3_NFB4EOF_ERR_EN	DMAIC_2_NFB4EOF_ERR_EN	DMAIC_1_NFB4EOF_ERR_EN	DMAIC_0_NFB4EOF_ERR_EN
	W																

Table 44-16. IPU Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x53FC_0038 (IPU_INT_CTRL_5)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	SAHB_ADDR_ERR_EN
	W																
	R	DI_LLA_LOCK_ERR_EN	DI_ADC_LOCK_ERR_EN	VF_FRM_LOST_ERR_EN	ENC_FRM_LOST_ERR_EN	BAYER_FRM_LOST_ERR_EN	SDC_MSKD_ERR_EN	SDC_FGD_ERR_EN	SDC_BGD_ERR_EN	AHB_M2_ERR_EN	AHB_M1_ERR_EN	ADC_SYS2_TEARING_ERR_EN	ADC_SYS1_TEARING_ERR_EN	ADC_PP_TEARING_ERR_EN	VF_BUF_OVF_ERR_EN	ENC_BUF_OVF_ERR_EN	BAYER_BUF_OVF_ERR_EN
	W																
0x53FC_003C (IPU_INT_STAT_1)	R	DMAPF_7_EOF	DMAPF_6_EOF	DMAPF_5_EOF	DMAPF_4_EOF	DMAPF_3_EOF	DMAPF_2_EOF	DMAPF_1_EOF	DMAPF_0_EOF	DMAADC_7_EOF	DMAADC_6_EOF	DMAADC_5_EOF	DMAADC_4_EOF	DMAADC_3_EOF	DMAADC_2_EOF	DMASDC_3_EOF	DMASDC_2_EOF
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
	R	DMASDC_1_EOF	DMASDC_0_EOF	DMAIC_13_EOF	DMAIC_12_EOF	DMAIC_11_EOF	DMAIC_10_EOF	DMAIC_9_EOF	DMAIC_8_EOF	DMAIC_7_EOF	DMAIC_6_EOF	DMAIC_5_EOF	DMAIC_4_EOF	DMAIC_3_EOF	DMAIC_2_EOF	DMAIC_1_EOF	DMAIC_0_EOF
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c

Table 44-16. IPU Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x53FC_0040 (IPU_INT_STAT_2)	R	DMAPF_7_NFACK	DMAPF_6_NFACK	DMAPF_5_NFACK	DMAPF_4_NFACK	DMAPF_3_NFACK	DMAPF_2_NFACK	DMAPF_1_NFACK	DMAPF_0_NFACK	DMAADC_7_NFACK	DMAADC_6_NFACK	DMAADC_5_NFACK	DMAADC_4_NFACK	DMAADC_3_NFACK	DMAADC_2_NFACK	DMASDC_3_NFACK	DMASDC_2_NFACK
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
	R	DMASDC_1_NFACK	DMASDC_0_NFACK	DMAIC_13_NFACK	DMAIC_12_NFACK	DMAIC_11_NFACK	DMAIC_10_NFACK	DMAIC_9_NFACK	DMAIC_8_NFACK	DMAIC_7_NFACK	DMAIC_6_NFACK	DMAIC_5_NFACK	DMAIC_4_NFACK	DMAIC_3_NFACK	DMAIC_2_NFACK	DMAIC_1_NFACK	DMAIC_0_NFACK
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
0x53FC_0044 (IPU_INT_STAT_3)	R	0	0	0	0	0	0	0	0	STOP_MODE_ACK	ADC_SYS2_EOF	ADC_SYS1_EOF	ADC_PP_EOF	ADC_PRP_EOF	ADC_DISP12_VSYNC	ADC_DISP0_VSYNC	SDC_DISP3_VSYNC
	W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
	R	DMAADC_3_SBUF_END	DMAADC_2_SBUF_END	DMASDC_2_SBUF_END	DMASDC_1_SBUF_END	DMASDC_0_SBUF_END	DMAIC_6_SBUF_END	DMAIC_5_SBUF_END	DMAIC_4_SBUF_END	DMAIC_3_SBUF_END	CSI_EOF	CSI_NF	SERIAL_DATA_FINISH	SDC_MSK_EOF	SDC_FG_EOF	SDC_BG_EOF	BRK_RQ_STAT
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c

Table 44-16. IPU Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x53FC_0048 (IPU_INT_STAT_4)	R	DMAPF_7_NFB4EOF_ERR	DMAPF_6_NFB4EOF_ERR	DMAPF_5_NFB4EOF_ERR	DMAPF_4_NFB4EOF_ERR	DMAPF_3_NFB4EOF_ERR	DMAPF_2_NFB4EOF_ERR	DMAPF_1_NFB4EOF_ERR	DMAPF_0_NFB4EOF_ERR	DMAADC_7_NFB4EOF_ERR	DMAADC_6_NFB4EOF_ERR	DMAADC_5_NFB4EOF_ERR	DMAADC_4_NFB4EOF_ERR	DMAADC_3_NFB4EOF_ERR	DMAADC_2_NFB4EOF_ERR	DMASDC_3_NFB4EOF_ERR	DMASDC_2_NFB4EOF_ERR
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
	R	DMASDC_1_NFB4EOF_ERR	DMASDC_0_NFB4EOF_ERR	DMAIC_13_NFB4EOF_ERR	DMAIC_12_NFB4EOF_ERR	DMAIC_11_NFB4EOF_ERR	DMAIC_10_NFB4EOF_ERR	DMAIC_9_NFB4EOF_ERR	DMAIC_8_NFB4EOF_ERR	DMAIC_7_NFB4EOF_ERR	DMAIC_6_NFB4EOF_ERR	DMAIC_5_NFB4EOF_ERR	DMAIC_4_NFB4EOF_ERR	DMAIC_3_NFB4EOF_ERR	DMAIC_2_NFB4EOF_ERR	DMAIC_1_NFB4EOF_ERR	DMAIC_0_NFB4EOF_ERR
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
0x53FC_004C (IPU_INT_STAT_5)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	SAHB_ADDR_ERR
	W																
	R	DI_LLA_LOCK_ERR	DI_ADC_LOCK_ERR	VF_FRM_LOST_ERR	ENC_FRM_LOST_ERR	BAYER_FRM_LOST_ERR	SDC_MSKD_ERR	SDC_FGD_ERR	SDC_BGD_ERR	AHB_M2_ERR	AHB_M1_ERR	ADC_SYS2_TEARING_ERR	ADC_SYS1_TEARING_ERR	ADC_PP_TEARING_ERR	VF_BUF_OVF_ERR	ENC_BUF_OVF_ERR	BAYER_BUF_OVF_ERR
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c

Table 44-16. IPU Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x53FC_0050 (IPU_BRK_CTRL_1)	R	BRK_EVNT_NUM								BRK_GRP_SEL		BRK_SIG_SEL					
	W																
	R	0	0	0	SDC_DBG_MASK_DIS	BRK_SIG_COND_EN	BRK_COL_COND_EN	BRK_ROW_COND_EN	BRK_CHA_COND_EN	BRK_RQ_MODE		DBG_ENTER_MODE		0	DBG_EXIT	FRC_DGB	BRK_EN
	W																
0x53FC_0054 (IPU_BRK_CTRL_2)	R	CSI_CHA_EN	BRK_CHA_NUM					0	BRK_COL_NUM[11:3]								
	W																
	R	BRK_COL_NUM[2:0]			BRK_ROW_NUM												
	W																
0x53FC_0058 (IPU_BRK_STAT)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	BRK_SRC	MCU_DBGRQ	IPU_BREAK_ACK
	W																
0x53FC_005C (IPU_DIAGB_CTRL)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
0x53FC_0060 (CSI_SENS_CONF)	R	0	0	0	0	0	0	0	0	DIV_RATIO							
	W																
	R	EXT_VSYNC	0	0	0	DATA_WITDH		SENS_DATA_FORMAT		SENS_CLK_SRC	0	SENS_PRTCL		SENS_PIX_CLK_POL	DATA_POL	HSYNC_POL	VSYNC_POL
	W																

Table 44-16. IPU Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x53FC_0064 (CSI_SENS_FRM_SIZE)	R	0	0	0	0	SENS_FRM_HEIGHT											
	W																
	R	0	0	0	0	SENS_FRM_WIDTH											
	W																
0x53FC_0068 (CSI_ACT_FRM_SIZE)	R	0	0	0	0	ACT_FRM_HEIGHT											
	W																
	R	0	0	0	0	ACT_FRM_WIDTH											
	W																
0x53FC_006C (CSI_OUT_FRM_CTRL)	R	0	0	HORZ_DWNS	VERT_DWNS	0	IC_TV_MODE	SKIP_VF				SKIP_ENC					
	W																
	R	HSC								VSC							
	W																
0x53FC_0070 (CSI_TST_CTRL)	R	0	0	0	0	0	0	0	TEST_GEN_MODE	PG_B_VALUE							
	W																
	R	PG_G_VALUE								PG_R_VALUE							
	W																
0x53FC_0074 (CSI_CCIR_CODE_1)	R	0	0	0	0	0	0	0	CCIR_ERR_DET_EN	0	0	STRT_FLD0_ACTV			END_FLD0_ACTV		
	W																
	R	0	0	0	0	STRT_FLD0_BLNK_2ND			END_FLD0_BLNK_2ND			STRT_FLD0_BLNK_1ST			END_FLD0_BLNK_1ST		
	W																
0x53FC_0078 (CSI_CCIR_CODE_2)	R	0	0	0	0	0	0	0	0	0	0	STRT_FLD1_ACTV			END_FLD1_ACTV		
	W																
	R	0	0	0	0	STRT_FLD1_BLNK_2ND			END_FLD1_BLNK_2ND			STRT_FLD1_BLNK_1ST			END_FLD1_BLNK_1ST		
	W																

Table 44-16. IPU Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x53FC_007C (CSI_CCIR_CODE_3)	R	0	0	0	0	0	0	0	0	CCIR_PRECOM[23:16]							
	W																
	R	CCIR_PRECOM[15:0]															
	W																
0x53FC_0080 (CSI_FLASH_STROBE_1)	R	SENS_ROW_DURATION															
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CLOCK_SEL
	W																
0x53FC_0084 (CSI_FLASH_STROBE_2)	R	STROBE_DURATION															
	W																
	R	STROBE_START_TIME													0	STROBE_POL	STROBE_EN
	W																
0x53FC_0088 (IC_CONF)	R	CSI_MEM_WR_EN	RWS_EN	IC_KEY_COLOR_EN	IC_GLB_LOC_A	0	0	0	0	0	0	0	PP_ROT_EN	PP_CMB	PP_CSC2	PP_CSC1	PP_EN
	W																
	R	0	0	0	PRPVF_ROT_EN	PRPVF_CMB	PRPVF_CSC2	PRPVF_CSC1	PRPVF_EN	0	0	0	0	0	PRPENC_ROT_EN	PRPENC_CSC1	PRPENC_EN
	W																
0x53FC_008C (IC_PRP_ENC_RSC)	R	PRPENC_DS_R_V		PRPENC_RS_R_V													
	W																
	R	PRPENC_DS_R_H		PRPENC_RS_R_H													
	W																

Table 44-16. IPU Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x53FC_0090 (IC_PRP_VF_RSC)	R	PRPVF_DS_R_V		PRPVF_RS_R_V													
	W																
	R	PRPVF_DS_R_H		PRPVF_RS_R_H													
	W																
0x53FC_0094 (IC_PP_RSC)	R	PP_DS_R_V		PP_RS_R_V													
	W																
	R	PP_DS_R_H		PP_RS_R_H													
	W																
0x53FC_0098 (IC_CMBP_1)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	IC_PP_ALPHA_V								IC_PRPVF_ALPHA_V							
	W																
0x53FC_009C (IC_CMBP_2)	R	0	0	0	0	0	0	0	0	IC_KEY_COLOR_R							
	W																
	R	IC_KEY_COLOR_G								IC_KEY_COLOR_B							
	W																
0x53FC_00A0 (PF_CONF)	R	0	0	0	0	0	0	0	0	0	0	H264_Y_PAUSE_ROW					
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	H264_Y_PAUSE_EN	0	PF_TYPE		
	W																
0x53FC_00A4 (IDMAC_CONF)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	SINGLE_AHB_M_EN	0	SRCNT			0	0	PRYM	
	W																

Table 44-16. IPU Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x53FC_00A8 (IDMAC_CHA_EN)	R	DMAPF_7_CHAN_EN	DMAPF_6_CHAN_EN	DMAPF_5_CHAN_EN	DMAPF_4_CHAN_EN	DMAPF_3_CHAN_EN	DMAPF_2_CHAN_EN	DMAPF_1_CHAN_EN	DMAPF_0_CHAN_EN	DMAADC_7_CHAN_EN	DMAADC_6_CHAN_EN	DMAADC_5_CHAN_EN	DMAADC_4_CHAN_EN	DMAADC_3_CHAN_EN	DMAADC_2_CHAN_EN	DMASDC_3_CHAN_EN	DMASDC_2_CHAN_EN
	W																
	R	DMASDC_1_CHAN_EN	DMASDC_0_CHAN_EN	DMAIC_13_CHAN_EN	DMAIC_12_CHAN_EN	DMAIC_11_CHAN_EN	DMAIC_10_CHAN_EN	DMAIC_9_CHAN_EN	DMAIC_8_CHAN_EN	DMAIC_7_CHAN_EN	DMAIC_6_CHAN_EN	DMAIC_5_CHAN_EN	DMAIC_4_CHAN_EN	DMAIC_3_CHAN_EN	DMAIC_2_CHAN_EN	DMAIC_1_CHAN_EN	DMAIC_0_CHAN_EN
	W																
0x53FC_00Ac (IDMAC_CHA_PRI)	R	DMAPF_7_PRI	DMAPF_6_PRI	DMAPF_5_PRI	DMAPF_4_PRI	DMAPF_3_PRI	DMAPF_2_PRI	DMAPF_1_PRI	DMAPF_0_PRI	DMAADC_7_PRI	DMAADC_6_PRI	DMAADC_5_PRI	DMAADC_4_PRI	DMAADC_3_PRI	DMAADC_2_PRI	DMASDC_3_PRI	DMASDC_2_PRI
	W																
	R	DMASDC_1_PRI	DMASDC_0_PRI	DMAIC_13_PRI	DMAIC_12_PRI	DMAIC_11_PRI	DMAIC_10_PRI	DMAIC_9_PRI	DMAIC_8_PRI	DMAIC_7_PRI	DMAIC_6_PRI	DMAIC_5_PRI	DMAIC_4_PRI	DMAIC_3_PRI	DMAIC_2_PRI	DMAIC_1_PRI	DMAIC_0_PRI
	W																
0x53FC_00B0 (IDMAC_CHA_BUSY)	R	DMAPF_7_BUSY	DMAPF_6_BUSY	DMAPF_5_BUSY	DMAPF_4_BUSY	DMAPF_3_BUSY	DMAPF_2_BUSY	DMAPF_1_BUSY	DMAPF_0_BUSY	DMAADC_7_BUSY	DMAADC_6_BUSY	DMAADC_5_BUSY	DMAADC_4_BUSY	DMAADC_3_BUSY	DMAADC_2_BUSY	DMASDC_3_BUSY	DMASDC_2_BUSY
	W																
	R	DMASDC_1_BUSY	DMASDC_0_BUSY	DMAIC_13_BUSY	DMAIC_12_BUSY	DMAIC_11_BUSY	DMAIC_10_BUSY	DMAIC_9_BUSY	DMAIC_8_BUSY	DMAIC_7_BUSY	DMAIC_6_BUSY	DMAIC_5_BUSY	DMAIC_4_BUSY	DMAIC_3_BUSY	DMAIC_2_BUSY	DMAIC_1_BUSY	DMAIC_0_BUSY
	W																

Table 44-16. IPU Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x53FC_00B4 (SDC_COM_CONF)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	COC		
	W																
	R	DUAL_MODE	SAVE_REFR_EN	0	SHARP	0	0	BG_EN	MASK_EN	SDC_KEY_COLOR_EN	SDC_GLB_LOC_A	GWSEL	FG_EN	FG_MCP_FORM	BG_MCP_FORM	SDC_MODE	
	W																
0x53FC_00B8 (SDC_GRAPH_WIND_CTRL)	R	SDC_ALPHA_V								SDC_KEY_COLOR_R							
	W																
	R	SDC_KEY_COLOR_G								SDC_KEY_COLOR_B							
	W																
0x53FC_00BC (SDC_FG_POS)	R	0	0	0	0	0	0	FGXP									
	W																
	R	0	0	0	0	0	0	FGYP									
	W																
0x53FC_00C0 (SDC_BG_POS)	R	0	0	0	0	0	0	BGXP									
	W																
	R	0	0	0	0	0	0	BGYP									
	W																
0x53FC_00C4 (SDC_CUR_POS)	R	0	CXW					CXP									
	W																
	R	0	CYH					CYP									
	W																
0x53FC_00C8 (SDC_CUR_BLINK_PWM_CTRL)	R	0	0	0	0	0	SCR		CC_EN	PWM							
	W																
	R	BK_EN	0	0	0	0	0	0	0	BKDIV							
	W																
0x53FC_00CC (SDC_CUR_MAP)	R	0	0	0	0	0	0	0	0	CUR_COL_R							
	W																
	R	CUR_COL_G								CUR_COL_B							
	W																

Table 44-16. IPU Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x53FC_00D0 (SDC_HOR_CONF)	R	H_SYNC_WIDTH						SCREEN_WIDTH									
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	H_SYNC_DELAY			
	W																
0x53FC_00D4 (SDC_VER_CONF)	R	V_SYNC_WIDTH						SCREEN_HEIGHT									
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	V_SYNC_WIDTH_L
	W																
0x53FC_00D8 (SDC_SHARP_CONF_1)	R	0	0	0	0	0	0	REV_TOGGLE_DELAY									
	W																
	R	PS_FALL_DELAY								CLS_RISE_DELAY							
	W																
0x53FC_00DC (SDC_SHARP_CONF_2)	R	0	0	0	0	0	0	PS_RISE_DELAY									
	W																
	R	0	0	0	0	0	0	CLS_FALL_DELAY									
	W																
0x53FC_00E0 (ADC_CONF)	R	SYS2_DATA_MAP		SYS2_ADDR_INC		SYS2_DISP_NUM		SYS2_MODE		SYS1_DATA_MAP		SYS1_ADDR_INC		SYS1_DISP_NUM		SYS1_MODE	
	W																
	R	SYS2_NO_TEARING	SYS1_NO_TEARING	PP_NO_TEARING	PP_DATA_MAP	PP_ADDR_INC		PP_DISP_NUM		PRP_DATA_MAP		PRP_ADDR_INC		PRP_DISP_NUM		MCU_CHAN_EN	PP_CHAN_EN	PRP_CHAN_EN
	W																

Table 44-16. IPU Register Summary (continued)

Name		31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0x53FC_00E4 (ADC_SYSCHA1_SA)	R	SYS1_START_TIME | SYS1_CHAN_SA[22:16]
	W	
	R	SYS1_CHAN_SA[15:0]
	W	
0x53FC_00E8 (ADC_SYSCHA2_SA)	R	SYS2_START_TIME | SYS2_CHAN_SA[22:16]
	W	
	R	SYS2_CHAN_SA[15:0]
	W	
0x53FC_00EC (ADC_PRPCHAN_SA)	R	0 0 0 0 0 0 0 0 0 | PRP_CHAN_SA[22:16]
	W	
	R	PRPCHAN_SA[15:0]
	W	
0x53FC_00F0 (ADC_PPCHAN_SA)	R	PP_START_TIME | PP_CHAN_SA[22:16]
	W	
	R	PPCHAN_SA[15:0]
	W	
0x53FC_00F4 (ADC_DISP0_CONF)	R	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
	W	
	R	MCU_DISP0_DATA_MAP | MCU_DISP0_DATA_WIDTH | DISP0_TYPE | DISP0_SL
	W	
0x53FC_00F8 (ADC_DISP0_RD_AP)	R	0 0 0 0 0 0 0 0 | DISP0_ACK_PTRN[23:16]
	W	
	R	DISP0_ACK_PTRN[15:0]
	W	

Table 44-16. IPU Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x53FC_00FC (ADC_DISP0_RDM)	R	0	0	0	0	0	0	0	0	DISP0_MASK_ACK_DATA[23:16]							
	W																
	R	DISP0_MASK_ACK_DATA[15:0]															
	W																
0x53FC_0100 (ADC_DISP0_SS)	R	0	0	0	0	0	0	SCREEN0_HEIGHT									
	W																
	R	0	0	0	0	0	0	SCREEN0_WIDTH									
	W																
0x53FC_0104 (ADC_DISP1_CONF)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	MCU_DISP1_DATA_MAP	MCU_DISP0_DATA_WIDTH	DISP0_TYPE				DISP0_SL									
	W																
0x53FC_0108 (ADC_DISP1_RD_AP)	R	0	0	0	0	0	0	0	0	DISP1_ACK_PTRN[23:16]							
	W																
	R	DISP1_ACK_PTRN[15:0]															
	W																
0x53FC_010C (ADC_DISP1_RDM)	R	0	0	0	0	0	0	0	0	DISP1_MASK_ACK_DATA[23:16]							
	W																
	R	DISP1_MASK_ACK_DATA[15:0]															
	W																
0x53FC_0110 (ADC_DISP12_SS)	R	0	0	0	0	0	0	SCREEN12_HEIGHT									
	W																
	R	0	0	0	0	0	0	SCREEN12_WIDTH									
	W																

Table 44-16. IPU Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x53FC_0114 (ADC_DISP2_CONF)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	MCU_DISP2_DATA_MAP	MCU_DISP0_DATA_WIDTH	DISP0_TYPE		DISP0_SL											
	W																
0x53FC_0118 (ADC_DISP2_RD_AP)	R	0	0	0	0	0	0	0	0	DISP2_ACK_PTRN[23:16]							
	W																
	R	DISP2_ACK_PTRN[15:0]															
	W																
0x53FC_011C (ADC_DISP2_RDM)	R	0	0	0	0	0	0	0	0	DISP2_MASK_ACK_DATA[23:16]							
	W																
	R	DISP2_MASK_ACK_DATA[15:0]															
	W																
0x53FC_0120 (ADC_DISP_VSYNC)	R	0	DISP12_VSYNC_WIDTH_L	DISP12_VSYNC_WIDTH						0	DISP0_VSYNC_WIDTH_L	DISP0_VSYNC_WIDTH					
	W																
	R	DISP_LN_WT										0	DISP12_VSYNC_SEL	DISP12_VSYNC_MODE		DISPL0_VSYNC_MODE	
	W																

Table 44-16. IPU Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x53FC_0124 (DI_DISP_IF_CONF)	R	0	DISP012_DEAD_CLK_NUM				DISP3_CLK_IDLE	DISP3_CLK_SEL	DISP3_DATAMSK	0	0	DISP2_PAR_BURST_MODE		DISP2_IF_MODE			DISP2_EN
	W																
	R	0	0	DISP1_PAR_BURST_MODE		DISP1_IF_MODE			DISP1_EN	0	0	0	DISP0_PAR_BURST_MODE		DISP0_IF_MODE		DISP0_EN
	W																
0x53FC_0128 (DI_DISP_SIG_POL)	R	D2_BCLK_POL	D1_BCLK_POL	D0_BCLK_POL	D3_VSYNC_POL	D3_HSYNC_POL	D3_DRDY_SHARP_POL	D3_CLK_POL	D3_DATA_POL	D2_SER_RS_POL	D2_SD_CLK_POL	D2_SD_D_POL	D2_RD_POL	D2_WR_POL	D2_PAR_RS_POL	D2_CS_POL	D2_DATA_POL
	W																
	R	D1_SER_RS_POL	D1_SD_CLK_POL	D1_SD_D_POL	D1_RD_POL	D1_WR_POL	D1_PAR_RS_POL	D1_CS_POL	D1_DATA_POL	0	D12_VSYNC_POL	D0_VSYNC_POL	D0_RD_POL	D0_WR_POL	D0_PAR_RS_POL	D0_CS_POL	D0_DATA_POL
	W																

Table 44-16. IPU Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x53FC_012C (DI_SER_DISP1_CONF)	R	0	0	0	0	0	0	0	DISP1_SER_BURST_MODE	0	0	0	DISP1_SER_BIT_NUM				
	W																
	R	DISP1_PREAMBLE								0	DISP1_PREAMBLE_LENGTH			0	DISP1_RW_CONFIG		DISP1_PREAMBLE_EN
	W																
0x53FC_0130 (DI_SER_DISP2_CONF)	R	0	0	0	0	0	0	0	DISP2_SER_BURST_MODE	0	0	0	DISP2_SER_BIT_NUM				
	W																
	R	DISP2_PREAMBLE								0	DISP2_PREAMBLE_LENGTH			0	DISP2_RW_CONFIG		DISP2_PREAMBLE_EN
	W																
0x53FC_0134 (DI_HSP_CLK_PER)	R	0	0	0	0	0	0	0	0	0	HSP_CLK_PERIOD_2						
	W																
	R	0	0	0	0	0	0	0	0	0	HSP_CLK_PERIOD_1						
	W																

Table 44-16. IPU Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x53FC_0138 (DI_DISP0_TIME_CONF_1)	R	DISP0_IF_CLK_DOWN_WR										DISP0_IF_CLK_UP_WR[9:4]					
	W																
	R	DISP0_IF_CLK_UP_WR[3:0]				DISP0_IF_CLK_PER_WR											
	W																
0x53FC_013C (DI_DISP0_TIME_CONF_2)	R	DISP0_IF_CLK_DOWN_RD										DISP0_IF_CLK_UP_RD[9:4]					
	W																
	R	DISP0_IF_CLK_UP_RD[3:0]				DISP0_IF_CLK_PER_RD											
	W																
0x53FC_0140 (DI_DISP0_TIME_CONF_3)	R	0	0	DISP0_RD_WAIT_ST				0	0	DISP0_READ_EN							
	W																
	R	0	0	0	0	DISP0_PIX_CLK_PER											
	W																

Table 44-16. IPU Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x53FC_0144 (DI_DISP1_TIME_CONF_1)	R	DISP1_IF_CLK_DOWN_WR										DISP1_IF_CLK_UP_WR[9:4]					
	W																
	R	DISP1_IF_CLK_UP_WR[3:0]				DISP1_IF_CLK_PER_WR											
	W																
0x53FC_0148 (DI_DISP1_TIME_CONF_2)	R	DISP1_IF_CLK_DOWN_RD										DISP1_IF_CLK_UP_RD[9:4]					
	W																
	R	DISP1_IF_CLK_UP_RD[3:0]				DISP1_IF_CLK_PER_RD											
	W																
0x53FC_014C (DI_DISP1_TIME_CONF_3)	R	0	0	DISP1_RD_WAIT_ST				0	0	DISP1_READ_EN							
	W																
	R	0	0	0	0	DISP12_PIX_CLK_PER											
	W																

Table 44-16. IPU Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x53FC_0150 (DI_DISP2_TIME_CONF_1)	R	DISP2_IF_CLK_DOWN_WR										DISP2_IF_CLK_UP_WR[9:4]					
	W																
	R	DISP2_IF_CLK_UP_WR[3:0]				DISP2_IF_CLK_PER_WR											
	W																
0x53FC_0154 (DI_DISP2_TIME_CONF_2)	R	DISP2_IF_CLK_DOWN_RD										DISP2_IF_CLK_UP_RD[9:4]					
	W																
	R	DISP2_IF_CLK_UP_RD[3:0]				DISP2_IF_CLK_PER_RD											
	W																
0x53FC_0158 (DI_DISP2_TIME_CONF_3)	R	0	0	DISP2_RD_WAIT_ST		0	0	DISP2_READ_EN									
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																

Table 44-16. IPU Register Summary (continued)

Name		31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0x53FC_015C (DI_DISP3_TIME_CONF)	R	DISP3_IF_CLK_DOWN_WR; DISP3_IF_CLK_UP_WR[9:4]
	W	
	R	DISP3_IF_CLK_UP_WR[3:0]; DISP3_IF_CLK_PER_WR
	W	
0x53FC_0160 (DI_DISP0_DB0_MAP)	R	0; MD00_OFFS2; MD00_OFFS1; MD00_OFFS0
	W	
	R	MD00_M7; MD00_M6; MD00_M5; MD00_M4; MD00_M3; MD00_M2; MD00_M1; MD00_M0
	W	
0x53FC_0164 (DI_DISP0_DB1_MAP)	R	0; MD01_OFFS2; MD01_OFFS1; MD01_OFFS0
	W	
	R	MD01_M7; MD01_M6; MD01_M5; MD01_M4; MD01_M3; MD01_M2; MD01_M1; MD01_M0
	W	
0x53FC_0168 (DI_DISP0_DB2_MAP)	R	0; MD02_OFFS2; MD02_OFFS1; MD02_OFFS0
	W	
	R	MD02_M7; MD02_M6; MD02_M5; MD02_M4; MD02_M3; MD02_M2; MD02_M1; MD02_M0
	W	
0x53FC_016C (DI_DISP0_CB0_MAP)	R	0; MC00_OFFS2; MC00_OFFS1; MC00_OFFS0
	W	
	R	MC00_M7; MC00_M6; MC00_M5; MC00_M4; MC00_M3; MC00_M2; MC00_M1; MC00_M0
	W	
0x53FC_0170 (DI_DISP0_CB1_MAP)	R	0; MC01_OFFS2; MC01_OFFS1; MC01_OFFS0
	W	
	R	MC01_M7; MC01_M6; MC01_M5; MC01_M4; MC01_M3; MC01_M2; MC01_M1; MC01_M0
	W	

Table 44-16. IPU Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x53FC_0174 (DI_DISP0_CB2_MAP)	R	0	MC02_OFFS2					MC02_OFFS1					MC02_OFFS0				
	W																
	R	MC02_M7		MC02_M6		MC02_M5		MC02_M4		MC02_M3		MC02_M2		MC02_M1		MC02_M0	
	W																
0x53FC_0178 (DI_DISP1_DB0_MAP)	R	0	MD10_OFFS2					MD10_OFFS1					MD10_OFFS0				
	W																
	R	MD10_M7		MD10_M6		MD10_M5		MD10_M4		MD10_M3		MD10_M2		MD10_M1		MD10_M0	
	W																
0x53FC_017C (DI_DISP1_DB1_MAP)	R	0	MD11_OFFS2					MD11_OFFS1					MD11_OFFS0				
	W																
	R	MD11_M7		MD11_M6		MD11_M5		MD11_M4		MD11_M3		MD11_M2		MD11_M1		MD11_M0	
	W																
0x53FC_0180 (DI_DISP1_DB2_MAP)	R	0	MD12_OFFS2					MD12_OFFS1					MD12_OFFS0				
	W																
	R	MD12_M7		MD12_M6		MD12_M5		MD12_M4		MD12_M3		MD12_M2		MD12_M1		MD12_M0	
	W																
0x53FC_0184 (DI_DISP1_CB0_MAP)	R	0	MC10_OFFS2					MC10_OFFS1					MC10_OFFS0				
	W																
	R	MC10_M7		MC10_M6		MC10_M5		MC10_M4		MC10_M3		MC10_M2		MC10_M1		M0MC10_	
	W																
0x53FC_0188 (DI_DISP1_CB1_MAP)	R	0	MC11_OFFS2					MC11_OFFS1					MC11_OFFS0				
	W																
	R	MC11_M7		MC11_M6		MC11_M5		MC11_M4		MC11_M3		MC11_M2		MC11_M1		M0MC11_	
	W																
0x53FC_018C (DI_DISP1_CB2_MAP)	R	0	MC12_OFFS2					MC12_OFFS1					MC12_OFFS0				
	W																
	R	MC12_M7		MC12_M6		MC12_M5		MC12_M4		MC12_M3		MC12_M2		MC12_M1		MC12_M0	
	W																
0x53FC_0190 (DI_DISP2_DB0_MAP)	R	0	MD20_OFFS2					MD20_OFFS1					MD20_OFFS0				
	W																
	R	MD20_M7		MD20_M6		MD20_M5		MD20_M4		MD20_M3		MD20_M2		MD20_M1		MD20_M0	
	W																

Table 44-16. IPU Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x53FC_0194 (DI_DISP2_DB1_MAP)	R	0	MD21_OFFS2					MD21_OFFS1					MD21_OFFS0				
	W																
	R	MD21_M7		MD21_M6		MD21_M5		MD21_M4		MD21_M3		MD21_M2		MD21_M1		MD21_M0	
	W																
0x53FC_0198 (DI_DISP2_DB2_MAP)	R	0	MD22_OFFS2					MD22_OFFS1					MD22_OFFS0				
	W																
	R	MD22_M7		MD22_M6		MD22_M5		MD22_M4		MD22_M3		MD22_M2		MD22_M1		MD22_M0	
	W																
0x53FC_019C (DI_DISP2_CB0_MAP)	R	0	MC20_OFFS2					MC20_OFFS1					MC20_OFFS0				
	W																
	R	MC20_M7		MC20_M6		MC20_M5		MC20_M4		MC20_M3		MC20_M2		MC20_M1		MC20_M0	
	W																
0x53FC_01A0 (DI_DISP2_CB1_MAP)	R	0	MC21_OFFS2					MC21_OFFS1					MC21_OFFS0				
	W																
	R	MC21_M7		MC21_M6		MC21_M5		MC21_M4		MC21_M3		MC21_M2		MC21_M1		MC21_M0	
	W																
0x53FC_01A4 (DI_DISP2_CB2_MAP)	R	0	MC22_OFFS2					MC22_OFFS1					MC22_OFFS0				
	W																
	R	MC22_M7		MC22_M6		MC22_M5		MC22_M4		MC22_M3		MC22_M2		MC22_M1		MC22_M0	
	W																
0x53FC_01A8 (DI_DISP3_B0_MAP)	R	0	M30_OFFS2					M30_OFFS1					M30_OFFS0				
	W																
	R	M30_M7		M30_M6		M30_M5		M30_M4		M30_M3		M30_M2		M30_M1		M30_M0	
	W																
0x53FC_01AC (DI_DISP3_B1_MAP)	R	0	M31_OFFS2					M31_OFFS1					M31_OFFS0				
	W																
	R	M31_M7		M31_M6		M31_M5		M31_M4		M31_M3		M31_M2		M31_M1		M31_M0	
	W																
0x53FC_01B0 (DI_DISP3_B2_MAP)	R	0	M32_OFFS2					M32_OFFS1					M32_OFFS0				
	W																
	R	M32_M7		M32_M6		M32_M5		M32_M4		M32_M3		M32_M2		M32_M1		M32_M0	
	W																

Table 44-16. IPU Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x53FC_01B4 (DI_DISP_ACC_CC)	R	0	0	DISP3_IF_CLK_CNT_D		DISP2_IF_CLK_CNT_C		DISP2_IF_CLK_CNT_D		DISP1_IF_CLK_CNT_C		DISP1_IF_CLK_CNT_D		DISP0_IF_CLK_CNT_C		DISP0_IF_CLK_CNT_D	
	W																
0x53FC_01B8 (DI_DISP_LLA_CONF)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	DRCT_BE_MODE	DRCT_MAP_DC	DRCT_LOCK	DRCT_DISP_NUM		DRCT_RS
	W																
0x53FC_01BC (DI_DISP_LLA_DATA)	R	0	0	0	0	0	0	0	0	LLA_DATA[23:16]							
	W																
	R	LLA_DATA[15:0]															
	W																

44.3.3 Register Descriptions

44.3.3.1 IPU Common Registers

44.3.3.1.1 IPU Configuration Register (IPU_CONF)

This register contains general configuration parameters of IPU. It controls all units' clocks and enables.

0x53FC_0000 (IPU_CONF) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	PXL_ENDIAN	DU_EN	DI_EN	ADC_EN	SDC_EN	PF_EN	ROT_EN	IC_EN	CSI_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-8. IPU Configuration Register (IPU_CONF)

Table 44-17. IPU_CONF Field Descriptions

Field	Description
31–9	Reserved
8 PXL_ENDIAN	Pixel Endianness. This bit defines the pixel Endianness. 0 Little endianness 1 Big Endianness
7 DU_EN	Debug Unit enable. This bit enables the Debug Unit. 0 DU is disabled. 1 DU is enabled.
6 DI_EN	Display Interface enable. This bit enables the Display Interface. 0 DI is disabled. 1 DI is enabled.
5 ADC_EN	Asynchronous Display Controller enable. This bit enables the Asynchronous Display Controller. 0 ADC is disabled. 1 ADC is enabled.
4 SDC_EN	Synchronous Display Controller enable. This bit enables the Synchronous Display Controller. 0 SDC is disabled. 1 SDC is enabled.
3 PF_EN	Postfilter enable. This bit enables the Postfilter. 0 PF is disabled. 1 PF is enabled.
2 ROT_EN	Rotation Unit enable. This bit enables the Rotation Unit. 0 ROT is disabled. 1 ROT is enabled.

Table 44-17. IPU_CONF Field Descriptions (continued)

Field	Description
1 IC_EN	Image Converter enable. This bit enables the Image Converter. 0 IC is disabled. 1 IC is enabled.
0 CSI_EN	Camera Sensor interface enable. This enables the CSI unit. 0 CSI is disabled. 1 CSI is enabled.

44.3.3.1.2 IPU Channels Buffer 0 Ready Register (IPU_CHA_BUF0_RDY)

The register contains buffer 0 ready control information for 32 IPU's DMA channels. Writing "1" to each field will set each bit. Writing "0" to each field simultaneously will clear all the bits.

0x53FC_0004 (IPU_CHA_BUF0_RDY) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DMAPF_7_BUF0_RDY	DMAPF_6_BUF0_RDY	DMAPF_5_BUF0_RDY	DMAPF_4_BUF0_RDY	DMAPF_3_BUF0_RDY	DMAPF_2_BUF0_RDY	DMAPF_1_BUF0_RDY	DMAPF_0_BUF0_RDY	DMAADC_7_BUF0_RDY	DMAADC_6_BUF0_RDY	DMAADC_5_BUF0_RDY	DMAADC_4_BUF0_RDY	DMAADC_3_BUF0_RDY	DMAADC_2_BUF0_RDY	DMASDC_3_BUF0_RDY	DMASDC_2_BUF0_RDY
W	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DMASDC_1_BUF0_RDY	DMASDC_0_BUF0_RDY	DMAIC_13_BUF0_RDY	DMAIC_12_BUF0_RDY	DMAIC_11_BUF0_RDY	DMAIC_10_BUF0_RDY	DMAIC_9_BUF0_RDY	DMAIC_8_BUF0_RDY	DMAIC_7_BUF0_RDY	DMAIC_6_BUF0_RDY	DMAIC_5_BUF0_RDY	DMAIC_4_BUF0_RDY	DMAIC_3_BUF0_RDY	DMAIC_2_BUF0_RDY	DMAIC_1_BUF0_RDY	DMAIC_0_BUF0_RDY
W	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-9. IPU Channels Buffer 0 Ready Register (IPU_CHA_BUF0_RDY)

Table 44-18. IPU_CHA_BUF0_RDY

Field	Description
31–0 CHA#_BUF0_RDY[1]	Buffer 0 is ready. This bit indicates that MCU finished/writing reading buffer 0 in memory. 0 Buffer 0 is not ready. 1 Buffer 0 is ready.

[1] CHA# is the corresponding DMA channel indicated in Figure 44-9.

44.3.3.1.3 IPU Channels Buffer 1 Ready Register (IPU_CHA_BUF1_RDY)

The register contains buffer 1 ready control information for 32 IPU's DMA channels. Writing “1” to each field will set each bit. Writing “0” to each field simultaneously will clear all the bits.

0x53FC_0008 (IPU_CHA_BUF1_RDY) — Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DMAPF_7_BUF1_RDY	DMAPF_6_BUF1_RDY	DMAPF_5_BUF1_RDY	DMAPF_4_BUF1_RDY	DMAPF_3_BUF1_RDY	DMAPF_2_BUF1_RDY	DMAPF_1_BUF1_RDY	DMAPF_0_BUF1_RDY	DMAADC_7_BUF1_RDY	DMAADC_6_BUF1_RDY	DMAADC_5_BUF1_RDY	DMAADC_4_BUF1_RDY	DMAADC_3_BUF1_RDY	DMAADC_2_BUF1_RDY	DMASDC_3_BUF1_RDY	DMASDC_2_BUF1_RDY
W	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DMASDC_1_BUF1_RDY	DMASDC_0_BUF1_RDY	DMAIC_13_BUF1_RDY	DMAIC_12_BUF1_RDY	DMAIC_11_BUF1_RDY	DMAIC_10_BUF1_RDY	DMAIC_9_BUF1_RDY	DMAIC_8_BUF1_RDY	DMAIC_7_BUF1_RDY	DMAIC_6_BUF1_RDY	DMAIC_5_BUF1_RDY	DMAIC_4_BUF1_RDY	DMAIC_3_BUF1_RDY	DMAIC_2_BUF1_RDY	DMAIC_1_BUF1_RDY	DMAIC_0_BUF1_RDY
W	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s	w1s
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-10. IPU Channels Buffer 1 Ready Register (IPU_CHA_BUF1_RDY)

Table 44-19. IPU_CHA_BUF1_RDY Field Descriptions

Field	Description
31–0[1] CHA#_BUF1_RDY	Buffer 1 is ready. This bit indicates that MCU finished/writing reading buffer 1 in memory. 0 Buffer 1 is not ready. 1 Buffer 1 is ready.

1 CHA# is the corresponding DMA channel indicated in Figure 44-10.

44.3.3.1.4 IPU Channel Double Buffer Mode Select Register (IPU_CHA_DB_MODE_SEL)

The register contains double buffer mode select control information for 32 IPU's DMA channels.

0x53FC_000C (IPU_CHA_DB_MODE_SEL) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	DMAPF_5_DBMS	0	0	DMAPF_2_DBMS	0	0	0	0	0	0	DMAADC_3_DBMS	DMAADC_2_DBMS	0	DMASDC_2_DBMS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DMASDC_1_DBMS	DMASDC_0_DBMS	DMAIC_13_DBMS	DMAIC_12_DBMS	DMAIC_11_DBMS	DMAIC_10_DBMS	DMAIC_9_DBMS	DMAIC_8_DBMS	DMAIC_7_DBMS	DMAIC_6_DBMS	DMAIC_5_DBMS	DMAIC_4_DBMS	DMAIC_3_DBMS	DMAIC_2_DBMS	DMAIC_1_DBMS	DMAIC_0_DBMS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-11. IPU Channel Double Buffer Mode Select Register (IPU_CHA_DB_MODE_SEL)

Table 44-20. IPU_CHA_DB_MODE_SEL Field Descriptions

Field	Description
31–0 CHA#_DB_MODE_SEL[1]	Double buffer mode select. This bit indicates if a double buffer is used for this channel. 0 Double buffer is not used for this channel. 1 Double buffer is used for this channel.

1 CHA# is the corresponding DMA channel indicated in Figure 44-11.

44.3.3.1.5 IPU Channel Current Buffer Register (IPU_CHA_CUR_BUF)

The register contains current buffer status information for 32 IPU's DMA channels.

0x53FC_0010 (IPU_CHA_CUR_BUF) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DMAPF_7_CUR_BUF	DMAPF_6_CUR_BUF	DMAPF_5_CUR_BUF	DMAPF_4_CUR_BUF	DMAPF_3_CUR_BUF	DMAPF_2_CUR_BUF	DMAPF_1_CUR_BUF	DMAPF_0_CUR_BUF	DMAADC_7_CUR_BUF	DMAADC_6_CUR_BUF	DMAADC_5_CUR_BUF	DMAADC_4_CUR_BUF	DMAADC_3_CUR_BUF	DMAADC_2_CUR_BUF	DMASDC_3_CUR_BUF	DMASDC_2_CUR_BUF
W	w1r	w1r	w1r	w1r	w1r	w1r	w1r	w1r	w1r	w1r	w1r	w1r	w1r	w1r	w1r	w1r
Reset	0	0	[DMAPF_5_DBMS]	0	0	[DMAPF_2_DBMS]	0	0	0	0	0	0	[DMAADC_3_DBMS]	[DMAADC_2_DBMS]	0	[DMASDC_2_DBMS]

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DMASDC_1_CUR_BUF	DMASDC_0_CUR_BUF	DMAIC_13_CUR_BUF	DMAIC_12_CUR_BUF	DMAIC_11_CUR_BUF	DMAIC_10_CUR_BUF	DMAIC_9_CUR_BUF	DMAIC_8_CUR_BUF	DMAIC_7_CUR_BUF	DMAIC_6_CUR_BUF	DMAIC_5_CUR_BUF	DMAIC_4_CUR_BUF	DMAIC_3_CUR_BUF	DMAIC_2_CUR_BUF	DMAIC_1_CUR_BUF	DMAIC_0_CUR_BUF
W	w1r	w1r	w1r	w1r	w1r	w1r	w1r	w1r	w1r	w1r	w1r	w1r	w1r	w1r	w1r	w1r
Reset	[DMASDC_1_DBMS]	[DMASDC_0_DBMS]	[DMAIC_13_DBMS]	[DMAIC_12_DBMS]	[DMAIC_11_DBMS]	[DMAIC_10_DBMS]	[DMAIC_9_DBMS]	[DMAIC_8_DBMS]	[DMAIC_7_DBMS]	[DMAIC_6_DBMS]	[DMAIC_5_DBMS]	[DMAIC_4_DBMS]	[DMAIC_3_DBMS]	[DMAIC_2_DBMS]	[DMAIC_1_DBMS]	[DMAIC_0_DBMS]

Figure 44-12. IPU Channel Current Buffer Register (IPU_CHA_CUR_BUF)

Table 44-21. IPU_CHA_CUR_BUF Field Descriptions

Field	Description
31–0 CHA#_CUR_BUF[1]	Current buffer. This bit configures which buffer is in use by DMA when double buffer mode is selected. 0 Current buffer used by DMA is buffer 0. 1 Current buffer used by DMA is buffer 1.

[1] CHA# is the corresponding DMA channel indicated in Figure 44-12.

44.3.3.1.6 IPU Frame Synchronization Processing Flow Register (IPU_FS_PROC_FLOW)

The register contains IPU tasks status information.

0x53FC_0014 (IPU_FS_PROC_FLOW) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	PP_ROT_DEST_SEL			0	PP_DEST_SEL			0	PRPVF_ROT_DEST_SEL			0	PRPVF_DEST_SEL		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	PF_DEST_SEL		PP_ROT_SRC_SEL		PP_SRC_SEL		0	PRPVF_ROT_SRC_SEL	PRPENC_ROT_SRC_SEL	PRPENC_DEST_SEL	0	0	VF_IN_VALID	ENC_IN_VALID
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-13. IPU Frame Synchronization Processing Flow Register (IPU_FS_PROC_FLOW)

Table 44-22. IPU_FS_PROC_FLOW Field Descriptions

Field	Description
31	Reserved
30–28 PP_ROT_DEST_SEL	Rotation for post-processing destination select. This field select the destination of rotation for post-processing. Values: 000 MCU 001 Post-processing 010 ADC system channel 1 011 ADC system channel 2 100 SDC background channel 101 SDC foreground channel 110 RSV 111 RSV
27	Reserved
26–24 PP_DEST_SEL	Post-processing destination select. This field select the destination of Post-processing. Values: 000 MCU 001 Rotation for post-processing 010 ADC system channel 1 011 ADC system channel 2 100 SDC background channel 101 SDC foreground channel 110 ADC direct PP channel 111 RSV
23	Reserved
22–20 PRPVF_ROT_DEST_SEL	Rotation for view-finder destination select. This field select the destination of rotation for view-finder. Values: 000 MCU 001 RSV 010 ADC system channel 1 011 ADC system channel 2 100 SDC background channel 101 SDC foreground channel 110 RSV 111 RSV
19	Reserved
18–16 PRPVF_DEST_SEL	Pre-processing for view-finder destination select. This field select the destination of pre-processing for view-finder. Values: 000 MCU 001 Rotation for view-finder 010 ADC system channel 1 011 ADC system channel 2 100 SDC background channel 101 SDC foreground channel 110 ADC direct VF channel 111 RSV

Table 44-22. IPU_FS_PROC_FLOW Field Descriptions (continued)

Field	Description
15–14	Reserved
13–12 PF_DEST_SEL	Post-filtering destination select. This field select the destination of post-filtering task. Values: 00 MCU 01 Post-Processing 10 Rotation for post-processing 11 RSV
11–10 PP_ROT_SRC_SEL	Rotation for post processing source select. This field select the source of rotation for post-processing task. Values: 00 MCU 01 Post-processing 10 Post-filtering 11 RSV
9–8 PP_SRC_SEL	Post-processing source select. This field selects the source of the Post-processing task. Values: 00 MCU 01 Post-filtering 10 Rotation for post-processing 11 RSV
7	Reserved
6 PRPVF_ROT_SRC_SEL	Rotation for view-finder source select. This bit select the source of rotation for view-finder task. 0 View-finder rotation source is controlled by MCU. 1 View-finder rotation source is controlled by view-finder.
5 PRPENC_ROT_SRC_SEL	Rotation for encoding source select. This bit select the source of rotation for encoding task. 0 Encoding Rotation source is controlled by MCU. 1 Encoding rotation source is controlled by encoding.
4 PRPENC_DEST_SEL	Pre-processing for encoding destination select. This bit select the destination of encoding task. 0 Encoding destination is controlled by MCU. 1 Encoding destination is controlled by encoding rotation.
3–2	Reserved
1 VF_IN_VALID	View-finder input valid. Setting this bit indicates that the buffer in memory for viewfinder is validated by the MCU (valid only when RWS_EN is ‘1’). 0 View-finder should skip buffer in memory. 1 View-finder should use buffer in memory.
0 ENC_IN_VALID	Encoding input valid. Setting this bit indicates that the buffer in memory for encoding is validated by the MCU (valid only when RWS_EN is ‘1’). 0 Encoding should skip buffer in memory. 1 Encoding should use buffer in memory.

44.3.3.1.7 IPU Frame Synchronization Displaying Flow Register (IPU_FS_DISP_FLOW)

0x53FC_0018 (IPU_FS_DISP_FLOW) Access: User Read/Write

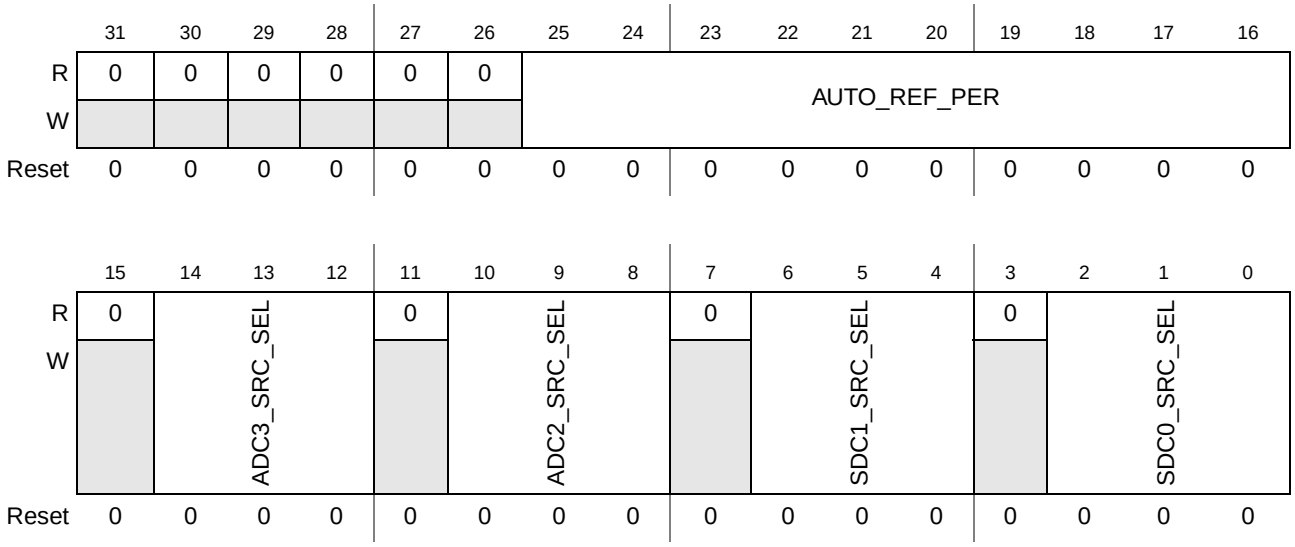


Figure 44-14. IPU Frame Synchronization Displaying Flow Register (IPU_FS_DISP_FLOW)

Table 44-23. IPU_FS_DISP_FLOW Field Descriptions

Field	Description
31–26	Reserved
25–16 AUTO_REF_PER	Autorefresh period minus 1. The actual value of autorefresh period is equal to the high speed clock (HSP_CLK) period multiplied by 217*(AUTO_REF_PER+1).
15	Reserved
14–12 ADC3_SRC_SEL	ADC system channel 2 source select. This field select the source of ADC system channel 2. Values: 000 MCU 001 Rotation for View-Finder 010 Rotation for Post-processing 011 View-Finder 100 Post-processing 101 Snooping for channel 2 110 Autorefresh 111 Autorefresh with snooping for channel 2
11	Reserved

Table 44-23. IPU_FS_DISP_FLOW Field Descriptions (continued)

Field	Description
10–8 ADC2_SRC_SEL	ADC system channel 1 source select. This field select the source of ADC system channel 1. Values: 000 MCU 001 Rotation for View-Finder 010 Rotation for Post-processing 011 View-Finder 100 Post-processing 101 Snooping for channel 1 110 Autorefresh 111 Autorefresh with snooping for channel 1
7	Reserved
6–4 SDC1_SRC_SEL	SDC foreground channel (1) source select. This field select the source of SDC foreground channel (1). Values: 000 MCU 001 Rotation for View-Finder 010 Rotation for Post-processing 011 View-Finder 100 Post-processing 101 RSV 110 RSV 111 RSV
3	Reserved
2–0 SDC0_SRC_SEL	SDC background channel (0) source select. This field select the source of SDC background channel (0). Values: 000 MCU 001 Rotation for View-Finder 010 Rotation for Post-processing 011 View-Finder 100 Post-processing 101 RSV 110 RSV 111 RSV

44.3.3.1.8 IPU Tasks Status Register (IPU_TASKS_STAT)

This register contains IPU tasks' status information.

0x53FC_001C (IPU_TASKS_STAT) Access: User Read-Only

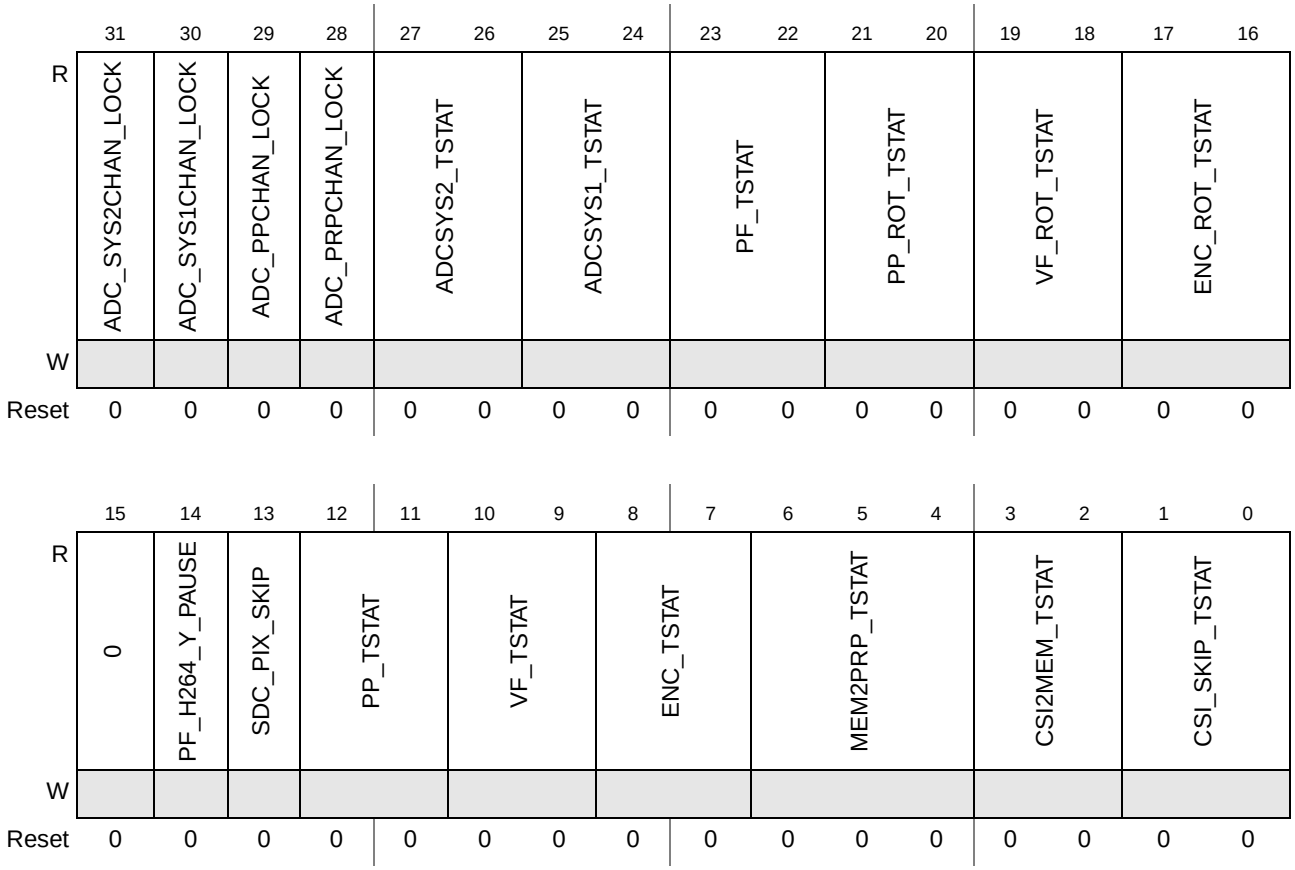


Figure 44-15. IPU Tasks Status Register (IPU_TASKS_STAT)

Table 44-24. IPU_TASKS_STAT Field Descriptions

Field	Description
31 ADC_SYS2CHAN_LOCK	ADC SYS2 Channel Locked. This bit is the Status of ADC SYS2 Channel. 0 Channel is unlocked (not busy). 1 Channel is locked (busy).
30 ADC_SYS1CHAN_LOCK	ADC SYS1 Channel Locked. This bit is the Status of ADC SYS1 Channel. 0 Channel is unlocked (not busy). 1 Channel is locked (busy).
29 ADC_PPCHAN_LOCK	ADC Post-Processing Channel Locked. This bit is the Status of ADC Post-Processing Channel. 0 Channel is unlocked (not busy). 1 Channel is locked (busy).
28 ADC_PRPCHAN_LOCK	ADC Pre-Processing Channel Locked. This bit is the Status of ADC Preprocessing Channel. 0 Channel is unlocked (not busy). 1 Channel is locked (busy).

Table 44-24. IPU_TASKS_STAT Field Descriptions (continued)

Field	Description
27–26 ADCSYS2_TSTAT	ADC system channel 2 task status. These bits indicate ADC system channel 2 task status. Values: 00 IDLE 01 ACTIVE 10 W4RDY 11 RSV
25–24 ADCSYS1_TSTAT	ADC system channel 1 task status. These bits indicate ADC system channel 1 task status. Values: 00 IDLE 01 ACTIVE 10 W4RDY 11 RSV
23–22 PF_TSTAT	Post-flittering task status. These bits indicate post-flittering task status Values: 00 IDLE 01 ACTIVE 10 W4RDY 11 RSV
21–20 PP_ROT_TSTAT	Rotation for post-processing task status. These bits indicate rotation for post-processing task status. Values: 00 IDLE 01 ACTIVE 10 W4RDY 11 RSV
19–18 VF_ROT_TSTAT	Rotation for View-Finder task status. These bits indicate Rotation for View-Finder task status. Values: 00 IDLE 01 ACTIVE 10 W4RDY 11 RSV
17–16 ENC_ROT_TSTAT	Rotation for encoding task status. These bits indicate rotation for encoding task status. Values: 00 IDLE 01 ACTIVE 10 W4RDY 11 RSV
15	Reserved
14 PF_H264_Y_PAUSE	Status bit indicates pause in the PF operation in H.264 mode.
13 SDC_PIX_SKIP	Status bit indicates skipping pixels on synchronous display after overrun error in the SDC.

Table 44-24. IPU_TASKS_STAT Field Descriptions (continued)

Field	Description
12–11 PP_TSTAT	Post-processing task status. These bits indicate Post-processing task status. Values: 00 IDLE 01 ACTIVE 10 W4RDY 11 RSV
10–9 VF_TSTAT	Pre-processing for view-finder task status. These bits indicate this task status. Values: 00 IDLE 01 ACTIVE 10 PAUSE 11 RSV
8–7 ENC_TSTAT	Encoding directly from CSI task status. These bits indicate the direct transferring from CSI to encoding task status. Values: 00 IDLE 01 ACTIVE 10 PAUSE 11 RSV
6–4 MEM2PRP_TSTAT	Memory to PRP task status. These bits indicate the indirect transferring for Pre-processing from memory tasks status. Values: 000 IDLE 001 BOTH_ACTIVE 010 ENC_ACTIVE 011 VF_ACTIVE 100 BOTH_PAUSE 101 RSV 110 RSV 111 RSV
3–2 CSI2MEM_TSTAT	CSI directly to memory task status. These bits indicate the transferring from CSI direct to memory task status. Values: 00 IDLE 01 ACTIVE 10 PAUSE 11 RSV
1–0 CSI_SKIP_TSTAT	Next VF and ENC frames skipping status in the CSI. Values: 00 Next ENC frame and next VF frame would be skipped. 01 Next ENC frame would be skipped, and next VF frame would be valid. 10 Next VF frame would be skipped, and next ENC frame would be valid. 11 Next ENC frame and next VF frame would be valid.

44.3.3.1.9 IPU Internal Memory Access Address and Data Registers (IPU_IMA_ADDR and IPU_IMA_DATA)

The IPU_IMA_ADDR register contains an address of 32-word to be written to or read from one of internal IPU memories.

0x53FC_0020 (IPU_IMA_ADDR) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	MEM_NU			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ROW_NU													WORD_NU		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-16. IPU Internal Memory Access Address Register (IPU_IMA_ADDR)

Table 44-25. IPU_IMA_ADDR Field Descriptions

Field	Description
31–20	Reserved
19–16 MEM_NU	Memory number. This field selects the accessed memory.
15–3 ROW_NU	Row number. This field selects a row in the accessed memory. The row number begins from 0.
2–0 WORD_NU	Word number. This field selects a 32-bit word inside a row in the accessed memory. The word number begins from 0.

Table 44-26 shows the MEM_NU, ROW_NU, and WORD_NU values.

Table 44-26. MEM_NU, ROW_NU, and WORD_NU Values

MEM_NU	Memory Name	Submodule	Maximum ROW_NU	Row Width/ Maximum WORD_NU
0000	Task Parameter Memory	IC	2239	48 (1)
0001	Channel Parameter Memory	IDMAC	63	132 (4)
0010	Look-Up Table Memory	IDMAC	255	32 (0)
0011	Template Memory	ADC	383	30 (0)
0100	Asynchronous Display Buffer Memory	ADC	223	64 (1)
0101	Downsizing Output Memory	IC	2175	48 (1)

Table 44-26. MEM_NU, ROW_NU, and WORD_NU Values (continued)

MEM_NU	Memory Name	Submodule	Maximum ROW_NU	Row Width/ Maximum WORD_NU
0110	Down Sizing Temporary Memory	IC	2175	36 (1)
0111	Input Buffer Memory	IC	95	64 (1)
1000	Main Processing Memory	IC	159	64 (1)
1001	Rotation Memory	IC	47	64 (1)
1010	Post Filter Memory	PF	271	64 (1)
1011	Synchronous Display Buffer Memory	SDC	127	64 (1)

When the MCU performs sequential write to or read from the IPU_IMA_DATA register, the WORD_NU and ROW_NU values increment automatically according to the selected memory size. Firstly WORD_NU sequentially increments. When it arrives end of memory row, it is cleared and ROW_NU increments by 1. After that WORD_NU increments again. So the memory is accessed via the IPU_IMA_DATA register like a FIFO. For write after read operation, the WORD_NU and ROW_NU values are not changed.

The IPU_IMA_DATA register contains a 32-bit data word written to or read from the selected memory according to the address defined in the IPU_IMA_ADDR register.

0x53FC_0024 (IPU_IMA_DATA) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	IMA_DATA[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IMA_DATA[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-17. IPU Internal Memory Access Data Register (IPU_IMA_DATA)

Table 44-27. IPU_IMA_DATA Field Description

Field	Description
31–16 IMA_DATA[31:16]	Contains a 16-bit data word written to or read from the selected memory according to the address defined in the IPU_IMA_ADDR register.
15–0 IMA_DATA[15:0]	

The IPU_IMA_ADDRESS and IPU_IMA_DATA registers are used for access to IPU internal memories during normal operation and in debug mode.

The IPU has several unmapped internal memories. Four of them should be configured by the MCU during setup before the IPU can properly operate. The four memories are:

- TPM—Task Parameter Memory which resides at IC sub module. Must be configured before enabling processing tasks. This memory can be accessed while tasks are enabled.
- CPM—Channel Parameter Memory which resides at IDMAC sub module. Must be configured before enabling processing tasks. This memory can be accessed while tasks are enabled.
- LUTM—Look-Up Table Memory which resides at IDMAC sub module. Can be configured before enabling tasks. Must not be accessed while operation.
- TM—ADC Template Memory which reside at ADC sub module. Can be configured before enabling tasks. Must not be accessed while operation.

These memories will be mapped internally and MCU will have to write the targeted address to IPU_IMA_ADDR and then perform write access to IPU_IMA_DATA. The Internal Memory Access unit which reside in the SBIST sub module of the Control Module, will translate the address which was written by MCU to IPU_IMA_ADDR and the write access to IPU_IMA_DATA into an internal memory access. At the end of the access IPU_IMA_ADDR increments for reducing IP accessed while configuring these memories sequentially.

The TPM, CPM, LUTM and TM memories can be accessed during IPU normal operation with the following restrictions:

- IC task parameters must not be changed in the TPM when the corresponding task is active.
- IDMAC channel parameters must not be changed in the CPM when the corresponding DMA channel is enabled excluding the base addresses (EBA0 and EBA1). One of these parameters can be changed during channel operation if it relates to the non-active double buffer.
- Look-up-table must not be changed in the LUTM when there is any enabled DMA channel which can use this table.
- Both read and write access to the TM is forbidden when there is any enabled ADC channel which can use templates.

The rest of memories can be accessed only in debug (freeze) mode. The tables in the following sections describe the four internal memory structures.

IC Task Parameter Memory

Table 44-28 presents IC task parameter memory details.

Table 44-28. IC Parameters

Address	Word	Parameter	Field	Description
x2d1	Encoding CSC1 matrix1 word1	C22	8:0	Coefficients of color conversion matrix1 for encoding task: $Z0 = 2^{SLCALE-1}*(X0*C00 + X1*C01 + X2*C02+A0);$ $Z1 = 2^{SLCALE-1}*(X0*C10 + X1*C11 + X2*C12+A1);$ $Z2 = 2^{SLCALE-1}*(X0*C20 + X1*C21 + X2*C22+A2);$ Coefficients format is $s.xxxxxxxx^{1}$;
		C11	17:9	
		C00	26:18	
		A0	39:27	Offset of color conversion matrix1 for encoding task: Offset format is sxxxxxxxxxx.xx
		SCALE	41:40	Scale of coefficients for color conversion matrix1 for encoding task:
		SAT_MODE	42:42	Saturation mode for color conversion matrix1 for encoding task: 0 --> (min, max) = (0, 255) 1 --> (min, max) = (16, 240) for Z1,Z2 1 --> (min, max) = (16, 235) for Z0
x2d2	Encoding CSC1 matrix1 word2	C20	8:0	Coefficients of color conversion matrix1 for encoding task: $Z0 = 2^{SLCALE-1}*(X0*C00 + X1*C01 + X2*C02+A0);$ $Z1 = 2^{SLCALE-1}*(X0*C10 + X1*C11 + X2*C12+A1);$ $Z2 = 2^{SLCALE-1}*(X0*C20 + X1*C21 + X2*C22+A2);$ Coefficients format is s.xxxxxxxx;
		C10	17:9	
		C01	26:18	
		A1	39:27	Offset of color conversion matrix1 for encoding task: Offset format is sxxxxxxxxxx.xx
x2d3	Encoding CSC1 matrix1 word3	C21	8:0	Coefficients of color conversion matrix1 for encoding task: $Z0 = 2^{SLCALE-1}*(X0*C00 + X1*C01 + X2*C02+A0);$ $Z1 = 2^{SLCALE-1}*(X0*C10 + X1*C11 + X2*C12+A1);$ $Z2 = 2^{SLCALE-1}*(X0*C20 + X1*C21 + X2*C22+A2);$ Coefficients format is s.xxxxxxxx;
		C12	17:9	
		C02	26:18	
		A2	39:27	Offset of color conversion matrix1 for encoding task: Offset format is sxxxxxxxxxx.xx

Table 44-28. IC Parameters (continued)

Address	Word	Parameter	Field	Description
x5a5	Viewfinder CSC1 matrix1 word1	C22	8:0	Coefficients of color conversion matrix1 for viewfinder task: Z0 = $2^{SLCALE-1}$*(X0*C00 + X1*C01 +X2*C02+A0); Z1 = $2^{SLCALE-1}$*(X0*C10 + X1*C11 +X2*C12+A1); Z2 = $2^{SLCALE-1}$*(X0*C20 + X1*C21 +X2*C22+A2); Coefficients format is s.xxxxxxxx;
		C11	17:9	
		C00	26:18	
		A0	39:27	Offset of color conversion matrix1 for viewfinder task: Offset format is sxxxxxxxxxx.xx
		SCALE	41:40	Scale of coefficients for color conversion matrix1 for viewfinder task
		SAT_MODE	42:42	Saturation mode for color conversion matrix1 for viewfinder task: 0 --> (min, max) = (0, 255) 1 --> (min, max) = (16, 240) for Z1,Z2 1 --> (min, max) = (16, 235) for Z0
x5a6	Viewfinder CSC1 matrix1 word2	C20	8:0	Coefficients of color conversion matrix1 for viewfinder task: Z0 = $2^{SLCALE-1}$*(X0*C00 + X1*C01 +X2*C02+A0); Z1 = $2^{SLCALE-1}$*(X0*C10 + X1*C11 +X2*C12+A1); Z2 = $2^{SLCALE-1}$*(X0*C20 + X1*C21 +X2*C22+A2); Coefficients format is s.xxxxxxxx;
		C10	17:9	
		C01	26:18	
		A1	39:27	Offset of color conversion matrix1 for viewfinder task: Offset format is sxxxxxxxxxx.xx
x5a7	Viewfinder CSC1 matrix1 word3	C21	8:0	Coefficients of color conversion matrix1 for viewfinder task: Z0 = $2^{SLCALE-1}$*(X0*C00 + X1*C01 +X2*C02+A0); Z1 = $2^{SLCALE-1}$*(X0*C10 + X1*C11 +X2*C12+A1); Z2 = $2^{SLCALE-1}$*(X0*C20 + X1*C21 +X2*C22+A2); Coefficients format is s.xxxxxxxx;
		C12	17:9	
		C02	26:18	
		A2	39:27	Offset of color conversion matrix1 for viewfinder task: Offset format is sxxxxxxxxxx.xx
x5a8	Viewfinder CSC2 matrix2 word1	C22	8:0	Coefficients of color conversion matrix2 for viewfinder task: Z0 = $2^{SLCALE-1}$*(X0*C00 + X1*C01 +X2*C02+A0); Z1 = $2^{SLCALE-1}$*(X0*C10 + X1*C11 +X2*C12+A1); Z2 = $2^{SLCALE-1}$*(X0*C20 + X1*C21 +X2*C22+A2); Coefficients format is s.xxxxxxxx;
		C11	17:9	
		C00	26:18	
		A0	39:27	Offset of color conversion matrix2 for viewfinder task: Offset format is sxxxxxxxxxx.xx
		SCALE	41:40	Scale of coefficients for color conversion matrix2 for viewfinder task.
		SAT_MODE	42:42	Saturation mode for color conversion matrix2 for viewfinder task: 0 --> (min, max) = (0, 255) 1 --> (min, max) = (16, 240) for Z1,Z2 1 --> (min, max) = (16, 235) for Z0

Table 44-28. IC Parameters (continued)

Address	Word	Parameter	Field	Description
x5a9	Viewfinder CSC2 matrix2 word2	C20	8:0	Coefficients of color conversion matrix2 for viewfinder task: $Z0 = 2^{SLCALE-1}*(X0*C00 + X1*C01 + X2*C02+A0);$ $Z1 = 2^{SLCALE-1}*(X0*C10 + X1*C11 + X2*C12+A1);$ $Z2 = 2^{SLCALE-1}*(X0*C20 + X1*C21 + X2*C22+A2);$ Coefficients format is s.xxxxxxxx;
		C10	17:9	
		C01	26:18	
		A1	39:27	Offset of color conversion matrix2 for viewfinder task: Offset format is sxxxxxxxxxx.xx
x5aa	Viewfinder CSC2 matrix2 word3	C21	8:0	Coefficients of color conversion matrix2 for viewfinder task: $Z0 = 2^{SLCALE-1}*(X0*C00 + X1*C01 + X2*C02+A0);$ $Z1 = 2^{SLCALE-1}*(X0*C10 + X1*C11 + X2*C12+A1);$ $Z2 = 2^{SLCALE-1}*(X0*C20 + X1*C21 + X2*C22+A2);$ Coefficients format is s.xxxxxxxx;
		C12	17:9	
		C02	26:18	
		A2	39:27	Offset of color conversion matrix2 for viewfinder task: Offset format is sxxxxxxxxxx.xx
x87c	Postprocessing CSC1 matrix1 word1	C22	8:0	Coefficients of color conversion matrix1 for postprocessing task: $Z0 = 2^{SLCALE-1}*(X0*C00 + X1*C01 + X2*C02+A0);$ $Z1 = 2^{SLCALE-1}*(X0*C10 + X1*C11 + X2*C12+A1);$ $Z2 = 2^{SLCALE-1}*(X0*C20 + X1*C21 + X2*C22+A2);$ Coefficients format is s.xxxxxxxx;
		C11	17:9	
		C00	26:18	
		A0	39:27	Offset of color conversion matrix1 for viewfinder task: Offset format is sxxxxxxxxxx.xx
		SCALE	41:40	Scale of coefficients for color conversion matrix1 for postprocessing task.
		SAT_MODE	42:42	Saturation mode for color conversion matrix1 for postprocessing task: 0 --> (min, max) = (0, 255) 1 --> (min, max) = (16, 240) for Z1,Z2 1 --> (min, max) = (16, 235) for Z0
x87d	Postprocessing CSC1 matrix1 word2	C20	8:0	Coefficients of color conversion matrix1 for postprocessing task: $Z0 = 2^{SLCALE-1}*(X0*C00 + X1*C01 + X2*C02+A0);$ $Z1 = 2^{SLCALE-1}*(X0*C10 + X1*C11 + X2*C12+A1);$ $Z2 = 2^{SLCALE-1}*(X0*C20 + X1*C21 + X2*C22+A2);$ Coefficients format is s.xxxxxxxx;
		C10	17:9	
		C01	26:18	
		A1	39:27	Offset of color conversion matrix1 for post-processing task: Offset format is sxxxxxxxxxx.xx

Table 44-28. IC Parameters (continued)

Address	Word	Parameter	Field	Description
x87e	Postprocessing CSC1 matrix1 word3	C21	8:0	Coefficients of color conversion matrix1 for postprocessing task: $Z0 = 2^{SLCALE-1}*(X0*C00 + X1*C01 + X2*C02+A0)$; $Z1 = 2^{SLCALE-1}*(X0*C10 + X1*C11 + X2*C12+A1)$; $Z2 = 2^{SLCALE-1}*(X0*C20 + X1*C21 + X2*C22+A2)$; Coefficients format is s.xxxxxxxx;
		C12	17:9	
		C02	26:18	
		A2	39:27	Offset of color conversion matrix1 for postprocessing task: Offset format is sxxxxxxxxxx.xx
x87f	Postprocessing CSC2 matrix2 word1	C22	8:0	Coefficients of color conversion matrix2 for postprocessing task: $Z0 = 2^{SLCALE-1}*(X0*C00 + X1*C01 + X2*C02+A0)$; $Z1 = 2^{SLCALE-1}*(X0*C10 + X1*C11 + X2*C12+A1)$; $Z2 = 2^{SLCALE-1}*(X0*C20 + X1*C21 + X2*C22+A2)$; Coefficients format is s.xxxxxxxx;
		C11	17:9	
		C00	26:18	
		A0	39:27	Offset of color conversion matrix2 for viewfinder task: Offset format is sxxxxxxxxxx.xx
		SCALE	41:40	Scale of coefficients for color conversion matrix2 for viewfinder task.
		SAT_MODE	42:42	Saturation mode for color conversion matrix2 for postprocessing task: 0 --> (min, max) = (0, 255) 1 --> (min, max) = (16, 240) for Z1,Z2 1 --> (min, max) = (16, 235) for Z0
x880	Postprocessing CSC2 matrix2 word2	C20	8:0	Coefficients of color conversion matrix2 for postprocessing task: $Z0 = 2^{SLCALE-1}*(X0*C00 + X1*C01 + X2*C02+A0)$; $Z1 = 2^{SLCALE-1}*(X0*C10 + X1*C11 + X2*C12+A1)$; $Z2 = 2^{SLCALE-1}*(X0*C20 + X1*C21 + X2*C22+A2)$; Coefficients format is s.xxxxxxxx;
		C10	17:9	
		C01	26:18	
		A1	39:27	Offset of color conversion matrix2 for postprocessing task: Offset format is sxxxxxxxxxx.xx
x881	Postprocessing CSC2 matrix2 word3	C21	8:0	Coefficients of color conversion matrix2 for postprocessing task: $Z0 = 2^{SLCALE-1}*(X0*C00 + X1*C01 + X2*C02+A0)$; $Z1 = 2^{SLCALE-1}*(X0*C10 + X1*C11 + X2*C12+A1)$; $Z2 = 2^{SLCALE-1}*(X0*C20 + X1*C21 + X2*C22+A2)$; Coefficients format is s.xxxxxxxx;
		C12	17:9	
		C02	26:18	
		A2	39:27	Offset of color conversion matrix2 for postprocessing task: Offset format is sxxxxxxxxxx.xx

[1] s - sign position, x - binary digit position

IDMAC Channel Parameter Memory

The following tables show, the IDMAC channel programing bits. There are a total of 32 IDMAC channels that can be configured into 2 modes: non-interleaved and interleaved. See Table 44-29 and Table 44-30 for non-interleaved configuration, variable and constant, respectively. See Table 44-31 and Table 44-32 for interleaved configuration, variable and constant, respectively. Each channel has 2 132 configuration bit words that are located in the channel parameter memory, thus for example channel N's parameter data will be divided into two 132-bit words at address 2*N and 2*N + 1. Configuration data in 2*N addressees are divided into two parts: variable and constant data. Variable data should be initialized to zero except for the NSB bit which should initialized to one. Constant data should be configured according to the users needs. The ratio between variable bits and constant bits in 2*N address words are different for non-interleaved and interleaved configuration. In 2*N + 1 address words, all data is constant.

Table 44-29. Variable Channel Parameters for RGB/YUV Non-Interleaved Configuration

Address	Word	Parameter	Field	Description
2*N	Word0	XV	W0[9:0]	Variable coordinates for determining next block address. {X1,Y1} and {X2,Y2} coordinates will be determined according to {XV,YV} upon restart of channel. These coordinates are used for RGB and Y:U:V (Y pointer) formats.
		YV	W0[19:10]	
		XB	W0[31:20]	Variable coordinates for determining address within the block. These coordinates are used for Y:U:V (Y pointer) and RGB formats. Need 20 bits for 1D transfer support.
		YB	W0[43:32]	
		-	W0[45:44]	Reserved.
		NSB	W0[46]	This bit determines if the next value for {xb,yb} should be taken from {xb,yb} saved in channel parameter memory or from new {x1,y1}/{x2,y2}.
		LNPB	W0[52:47]	Holds the value of the last burst size incase size of row is not aligned to burst size. These bits are updated by Addressing Arithmetic Unit one channel access before the end of row access and therefore saved at the end of that channel event.

Table 44-30. Constant Channel Parameters for RGB/YUV Non-Interleaved Configuration

Address	Word	Parameter	Field	Description
2*N	Word0	UBO	W0[78:53]	Double buffer destination address offset for Y:U:V (U pointer) formats.
		VBO	W0[104:79]	Double buffer destination address offset for Y:U:V (V pointer) format.
		-	W0[107:105]	Reserved.

Table 44-30. Constant Channel Parameters for RGB/YUV Non-Interleaved Configuration (continued)

Address	Word	Parameter	Field	Description
2*N	Word0	FW	W0[119:108]	Frame width minus 1 (in pixels). 000000000000 = 0001 pixels 000000000001 = 0002 pixels ..... 111111111111 = 4096 pixels
		FH	W0[131:120]	Frame height minus 1 (in rows). 000000000000 = 0001 pixels 000000000001 = 0002 pixels ..... 111111111111 = 4096 pixels
2*N + 1	Word1	EBA0	W1[31:0]	Double buffer page 0 destination address for RGB and Y:U:V formats. Must be aligned to 32-bit boundaries.
		EBA1	W1[63:32]	Double buffer page 1 destination address for RGB and Y:U:V formats. Must be aligned to 32-bit boundaries.
2*N + 1	Word1	BPP	W1[66:64]	Determines the size of each pixel in bits. 011 = 8 bits per color component (Only valid value) All other BPP values will cause unknown results.
2*N + 1	Word1	SL	W1[80:67]	Stride line value minus 1 for Y component buffer in bytes (number of maximum bytes in a row according to memory limitations). For U and V component buffers, the stride line value is (SL+1)/2. 00000000000000 = 00001 bytes 00000000000001 = 00002 bytes ..... 11111111111111 = 16384 bytes
		PFS	W1[83:81]	Selects between the following formats: Y:U:V 4.4.4, Y:U:V 4.2.2, Y:U:V 4.2.0 non-interleaved and no packing/unpacking for all 3 formats. Decoded pixels: 001 = Y:U:V Non-interleaved 4:4:4 010 = Y:U:V Non-interleaved 4:2:2 011 = Y:U:V Non-interleaved 4:2:0 All other PFS values will cause unknown results.

Table 44-30. Constant Channel Parameters for RGB/YUV Non-Interleaved Configuration (continued)

Address	Word	Parameter	Field	Description
2*N+1	Word1	BAM	W1[86:84]	Determines mode of block arrangement on frame, for supporting all block rotation and flip constellations. BAM[0] = 0 -> No vertical flip BAM[0] = 1 -> Vertical flip BAM[1] = 0 -> No horizontal flip BAM[1] = 1 -> Horizontal flip BAM[2] = 0 -> No 90 degree rotation BAM[2] = 1 -> 90 degree rotation The BAM bits controls rotation/flipping both for the IDMAC and the IC. For the channels 10,11 and 13 all the BAM bits are valid. In this case, the IDMAC performs rotation/flipping of position of 8*8 blocks, and the IC rotation unit performs rotation/flipping of pixels inside the 8*8 blocks. The BAM[1] is also valid for the channels 0, 1, 2. The IC can perform horizontal flip for these channels. The BAM[0] is valid for all the channels (excluding the channels 24-31). This bit controls vertical flip done by the IDMAC.
		-	W1[88:87]	Reserved
2*N + 1	Word1	NPB	W1[94:89]	Burst size minus 1 (in pixels). The following are valid numbers of pixels in a memory burst access according to the BPP parameter: 000000 = 01 pixels in each burst 000001 = 02 pixels in each burst ..... 111111 = 64 pixels in each burst Range: 8bpp => 4, 8, 12, 16 pixels All other values of NPB are not valid. For channels 0 to 9 and 12, NPB must be set to 8 pixels or 16 pixels. For channels 10, 11 and 13, NPB must be set to 8 pixels. All other values for NPB for channels 0 to 13 are not valid. For channels from 18 to 23, NPB must be set to 8 or 16 pixels.
		-	W1[95]	Reserved.
		SAT	W1[97:96]	Determines what are access type (32 bit, 16 bit and 8 bit) used by the MCU to write/read data to/from the system memory. 00 => 08 bit 01 => 16 bit 10 => 32 bit 11 => Reserved
2*N + 1	Word1	-	W0[131:98]	Reserved.

Table 44-31. Variable Channel for Parameters for RGB/YUV Interleaved Configuration

Address	Word	Parameter	Field	Description
2*N	Word0	XV	W0[9:0]	Variable coordinates for determining next block address. {X1,Y1} and {X2,Y2} coordinates will be determined according to {XV,YV} upon restart of channel. These coordinates are used for RGB and Y:U:V (Y pointer) formats.
		YV	W0[19:10]	
		XB	W0[31:20]	Variable coordinates for determining address within the block. These coordinates are used for RGB and Y:U:V (Y pointer) formats. Need 20 bits for 1D transfer support.
		YB	W0[43:32]	
		SCE	W0[44]	Enables SX and SY to be incremented after EOF. **Note:** Users should not write to this bit.
2*N	Word0	-	W0[45]	Reserved.
		NSB	W0[46]	This bit determines if the next value for {xb,yb} should be taken from {xb,yb} saved in channel parameter memory or from new {x1,y1}/{x2,y2}.
		LNPB	W0[52:47]	Holds the value of the last burst size incase size of row is not aligned to burst size. These bits are updated by Addressing Arithmetic Unit one channel access before the end of row access and therefore saved at the end of that channel event.
		SX	W0[62:53]	Holds the temporary count for the Scroll X in between frame
		SY	W0[72:63]	Holds the temporary count for the Scroll Y in between frame
		NS	W0[82:73]	This variable holds the total number of scrolls

Table 44-32. Constant Channel Addressing Parameters for RGB/YUV Interleaved Configuration

Address	Word	Parameter	Field	Description
2*N	Word0	SM	W0[92:83]	Maximal number of scrolling steps minus 1 (in frames). 0000000000 = 0001 0000000001 = 0002 ..... 1111111111 = 1024

Table 44-32. Constant Channel Addressing Parameters for RGB/YUV Interleaved Configuration (continued)

Address	Word	Parameter	Field	Description
2*N	Word0	SDX	W0[97:93]	Coded horizontal scrolling increment (in pixels). 00000 = 00 pixels 00001 = 01 pixel 00010 = 02 pixels ..... 11110 = 30 pixels 11111 = 32 pixels
		SDY	W0[102:98]	Coded vertical scrolling increment (in rows). 00000 = 00 rows 00001 = 01 row 00010 = 02 rows ..... 11110 = 30 rows 11111 = 32 rows
2*N	Word0	SDRX	W0[103]	Horizontal scrolling direction. 0 => Next frame will be right of current 1 => Next frame will be left of current
		SDRY	W0[104]	Vertical scrolling direction. 0 => Next frame will be down of current 1 => Next frame will be up of current
2*N	Word0	SCRQ	W0[105]	Request to start scrolling. Scrolling will start from next frame. 0 => No scrolling request 1 => Scrolling request
		-	W0[107:106]	Reserved.
2*N	Word0	FW	W0[119:108]	Frame width minus 1 (in pixels). 000000000000 = 0001 pixels 000000000001 = 0002 pixels ..... 111111111111 = 4096 pixels
		FH	W0[131:120]	Frame height minus 1 (in rows). 000000000000 = 0001 pixels 000000000001 = 0002 pixels ..... 111111111111 = 4096 pixels
2*N + 1	Word1	EBA0	W1[31:0]	Double buffer page 0 destination address for RGB and Y:U:V formats. Must be aligned to 32-bit boundaries.
		EBA1	W1[63:32]	Double buffer page 1 destination address for RGB and Y:U:V formats. Must be aligned to 32-bit boundaries.

Table 44-32. Constant Channel Addressing Parameters for RGB/YUV Interleaved Configuration (continued)

Address	Word	Parameter	Field	Description
2*N + 1	Word1	BPP	W1[66:64]	Determines the size of each pixel in bits. BPP. Values: 000 32 Bits per pixel 001 24 Bits per pixel 010 16 Bits per pixel 011 08 Bits per pixel 100 04 Bits per pixel 101 01 Bits per pixel 110 Reserved 111 Reserved
2*N + 1	Word1	SL	W1[80:67]	Stride line in bytes minus 1 (number of maximum bytes in a row according to memory limitations). Must be a multiple of pixel size. Values: 00000000000000 = 00001 bytes 00000000000001 = 00002 bytes ..... 11111111111111 = 16384 bytes
		PFS	W1[83:81]	Selects between the following formats: R:G:B/Y:U:V interleaved with and without packing/unpacking. Decoded pixels. Values: 000 = Code 100 = R:G:B Packing/Unpacking 110 = Y:U:V Interleaved 4:2:2 111 = Generic Data All other PFS values will cause unknown results.
2*N+1	Word1	BAM	W1[86:84]	Determines mode of block arrangement on frame, for supporting all block rotation and flip constellations. The BAM bits controls rotation/flipping both for the IDMAC and the IC. For the channels 10,11 and 13 all the BAM bits are valid. In this case, the IDMAC performs rotation/flipping of position of 8*8 blocks, and the IC rotation unit performs rotation/flipping of pixels inside the 8*8 blocks. The BAM[1] is also valid for the channels 0, 1, 2. The IC can perform horizontal flip for these channels. The BAM[0] is valid for all the channels (excluding the channels 24-31). This bit controls vertical flip done by the IDMAC. BAM[0] = 0 -> No vertical flip BAM[0] = 1 -> Vertical flip BAM[1] = 0 -> No horizontal flip BAM[1] = 1 -> Horizontal flip BAM[2] = 0 -> No 90 degree rotation BAM[2] = 1 -> 90 degree rotation
		-	W1[88:87]	Reserved

Table 44-32. Constant Channel Addressing Parameters for RGB/YUV Interleaved Configuration (continued)

Address	Word	Parameter	Field	Description
2*N + 1	Word1	NPB	W1[94:89]	Burst size minus 1 (in pixels). The following are valid numbers of pixels in a memory burst access according to the bpp parameter: 000000 = 01 pixels in each burst 000001 = 02 pixels in each burst ..... 111111 = 64 pixels in each burst Range: 32bpp => 1 -> 08 pixels 24bpp => 1 -> 10 pixels 16bpp => 1 -> 16 pixels 08bpp => 1 -> 32 pixels 04bpp => 1 -> 32 pixels 01bpp => 1 -> 64 pixels (only 64 or 32 bpp is allowed) For channels from 0 to 9 and 12, NPB must be set to 8 pixels or 16 pixels. For channels from 10, 11 and 13, NPB must be set to 8 pixels. All other values for NPB for channels 0 to 13 will cause unknown results. For channels from 18 to 23, NPB must be set to 8 or 16 pixels.
		-	W1[95]	Reserved.
		SAT	W1[97:96]	Determines what are access type (32 bit, 16 bit and 8 bit) used by the MCU to write/read data to/from the system memory. 00 => 08 bit 01 => 16 bit 10 => 32 bit 11 => Reserved
2*N + 1	Word1	SCC	W1[98]	Scrolling wrap-around enable. 0 => Scrolling will stop after NS reaches SM+1 1 => Scrolling will start from the beginning after NS reaches SM+1

Table 44-33. Constant Channel Formatting Parameters for YUV/RGB Interleaved Configuration

Address	Word	Parameter	Field	Description
2*N + 1	Word1	OFS0	W1[103:99]	Packing/unpacking parameter. Offset between MSB position of the color component 0 (red color) and MSB position of packed pixel. The color component 0 occupies the most significant bits of the unpacked pixel (see Figure 44-145). 00000 = No offset 00001 = u => 1 bit shift left, p => 1 bit shift right ..... 11111 = u => 31 bits left, p => 31 bits right where * u = unpacking, p = packing, sleft = shift left, sright = shift right For write specified channels, the allowed values of the OFS0 are from 00000 to 10111

Table 44-33. Constant Channel Formatting Parameters for YUV/RGB Interleaved Configuration (continued)

Address	Word	Parameter	Field	Description
2*N + 1	Word1	OFS1	W1[108:104]	Packing/unpacking parameter. Offset between MSB position of the color component 1 (green color) and MSB position of packed pixel. The color component 1 occupies the middle left bits of the unpacked pixel (see Figure 44-145). 00000 = No offset 00001 = u => 1 bit sleft, p => 1 bit sright ..... 11111 = u => 31 bit sleft, p => 31 bit sright * u = unpacking, p = packing, sleft = shift left, sright = shift right For write specified channels, the allowed values of the OFS1 are from 00000 to 10111
		OFS2	W1[113:109]	Packing/unpacking parameter. Offset between MSB position of the color component 2 (blue color) and MSB position of packed pixel. The color component 2 occupies the middle right bits of the unpacked pixel (see Figure 44-145). 00000 = No offset 00001 = u => 1 bit sleft, p => 1 bit sright ..... 11111 = u => 31 bit sleft, p => 31 bit sright * u = unpacking, p = packing, sleft = shift left, sright = shift right For write specified channels, the allowed values of the OFS2 are from 00000 to 10111
		OFS3	W1[118:114]	Packing/unpacking parameter. Offset between MSB position of the color component 3 (local alpha value) and MSB position of packed pixel. The color component 3 occupies the least significant bits of the unpacked pixel (see Figure 44-145). 00000 = No offset 00001 = u => 1 bit sleft, p => 1 bit sright ..... 11111 = u => 31 bit sleft, p => 31 bit sright * u = unpacking, p = packing, sleft = shift left, sright = shift right For write specified channels, the OFS3 is ignored (the real offset is always 11000)
		WID0	W1[121:119]	Packing/unpacking parameter. Color component 0 (red color) width minus 1. The color component 0 occupies the most significant bits of the unpacked pixel (see Figure 44-145). 000 = 1 bits 001 = 2 bits ..... 111 = 8 bits

Table 44-33. Constant Channel Formatting Parameters for YUV/RGB Interleaved Configuration (continued)

Address	Word	Parameter	Field	Description
2*N + 1	Word1	WID1	W1[124:122]	Packing/unpacking parameter. Color component 1 (green color) width minus 1. The color component 1 occupies the middle left bits of the unpacked pixel (see Figure 44-145). 000 = 1 bits 000 = 2 bits ..... 111 = 8 bits
		WID2	W1[127:125]	Packing/unpacking parameter. Color component 2 (blue color) width minus 1. The color component 1 occupies the middle right bits of the unpacked pixel (see Figure 44-145). 000 = 1 bits 001 = 2 bits ..... 111 = 8 bits
		WID3	W1[130:128]	Packing/unpacking parameter. Color component 3 (alpha value) width minus 1. The color component 3 occupies the least significant bits of the unpacked pixel (see Figure 44-145). 000 = 1 bits 001 = 2 bits ..... 111 = 8 bits For write specified channels, the WID3 is ignored (the real width is always 8 bits)
		DEC_SEL	W1[131]	Upon 4bpp, selects between two look-up tables 0 = addresses 0 to 15 1 = addresses 128 to 143
Interleaved configuration bit count: 257 bits				

IDMAC Look-Up Table Memory

Table 44-34. Look-Up Table Memory Structure

Address	Word	Parameter	Field	Description
0	Word0	Decoded Pixel 0 for both 8 and 4 bit coded configuration	W[31:0]	When working in coded pixel format, the data read from the memory is the decoded value of pixel according to address given. In addition, in case of 4 bit code configuration, when DEC_SEL = 0, the 4 bit decoded values are read from address 0->15 and when dec_sel=1 the 4 bit decode values are read from 128->143.
...	...	...		
15	Word15	Decoded Pixel 15 for both 8 and 4 bit coded configuration		
16	Word16	Decoded Pixel 16 for 8 bit coded configuration only		
...	...	...		
127	Word127	Decoded Pixel 127 for 8 bit coded configuration only		
128	Word 128	Decoded Pixel 0 for 4 bit and 128 for 8 bit configuration		
...	...	...		
143	Word143	Decoded Pixel 15 for 4 bit and 143 for 8 bit configuration		
...	...	...		
255	Word255	Decoded Pixel 255 for 8 bit coded configuration only		

ADC Template Memory

Table 44-35. ADC Template Memory Structure

Address	Word	Parameter	Field	Description	
0	Display 0 Write Template Command [29:0]	D	[23:0]	Command/data. Usually, the stream means display's command. It is relevant with WR_CMND or RD_ACK opcodes only.	
		OC	[26:24]	Opcode for stream control:	
				000	RD_DATA—Read data from the display
				001	RD_ACK—Wait for display acknowledge
				010	RD_WAIT—Wait for given number of display clock cycles
				011	WR_XADDR—Send X address or 16 lsb or full address to the display
				100	WR_YADDR—Send Y address or msb of full address to the display
				101	WR_ADDR—Send whole address to the display
				110	WR_CMND—Send command field to the display
				111	WR_DATA—Send data from the Buffer Memory to the display
		FC	[28:27]	Flow control flags:	
				00	Step-by-step
				01	Stop—Template last command, no post command. It has to be located on last command
				10	Jump—Flag for non stopping new template execute. It is used for pipeline loading of new template command after post commands of current template. It has to be located in next to last command, including post command part.
				11	Reserved
		RS	29	Register select value for display's registers	
. . . .					
. . . .					
. . . .					
. . . .					

Table 44-35. ADC Template Memory Structure (continued)

Address	Word	Parameter	Field	Description	
h20	Display 0 Read Template Command [29:0]	D	[23:0]	Command/data. Usually, the stream means display's command. It is relevant with WR_CMND or RD_ACK opcodes only.	
		OC	[26:24]	Opcode for stream control:	
				000	RD_DATA—Read data from the display
				001	RD_ACK—Wait for display acknowledge
				010	RD_WAIT— Wait for given number of display clock cycles
				011	WR_XADDR—Send X address or 16 lsb or full address to the display
				100	WR_YADDR—Send Y address or msb of full address to the display
				101	WR_ADDR—Send whole address to the display
				110	WR_CMND—Send command field to the display
				111	WR_DATA—Send data from the Buffer Memory to the display
		FC	[28:27]	Flow control flags:	
				00	Step-by-step
				01	Stop—Template last command, no post command. It has to be located on last command
				10	Jump—Flag for non stopping new template execute. It is used for pipeline loading of new template command after post commands of current template. It has to be located in next to last command, including post command part.
				11	Reserved
		RS	29	Register select value for display's registers	
. . . .					
. . . .					
. . . .					
. . . .					

Table 44-35. ADC Template Memory Structure (continued)

Address	Word	Parameter	Field	Description	
h40	Display 1 Write Template Command [29:0]	D	[23:0]	Command/data. Usually, the stream means display's command. It is relevant with WR_CMND or RD_ACK opcodes only.	
		OC	[26:24]	Opcode for stream control:	
				000	RD_DATA—Read data from the display
				001	RD_ACK—Wait for display acknowledge
				010	RD_WAIT—Wait for given number of display clock cycles
				011	WR_XADDR—Send X address or 16 lsb or full address to the display
				100	WR_YADDR—Send Y address or msb of full address to the display
				101	WR_ADDR—Send whole address to the display
				110	WR_CMND—Send command field to the display
				111	WR_DATA—Send data from the Buffer Memory to the display
		FC	[28:27]	Flow control flags:	
				00	Step-by-step
				01	Stop—Template last command, no post command. It has to be located on last command
				10	Jump—Flag for non stopping new template execute. It is used for pipeline loading of new template command after post commands of current template. It has to be located in next to last command, including post command part.
				11	Reserved
		RS	29	Register select value for display's registers	
. . . .					
. . . .					
. . . .					
. . . .					

Table 44-35. ADC Template Memory Structure (continued)

Address	Word	Parameter	Field	Description	
h60	Display 1 Read Template Command [29:0]	D	[23:0]	Command/data. Usually, the stream means display's command. It is relevant with WR_CMND or RD_ACK opcodes only.	
		OC	[26:24]	Opcode for stream control:	
				000	RD_DATA—Read data from the display
				001	RD_ACK—Wait for display acknowledge
				010	RD_WAIT—Wait for given number of display clock cycles
				011	WR_XADDR—Send X address or 16 lsb or full address to the display
				100	WR_YADDR—Send Y address or msb of full address to the display
				101	WR_ADDR—Send whole address to the display
				110	WR_CMND—Send command field to the display
				111	WR_DATA—Send data from the Buffer Memory to the display
		FC	[28:27]	Flow control flags:	
				00	Step-by-step
				01	Stop—Template last command, no post command. It has to be located on last command
				10	Jump—Flag for non stopping new template execute. It is used for pipeline loading of new template command after post commands of current template. It has to be located in next to last command, including post command part.
				11	Reserved
		RS	29	Register select value for display's registers	
. . . .					
. . . .					
. . . .					
. . . .					

Table 44-35. ADC Template Memory Structure (continued)

Address	Word	Parameter	Field	Description	
h80	Display 2 Write Template Command [29:0]	D	[23:0]	Command/data. Usually, the stream means display's command. It is relevant with WR_CMND or RD_ACK opcodes only.	
		OC	[26:24]	Opcode for stream control:	
				000	RD_DATA—Read data from the display
				001	RD_ACK—Wait for display acknowledge
				010	RD_WAIT—Wait for given number of display clock cycles
				011	WR_XADDR—Send X address or 16 lsb or full address to the display
				100	WR_YADDR—Send Y address or msb of full address to the display
				101	WR_ADDR—Send whole address to the display
				110	WR_CMND—Send command field to the display
				111	WR_DATA—Send data from the Buffer Memory to the display
		FC	[28:27]	Flow control flags:	
				00	Step-by-step
				01	Stop—Template last command, no post command. It has to be located on last command
				10	Jump—Flag for non stopping new template execute. It is used for pipeline loading of new template command after post commands of current template. It has to be located in next to last command, including post command part.
				11	Reserved
		RS	29	Register select value for display's registers	
. . . .					
. . . .					
. . . .					
. . . .					

Table 44-35. ADC Template Memory Structure (continued)

Address	Word	Parameter	Field	Description	
hA0	Display 2 Read Template Command [29:0]	D	[23:0]	Command/data. Usually, the stream means display's command. It is relevant with WR_CMND or RD_ACK opcodes only.	
		OC	[26:24]	Opcode for stream control:	
				000	RD_DATA—Read data from the display
				001	RD_ACK—Wait for display acknowledge
				010	RD_WAIT—Wait for given number of display clock cycles
				011	WR_XADDR—Send X address or 16 lsb or full address to the display
				100	WR_YADDR—Send Y address or msb of full address to the display
				101	WR_ADDR—Send whole address to the display
				110	WR_CMND—Send command field to the display
				111	WR_DATA—Send data from the Buffer Memory to the display
		FC	[28:27]	Flow control flags:	
				00	Step-by-step
				01	Stop—Template last command, no post command. It has to be located on last command
				10	Jump—Flag for non stopping new template execute. It is used for pipeline loading of new template command after post commands of current template. It has to be located in next to last command, including post command part.
				11	Reserved
		RS	29	Register select value for display's registers	
. . . .					
. . . .					
. . . .					
. . . .					

44.3.3.1.10 IPU Interrupt Control Register 1 (IPU_INT_CTRL_1)

This register contains part of IPU interrupts controls. All the controls of EOF of DMA Channels interrupts can be found in this register.

0x53FC_0028 (IPU_INT_CTRL_1) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DMAPF_7_EOF_EN	DMAPF_6_EOF_EN	DMAPF_5_EOF_EN	DMAPF_4_EOF_EN	DMAPF_3_EOF_EN	DMAPF_2_EOF_EN	DMAPF_1_EOF_EN	DMAPF_0_EOF_EN	DMAADC_7_EOF_EN	DMAADC_6_EOF_EN	DMAADC_5_EOF_EN	DMAADC_4_EOF_EN	DMAADC_3_EOF_EN	DMAADC_2_EOF_EN	DMASDC_3_EOF_EN	DMASDC_2_EOF_EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DMASDC_1_EOF_EN	DMASDC_0_EOF_EN	DMAIC_13_EOF_EN	DMAIC_12_EOF_EN	DMAIC_11_EOF_EN	DMAIC_10_EOF_EN	DMAIC_9_EOF_EN	DMAIC_8_EOF_EN	DMAIC_7_EOF_EN	DMAIC_6_EOF_EN	DMAIC_5_EOF_EN	DMAIC_4_EOF_EN	DMAIC_3_EOF_EN	DMAIC_2_EOF_EN	DMAIC_1_EOF_EN	DMAIC_0_EOF_EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-18. IPU Interrupt Control Register 1 (IPU_INT_CTRL_1)

Table 44-36. IPU_INT_CTRL_1 Field Descriptions

Field	Description
31–0 *_EOF_EN[1]	Enable End of Frame of Channel interrupt. This bit is the control of End Of Frame of Channel #n. 0 Interrupt is disabled. 1 Interrupt is enabled.

[1] Indicates the corresponding DMA channel number noted in Figure 44-18.

44.3.3.1.11 IPU Interrupt Control Register 2 (IPU_INT_CTRL_2)

This register contains part of IPU interrupts controls. All the controls of NFACK of DMA Channels interrupts can be found in this register.

0x53FC_002C (IPU_INT_CTRL_2) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DMAPF_7_NFACK_EN	DMAPF_6_NFACK_EN	DMAPF_5_NFACK_EN	DMAPF_4_NFACK_EN	DMAPF_3_NFACK_EN	DMAPF_2_NFACK_EN	DMAPF_1_NFACK_EN	DMAPF_0_NFACK_EN	DMAADC_7_NFACK_EN	DMAADC_6_NFACK_EN	DMAADC_5_NFACK_EN	DMAADC_4_NFACK_EN	DMAADC_3_NFACK_EN	DMAADC_2_NFACK_EN	DMASDC_3_NFACK_EN	DMASDC_2_NFACK_EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DMASDC_1_NFACK_EN	DMASDC_0_NFACK_EN	DMAIC_13_NFACK_EN	DMAIC_12_NFACK_EN	DMAIC_11_NFACK_EN	DMAIC_10_NFACK_EN	DMAIC_9_NFACK_EN	DMAIC_8_NFACK_EN	DMAIC_7_NFACK_EN	DMAIC_6_NFACK_EN	DMAIC_5_NFACK_EN	DMAIC_4_NFACK_EN	DMAIC_3_NFACK_EN	DMAIC_2_NFACK_EN	DMAIC_1_NFACK_EN	DMAIC_0_NFACK_EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-19. IPU Interrupt Control Register 2 (IPU_INT_CTRL_2)

Table 44-37. IPU_INT_CTRL_2 Field Descriptions

Field	Description
31–0 *_NFACK_EN [1]	Enable new frame and ACK of channel interrupt. This bit is the control of new frame and ACK of channel interrupt. 0 Interrupt is disabled. 1 Interrupt is enabled.

[1] Indicates the corresponding DMA channel number noted in Figure 44-19.

44.3.3.1.12 IPU Interrupt Control Register 3 (IPU_INT_CTRL_3)

This register contains part of IPU interrupts controls.

0x53FC_0030
(IPU_INT_CTRL_3)

Access: User Read/Write

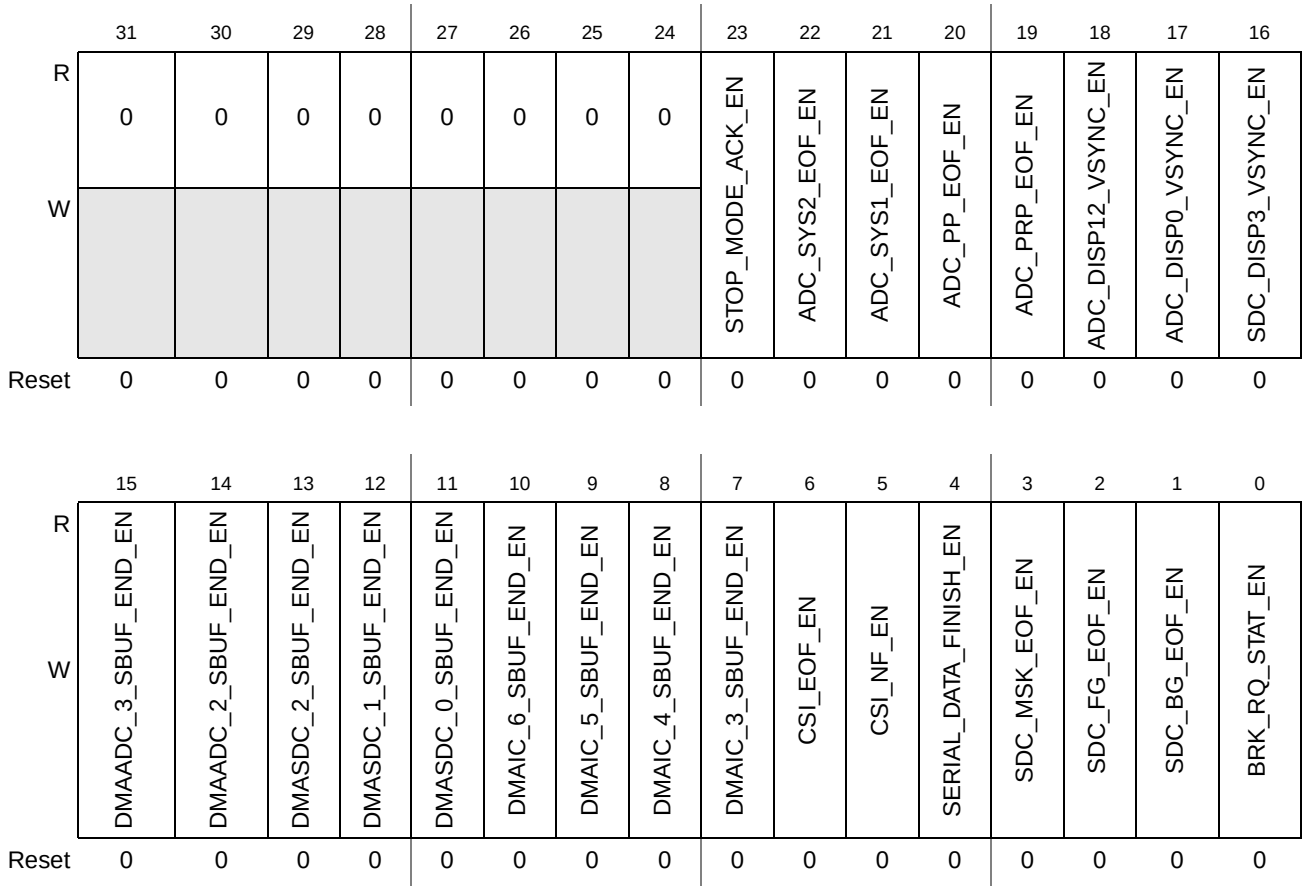


Figure 44-20. IPU Interrupt Control Register 3 (IPU_INT_CTRL_3)

Table 44-38. IPU_INT_CTRL_3 Field Descriptions

Field	Description
31–24	Reserved
23 STOP_MODE_ACK_EN	Control of stop mode interrupt. This bit enables the stop mode interrupt. The interrupt is generated when the IPU send IPG_STOP_ACK signal. 0 Interrupt is disabled. 1 Interrupt is enabled.
22 ADC_SYS2_EOF_EN	Control of end-of-frame interrupt for the ADC system 2 channel. This bit enables end-of-frame interrupt for the ADC system 2 channel. 0 Interrupt is disabled. 1 Interrupt is enabled.
21 ADC_SYS1_EOF_EN	Control of end-of-frame interrupt for the ADC system 1 channel. This bit enables end-of-frame interrupt for the ADC system 1 channel. 0 Interrupt is disabled. 1 Interrupt is enabled.

Table 44-38. IPU_INT_CTRL_3 Field Descriptions (continued)

Field	Description
20 ADC_PP_EOF_EN	Control of end-of-frame interrupt for the ADC postprocessing channel. This bit enables end-of-frame interrupt for the ADC postprocessing channel. 0 Interrupt is disabled. 1 Interrupt is enabled.
19 ADC_PRP_EOF_EN	Control of end-of-frame interrupt for the ADC preprocessing channel. This bit enables end-of-frame interrupt for the ADC preprocessing channel. 0 Interrupt is disabled. 1 Interrupt is enabled.
18 ADC_DISP12_VSYNC_EN	Control of VSYNC interrupt for displays 1 and 2. This bit enables interrupt for displays 1 and 2. 0 Interrupt is disabled. 1 Interrupt is enabled.
17 ADC_DISP0_VSYNC_EN	Control of VSYNC interrupt for display 0. This bit enables interrupt for display 0. 0 Interrupt is disabled. 1 Interrupt is enabled.
16 SDC_DISP3_VSYNC_EN	Control of VSYNC interrupt for display 3. This bit enables interrupt for display 3. 0 Interrupt is disabled. 1 Interrupt is enabled.
15 DMAADC_3_SBUF_END_EN	Control of DMAADC channel 3 scroll buffer end interrupt. This bit is the control of scroll buffer end of input system 2 ADC channel. 0 Interrupt is disabled. 1 Interrupt is enabled.
14 DMAADC_2_SBUF_END_EN	Control of DMAADC channel 2 scroll buffer end interrupt. This bit is the control of scroll buffer end of input system 1 ADC channel. 0 Interrupt is disabled. 1 Interrupt is enabled.
13 DMASDC_2_SBUF_END_EN	Control of DMASDC channel 2 scroll buffer end interrupt. This bit is the control of scroll buffer end of input mask SDC channel. 0 Interrupt is disabled. 1 Interrupt is enabled.
12 DMASDC_1_SBUF_END_EN	Control of DMASDC channel 1 scroll buffer end interrupt. This bit is the control of scroll buffer end of input background SDC channel. 0 Interrupt is disabled. 1 Interrupt is enabled.
11 DMASDC_0_SBUF_END_EN	Control of DMASDC channel 0 scroll buffer end interrupt. This bit is the control of scroll buffer end of input foreground SDC channel. 0 Interrupt is disabled. 1 Interrupt is enabled.
10 DMAIC_6_SBUF_END_EN	Control of DMAIC channel 6 scroll buffer end interrupt. This bit is the control of scroll buffer end of input graphics for PRPVF channel. 0 Interrupt is disabled. 1 Interrupt is enabled.

Table 44-38. IPU_INT_CTRL_3 Field Descriptions (continued)

Field	Description
9 DMAIC_5_SBUF_END_EN	Control of DMAIC channel 5 scroll buffer end interrupt. This bit is the control of scroll buffer end of input VIDEO for PP task. 0 Interrupt is disabled. 1 Interrupt is enabled.
8 DMAIC_4_SBUF_END_EN	Control of DMAIC channel 4 scroll buffer end interrupt. This bit is the control of scroll buffer end of input graphics for PRPPP task. 0 Interrupt is disabled. 1 Interrupt is enabled.
7 DMAIC_3_SBUF_END_EN	Control of DMAIC channel 3 scroll buffer end interrupt. This bit is the control of scroll buffer end of input graphics for PRPVF task. 0 Interrupt is disabled. 1 Interrupt is enabled.
6 CSI_EOF_EN	Enable CSI_EOF interrupt. This bit is the control of CSI_EOF interrupt. 0 Interrupt is disabled. 1 Interrupt is enabled.
5 CSI_NF_EN	Enable CSI_NF interrupt. This bit is the control of CSI_NF interrupt. 0 Interrupt is disabled. 1 Interrupt is enabled.
4 SERIAL_DATA_FINISH_EN	Enable the DI serial data finish interrupt. This bit is the control of DI serial data finish interrupt. 0 Interrupt is disabled. 1 Interrupt is enabled.
3 SDC_MSK_EOF_EN	Enable SDC Mask EOF interrupt. This bit is the control of SDC Mask EOF interrupt. 0 Interrupt is disabled. 1 Interrupt is enabled.
2 SDC_FG_EOF_EN	Enable SDC Foreground EOF interrupt. This bit is the control of SDC Foreground EOF interrupt. 0 Interrupt is disabled. 1 Interrupt is enabled.
1 SDC_BG_EOF_EN	Enable SDC Background EOF interrupt. This bit is the control of SDC Background EOF interrupt. 0 Interrupt is disabled. 1 Interrupt is enabled.
0 BRK_RQ_STAT_EN	Enable BRK_RQ interrupt. This bit is the control of BRK_RQ interrupt. 0 Interrupt is disabled. 1 Interrupt is enabled.

44.3.3.1.13 IPU Interrupt Control Register 4 (IPU_INT_CTRL_4)

This register contains part of IPU interrupts controls. All the controls of NFB4EOF_ERR Channels interrupts can be found in this register.

0x53FC_0034
(IPU_INT_CTRL_4)

Access: User Read/Write

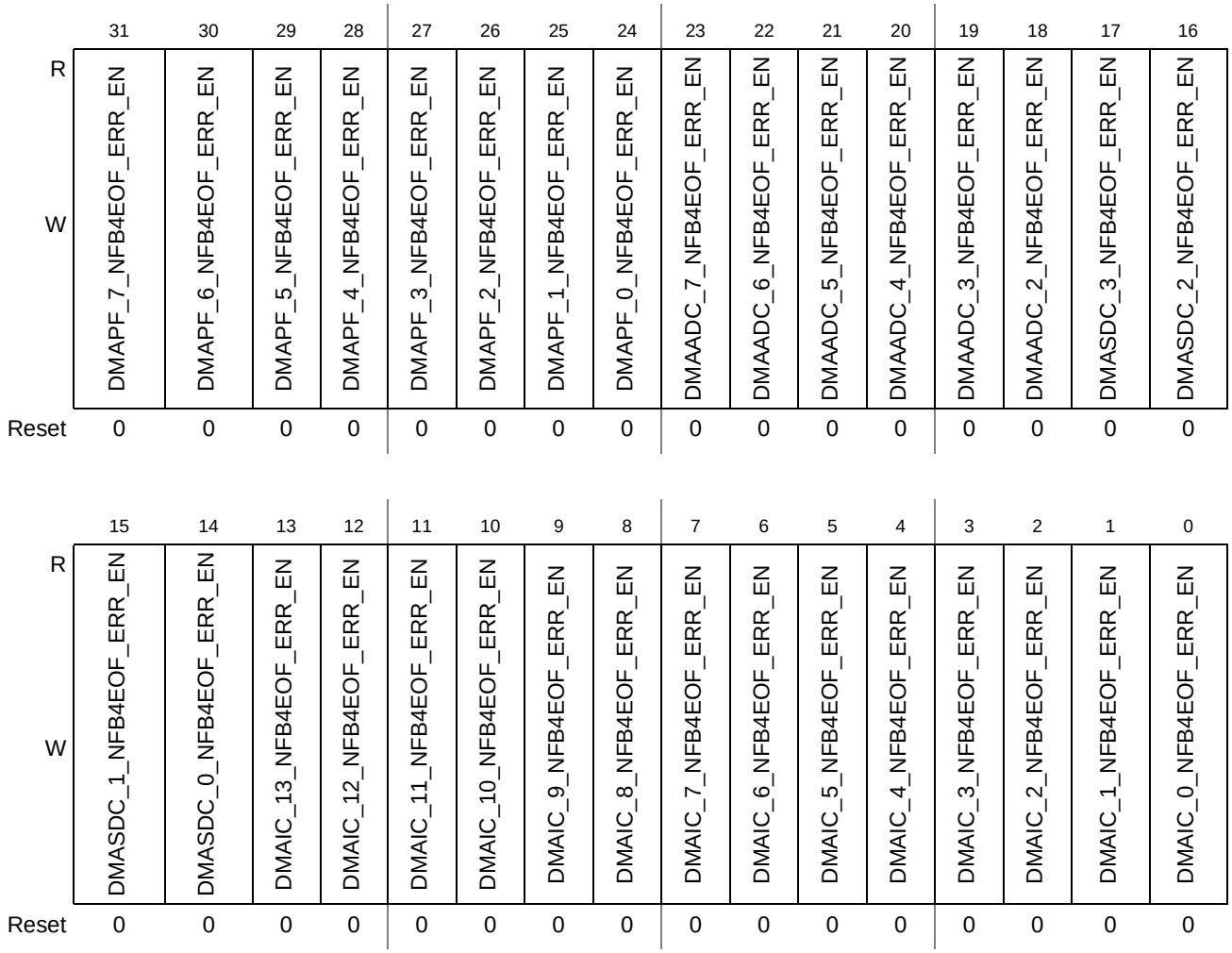


Figure 44-21. IPU Interrupt Control Register 4 (IPU_INT_CTRL_4)

Table 44-39. IPU_INT_CTRL_4 Field Descriptions

Field	Description
31–0 *_NFB4EOF_ERR_EN[1]	The control of NFB4EOF error enable. This bit is the control of new frame and ACK of channel interrupt. 0 Interrupt is disabled. 1 Interrupt is enabled.

[1] Indicates the corresponding DMA channel number noted in Figure 44-21.

44.3.3.1.14 IPU Interrupt Control Register 5 (IPU_INT_CTRL_5)

This register contains part of IPU interrupts controls.

0x53FC_0038
(IPU_INT_CTRL_5)

Access: User Read/Write

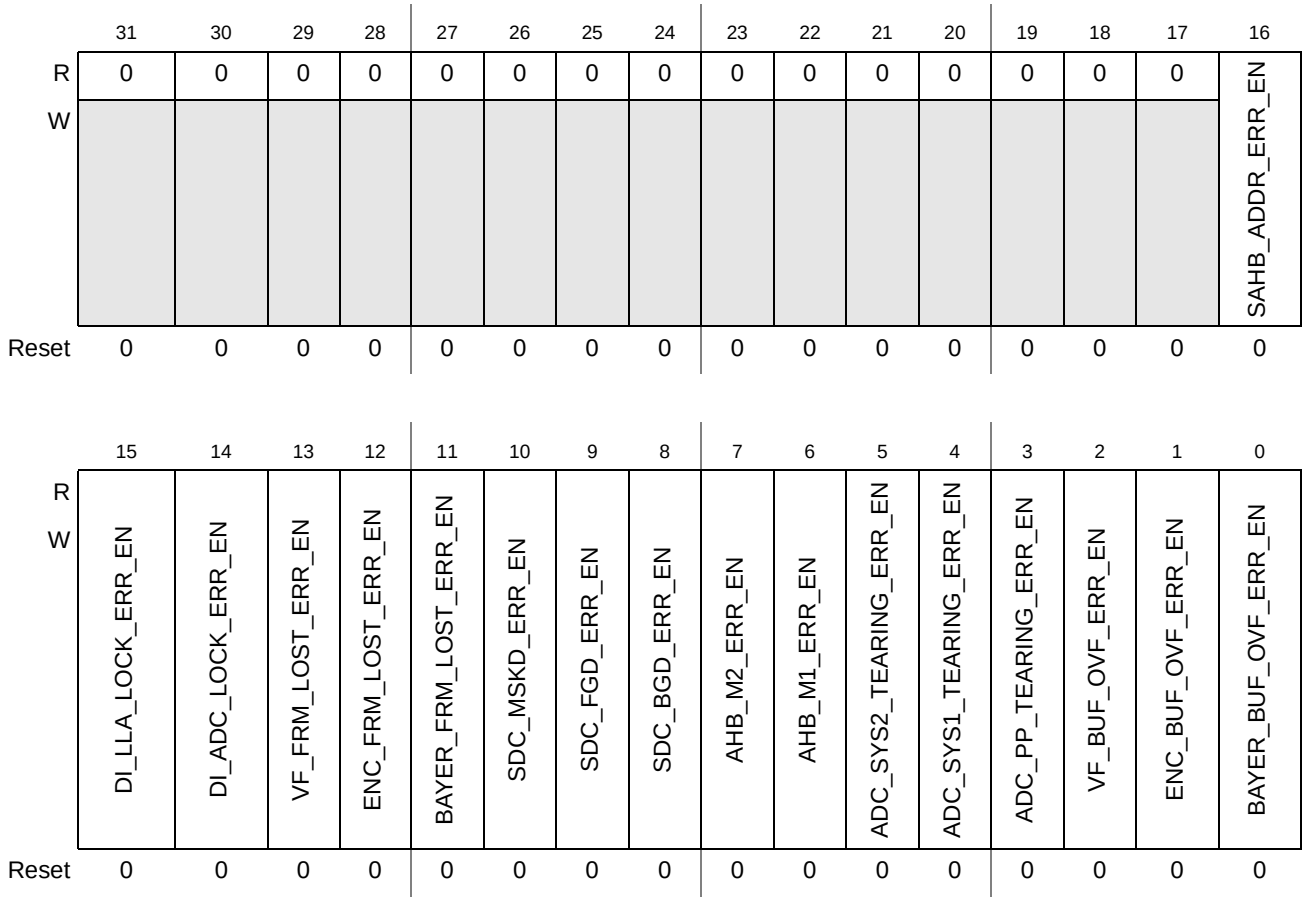


Figure 44-22. IPU Interrupt Control Register 5 (IPU_INT_CTRL_5)

Table 44-40. IPU_INT_CTRL_5 Field Descriptions

Field	Description
31–17	Reserved
16 SAHB_ADDR_ERR_EN	Enable SAHB_ADDR_ERR interrupt. This bit enables the SAHB_ADDR_ERR interrupt. The interrupt is generated if the slave AHB address most significant bits do not match the base address defined by the AHB_S_BA pins. 0 Interrupt is disabled. 1 Interrupt is enabled.
15 DI_LLA_LOCK_ERR_EN	Enable DI_LLA_LOCK_ERR interrupt. This bit enables the DI_LLA_LOCK_ERR interrupt. The interrupt is generated if the DI has not been released by low level access when the SDC begins data transfer to a synchronous interface. 0 Interrupt is disabled. 1 Interrupt is enabled.

Table 44-40. IPU_INT_CTRL_5 Field Descriptions (continued)

Field	Description
14 DI_ADC_LOCK_ERR_EN	Enable DI_ADC_LOCK_ERR interrupt. This bit enables the DI_ADC_LOCK_ERR interrupt. The interrupt is generated if the DI has not been released by the ADC when the SDC begins data transfer to a synchronous interface. 0 Interrupt is disabled. 1 Interrupt is enabled.
13 VF_FRM_LOST_ERR_EN	Enable VF_FRM_LOST_ERR interrupt. This bit is the enable VF_FRM_LOST_ERR interrupt. 0 Interrupt is disabled. 1 Interrupt is enabled.
12 ENC_FRM_LOST_ERR_EN	Enable ENC_FRM_LOST_ERR interrupt. This bit is the enable ENC_FRM_LOST_ERR interrupt. 0 Interrupt is disabled. 1 Interrupt is enabled.
11 BAYER_FRM_LOST_ERR_EN	Enable BAYER_FRM_LOST_ERR interrupt. This bit is the enable BAYER_FRM_LOST_ERR interrupt. 0 Interrupt is disabled. 1 Interrupt is enabled.
10 SDC_MSK_ERR_EN	Enable SDC_MSK_ERR interrupt. This bit is the control of SDC_MSK_ERR interrupt. 0 Interrupt is disabled. 1 Interrupt is enabled.
9 SDC_FGD_ERR_EN	Enable SDC_FGD_ERR interrupt. This bit is the control of SDC_FGD_ERR interrupt. 0 Interrupt is disabled. 1 Interrupt is enabled.
8 SDC_BGD_ERR_EN	Enable SDC_BGD_ERR interrupt. This bit is the control of SDC_BGD_ERR interrupt. 0 Interrupt is disabled. 1 Interrupt is enabled.
7 AHB_M2_ERR_EN	Enable AHB_M2_ERR interrupt. This bit enables the AHB_M2_ERR interrupt. The interrupt is generated if master 2 AHB error appears. 0 Interrupt is disabled. 1 Interrupt is enabled.
6 AHB_M1_ERR_EN	Enable AHB_M1_ERR interrupt. This bit enables the AHB_M1_ERR interrupt. The interrupt is generated if master 1 AHB error appears. 0 Interrupt is disabled. 1 Interrupt is enabled.
5 ADC_SYS2_TEARING_ERR_EN	Enable ADC_SYS2_TEARING_ERR interrupt. This bit enables the ADC_SYS2_TEARING_ERR interrupt. The interrupt is generated if tearing takes place for the system 2 ADC channel. 0 Interrupt is disabled. 1 Interrupt is enabled.

Table 44-40. IPU_INT_CTRL_5 Field Descriptions (continued)

Field	Description
4 ADC_SYS1_TEARING_ERR_EN	Enable ADC_SYS1_TEARING_ERR interrupt. This bit enables the ADC_SYS1_TEARING_ERR interrupt. The interrupt is generated if tearing takes place for the system 1 ADC channel. 0 Interrupt is disabled. 1 Interrupt is enabled.
3 ADC_PP_TEARING_ERR_EN	Enable ADC_PP_TEARING_ERR interrupt. This bit enables the ADC_PP_TEARING_ERR interrupt. The interrupt is generated if tearing takes place for the postprocessing ADC channel. 0 Interrupt is disabled. 1 Interrupt is enabled.
2 VF_BUF_OVF_ERR_EN	Enable VF_BUF_OVF_ERR interrupt. This bit is the control of VF_BUF_OVF_ERR interrupt. 0 Interrupt is disabled. 1 Interrupt is enabled.
1 ENC_BUF_OVF_ERR_EN	Enable ENC_BUF_OVF_ERR interrupt. This bit is the control of ENC_BUF_OVF_ERR interrupt. 0 Interrupt is disabled. 1 Interrupt is enabled.
0 BAYER_BUF_OVF_ERR_EN	Enable BAYER_BUF_OVF_ERR interrupt. This bit is the control of BAYER_BUF_OVF_ERR interrupt. 0 Interrupt is disabled. 1 Interrupt is enabled.

44.3.3.1.15 IPU Interrupt Status Register 1 (IPU_INT_STAT_1)

This register contains part of IPU interrupt status. All end of frame of DMA channels can be found in this register.

0x53FC_003C
(IPU_INT_STAT_1)

Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DMAPF_7_EOF	DMAPF_6_EOF	DMAPF_5_EOF	DMAPF_4_EOF	DMAPF_3_EOF	DMAPF_2_EOF	DMAPF_1_EOF	DMAPF_0_EOF	DMAADC_7_EOF	DMAADC_6_EOF	DMAADC_5_EOF	DMAADC_4_EOF	DMAADC_3_EOF	DMAADC_2_EOF	DMASDC_3_EOF	DMASDC_2_EOF
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DMASDC_1_EOF	DMASDC_0_EOF	DMAIC_13_EOF	DMAIC_12_EOF	DMAIC_11_EOF	DMAIC_10_EOF	DMAIC_9_EOF	DMAIC_8_EOF	DMAIC_7_EOF	DMAIC_6_EOF	DMAIC_5_EOF	DMAIC_4_EOF	DMAIC_3_EOF	DMAIC_2_EOF	DMAIC_1_EOF	DMAIC_0_EOF
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-23. IPU Interrupt Status Register 1 (IPU_INT_STAT_1)

Table 44-41. IPU_INT_STAT_1 Field Descriptions

Field	Description
31–0 *_EOF -[1]	End of frame of <block> and channel <num>. This bit is the status of end of frame of <block> and channel <num>. 0 Interrupt is cleared. 1 Interrupt is requested.

[1] Indicates the corresponding DMA channel number noted in Figure 44-23.

44.3.3.1.16 IPU Interrupt Status Register 2 (IPU_INT_STAT_2)

This register contains part of IPU interrupt status. All NFACK of DMA Channels can be found in this register.

0x53FC_0040 (IPU_INT_STAT_2) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DMAPF_7_NFACK	DMAPF_6_NFACK	DMAPF_5_NFACK	DMAPF_4_NFACK	DMAPF_3_NFACK	DMAPF_2_NFACK	DMAPF_1_NFACK	DMAPF_0_NFACK	DMAADC_7_NFACK	DMAADC_6_NFACK	DMAADC_5_NFACK	DMAADC_4_NFACK	DMAADC_3_NFACK	DMAADC_2_NFACK	DMASDC_3_NFACK	DMASDC_2_NFACK
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DMASDC_1_NFACK	DMASDC_0_NFACK	DMAIC_13_NFACK	DMAIC_12_NFACK	DMAIC_11_NFACK	DMAIC_10_NFACK	DMAIC_9_NFACK	DMAIC_8_NFACK	DMAIC_7_NFACK	DMAIC_6_NFACK	DMAIC_5_NFACK	DMAIC_4_NFACK	DMAIC_3_NFACK	DMAIC_2_NFACK	DMAIC_1_NFACK	DMAIC_0_NFACK
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-24. IPU Interrupt Status Register 2 (IPU_INT_STAT_2)

Table 44-42. IPU_INT_STAT_2 Field Descriptions

Field	Description
31–0 *_NFACK[1]	NFACK of <block> and Channel <num>. This bit is the status of NFACK of <block> and Channel <num>. 0 Interrupt is cleared. 1 Interrupt is requested.

1 Indicates the corresponding DMA channel number noted in Figure 44-24.

44.3.3.1.17 IPU Interrupt Status Register 3 (IPU_INT_STAT_3)

This register contains part of the IPU interrupt status.

0x53FC_0044 (IPU_INT_STAT_3) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	STOP_MODE_ACK	ADC_SYS2_EOF	ADC_SYS1_EOF	ADC_PP_EOF	ADC_PRP_EOF	ADC_DISP12_VSYNC	ADC_DISP0_VSYNC	SDC_DISP3_VSYNC
W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DMAADC_3_SBUF_END	DMAADC_2_SBUF_END	DMASDC_2_SBUF_END	DMASDC_1_SBUF_END	DMASDC_0_SBUF_END	DMAIC_6_SBUF_END	DMAIC_5_SBUF_END	DMAIC_4_SBUF_END	DMAIC_3_SBUF_END	CSI_EOF	CSI_NF	SERIAL_DATA_FINISH	SDC_MSK_EOF	SDC_FG_EOF	SDC_BG_EOF	BRK_RQ_STAT
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-25. IPU Interrupt Status Register 3 (IPU_INT_STAT_3)

Table 44-43. IPU_INT_STAT_3 Field Descriptions

Field	Description
31–24	Reserved
23 STOP_MODE_ACK	Control of stop mode interrupt. This bit is the status of the stop mode interrupt. The interrupt is generated when the IPU send IPG_STOP_ACK signal. 0 Interrupt is cleared. 1 Interrupt is requested.
22 ADC_SYS2_EOF	Status of end-of-frame interrupt for the ADC system 2 channel. This bit is the status of end-of-frame interrupt for the ADC system 2 channel. The interrupt is generated after the ADC has completed data transfer. 0 Interrupt is cleared. 1 Interrupt is requested.

Table 44-43. IPU_INT_STAT_3 Field Descriptions (continued)

Field	Description
21 ADC_SYS1_EOF	Status of end-of-frame interrupt for the ADC system 1 channel. This bit is the status of end-of-frame interrupt for the ADC system 1 channel. The interrupt is generated after the ADC has completed data transfer. 0 Interrupt is cleared. 1 Interrupt is requested.
20 ADC_PP_EOF	Status of end-of-frame interrupt for the ADC postprocessing channel. This bit is the status of end-of-frame interrupt for the ADC postprocessing channel. The interrupt is generated after the ADC has completed data transfer. 0 Interrupt is cleared. 1 Interrupt is requested.
19 ADC_PRP_EOF	Status of end-of-frame interrupt for the ADC preprocessing channel. This bit is the status of end-of-frame interrupt for the ADC preprocessing channel. The interrupt is generated after the ADC has completed data transfer. 0 Interrupt is cleared. 1 Interrupt is requested.
18 ADC_DISP12_VSYNC	Status of VSYNC interrupt for displays 1 and 2. This bit is the status of interrupt for displays 1 and 2. 0 Interrupt is cleared. 1 Interrupt is requested.
17 ADC_DISP0_VSYNC	Status of VSYNC interrupt for display 0. This bit is the status of interrupt for display 0. 0 Interrupt is cleared. 1 Interrupt is requested.
16 SDC_DISP3_VSYNC	Status of VSYNC interrupt for display 3. This bit is the status of interrupt for display 3. 0 Interrupt is cleared. 1 Interrupt is requested.
15 DMAADC_3_SBUF_END	Status of DMASDC channel 3 scroll buffer end interrupt. This bit is the status of scroll buffer end of input system 2 ADC channel. 0 Interrupt is cleared. 1 Interrupt is requested.
14 DMAADC_2_SBUF_END	Status of DMAADC channel 2 scroll buffer end interrupt. This bit is the status of scroll buffer end of input system 1 ADC channel. 0 Interrupt is cleared. 1 Interrupt is requested.
13 DMASDC_2_SBUF_END	Status of DMASDC channel 2 scroll buffer end interrupt. This bit is the status of scroll buffer end of input mask SDC channel. 0 Interrupt is cleared. 1 Interrupt is requested.
12 DMASDC_1_SBUF_END	Status of DMASDC channel 1 scroll buffer end interrupt. This bit is the status of scroll buffer end of input background SDC channel. 0 Interrupt is cleared. 1 Interrupt is requested.

Table 44-43. IPU_INT_STAT_3 Field Descriptions (continued)

Field	Description
11 DMASDC_0_SBUF_END	Status of DMASDC channel 0 scroll buffer end interrupt. This bit is the status of scroll buffer end of input foreground SDC channel. 0 Interrupt is cleared. 1 Interrupt is requested.
10 DMAIC_6_SBUF_END	Status of DMAIC channel 6 scroll buffer end interrupt. This bit is the status of scroll buffer end of input graphics for PRPVF channel. 0 Interrupt is cleared. 1 Interrupt is requested.
9 DMAIC_5_SBUF_END	Status of DMAIC channel 5 scroll buffer end interrupt. This bit is the status of scroll buffer end of input VIDEO for PP task. 0 Interrupt is cleared. 1 Interrupt is requested.
8 DMAIC_4_SBUF_END	Status of DMAIC channel 4 scroll buffer end interrupt. This bit is the status of scroll buffer end of input graphics for PRPPP task. 0 Interrupt is cleared. 1 Interrupt is requested.
7 DMAIC_3_SBUF_END	Status of DMAIC channel 3 scroll buffer end interrupt. This bit is the status of scroll buffer end of input graphics for PRPVF task. 0 Interrupt is cleared. 1 Interrupt is requested.
6 CSI_EOF	Status of CSI_EOF interrupt. This bit is the status of CSI_EOF interrupt. 0 Interrupt is cleared. 1 Interrupt is requested.
5 CSI_NF	Status of CSI_NF interrupt. This bit is the status of CSI_NF interrupt. 0 Interrupt is cleared. 1 Interrupt is requested.
4 SERIAL_DATA_FINISH	Status the DI serial data finish interrupt. This bit is the status of DI serial data finish interrupt. 0 Interrupt is cleared. 1 Interrupt is requested.
3 SDC_MSK_EOF	Status of SDC mask EOF interrupt. This bit is the status of SDC Mask EOF interrupt. 0 Interrupt is cleared. 1 Interrupt is requested.
2 SDC_FG_EOF	Status of SDC foreground EOF interrupt. This bit is the status of SDC foreground EOF interrupt. 0 Interrupt is cleared. 1 Interrupt is requested.
1 SDC_BG_EOF	Status of SDC background EOF interrupt. This bit is the status of SDC background EOF interrupt. 0 Interrupt is cleared. 1 Interrupt is requested.
0 BRK_RQ_STAT	Status of BRK_RQ interrupt. This bit is the status of BRK_RQ interrupt. 0 Interrupt is cleared. 1 Interrupt is requested.

44.3.3.1.18 IPU Interrupt Status Register 4 (IPU_INT_STAT_4)

This register contains part of IPU interrupt status. All _NFB4EOF_ERR of DMA Channels can be found in this register.

0x53FC_0048 (IPU_INT_STAT_4) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DMAPF_7_NFB4EOF_ERR	DMAPF_6_NFB4EOF_ERR	DMAPF_5_NFB4EOF_ERR	DMAPF_4_NFB4EOF_ERR	DMAPF_3_NFB4EOF_ERR	DMAPF_2_NFB4EOF_ERR	DMAPF_1_NFB4EOF_ERR	DMAPF_0_NFB4EOF_ERR	DMAADC_7_NFB4EOF_ERR	DMAADC_6_NFB4EOF_ERR	DMAADC_5_NFB4EOF_ERR	DMAADC_4_NFB4EOF_ERR	DMAADC_3_NFB4EOF_ERR	DMAADC_2_NFB4EOF_ERR	DMASDC_3_NFB4EOF_ERR	DMASDC_2_NFB4EOF_ERR
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DMASDC_1_NFB4EOF_ERR	DMASDC_0_NFB4EOF_ERR	DMAIC_13_NFB4EOF_ERR	DMAIC_12_NFB4EOF_ERR	DMAIC_11_NFB4EOF_ERR	DMAIC_10_NFB4EOF_ERR	DMAIC_9_NFB4EOF_ERR	DMAIC_8_NFB4EOF_ERR	DMAIC_7_NFB4EOF_ERR	DMAIC_6_NFB4EOF_ERR	DMAIC_5_NFB4EOF_ERR	DMAIC_4_NFB4EOF_ERR	DMAIC_3_NFB4EOF_ERR	DMAIC_2_NFB4EOF_ERR	DMAIC_1_NFB4EOF_ERR	DMAIC_0_NFB4EOF_ERR
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-26. IPU Interrupt Status Register 4 (IPU_INT_STAT_4)

Table 44-44. IPU_INT_STAT_4 Field Descriptions

Field	Description
31–0 *_NFB4EOF_ERR[1]	_NFB4EOF_ERR of <block> and channel <num>. This bit is the status of _NFB4EOF_ERR of <block> and channel <num>. 0 Interrupt is cleared. 1 Interrupt is requested.

1 Indicates the corresponding DMA channel number noted in Figure 44-26.

44.3.3.1.19 IPU Interrupt Status Register 5 (IPU_INT_STAT_5)

This register contains part of IPU interrupts status.

0x53FC_004C (IPU_INT_STAT_5) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	SAHB_ADDR_ERR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DI_LLA_LOCK_ERR	DI_ADC_LOCK_ERR	VF_FRM_LOST_ERR	ENC_FRM_LOST_ERR	BAYER_FRM_LOST_ERR	SDC_MSKD_ERR	SDC_FGD_ERR	SDC_BGD_ERR	AHB_M2_ERR	AHB_M1_ERR	ADC_SYS2_TEARING_ERR	ADC_SYS1_TEARING_ERR	ADC_PP_TEARING_ERR	VF_BUF_OVF_ERR	ENC_BUF_OVF_ERR	BAYER_BUF_OVF_ERR
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-27. IPU Interrupt Status Register 5 (IPU_INT_STAT_5)

Table 44-45. IPU_INT_STAT_5 Field Descriptions

Field	Description
30–17	Reserved
16 SAHB_ADDR_ERR	Status of the SAHB_ADDR_ERR interrupt. This bit indicates a status of the SAHB_ADDR_ERR interrupt. The interrupt is generated if the slave AHB address most significant bits do not match the base address defined by the AHB_S_BA pins. 0 Interrupt is cleared. 1 Interrupt is requested.
15 DI_LLA_LOCK_ERR	Status of the DI_LLA_LOCK_ERR interrupt. This bit indicates a status of the DI_LLA_LOCK_ERR interrupt. The interrupt is generated if the DI has not been released by low level access when the SDC begins data transfer to a synchronous interface. 0 Interrupt is cleared. 1 Interrupt is requested.

Table 44-45. IPU_INT_STAT_5 Field Descriptions (continued)

Field	Description
14 DI_ADC_LOCK_ERR	Status of DI_ADC_LOCK_ERR interrupt. This bit indicates a status of the DI_ADC_LOCK_ERR interrupt. The interrupt is generated if the DI has not been released by the ADC when the SDC begins data transfer to a synchronous interface. 0 Interrupt is cleared. 1 Interrupt is requested.
13 VF_FRM_LOST_ERR	Status of VF_FRM_LOST_ERR interrupt. This bit is the Status of VF_FRM_LOST_ERR interrupt. 0 Interrupt is cleared. 1 Interrupt is requested.
12 ENC_FRM_LOST_ERR	Status of ENC_FRM_LOST_ERR interrupt. This bit is the Status of ENC_FRM_LOST_ERR interrupt. 0 Interrupt is cleared. 1 Interrupt is requested.
11 BAYER_FRM_LOST_ERR	Status of BAYER_FRM_LOST_ERR interrupt. This bit is the Status of BAYER_FRM_LOST_ERR interrupt. 0 Interrupt is cleared. 1 Interrupt is requested.
10 SDC_MSKD_ERR	Status of SDC_MSK_ERR interrupt. This bit is the Status of SDC_MSK_ERR interrupt. 0 Interrupt is cleared. 1 Interrupt is requested.
9 SDC_FGD_ERR	Status of SDC_FGD_ERR interrupt. This bit is the status of SDC_FGD_ERR interrupt. 0 Interrupt is cleared. 1 Interrupt is requested.
8 SDC_BGD_ERR	Status of SDC_BGD_ERR interrupt. This bit is the status of SDC_BGD_ERR interrupt. 0 Interrupt is cleared. 1 Interrupt is requested.
7 AHB_M2_ERR	Status of AHB_M2_ERR interrupt. This bit the status of the AHB_M2_ERR interrupt. The interrupt is generated if master 2 AHB error appears. 1 Interrupt is enabled. 0 Interrupt is disabled.
6 AHB_M1_ERR	Status of AHB_M1_ERR interrupt. This bit the status of the AHB_M1_ERR interrupt. The interrupt is generated if master 1 AHB error appears. 0 Interrupt is disabled. 1 Interrupt is enabled.
5 ADC_SYS2_TEARING_ERR	Status of ADC_SYS2_TEARING_ERR interrupt. This bit the status of the ADC_SYS2_TEARING_ERR interrupt. The interrupt is generated if tearing takes place for the system 2 ADC channel. 0 Interrupt is disabled. 1 Interrupt is enabled.

Table 44-45. IPU_INT_STAT_5 Field Descriptions (continued)

Field	Description
4 ADC_SYS1_TEARING_ERR	Status of ADC_SYS1_TEARING_ERR interrupt. This bit the status of the ADC_SYS1_TEARING_ERR interrupt. The interrupt is generated if tearing takes place for the system 1 ADC channel. 0 Interrupt is disabled. 1 Interrupt is enabled.
3 ADC_PP_TEARING_ERR	Status of ADC_PP_TEARING_ERR interrupt. This bit the status of the ADC_PP_TEARING_ERR interrupt. The interrupt is generated if tearing takes place for the postprocessing ADC channel. 0 Interrupt is disabled. 1 Interrupt is enabled.
2 VF_BUF_OVF_ERR	Status of VF_BUF_OVF_ERR interrupt. This bit is the status of VF_BUF_OVF_ERR interrupt. 0 Interrupt is cleared. 1 Interrupt is requested.
1 ENC_BUF_OVF_ERR	Status of ENC_BUF_OVF_ERR interrupt. This bit is the status of ENC_BUF_OVF_ERR interrupt. 0 Interrupt is cleared. 1 Interrupt is requested.
0 BAYER_BUF_OVF_ERR	Status of BAYER_BUF_OVF_ERR interrupt. This bit is the status of BAYER_BUF_OVF_ERR interrupt. 0 Interrupt is cleared. 1 Interrupt is requested.

44.3.3.1.20 IPU Break Control Register 1 (IPU_BRK_CTRL_1)

This register is the first control register of debug unit in control module. It contain the following control bits for debug unit:

0x53FC_0050 (IPU_BRK_CTRL_1) Access: User Read/Write

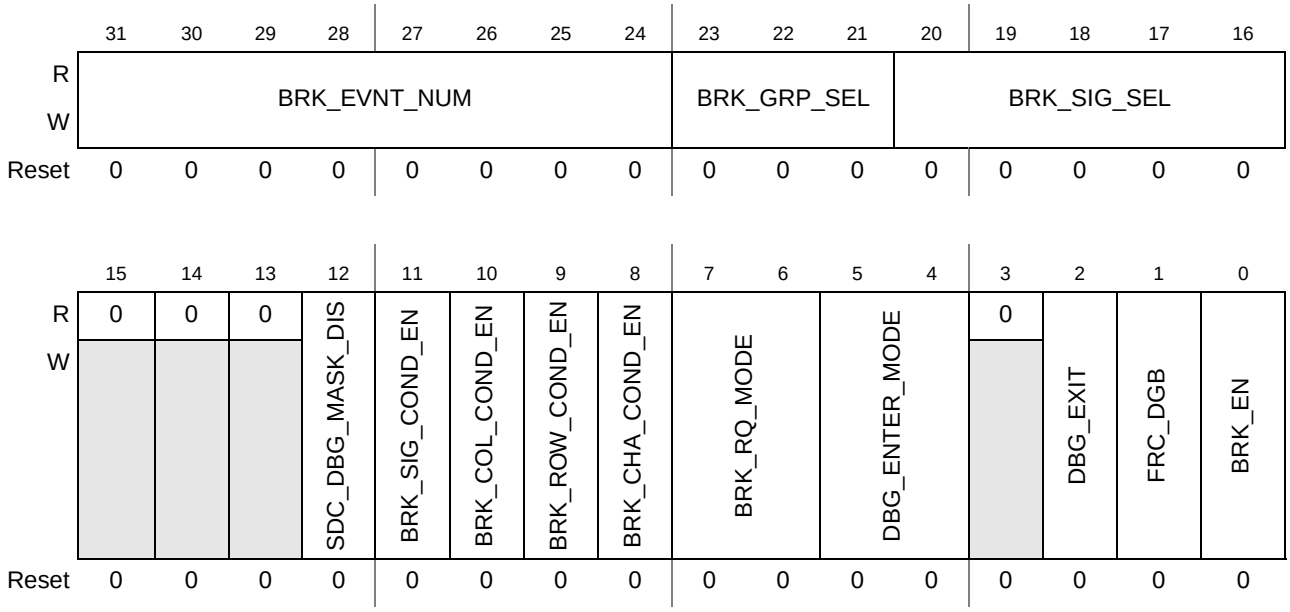


Figure 44-28. IPU Break Control Register 1 (IPU_BRK_CTRL_1)

Table 44-46. IPU_BRK_CTRL_1 Field Descriptions

Field	Description
31–24 BRK_EVNT_NUM	Break event number minus 1. This field is programmed by user to break on a specific break event number.
23–21 BRK_GRP_SEL	Break group select. This field selects a group of break sources. Each group reflects signals that set status bits in the status registers. List and location of signals inside a group matches the corresponding status register. Break Sources Groups/Status Registers: Group 0/IPU_INT_STAT_1 Group 1/IPU_INT_STAT_2 Group 2/IPU_INT_STAT_3 Group 3/IPU_INT_STAT_3 Group 4/IPU_INT_STAT_3
20–16 BRK_SIG_SEL	Break signal select. This field selects which of 32 break sources in the selected group will cause a break. Values: 0 DMA stop 1 Freeze 2 Interrupt 3 RSV
15–13	Reserved
12 SDC_DBG_MASK_DIS	SDC debug mode channel mask disable. This bit disables the masking of SDC channels when debug request arrives. 0 SDC channels are masked when debug request arrives. 1 SDC channels are not masked when debug request arrives.

Table 44-46. IPU_BRK_CTRL_1 Field Descriptions (continued)

Field	Description
11 BRK_SIG_COND_EN	Break on signal posedge enable. This bit when set enables break on signal posedge. 0 Signal posedge break is disabled. 1 Signal posedge break is enabled.
10 BRK_COL_COND_EN	Break on column number enable. This bit when set enables break on channel number. 0 Column break is disabled. 1 Column break is enabled.
9 BRK_ROW_COND_EN	Break on row number enable. This bit when set enables break on channel number. 0 Row break is disabled. 1 Row break is enabled.
8 BRK_CHA_COND_EN	Break on channel number enable. This bit when set enables break on channel number. 0 Channel break is disabled. 1 Channel break is enabled.
7–6 BRK_RQ_MODE	Break request mode. This field selects the mode the break request as followed. 0 DMA stop 1 Freeze 2 Interrupt 3 RSV
5–4 DBG_ENTER_MODE	Debug entering mode. These bits defines the dependency between MCU and IPU regarding debug entering. DBG_ENTER_MODE[1]—defines the dependency of MCU on IPU. DBG_ENTER_MODE[0]—defines the dependency of IPU on MCU. Values: 00 IPU internal break force MCU to enter debug. MCU entering debug force IPU to enter debug. 01 IPU internal break force MCU to enter debug. MCU entering debug DOES NOT force IPU to enter debug. 10 IPU internal break DOES NOT force MCU to enter debug. MCU entering debug force IPU to enter debug. 11 IPU internal break DOES NOT force MCU to enter debug. MCU entering debug DOES NOT force IPU to enter debug.
3	Reserved
2 DBG_EXIT	Exit debug mode 0 Stay in debug mode. 1 Exit debug mode.
1 FRC_DBG	Force entering debug mode. This bit forces entering debug mode by the IPU. 0 Disable unconditional entering debug mode. 1 Enable unconditional entering debug mode.
0 BRK_EN	Break enable. This bit enables the internal break. 0 Internal break is disabled. 1 Internal break is enabled.

44.3.3.1.21 IPU Break Control Register 2 (IPU_BRK_CTRL_2)

This register is the second control register of debug unit in control module. It contain the following control bits for debug unit:

0x53FC_0054 (IPU_BRK_CTRL_2) Access: User Read/Write

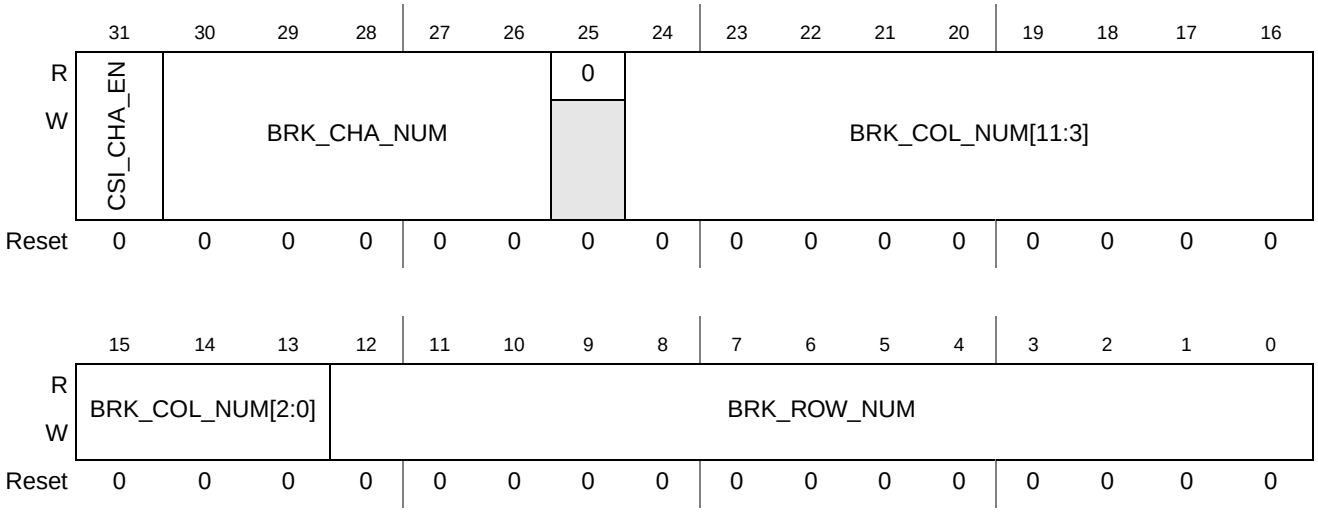


Figure 44-29. IPU Break Control Register 2 (IPU_BRK_CTRL_2)

Table 44-47. IPU_BRK_CTRL_2 Field Descriptions

Field	Description
31 CSI_CHA_EN	Enable CSI break condition. This bit enables break when CSI output frame column and row numbers matches the selected values. 0 enable CSI break condition. 1 enable CSI break condition (DMA break conditions are ignored)
30–26 BRK_CHA_NUM	DMA channel number for break condition. This field is programmed by user to break on a specific channel number. The number starts from 0.
25	Reserved
24–16 BRK_COL_NUM[11:3]	Frame column number for break condition. This field is programmed by user to break on a specific column number. The number starts from 0.
15–13 BRK_COL_NUM[2:0]	
12–0 BRK_ROW_NUM	Frame row number for break condition. This field is programmed by user to break on a specific row number. The number starts from 0.

44.3.3.1.22 IPU Break Status Register (IPU_BRK_STAT)

0x53FC_0058 (IPU_BRK_STAT) Access: User Read-Only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	BRK_SRC	MCU_DBGRQ	IPU_BREAK_ACK
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-30. IPU Break Status Register (IPU_BRK_STAT)

Table 44-48. IPU_BRK_STAT Field Descriptions

Field	Description
31–3	Reserved
2 BRK_SRC	IPU break acknowledge. This bit indicates that IPU is in break. 0 IPU is out of break mode. 1 IPU is in break mode.
1 MCU_DBGRQ	MCU debug request. IPU requests MCU to enter into debug mode thought this bit. 0 debug request from IPU to MCU is negated. 1 debug request from IPU to MCU is asserted.
0 IPU_BREAK_ACK	Break source. This bit indicates whether break is caused by internal or external source. 0 external source 1 internal source

44.3.3.1.23 IPU Diagnostic Bus Control Register (IPU_DIAGB_CTRL)

This register controls the Diagnostic Bus configuration.

0x53FC_005C (IPU_DIAGB_CTRL) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R												MON_GRP_SEL				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-31. IPU Diagnostic Bus Control Register (IPU_DIAGB_CTRL)

Table 44-49. IPU_DIAGB_CTRL Field Descriptions

Field	Description
31–5	Reserved
4–0 MON_GRP_SEL	Select group of signals to be output via the Diagnostic Bus.

Table 44-50. Diagnostic Bus Groups

MON_GRP_SEL	Group Description	Pin Number	Description
00000	ADC monitoring signals	0	ADC busy
		1	SYS1 channel FIFO full
		2	SYS1 channel FIFO empty
		3	SYS2 channel FIFO full
		4	SYS2 channel FIFO empty
		5	CMND1 channel FIFO full
		6	CMND1 channel FIFO empty
		7	CMND2 channel FIFO full
		8	CMND2 channel FIFO empty
		9	PRP channel FIFO full
		10	PRP channel FIFO empty
		11	PP channel FIFO full
		12	PP channel FIFO empty
		13	MCU write channel FIFO full
		14	MCU write channel FIFO empty
		15	Read strobe of template word
		16	Template memory address bit 0
		17	Template memory address bit 1
		18	Template memory address bit 2
		19	Template memory address bit 3
		20	Template memory address bit 4
		21	Template memory address bit 5
		22	Template memory address bit 6
		23	Template memory address bit 7
		24	New frame start for PP channel
		25	New frame start for SYS1 channel
		26	New frame start for SYS2 channel
		27	Memory access wait signal for MCU
		28	Memory access wait signal for internal source/destination
		31:29	Current channel: 000 - PRP, 001 - PP, 010- SYS1, 011 - SYS2, 100 - CMND1, 101 - CMND2, 110 - MCU write, 111 - MCU read

Table 44-50. Diagnostic Bus Groups (continued)

MON_GRP_SEL	Group Description	Pin Number	Description
00001	CSI monitoring group	0	Frame vertical synchronization
		5:1	Internal state machine states
		6	CSI end of line
		7	CSI end of frame
		8	CSI new frame
		9	CSI new frame for ADC
		10	Field number in interlace mode
		11	Current frame is skipped by encoder task
		12	Current frame is skipped by viewfinder task
		13	Frame horizontal synchronization
		31:14	Reserved

Table 44-50. Diagnostic Bus Groups (continued)

MON_GRP_SEL	Group Description	Pin Number	Description
00010	DI monitoring group	0	Load parallel data from display
		1	Load serial data from display
		2	LLA lock bit for parallel interface
		3	LLA lock bit for serial interface
		6:4	Parallel interface state 0- idle, 1- SDC write, 2- LLA write, 3- LLA read, 4- ADC write, 5- ADC read, 6- wait for finish ADC serial
		9:7	Serial interface state 0- idle, 1—ADC write, 2—ADC read, 3—LLA write, 4—LLA read
		10	DI idle state
		11	LLA serial finish
		12	Indicator that SDC will send data one row after
		13	ADC lock for parallel interface
		14	Pixel clock enable for display 0
		15	Pixel clock enable for display 1 and 2
		16	Pixel clock enable for display 3
		17	Display clock enable for display 0
		18	Display clock enable for display 1
		19	Display clock enable for display 2
		20	Display clock enable for display 3
		21	Window mask for display 3
		22	LLA request
		23	LLA write/read select
		25:24	LLA display number
		26	LLA data/command mapping
		27	ADC request
		28	ADC write/read select
		30:29	ADC display number
		31	ADC data/command mapping

Table 44-50. Diagnostic Bus Groups (continued)

MON_GRP_SEL	Group Description	Pin Number	Description
00011	IC monitoring group	0	Task 2 graphics buffer empty
		1	Task 3 graphics buffer empty
		2	Task 1 resizing input buffer not empty
		3	Task 2 resizing input buffer not empty
		4	Task 3 resizing input buffer not empty
		5	Task 1 resizing output buffer full
		6	Task 2 resizing output buffer full
		7	Task 3 resizing output buffer full
		8	Task 1 downsizing output buffer full
		9	Task 2 downsizing output buffer full
		10	Task 3 downsizing output buffer full
		11	Task 1 downsizing out buffer not empty
		12	Task 2 downsizing output buffer not empty
		13	Task 3 downsizing output buffer not empty
		14	Current resizing output buffer full
		15	Current resizing input buffer full
		16	Resizing task number [0
		17	Resizing task number [1
		18	Downsizing task number [0]
		19	Downsizing task number [1]
		20	Rotation task number [0]
		21	Rotation task number [1]
		22	Task 1 end of line
		23	Task 2 end of line
		24	Task 3 end of line
		25	Task 1 downsizing new frame
		26	Task 2 downsizing new frame
		27	Task 3 downsizing new frame
		28	Task 1 resizing new frame
		29	Task 2 resizing new frame
		30	Task 3 resizing new frame
		31	Reserved

Table 44-50. Diagnostic Bus Groups (continued)

MON_GRP_SEL	Group Description	Pin Number	Description
00100	PF monitoring group	6:0	PF state machine state
		9:7	X position status
		11:10	Y position status
		13:12	Current color component
		14	First column indicator
		15	First row indicator
		16	Input buffer ready
		17	Parameters buffer ready
		18	BS buffer ready
		19	New input buffer request
		20	New parameters buffer request
		21	New BS buffer request
		22	Parameters FIFO not full
		23	Bs FIFO not full
		24	Y input FIFO not full
		25	U input FIFO not full
		26	V input FIFO not full
		27	Y output FIFO not empty
		28	U output FIFO not empty
		29	V output FIFO not empty
		31:30	Reserved

Table 44-50. Diagnostic Bus Groups (continued)

MON_GRP_SEL	Group Description	Pin Number	Description
00101	SDC monitoring group	0	Cursor blinking control
		1	Cursor enable window including blinking
		2	FG enable window
		3	BG enable window
		4	BG FIFO empty
		5	FG FIFO empty
		6	Mask FIFO empty
		7	BG FIFO full
		8	FG FIFO full
		9	Mask FIFO full
		10	Screen last row
		11	Screen row last pixel
		12	Clear vertical and horizontal counters (from ADC)
		13	VSYNC
		14	HSYNC
		15	Indicator that SDC will send data one row after
		16	SPL, SHARP signal
		17	PS, SHARP signal
		18	REV, SHARP signal
		19	CLS, SHARP signal
		20	Even field indicator in TV mode
		21	Pixel clock enable
		22	BG end of frame
		23	FG end of frame
		24	Mask end of frame
		25	BG end of line
		26	FG end of line
		27	Mask end of line
		28	BG data delayed
		29	FG data delayed
		30	Mask data delayed
		31	Mask enable window

Table 44-50. Diagnostic Bus Groups (continued)

MON_GRP_SEL	Group Description	Pin Number	Description
00110	IDMAC channels end of transfer monitoring group	0	DMAIC_0 channel end of transfer
		1	DMAIC_1 channel end of transfer
		2	DMAIC_2 channel end of transfer
		3	DMAIC_3 channel end of transfer
		4	DMAIC_4 channel end of transfer
		5	DMAIC_5 channel end of transfer
		6	DMAIC_6 channel end of transfer
		7	DMAIC_7 channel end of transfer
		8	DMAIC_8 channel end of transfer
		9	DMAIC_9 channel end of transfer
		10	DMAIC_10 channel end of transfer
		11	DMAIC_11 channel end of transfer
		12	DMAIC_12 channel end of transfer
		13	DMAIC_13 channel end of transfer
		14	DMASDC_0 channel end of transfer
		15	DMASDC_1 channel end of transfer
		16	DMASDC_2 channel end of transfer
		17	DMASDC_3 channel end of transfer
		18	DMAADC_2 channel end of transfer
		19	DMAADC_3 channel end of transfer
		20	DMAADC_4 channel end of transfer
		21	DMAADC_5 channel end of transfer
		11	DMAADC_6 channel end of transfer
		23	DMAADC_7 channel end of transfer
		24	DMAPF_0 channel end of transfer
		25	DMAPF_1 channel end of transfer
		26	DMAPF_2 channel end of transfer
		27	DMAPF_3 channel end of transfer
		28	DMAPF_4 channel end of transfer
		29	DMAPF_5 channel end of transfer
		30	DMAPF_6 channel end of transfer
		31	DMAPF_7 channel end of transfer

Table 44-50. Diagnostic Bus Groups (continued)

MON_GRP_SEL	Group Description	Pin Number	Description
00111	IDMAC channels end of line monitoring group	0	DMAIC_0 channel end of line
		1	DMAIC_1 channel end of line
		2	DMAIC_2 channel end of line
		3	DMAIC_3 channel end of line
		4	DMAIC_4 channel end of line
		5	DMAIC_5 channel end of line
		6	DMAIC_6 channel end of line
		7	DMAIC_7 channel end of line
		8	DMAIC_8 channel end of line
		9	DMAIC_9 channel end of line
		10	DMAIC_10 channel end of line
		11	DMAIC_11 channel end of line
		12	DMAIC_12 channel end of line
		13	DMAIC_13 channel end of line
		14	DMASDC_0 channel end of line
		15	DMASDC_1 channel end of line
		16	DMASDC_2 channel end of line
		17	DMASDC_3 channel end of line
		18	DMAADC_2 channel end of line
		19	DMAADC_3 channel end of line
		20	DMAADC_4 channel end of line
		21	DMAADC_5 channel end of line
		11	DMAADC_6 channel end of line
		23	DMAADC_7 channel end of line
		24	DMAPF_0 channel end of line
		25	DMAPF_1 channel end of line
		26	DMAPF_2 channel end of line
		27	DMAPF_3 channel end of line
		28	DMAPF_4 channel end of line
		29	DMAPF_5 channel end of line
		30	DMAPF_6 channel end of line
		31	DMAPF_7 channel end of line

Table 44-50. Diagnostic Bus Groups (continued)

MON_GRP_SEL	Group Description	Pin Number	Description
01000	IDMAC channels last burst in frame monitoring group	0	DMAIC_0 Last burst in frame
		1	DMAIC_1 Last burst in frame
		2	DMAIC_2 Last burst in frame
		3	DMAIC_3 Last burst in frame
		4	DMAIC_4 Last burst in frame
		5	DMAIC_5 Last burst in frame
		6	DMAIC_6 Last burst in frame
		7	DMAIC_7 Last burst in frame
		8	DMAIC_8 Last burst in frame
		9	DMAIC_9 Last burst in frame
		10	DMAIC_10 Last burst in frame
		11	DMAIC_11 Last burst in frame
		12	DMAIC_12 Last burst in frame
		13	DMAIC_13 Last burst in frame
		14	DMASDC_0 Last burst in frame
		15	DMASDC_1 Last burst in frame
		16	DMASDC_2 Last burst in frame
		17	DMASDC_3 Last burst in frame
		18	DMAADC_2 Last burst in frame
		19	DMAADC_3 Last burst in frame
		20	DMAIC_10 Last burst in frame
		21	DMAADC_5 Last burst in frame
		11	DMAADC_6 Last burst in frame
		23	DMAADC_7 Last burst in frame
		24	DMAPF_0 Last burst in frame
		25	DMAPF_1 Last burst in frame
		26	DMAPF_2 Last burst in frame
		27	DMAPF_3 Last burst in frame
		28	DMAPF_4 Last burst in frame
		29	DMAPF_5 Last burst in frame
		30	DMAPF_6 Last burst in frame
		31	DMAPF_7 Last burst in frame

Table 44-50. Diagnostic Bus Groups (continued)

MON_GRP_SEL	Group Description	Pin Number	Description
01001	IDMAC channels new frame acknowledge monitoring group	0	DMAIC_0 new frame acknowledge
		1	DMAIC_1 new frame acknowledge
		2	DMAIC_2 new frame acknowledge
		3	DMAIC_3 new frame acknowledge
		4	DMAIC_4 new frame acknowledge
		5	DMAIC_5 new frame acknowledge
		6	DMAIC_6 new frame acknowledge
		7	DMAIC_7 new frame acknowledge
		8	DMAIC_8 new frame acknowledge
		9	DMAIC_9 new frame acknowledge
		10	DMAIC_0 new frame acknowledge
		11	DMAIC_11 new frame acknowledge
		12	DMAIC_12 new frame acknowledge
		13	DMAIC_13 new frame acknowledge
		14	DMASDC_0 new frame acknowledge
		15	DMASDC_1 new frame acknowledge
		16	DMASDC_2 new frame acknowledge
		17	DMASDC_3 new frame acknowledge
		18	DMAADC_2 new frame acknowledge
		19	DMAADC_3 new frame acknowledge
		20	DMAADC_4 new frame acknowledge
		21	DMAADC_5 new frame acknowledge
		11	DMAADC_6 new frame acknowledge
		23	DMAADC_7 new frame acknowledge
		24	DMAPF_0 new frame acknowledge
		25	DMAPF_1 new frame acknowledge
		26	DMAPF_2 new frame acknowledge
		27	DMAPF_3 new frame acknowledge
		28	DMAPF_4 new frame acknowledge
		29	DMAPF_5 new frame acknowledge
		30	DMAPF_6 new frame acknowledge
		31	DMAPF_7 new frame acknowledge

Table 44-50. Diagnostic Bus Groups (continued)

MON_GRP_SEL	Group Description	Pin Number	Description
01010	IDMAC monitoring group	4:0	Current channel
		9:5	Highest priority channel
		10	Next address calculation finish
		11	Inner channel 0 AHB status bit
		12	Inner channel 0 IBI status bit
		13	Inner channel 1 AHB status bit
		14	Inner channel 1 IBI status bit
		15	Transfer complete indicator for AHB 0
		16	Transfer complete indicator for AHB 1
		17	Transfer complete indicator for IBI
		18	Master AHB 0 is in IDLE
		19	Master AHB 1 is in IDLE
		20	IBI state machine is in IDLE
		21	Memory/parameter main pointer
		22	Inner channel 0 read indicator
		23	Inner channel 1 read indicator
		31:24	Reserved
01011	IPU_CHA_BUF0_RDY register monitoring group	31:0	IPU_CHA_BUF0_RDY register bits
01100	IPU_CHA_BUF1_RDY register monitoring group	31:0	IPU_CHA_BUF1_RDY register bits
01101	IPU_CHA_CUR_BUF register monitoring group	31:0	IPU_CHA_CUR_BUF register bits
01110	IPU_TASKS_STAT register monitoring group	31:0	IPU_TASKS_STAT register bits
01111	IPU_INT_STAT_3 register monitoring group	31:0	IPU_INT_STAT_3 register bits
10000	IPU_INT_STAT_4 register monitoring group	31:0	IPU_INT_STAT_4 register bits
10001	IPU_INT_STAT_5 register monitoring group	31:0	IPU_INT_STAT_5 register bits
10010	IDMAC_CHA_BUSY register monitoring group	31:0	IDMAC_CHA_BUSY register bits

44.3.3.2 CSI Registers

44.3.3.2.1 CSI Sensor Configuration Register (CSI_SENS_CONF)

This register controls the sensor configuration.

0x53FC_0060 (CSI_SENS_CONF) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	DIV_RATIO							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	EXT_VSYNC	0	0	0	DATA_WITDH		SENS_DATA_FORMAT		SENS_CLK_SRC	0	SENS_PRTCL		SENS_PIX_CLK_POL	DATA_POL	HSYNC_POL	VSYNC_POL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-32. CSI Sensor Configuration Register (CSI_SENS_CONF)

Table 44-51. CSI_SENS_CONF Field Descriptions

Field	Description
31–24	Reserved
23–16 DIV_RATIO	Clock division ratio minus 1. This field defines the division ratio of HSP_CLK into SENSB_MCLK: SENSB_MCLK rate = HSP_CLK rate /(DIV_RATIO+1)
15 EXT_VSYNC	External VSYNC enable. This bits select between external and internal VSYNC. 0 Internal VSYNC mode. 1 External VSYNC mode.
14–12	Reserved
11–10 DATA_WIDTH	Data width. This fields defines the number of bits per color. Values: 00 two 4-bit words per color 01 8 bits per color 10 10 bits per color 11 15 bits Bayer or generic data

Table 44-51. CSI_SENS_CONF Field Descriptions (continued)

Field	Description
9–8 SENS_DATA_FORMAT	Data format from the sensor. This field defines the data format for the input of the CSI sensor. Values: 00 RGB or YUV444 01 Reserved 10 YUV422 (UYVY...) 11 Bayer or generic data
7 SENS_CLK_SRC	Sensor clock source. This bit defines the clock source input 0 SENSB_SENS_CLK clock. 1 HSP_CLK clock after division.
6	Reserved
5–4 SENS_PRTCL	Sensor protocol. This bit defines the sensor timing/data mode protocol. Values: 00 Gated clock mode 01 Non-gated clock mode 10 CCIR progressive mode 11 CCIR interlaced mode
3 SENS_PIX_CLK_POL	Invert pixel clock input. This bit selects the polarity of pixel clock. 0 pixel clock is directly applied to internal circuitry. 1 pixel clock is inverted before applied to internal circuitry.
2 DATA_POL	Invert data input. This bit selects the polarity of data input. 0 data lines are directly applied to internal circuitry. 1 data lines are inverted before applied to internal circuitry.
1 HSYNC_POL	Invert IPP_IND_SENSB_HSYNC input. This bit selects the polarity of IPP_IND_SENSB_HSYNC signal. 0 IPP_IND_SENSB_HSYNC is directly applied to internal circuitry. 1 IPP_IND_SENSB_HSYNC is inverted before applied to internal circuitry.
0 VSYNC_POL	Invert IPP_IND_SENSB_VSYNC input. This bit selects the polarity of IPP_IND_SENSB_VSYNC signal. 0 IPP_IND_SENSB_VSYNC is not inverted before applied to internal circuitry. 1 IPP_IND_SENSB_VSYNC is inverted before applied to internal circuitry.

44.3.3.2.2 CSI Sensor Frame Size Register (CSI_SENS_FRM_SIZE)

This register controls the sensor frame size.

0x53FC_0064 (CSI_SENS_FRM_SIZE) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	SENS_FRM_HEIGHT											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	SENS_FRM_WIDTH											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-33. CSI Sensor Frame Size Register (CSI_SENS_FRM_SIZE)

Table 44-52. CSI_SENS_FRM_SIZE Field Descriptions

Field	Description
31–28	Reserved
27–16 SENS_FRM_HEIGHT	Sensor frame height minus 1. This field defines the sensor frame rows number minus 1.
15–12	Reserved
11–0 SENS_FRM_WIDTH	Sensor frame width minus 1. This field defines the sensor frame column number minus 1.

44.3.3.2.3 CSI Actual Frame Size Register (CSI_ACT_FRM_SIZE)

This register controls the actual frame size.

0x53FC_0068 (CSI_ACT_FRM_SIZE) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	ACT_FRM_HEIGHT											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	ACT_FRM_WIDTH											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-34. CSI Actual Frame Size Register (CSI_ACT_FRM_SIZE)

Table 44-53. CSI_ACT_FRM_SIZE Field Descriptions

Field	Description
31–28	Reserved
27–16 ACT_FRM_HEIGHT	Actual frame height minus 1. This field defines the CSI output frame rows number minus 1.
15–12	Reserved
11–0 ACT_FRM_WIDTH	Actual frame width minus 1. This field defines the CSI output frame columns number minus 1.

44.3.3.2.4 CSI Output Frame Control Register (CSI_OUT_FRM_CTRL)

This register controls the output frame parameters.

0x53FC_006C (CSI_OUT_FRM_CTRL) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	HORZ_DWNS	VERT_DWNS	0	IC_TV_MODE	SKIP_VF						SKIP_ENC			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HSC								VSC							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-35. CSI Output Frame Control Register (CSI_OUT_FRM_CTRL)

Table 44-54. CSI_OUT_FRM_CTRL Field Descriptions

Field	Description
31–30	Reserved
29 HORZ_DWNS	Enable horizontal downsizing (decimation) by 2. 0 Downsizing disabled 1 Downsizing enabled
28 VERT_DWNS	Enable vertical downsizing (decimation) by 2. 0 Downsizing disabled 1 Downsizing enabled
27	Reserved

Table 44-54. CSI_OUT_FRM_CTRL Field Descriptions (continued)

Field	Description
26 IC_TV_MODE	Convert interlaced frame format to progressive frame format when writing to the system memory. 0 Disable format conversion 1 Enable format conversion
25–21 SKIP_VF	Skip viewfinder frame. This field initiated by MCU, is shifted each new frame and according to the bits, frames are skipped for view-finder task—each bit meaning is: 0 Frame for view-finder is valid. 1 Frame for view-finder is skipped.
20–16 SKIP_ENC	Skip encoder frame. This field initiated by MCU, is shifted each new frame and according to the bits, frames are skipped for encoding task—each bit meaning is: 0 Frame for encoding is valid. 1 Frame for encoding is skipped.
15–8 HSC	Horizontal skip. This field defines the number of columns to skip.
7–0 VSC	Vertical skip. This field defines the number of rows to skip.

44.3.3.2.5 CSI Test Control Register (CSI_TST_CTRL)

This register controls the sensor pattern generating in test mode.

0x53FC_0070 (CSI_TST_CTRL) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	TEST_GEN_MODE	PG_B_VALUE							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PG_G_VALUE								PG_R_VALUE							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-36. CSI Test Control Register (CSI_TST_CTRL)

Table 44-55. CSI_TST_CTRL Field Descriptions

Field	Description
31–25	Reserved
24 TEST_GEN_MODE	Test generator mode. This bit activates the signal generation. 0 Test signal generator is inactive. 1 Test signal generator is active.
23–16 PG_B_VALUE	Pattern generator B value. This field selects the B value for the generated pattern of even pixel.
15–8 PG_G_VALUE	Pattern generator G value. This field selects the G value for the generated pattern of even pixel.
7–0 PG_R_VALUE	Pattern generator R value. This field selects the R value for the generated pattern of even pixel.

44.3.3.2.6 CSI CCIR Code Register 1 (CSI_CCIR_CODE_1)

This register controls the field 0 commands configuration.

0x53FC_0074 (CSI_CCIR_CODE_1) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	CCIR_ERR_DET_EN	0	0	STRT_FLD0_ACTV			END_FLD0_ACTV		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	STRT_FLD0_BLNK_2ND			END_FLD0_BLNK_2ND			STRT_FLD0_BLNK_1ST			END_FLD0_BLNK_1ST		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-37. CSI CCIR Code Register 1 (CSI_CCIR_CODE_1)

Table 44-56. CSI_CCIR_CODE_1 Field Descriptions

Field	Description
31–25	Reserved
24 CCIR_ERR_DET_EN	Enable error detection and correction for CCIR interlaced mode with protection bit. 0 Error detection and correction is disabled. 1 Error detection and correction is enabled.
31–25	Reserved

Table 44-56. CSI_CCIR_CODE_1 Field Descriptions (continued)

Field	Description
21–19 STRT_FLD0_ACTV	Start of field 0 active line command (interlaces mode). (In progressive mode, start of active line command mode).
18–16 END_FLD0_ACTV	End of field 0 active line command (interlaces mode). (In progressive mode, end of active line command mode).
15–12	Reserved
11–9 STRT_FLD0_BLNK_2ND	Start of field 0 second blanking line command (interlaces mode). (In progressive mode this field is ignored).
8–6 END_FLD0_BLNK_2ND	End of field 0 second blanking line command (interlaces mode). (In progressive mode this field is ignored).
5–3 STRT_FLD0_BLNK_1ST	Start of field 0 first blanking line command (interlaces mode). (In progressive mode this field indicates start of blanking line command).
2–0 END_FLD0_BLNK_1ST	End of field 0 first blanking line command (interlaces mode). (In progressive mode this field is ignored).

44.3.3.2.7 CSI CCIR Code Register 2 (CSI_CCIR_CODE_2)

This register controls the field 1 commands configuration.

0x53FC_0078 (CSI_CCIR_CODE_2) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	STRT_FLD1_ACTV			END_FLD1_ACTV		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	STRT_FLD1_BLNK_2ND			END_FLD1_BLNK_2ND			STRT_FLD1_BLNK_1ST			END_FLD1_BLNK_1ST		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-38. CSI CCIR Code Register 2 (CSI_CCIR_CODE_2)

Table 44-57. CSI_CCIR_CODE_2 Field Descriptions

Field	Description
31–22	Reserved
21–19 STRT_FLD1_ACTV	Start of field 1 active line command (interlaces mode). (In progressive mode this field is ignored).
18–16 END_FLD1_ACTV	End of field 1 active line command (interlaces mode). (In progressive mode this field is ignored).

Table 44-57. CSI_CCIR_CODE_2 Field Descriptions (continued)

Field	Description
15–12	Reserved
11–9 STRT_FLD1_BLNK_2ND	Start of field 1 second blanking line command (interlaces mode). (In progressive mode this field is ignored).
8–6 END_FLD1_BLNK_2ND	End of field 1 second blanking line command (interlaces mode). (In progressive mode this field is ignored).
5–3 STRT_FLD1_BLNK_1ST	Start of field 1 first blanking line command (interlaces mode). (In progressive mode this field is ignored).
2–0 END_FLD1_BLNK_1ST	End of field 1 first blanking line command (interlaces mode). (In progressive mode this field is ignored).

44.3.3.2.8 CSI CCIR Code Register 3 (CSI_CCIR_CODE_3)

This register controls the CCIR pre-command sequence.

0x53FC_007C (CSI_CCIR_CODE_3) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	CCIR_PRECOM[23:16]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CCIR_PRECOM[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-39. CSI CCIR Code Register 3 (CSI_CCIR_CODE_3)

Table 44-58. CSI_CCIR_CODE_3 Field Descriptions

Field	Description
31–24	Reserved
23–16 CCIR_PRECOM[23:16]	CCIR pre command. This field defines the sequence which comes before the CCIR command.
15–0 CCIR_PRECOM[15:0]	CCIR pre command. This field defines the sequence which comes before the CCIR command.

44.3.3.2.9 CSI Flash Strobe Register 1 (CSI_FLASH_STROBE_1)

This register controls the flash strobe generated by the CSI.

0x53FC_0080 (CSI_FLASH_STROBE_1) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SENS_ROW_DURATION															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CLOCK_SEL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-40. CSI Flash Strobe Register 1 (CSI_FLASH_STROBE_1)

Table 44-59. CSI_FLASH_STROBE_1 Field Descriptions

Field	Description
31–16 SENS_ROW_DURATION	Duration of sensor row minus 1. This field specifies scan duration for one sensor row expressed in SENSB_MCLK or SENSB_PIX_CLK periods. This value is used as an unit for definition of flash strobe start time and duration.
15–1	Reserved
0 CLOCK_SEL	Select clock for sensor row duration count. This bit selects SENSB_MCLK or SENSB_PIX_CLK clock for sensor row duration count. 0 Select SENSB_MCLK clock 1 Select SENSB_PIX_CLK clock

44.3.3.2.10 CSI Flash Strobe Register 2 (CSI_FLASH_STROBE_2)

This register controls the flash strobe generated by the CSI.

0x53FC_0084 (CSI_FLASH_STROBE_2) Access: User Read/Write

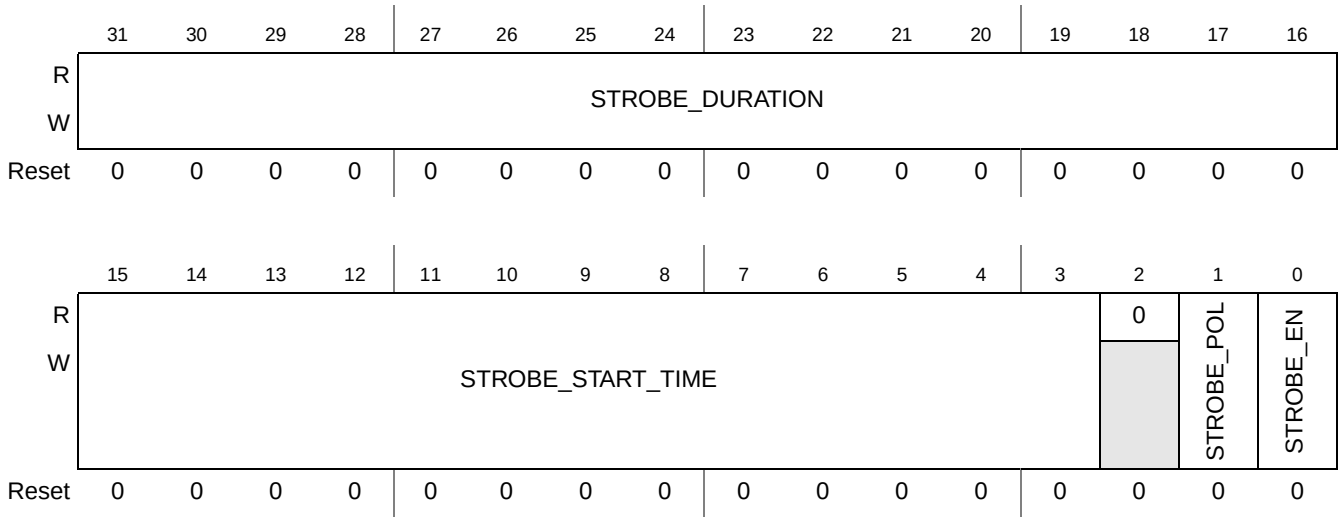


Figure 44-41. CSI Flash Strobe Register 2 (CSI_FLASH_STROBE_2)

Table 44-60. CSI_FLASH_STROBE_2 Field Descriptions

Field	Description
31–16 STROBE_DURATION	Strobe duration minus 1. This field defines flash strobe duration. The value is expressed in sensor image rows.
15–3 STROBE_START_TIME	Strobe start time minus 1. This field defines a time interval between beginning of the next sensor frame and flash strobe generation start. The value is expressed in sensor image rows. The minimal allowed value is 1.
2	Reserved
1 STROBE_POL	Strobe polarity control bit. This bit controls flash strobe polarity. 0 active low 1 active high
0 STROBE_EN	Strobe enable control/status bit. This bit enables flash strobe generation after beginning of the next sensor frame. Beginning of the sensor frame is defined as the first row start point. This bit should be set by the MCU. The IPU automatically clears the bit after strobe generation finish. 0 flash strobe disabled 1 flash strobe enabled

44.3.3.3 IC Registers

44.3.3.3.1 IC Configuration Register (IC_CONF)

This register contains control parameter for IC 3 tasks (pre-processing for encoding, pre-processing for view-finder and post processing).

0x53FC_0088 (IC_CONF) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CSI_MEM_WR_EN	RWS_EN	IC_KEY_COLOR_EN	IC_GLB_LOC_A	0	0	0	0	0	0	0	PP_ROT_EN	PP_CMB	PP_CSC2	PP_CSC1	PP_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	PRPVF_ROT_EN	PRPVF_CMB	PRPVF_CSC2	PRPVF_CSC1	PRPVF_EN	0	0	0	0	0	PRPENC_ROT_EN	PRPENC_CSC1	PRPENC_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-42. IC Configuration Register (IC_CONF)

Table 44-61. IC_CONF Field Descriptions

Field	Description
31 CSI_MEM_WR_EN	CSI direct memory write enable. This bit enables writing data from sensor directly to memory even when a raw sensor is not attached. 0 CSI direct writing to memory is disabled. 1 CSI direct writing to memory is enabled.
30 RWS_EN	Raw sensor enable. This bit indicate if a raw sensor is attached (bayer format). 0 Raw sensor is not attached. 1 Raw sensor is attached. This bit is used together with the CSI_MEM_WR_EN bit as follows: CSI_MEM_WR_EN=0, RWS_EN=0—data is fed from the CSI to the IC for processing; CSI_MEM_WR_EN=1, RWS_EN=0—data is fed from the CSI to the IC for processing and also for writing to the system memory; CSI_MEM_WR_EN=0, RWS_EN=1—data is fed from the CSI to the system memory (via the IC) and from the system memory to the IC for processing; CSI_MEM_WR_EN=1, RWS_EN=1—non-valid configuration.
29 IC_KEY_COLOR_EN	Key color enable. This bit enables the key color feature. 0 Key color is disabled. 1 Key color is enabled.

Table 44-61. IC_CONF Field Descriptions (continued)

Field	Description
28 IC_GLB_LOC_A	Global alpha. This bit select the source of alpha parameter. 0 Alpha parameter is local. 1 Alpha parameter is global.
27–21	Reserved
20 PP_ROT_EN	Post-processing rotation task enable. This bit enable post-processing rotation task. 0 Rotation is disabled. 1 Rotation is enabled.
19 PP_CMB	Post-processing task combining enable. This bit enables combining. Combining can be done only when first CSC is enabled (PP_CSC1=1). Otherwise the bit has no effect. 0 Combining is disabled. 1 Combining is enabled.
18 PP_CSC2	Post-processing task color conversion RGB-->YUV enable. This bit enables YUV-->RGB. 0 RGB-->YUV is disabled. 1 RGB-->YUV is enabled.
17 PP_CSC1	Post-processing task color conversion YUV-->RGB enable. This bit enables YUV-->RGB. 0 YUV-->RGB is disabled. 1 YUV-->RGB is enabled.
16 PP_EN	Post-processing task enable. This bit enables the post-processing task. 0 Task is disabled. 1 Task is enabled.
15–13	Reserved
12 PRPVF_ROT_EN	Preprocessing rotation task for viewfinder enable. This bit enable preprocessing rotation task for viewfinder. 0 Rotation is disabled. 1 Rotation is enabled.
11 PRPVF_CMB	Preprocessing task for view-finder combining enable. This bit enables combining. Combining can be done only when first CSC is enabled (PRPVF_CSC1=1). Otherwise the bit has no effect. 0 Combining is disabled. 1 Combining is enabled.
10 PRPVF_CSC2	Pre-processing task for view-finder second color conversion enable. This bit enables second color conversion. 0 Second color conversion is disabled. 1 Second color conversion is enabled.
9 PRPVF_CSC1	Pre-processing task for view-finder first color conversion enable. This bit enables first color conversion. 0 First color conversion is disabled. 1 First color conversion is enabled.
8 PRPVF_EN	Preprocessing task for view-finder enable. This bit enables the view-finder task. 0 Task is disabled. 1 Task is enabled.

Table 44-61. IC_CONF Field Descriptions (continued)

Field	Description
7–3	Reserved
2 PRPENC_ROT_EN	Preprocessing rotation task for encoding enable. This bit enable preprocessing rotation task for encoding. 0 Rotation is disabled. 1 Rotation is enabled.
1 PRPENC_CSC1	Preprocessing task for encoding color conversion enable. This bit enables color conversion. 0 Color conversion is disabled. 1 Color conversion is enabled.
0 PRPENC_EN	Preprocessing task for encoding enable. This bit enables the encoding task. 0 Task is disabled. 1 Task is enabled.

44.3.3.3.2 IC Preprocessing Encoder Resizing Coefficients Register (IC_PRP_ENC_RSC)

This register contains the resizing and downsizing parameters for preprocessing task for encoding.

0x53FC_008C (IC_PRP_ENC_RSC) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PRPENC_DS_R_V				PRPENC_RS_R_V											
Reset	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PRPENC_DS_R_H				PRPENC_RS_R_H											
Reset	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-43. IC Preprocessing Encoder Resizing Coefficients Register (IC_PRP_ENC_RSC)

Table 44-62. IC_PRP_ENC_RSC Field Descriptions

Field	Description
31–30 PRPENC_DS_R_V	Preprocessing task for encoding downsizing vertical ratio. This field contains the downsizing vertical coefficient of preprocessing for encoding.
29–16 PRPENC_RS_R_V	Preprocessing task for encoding resizing vertical ratio. This field contains the resizing vertical coefficient of preprocessing for encoding. Resizing ratio is equal to PRPENC_RS_R_V: M Where M = 2^13; SI—input size; SO—output size PRPENC_RS_R_V = floor(M*(SI-1)/(SO-1));

Table 44-62. IC_PRP_ENC_RSC Field Descriptions (continued)

Field	Description
15–14 PRPENC_DS_R_H	Preprocessing task for encoding downsizing horizontal ratio. This field contains the downsizing horizontal coefficient of preprocessing for encoding. Values: 00 1 01 2 10 4 11 RSV
13–0 PRPENC_RS_R_H	Preprocessing task for encoding resizing horizontal ratio. This field contains the resizing horizontal coefficient of preprocessing for encoding. Resizing ratio is equal to PRPENC_RS_R_H: M Where M = 2^13; SI—input size; SO—output size PRPENC_RS_R_H = floor(M*(SI-1)/(SO-1));

44.3.3.3.3 IC Preprocessing View-Finder Resizing Coefficients Register (IC_PRP_VF_RSC)

This register contains the resizing and downsizing parameters for preprocessing task for display.

0x53FC_0090 (IC_PRP_VF_RSC) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PRPVF_DS_R_V		PRPVF_RS_R_V													
Reset	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PRPVF_DS_R_H		PRPVF_RS_R_H													
Reset	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-44. IC Preprocessing View-Finder Resizing Coefficients Register (IC_PRP_VF_RSC)

Table 44-63. IC_PRP_VF_RSC Field Descriptions

Field	Description
31–30 PRPVF_DS_R_V	Preprocessing task for encoding downsizing vertical ratio. This field contains the downsizing vertical coefficient of preprocessing for view-finder.
29–16 PRPVF_RS_R_V	Preprocessing task for encoding resizing vertical ratio. This field contains the resizing vertical coefficient of preprocessing for view-finder. Resizing ratio is equal to PRPVF_RS_R_V: M Where M = 2^13; SI—input size; SO—output size PRPVF_RS_R_V = floor(M*(SI-1)/(SO-1);

Table 44-63. IC_PRP_VF_RSC Field Descriptions (continued)

Field	Description
15–14 PRPVF_DS_R_H	Preprocessing task for encoding downsizing horizontal ratio. This field contains the downsizing horizontal coefficient of preprocessing for view-finder. Values: 00 1 01 2 10 4 11 RSV
13–0 PRPVF_RS_R_H	Preprocessing task for view-finding resizing horizontal ratio. This field contains the resizing horizontal coefficient of preprocessing task for view-finder. Resizing ratio is equal to PRPVF_RS_R_H: M Where M = 2^13; SI—input size; SO—output size PRPVF_RS_R_H = floor(M*(SI-1)/(SO-1));

44.3.3.3.4 IC Post-Processing Resizing Coefficients Register (IC_PP_RSC)

This register contains the resizing and downsizing parameters for post-processing task for display.

0x53FC_0094 (IC_PP_RSC) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PP_DS_R_V		PP_RS_R_V													
Reset	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PP_DS_R_H		PP_RS_R_H													
Reset	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-45. IC Post-Processing Resizing Coefficients Register (IC_PP_RSC)

Table 44-64. IC_PP_RSC Field Descriptions

Field	Description
31–30 PP_DS_R_V	Post-processing task downsizing vertical ratio. This field contains the downsizing vertical coefficient of post-processing.
29–16 PP_RS_R_V	Post-processing task resizing vertical ratio. This field contains the resizing vertical coefficient of post-processing. Resizing ratio is equal to PP_RS_R_V: M Where M = 2^13; SI—input size; SO—output size PP_RS_R_V = floor(M*(SI-1)/(SO-1));

Table 44-64. IC_PP_RSC Field Descriptions (continued)

Field	Description
15–14 PP_DS_R_H	Post-processing task downsizing horizontal ratio. This field contains the downsizing horizontal coefficient of post-processing. 00 1 01 2 10 4 11 RSV
13–0 PP_RS_R_H	Post-processing task resizing horizontal ratio. This field contains the resizing horizontal coefficient of post-processing. Resizing ratio is equal to PP_RS_R_H: M Where M = 2^13; SI—input size; SO—output size PP_RS_R_H = floor(M*(SI-1)/(SO-1));

44.3.3.3.5 IC Combining Parameters Register 1 (IC_CMBP_1)

0x53FC_0098 (IC_CMBP_1) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IC_PP_ALPHA_V								IC_PRPVF_ALPHA_V							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-46. IC Combining Parameters Register 1 (IC_CMBP_1)

Table 44-65. IC_CMBP_1 Field Descriptions

Field	Description
31–16	Reserved
15–8 IC_PP_ALPHA_V	Post-processing task global alpha. This field contains the global alpha value of post-processing
7–0 IC_PRPVF_ALPHA_V	Preprocessing task for encoding global alpha. This field contains the global alpha value of preprocessing for encoding.

44.3.3.3.6 IC Combining Parameters Register 2 (IC_CMBP_2)

0x53FC_009C (IC_CMBP_2) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	IC_KEY_COLOR_R							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IC_KEY_COLOR_G								IC_KEY_COLOR_B							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-47. IC Combining Parameters Register 2 (IC_CMBP_2)

Table 44-66. IC_CMBP_2 Field Descriptions

Field	Description
31–24	Reserved
23–16 IC_KEY_COLOR_R	Key color red.
15–8 IC_KEY_COLOR_G	Key color green.
7–0 IC_KEY_COLOR_B	Key color blue.

44.3.3.4 Post Filter (PF) Registers

44.3.3.4.1 Post Filter (PF) Configuration Register (PF_CONF)

This register contains all the parameters needed for the Post Filter.

0x53FC_00A0 (PF_CONF) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	H264_Y_PAUSE_ROW					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	H264_Y_PAUSE_EN	0	PF_TYPE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-48. PF Configuration Register (PF_CONF)

Table 44-67. PF_CONF Field Descriptions

Field	Description
31–22	Reserved
21–16 H264_Y_PAUSE_ROW	Number of the last row in the Y input frame to be processed before pause in H.264 mode. This field has no effect when H264_Y_PAUSE_EN=0. The number starts from 0.
15–5	Reserved
4 H264_Y_PAUSE_EN	Enable PF pause in H.264 mode during processing Y component. 0 pause disabled 1 pause enabled
3	Reserved
2–0 PF_TYPE	Post filtering type. This field contains the type of the Post Filter. Values: 000 All PF tasks are Disabled 001 MPEG-4 Deblocking only 010 MPEG-4 Deringing only 011 MPEG-4 Deblocking and Deringing 100 H.264 Deblocking

44.3.3.5 IDMAC Registers

44.3.3.5.1 IDMAC Configuration Register (IDMAC_CONF)

This register control the DMA channel configuration.

0x53FC_00A4 (IDMAC_CONF) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	SINGLE_AHB_M_EN	0	SRCNT			0	0	PRYM	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-49. IDMAC Configuration Register (IDMAC_CONF)

Table 44-68. IDMAC_CONF Field Descriptions

Field	Description
31–9	Reserved
8 SINGLE_AHB_M_EN	Enable single master AHB. Has no effect (always single master AHB) if the SINGLE_AHB_M_MODE pin is asserted to 1.
7	Reserved
6–4 SRCNT	Service request counter. This field determines how many times a specific request will be serviced before selecting another channel. Values: 000 1 consecutive request 001 2 consecutive requests 010 3 consecutive requests 011 4 consecutive requests 100 5 consecutive requests 101 6 consecutive requests 110 7 consecutive requests 111 8 consecutive requests
3–2	Reserved
1–0 PRYM	Priority mode. This field determines in what manner IDMAC will move the highest priority channel. 00 Round robin. 01 Random 10 Read after write 11 Reserved.

44.3.3.5.2 IDMAC Channel Enable Register (IDMAC_CHA_EN)

This register control the DMA channels enabling.

0x53FC_00A8 (IDMAC_CHA_EN) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DMAPF_7_CHAN_EN	DMAPF_6_CHAN_EN	DMAPF_5_CHAN_EN	DMAPF_4_CHAN_EN	DMAPF_3_CHAN_EN	DMAPF_2_CHAN_EN	DMAPF_1_CHAN_EN	DMAPF_0_CHAN_EN	DMAADC_7_CHAN_EN	DMAADC_6_CHAN_EN	DMAADC_5_CHAN_EN	DMAADC_4_CHAN_EN	DMAADC_3_CHAN_EN	DMAADC_2_CHAN_EN	DMASDC_3_CHAN_EN	DMASDC_2_CHAN_EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DMASDC_1_CHAN_EN	DMASDC_0_CHAN_EN	DMAIC_13_CHAN_EN	DMAIC_12_CHAN_EN	DMAIC_11_CHAN_EN	DMAIC_10_CHAN_EN	DMAIC_9_CHAN_EN	DMAIC_8_CHAN_EN	DMAIC_7_CHAN_EN	DMAIC_6_CHAN_EN	DMAIC_5_CHAN_EN	DMAIC_4_CHAN_EN	DMAIC_3_CHAN_EN	DMAIC_2_CHAN_EN	DMAIC_1_CHAN_EN	DMAIC_0_CHAN_EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-50. IDMAC Channel Enable Register (IDMAC_CHA_EN)

Table 44-69. IDMAC_CHA_EN Field Descriptions

Field	Description
DMACH#_EN[1]	DMA channel enable. (#: 0-31). This field selects if channel # is enabled 0 Channel is disabled. 1 Channel is enabled.

[1] Indicates the corresponding DMA channel number noted in Figure 44-50.

44.3.3.5.3 IDMAC Channel Priority Register (IDMAC_CHA_PRI)

This register contains the DMA channel priorities.

0x53FC_00Ac (IDMAC_CHA_PRI) Access: User Read/Write

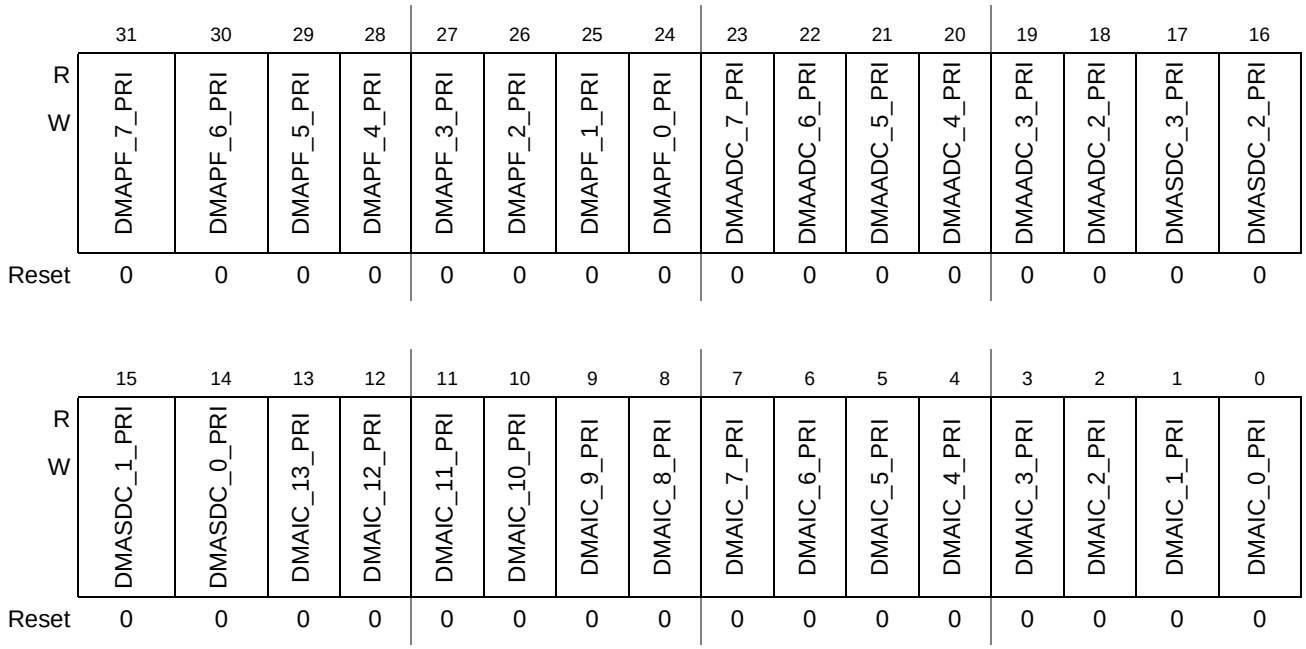


Figure 44-51. IDMAC Channel Priority Register (IDMAC_CHA_PRI)

Table 44-70. IDMAC_CHA_PRI Field Descriptions

Field	Description
DMACH#_PRI[1]	DMA channel priority. (#: 0..31). This field selects if channel # is enabled. 0 Channel has low priority. 1 Channel has high priority.

[1] Indicates the corresponding DMA channel number noted in Figure 44-51.

44.3.3.5.4 IDMAC Channel Busy Register (IDMAC_CHA_BUSY)

0x53FC_00B0 (IDMAC_CHA_BUSY) Access: User Read-Only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DMAPF_7_BUSY	DMAPF_6_BUSY	DMAPF_5_BUSY	DMAPF_4_BUSY	DMAPF_3_BUSY	DMAPF_2_BUSY	DMAPF_1_BUSY	DMAPF_0_BUSY	DMAADC_7_BUSY	DMAADC_6_BUSY	DMAADC_5_BUSY	DMAADC_4_BUSY	DMAADC_3_BUSY	DMAADC_2_BUSY	DMASDC_3_BUSY	DMASDC_2_BUSY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DMASDC_1_BUSY	DMASDC_0_BUSY	DMAIC_13_BUSY	DMAIC_12_BUSY	DMAIC_11_BUSY	DMAIC_10_BUSY	DMAIC_9_BUSY	DMAIC_8_BUSY	DMAIC_7_BUSY	DMAIC_6_BUSY	DMAIC_5_BUSY	DMAIC_4_BUSY	DMAIC_3_BUSY	DMAIC_2_BUSY	DMAIC_1_BUSY	DMAIC_0_BUSY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-52. IDMAC Channel Busy Register (IDMAC_CHA_BUSY)

Table 44-71. IDMAC_CHA_BUSY Field Descriptions

Field	Description
31–0 CHA#_BUSY[1]	Channel # is ready. This bit indicates that channel is busy (in the middle of frame). 0 Channel is not busy. 1 Channel is busy.

1 Indicates the corresponding DMA channel number noted in Figure 44-52.

44.3.3.6 SDC Registers

44.3.3.6.1 SDC Common Configuration Register (SDC_COM_CONF)

This register contains common configuration parameter for the SDC.

0x53FC_00B4 (SDC_COM_CONF) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	COC		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DUAL_MODE	SAVE_REFR_EN	0	SHARP	0	0	BG_EN	MASK_EN	SDC_KEY_COLOR_EN	SDC_GLB_LOC_A	GWSEL	FG_EN	FG_MCP_FORM	BG_MCP_FORM	SDC_MODE	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-53. SDC Common Configuration Register (SDC_COM_CONF)

Table 44-72. SDC_COM_CONF Field Descriptions

Field	Description
31–19	Reserved
18–16 COC	Cursor operation control. Controls the format of the cursor and the type of arithmetic operations. 000 Transparent, cursor is disabled. 001 Full cursor. 010 Reversed cursor. 011 AND between background and cursor. 100 Reserved 101 OR between background and cursor. 110 XOR between background and cursor. 111 Reserved
15 DUAL_MODE	Dual mode enable. Enables/disables dual mode. This bit define if synch interface sends data to smart display. 0 Disable dual mode. 1 Enable dual mode.
14 SAVE_REFR_EN	Enable saving refresh mode. This bit controls saving refresh mode for synchronous interface of smart displays. If saving refresh mode is enabled, display refresh is performed only when display buffer data or displayed windows or cursor position are changed 0 Disable saving refresh. 1 Enable saving refresh.
13	Reserved

Table 44-72. SDC_COM_CONF Field Descriptions (continued)

Field	Description
12 SHARP	Sharp panel enable. Enables/Disables signals for Sharp. 0 Disable Sharp signals. 1 Enable Sharp signals.
11–10	Reserved
9 BG_EN	Background enable. This bit enables the background channel. 0 Background channel is disabled, but control signals (HSYNC, VSYNC) are enabled. 1 Background channel is enabled.
8 MASK_EN	Mask enable. This bit enables the mask channel. 0 Mask channel is disabled. 1 Mask channel is enabled.
7 SDC_KEY_COLOR_EN	Graphic window color keying enable. Enable or disable graphic window color keying. 0 Disable color keying of graphic window 1 Enable color keying of graphic window
6 SDC_GLB_LOC_A	Graphic window alpha mode. Select the use of alpha to be global or local. 0 Local alpha. 1 Global alpha.
5 GWSEL	Graphic window select. Select graphic window to be on foreground or background. 0 Graphic window is background. 1 Graphic window is foreground.
4 FG_EN	Foreground enable. This bit enables the foreground channel. 0 Foreground channel is disabled. 1 Foreground channel is enabled.
3 FG_MCP_FORM	Foreground monochrome data format. This bit relevant only when SDC_MODE=00. It indicates if the pixels of foreground contains alpha. 0 8 bpp without alpha. 1 8 bpp and 8 bit alpha => 16 bit for pixel.
2 BG_MCP_FORM	Background monochrome data format. This bit relevant only when SDC_MODE=00. It indicates that the pixels of background contains alpha. 0 8 bpp without alpha. 1 8 bpp and 8 bit alpha => 16 bit for pixel.
1–0 SDC_MODE	SDC mode. This field selects the SDC output data format. 00 TFT monochrome 01 TFT color 10 YUV progressive 11 YUV interlaced

44.3.3.6.2 SDC Graphic Window Control Register (SDC_GRAPH_WIND_CTRL)

This register defines alpha and key color values.

0x53FC_00B8 (SDC_GRAPH_WIND_CTRL)

Access: User Read/Write

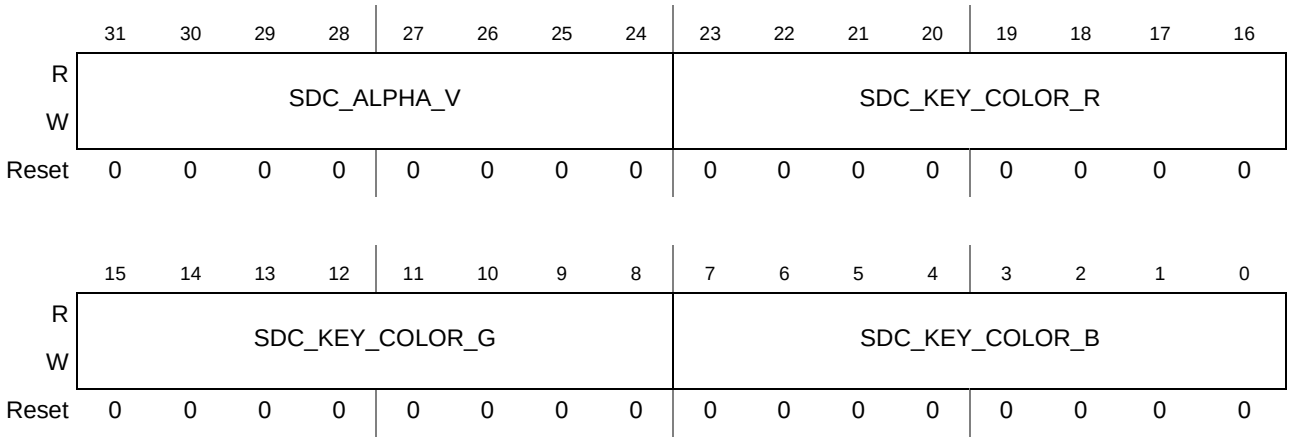


Figure 44-54. SDC Graphic Window Control Register (SDC_GRAPH_WIND_CTRL)

Table 44-73. SDC_GRAPH_WIND_CTRL Field Descriptions

Field	Description
31–24 SDC_ALPHA_V	Graphic window alpha value. Defines the alpha value of graphic window used for alpha blending between graphic window and background plane. The actual value the number, that used in combining calculations. If value = < 0.5- 1/256 (01111111) then actual value = value. If value >= 0.5 (10000000), then actual value = value + 1/256. For SDC_ALPHA_V values, see Table 44-74.
23–16 SDC_KEY_COLOR_R	Graphic window color keying red component. Defines the red component of graphic window color keying. 00000000 No Red .... 11111111 Full Red
15–8 SDC_KEY_COLOR_G	Graphic window color keying green component. Defines the green component of graphic window color keying. Values: 00000000 No green .... 11111111 Full green
7–0 SDC_KEY_COLOR_B	Graphic window color keying blue component. Defines the blue component of graphic window color keying. Values: 00000000 No blue .... 11111111 Full blue

Table 44-74. SDC_ALPHA_V Values

Value	Actual alpha value	Meaning
00000000	00000000	Graphic window totally transparent i.e. not displayed on LCD screen
.....	.....	.....
01111111	01111111	.....
10000000	10000001	.....
10000001	10000010	.....
.....	.....	.....
11111110	11111111	.....
11111111	100000000	Graphic window totally opaque i.e. overlay on LCD screen

44.3.3.6.3 SDC Foreground Window Position Register (SDC_FG_POS)

This register is used to determine the starting position of the foreground window relative to VSYNC.

0x53FC_00BC (SDC_FG_POS) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	FGXP									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	FGYP									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-55. SDC Foreground Window Position Register (SDC_FG_POS)

Table 44-75. SDC_FG_POS Field Descriptions

Field	Description
31–26	Reserved
25–16 FGXP	Foreground window X position. Specifies the number of pixel clock periods between the start of HSYNC and the beginning of the first data of the row. Starts from BGXP. A sum of FGXP and FW for the DMASDC_1 channel must be no greater than a sum of BGXP and FW for the DMASDC_0 channel. A sum of FGXP and FW for the DMASDC_1 channel must be less than SCREEN_WIDTH.

Table 44-75. SDC_FG_POS Field Descriptions (continued)

Field	Description
15–10	Reserved
9–0 FGYP	Foreground window Y position. Specifies the number of lines between the start of VSYNC and the beginning of the first data. Starts from BGYP. A sum of FGYP and FH for the DMASDC_1 channel must be no greater than a sum of BGYP and FH for the DMASDC_0 channel. A sum of FGYP and FH for the DMASDC_1 channel must be less than SCREEN_HEIGHT.

44.3.3.6.4 SDC Background Window Position Register (SDC_BG_POS)

This register is used to determine the starting position of the background window relative to VSYNC.

0x53FC_00C0 (SDC_BG_POS) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	BGXP									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	BGYP									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-56. SDC Background Window Position Register (SDC_BG_POS)

Table 44-76. SDC_BG_POS Field Descriptions

Field	Description
31–26	Reserved
25–16 BGXP	Background window X position. Specifies the number of pixel clock periods between the start of HSYNC and the beginning of the first data of the row. Starts from 0. A sum of BGXP and FW for the DMASDC_0 channel must be less than SCREEN_WIDTH.
15–10	Reserved
9–0 BGYP	Background window Y position. Specifies the number of lines between the start of VSYNC and the beginning of the first data. Starts from 0. A sum of BGYP and FH for the DMASDC_0 channel must be less than SCREEN_HEIGHT.

44.3.3.6.5 SDC Cursor Position Register (SDC_CUR_POS)

This register is used to determine the starting position of the cursor relative to VSYNC.

0x53FC_00C4 (SDC_CUR_POS) Access: User Read/Write

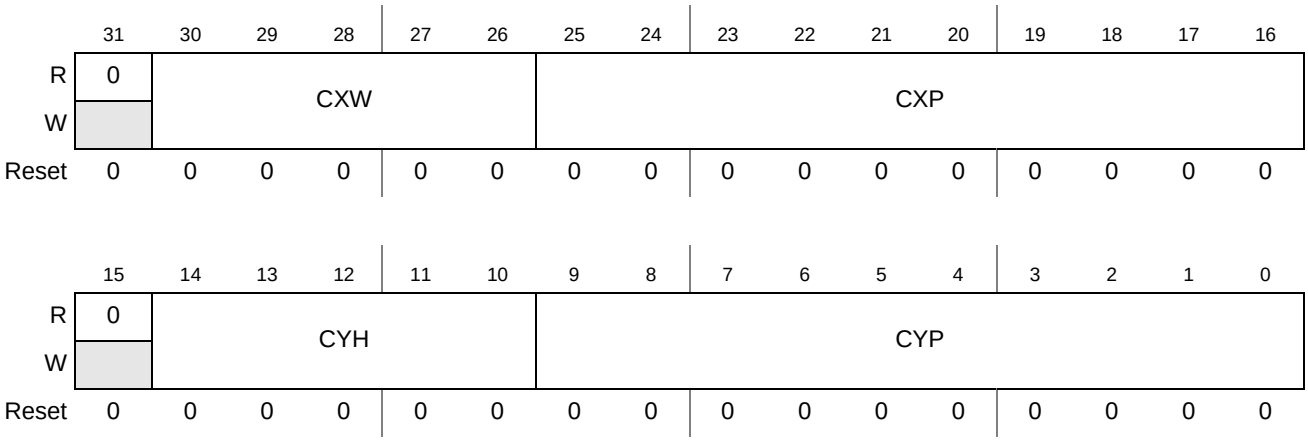


Figure 44-57. SDC Cursor Position Register (SDC_CUR_POS)

Table 44-77. SDC_CUR_POS Field Descriptions

Field	Description
31	Reserved
30–26 CXW	Cursor width minus 1. Specifies the width of the hardware cursor in pixels.
25–16 CXP	Cursor X position. Specifies the cursors horizontal starting position X expressed in pixels. Starts from BGXP. A sum of CXP and CXW must be no greater than a sum of BGXP and FW for the DMASDC_0 channel.
15	Reserved
14–10 CYH	Cursor height minus 1. Specifies the height of the hardware cursor in pixels.
9–0 CYP	Cursor Y position. Specifies the cursors vertical starting position Y expressed in rows. Starts from BGYP. A sum of CYP and CYH must be no greater than a sum of BGYP and FH for the DMASDC_0 channel.

44.3.3.6.6 SDC Cursor Blinking and PWM Contrast Control Register (SDC_CUR_BLINK_PWM_CTRL)

This register is used to control the blinking and the signal output at the IPP_DO_DISPB_CONTRAST pin, which controls the contrast of the LCD panel.

0x53FC_00C8 (SDC_CUR_BLINK_PWM_CTRL) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	SCR		CC_EN	PWM							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BK_EN	0	0	0	0	0	0	0	BKDIV							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-58. SDC Cursor Blinking and PWM Contrast Control Register (SDC_CUR_BLINK_PWM_CTRL)

Table 44-78. SDC_CUR_POS Field Descriptions

Field	Description
31–27	Reserved
26–25 SCR	Source select. Selects the input clock source for the PWM counter. The PWM output frequency is equal to the frequency of the input clock divided by 256. Values: 00 HSYNC 01 Pixel clock 10 HSP_CLK clock 11 Reserved
24 CC_EN	Contrast control enable. Enables/disables the contrast control function. 0 Contrast control is off. 1 Contrast control is on.
23–16 PWM	Pulse width minus 1. Controls the pulse width of the built-in pulse-width modulator, which controls the contrast of the LCD screen.
15 BK_EN	Blinking enable. Determines whether the blink enable cursor will blink or remain steady. 0 Blink is disabled. 1 Blink is enabled.
14–8	Reserved
7–0 BKDIV	Blink divisor minus 1. Sets the cursor blink rate. VSYNC is used to clock the 8-bit up counter. When the counter value equals BKDIV, the cursor toggles on/off.

44.3.3.6.7 SDC Color Cursor Mapping Register (SDC_CUR_MAP)

This register defines the cursor color.

0x53FC_00CC (SDC_CUR_MAP) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	CUR_COL_R							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CUR_COL_G								CUR_COL_B							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-59. SDC Color Cursor Mapping Register (SDC_CUR_MAP)

Table 44-79. SDC_CUR_MAP Field Descriptions

Field	Description
31–24	Reserved
23–16 CUR_COL_R	Cursor red field. Defines the red component of the cursor color in color mode. Values: 00000000 No Red. .... 11111111 Full Red.
15–8 CUR_COL_G	Cursor green field. Defines the green component of the cursor color in color mode. Values: 00000000 No Green. .... 11111111 Full Green.
7–0 CUR_COL_B	Cursor blue field. Defines the blue component of the cursor color in color mode. Values: 00000000 No Blue. .... 11111111 Full Blue.

44.3.3.6.8 SDC Horizontal Configuration Register (SDC_HOR_CONF)

This register defines the horizontal sync pulse timing.

0x53FC_00D0 (SDC_HOR_CONF) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	H_SYNC_WIDTH								SCREEN_WIDTH							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	H_SYNC_DELAY			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-60. SDC Horizontal Configuration Register (SDC_HOR_CONF)

Table 44-80. SDC_HOR_CONF Field Descriptions

Field	Description
31–26 H_SYNC_WIDTH	Horizontal synchronization pulse width minus 1. Specifies the number of pixel clock periods in active HSYNC.
25–16 SCREEN_WIDTH	Screen width minus 1. Specifies the number of pixel clock periods between the last HSYNC and the new HSYNC.
15–4	Reserved
3–0 H_SYNC_DELAY	Horizontal synchronization pulse delay. Specifies the number of pixel clock periods between VSYNC and HSYNC start points.

44.3.3.6.9 SDC Vertical Configuration Register (SDC_VER_CONF)

This register defines the vertical sync pulse timing.

0x53FC_00D4 (SDC_VER_CONF) Access: User Read/Write

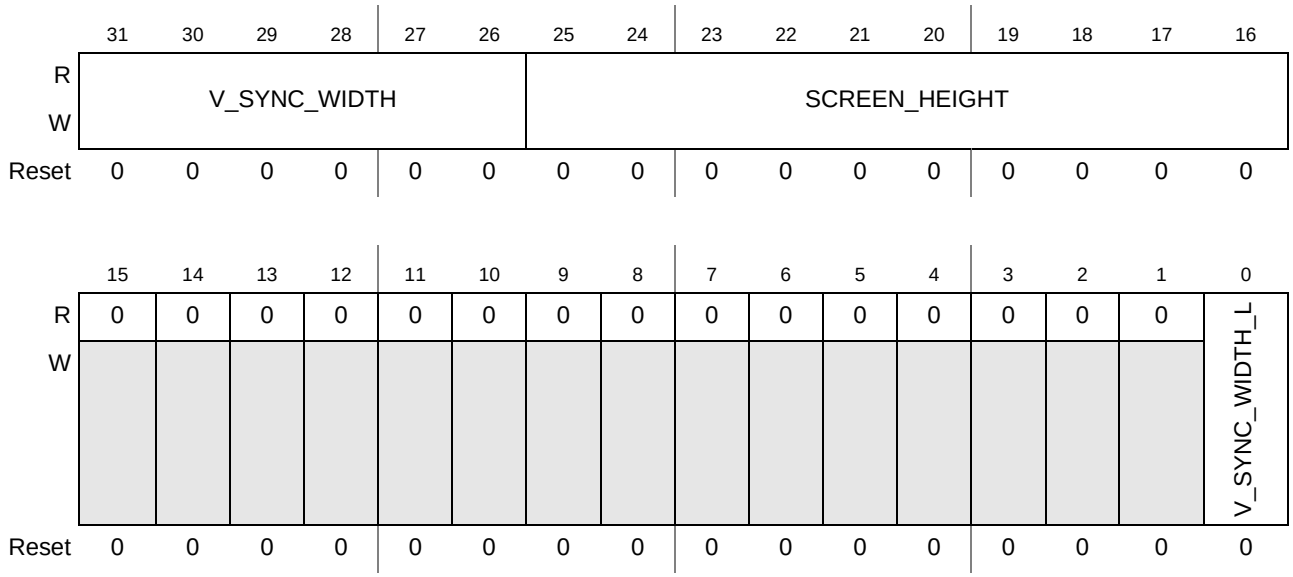


Figure 44-61. SDC Vertical Configuration Register (SDC_VER_CONF)

Table 44-81. SDC_VER_CONF Field Descriptions

Field	Description
31–26 V_SYNC_WIDTH	Vertical synchronization pulse width minus 1. Specifies the width (in pixels or rows depending of V_SYNC_WIDTH_L) of the VSYNC pulse.
25–16 SCREEN_HEIGHT	Screen height minus 1. Defines the number of rows between the last VSYNC pulse and the new VSYNC pulse.
15–1	Reserved
0 V_SYNC_WIDTH_L	Vertical synchronization pulse width units. Defines in which units (pixels or rows) defined VSYNC width. 0 in pixels 1 in rows

44.3.3.6.10 SDC Sharp Configuration Register 1 (SDC_SHARP_CONF_1)

0x53FC_00D8 (SDC_SHARP_CONF_1) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	REV_TOGGLE_DELAY									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PS_FALL_DELAY								CLS_RISE_DELAY							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-62. SDC Sharp Configuration Register 1 (SDC_SHARP_CONF_1)

Table 44-82. SDC_SHARP_CONF_1 Field Descriptions

Field	Description
31–26	Reserved
25–16 REV_TOGGLE_DELAY	REV toggle delay. Controls the transition delay of REV relative to the last data of the line. Values: 0000000000 = 0 pixel clock cycles . . . . 1111111111= 1023 pixel clock cycles
15–8 PS_FALL_DELAY	PS fall delay relative to LP rising edge. This parameter defines when PS pulse will fall. Values: 00000000 = 0 pixel clock cycles . . . . 11111111= 255 pixel clock cycles
7–0 CLS_RISE_DELAY	CLS rise delay relative to LP rising edge. This parameter defines when CLS pulse will rise. Values: 00000000 = 0 pixel clock cycles . . . . 11111111 = 255 pixel clock cycles

44.3.3.6.11 SDC Sharp Configuration Register 2 (SDC_SHARP_CONF_2)

0x53FC_00DC (SDC_SHARP_CONF_2) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	PS_RISE_DELAY									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	CLS_FALL_DELAY									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-63. SDC Sharp Configuration Register 2 (SDC_SHARP_CONF_2)

Table 44-83. SDC_SHARP_CONF_2 Field Descriptions

Field	Description
31–26	Reserved
25–16 PS_RISE_DELAY	PS rise delay relative to LP rising edge. This parameter defines when PS pulse will rise. Values 0000000000= 0 pixel clock cycles . . . . 1111111111= 1023 pixel clock cycles
15–10	Reserved
9–0 CLS_FALL_DELAY	CLS fall delay relative to LP rising edge. This parameter defines when CLS pulse will down. Values 0000000000= 0 pixel clock cycles . . . . 1111111111= 1023 pixel clock cycles

44.3.3.7 ADC Registers

44.3.3.7.1 ADC Configuration Register (ADC_CONF)

This register contain general control bits for configuring ADC.

0x53FC_00E0 (ADC_CONF) Access: User Read/Write

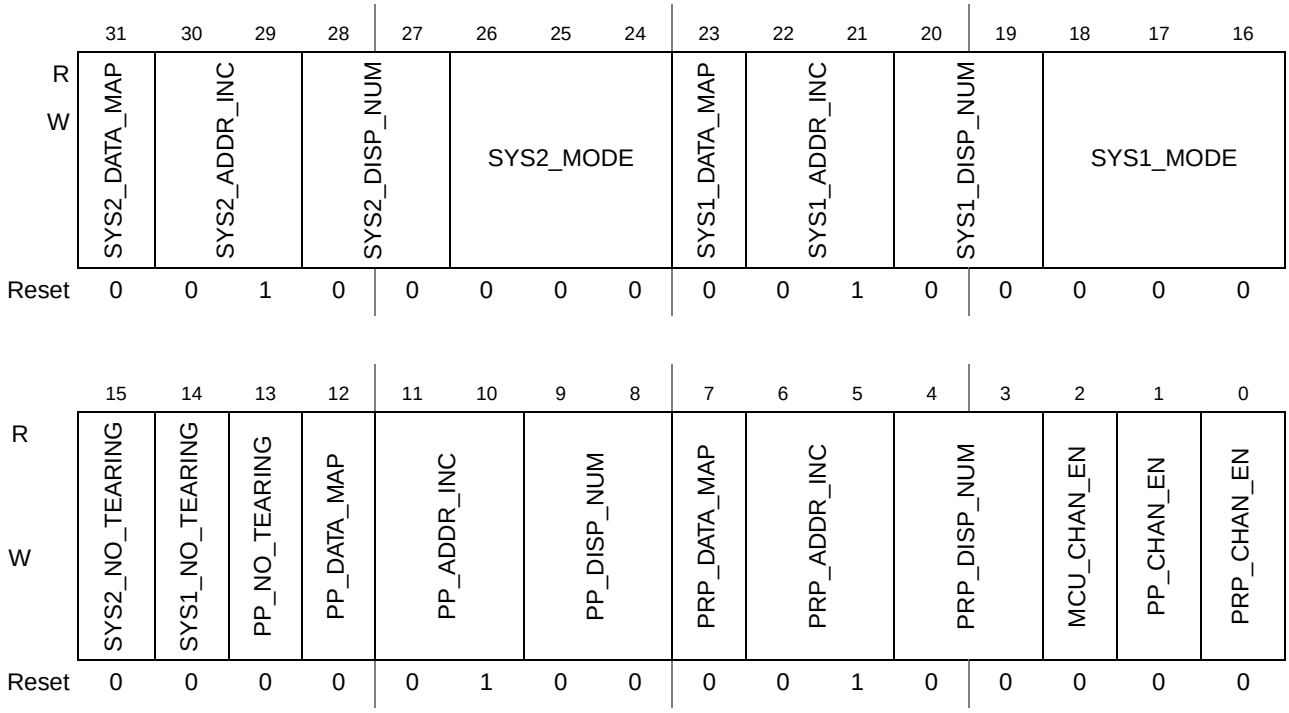


Figure 44-64. ADC Configuration Register (ADC_CONF)

Table 44-84. ADC_CONF Field Descriptions

Field	Description
31 SYS2_DATA_MAP	Select data mapping rule for the system channel 2. This field specifies data mapping rule in the DI for the system channel 2. 0 data mapping rule. 1 command mapping rule.
30–29 SYS2_ADDR_INC	Address increment for the system channel 2 (for full address display type). This field is ignored for XY address display type (increment is always 1). 00 Increment by 1—used for 8-bit access with pixel data 01 Increment by 2—used for 16-bit access with generic data (including byte enable) or pixel data 10 Reserved 11 Increment by 4—used for 32-bit access with generic data or pixel data
28–27 SYS2_DISP_NUM	Display number for system channel 2. This field contains pointer to display for system channel 2. 00 Display 0 01 Display 1 10 Display 2 11 Reserved

Table 44-84. ADC_CONF Field Descriptions (continued)

Field	Description
26–24 SYS2_MODE	Control sequence generation mode for system channel 2. 000 Disable system channel 2 001 Write in template mode, run template if display address is not sequential 010 Read in template mode, run template if display address is not sequential 011 Write in template mode, unconditionally run template for each access 100 Read in template mode, unconditionally run template for each access 101 Write data buffer without template, RS=0 110 Write data buffer without template, RS=1 111 Write in command buffer mode (commands and data)
23 SYS1_DATA_MAP	Select data mapping rule for the system channel 1. This field specifies data mapping rule in the DI for the system channel 1. 0 data mapping rule. 1 command mapping rule.
22–21 SYS1_ADDR_INC	Address increment for the system channel 1 (for full address display type). This field is ignored for XY address display type (increment is always 1. 00 Increment by 1—used for 8-bit access with pixel data 01 Increment by 2—used for 16-bit access with generic data (including byte enable) or pixel data 10 Reserved 11 Increment by 4—used for 32-bit access with generic data or pixel data
20–19 SYS1_DISP_NUM	Display number for system channel 1. This field contains pointer to display for system channel 1. 00 Display 0 01 Display 1 10 Display 2 11 Reserved
18–16 SYS1_MODE	Control sequence generation mode for system channel 1. 000 Disable system channel 1 001 Write in template mode, run template if display address is not sequential 010 Read in template mode, run template if display address is not sequential 011 Write in template mode, unconditionally run template for each access 100 Read in template mode, unconditionally run template for each access 101 Write data buffer without template, RS=0 110 Write data buffer without template, RS=1 111 Write in command buffer mode (commands and data)
15 SYS2_NO_TEARING	The system channel 2 data will be sent to display with display line synchronization to avoid tearing. This bit can ignored for the displays 1 or 2 if the DISP12_VSYNC_SEL bit in the ADC_DISP_VSYNC Register does not enable vertical synchronization for this display. 0 disable display synchronization to avoid tearing. 1 enable display line synchronization to avoid tearing.

Table 44-84. ADC_CONF Field Descriptions (continued)

Field	Description
14 SYS1_NO_TEARING	The system channel 1 data will be sent to display with display line synchronization to avoid tearing. This bit can ignored for the displays 1 or 2 if the DISP12_VSYNC_SEL bit in the ADC_DISP_VSYNC Register does not enable vertical synchronization for this display. 0 disable display synchronization to avoid tearing. 1 enable display line synchronization to avoid tearing.
13 PP_NO_TEARING	The PP channel will be sent to display with display line synchronization to avoid tearing. This bit can ignored for the displays 1 or 2 if the DISP12_VSYNC_SEL bit in the ADC_DISP_VSYNC Register does not enable vertical synchronization for this display. 0 disable display synchronization to avoid tearing 1 enable display line synchronization to avoid tearing
12 PP_DATA_MAP	Select data mapping rule for the post-processing channel. This field specifies data mapping rule in the DI for the post-processing channel. 0 data mapping rule. 1 command mapping rule.
11–10 PP_ADDR_INC	Post-processing channel display address increment per display access (for full address display type). This bit is ignored for XY address display type (increment is always 1). 00 Increment by 1—used for 8-bit access with pixel data 01 Increment by 2—used for 16-bit access with pixel data 10 Reserved 11 Increment by 4—used for 32-bit access with pixel data
9–8 PP_DISP_NUM	Post-processing channel display number. This field contains pointer to display of PP channel. 00 Display 0 01 Display 1 10 Display 2 11 Reserved
7 PRP_DATA_MAP	Select data mapping rule for the pre-processing channel. This field specifies data mapping rule in the DI for the pre-processing channel. 0 data mapping rule. 1 command mapping rule.
6–5 PRP_ADDR_INC	Pre-processing channel display address increment per display access (for full address display type). This bit is ignored for XY address display type (increment is always 1). 00 Increment by 1—used for 8-bit access with pixel data 01 Increment by 2—used for 16-bit access with pixel data 10 Reserved 11 Increment by 4—used for 32-bit access with pixel data
4–3 PRP_DISP_NUM	Pre-processing channel display number. This field contains pointer to display of PRP channel. 00 Display 0 01 Display 1 10 Display 2 11 Reserved

Table 44-84. ADC_CONF Field Descriptions (continued)

Field	Description
2 MCU_CHAN_EN	MCU channel enable. This bit enables MCU channel (direct access to display via the AHB slave interface). 0 MCU channel is disabled. 1 MCU channel is enabled.
1 PP_CHAN_EN	Enable post-processing channel. This bit enables post-processing channel. 0 PP channel is disabled. 1 PP channel is enabled.
0 PRP_CHAN_EN	Enable pre-processing channel. This bit enables pre-processing channel. 0 PRP Channel is disabled. 1 PRP channel is enabled.

44.3.3.7.2 ADC System Channel 1 Start Address Register (ADC_SYSCHA1_SA)

This register contains the start address of channel 1.

0x53FC_00E4 (ADC_SYSCHA1_SA) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SYS1_START_TIME								SYS1_CHAN_SA[22:16]							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SYS1_CHAN_SA[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-65. ADC System Channel 1 Start Address Register (ADC_SYSCHA1_SA)

Table 44-85. ADC_SYSCHA1_SA Field Descriptions

Field	Description
31–23 SYS1_START_TIME	Delay between display vertical synchronization pulse and start time point of system channel 1 window. The delay is defined in pairs of rows. It is used for tearing elimination
22–16 SYS1_CHAN_SA[22:16]	System channel 1 start address. This field contains the start address of channel 1. • In the case of full addressing display, Start address has to be done in full addressing. • In the case of XY addressing display, X is located in the bits [log2(DISP#_SL+1)-1 : 0] and Y is located in the bits [22 : log2(DISP#_SL+1)], where display stride line DISP#_SL+1 must be a power of 2.
15–0 SYS1_CHAN_SA[15:0]	

44.3.3.7.3 ADC System Channel 2 Start Address Register (ADC_SYSCHA2_SA)

This register contains the start address of channel 2.

0x53FC_00E8 (ADC_SYSCHA2_SA) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SYS2_START_TIME								SYS2_CHAN_SA[22:16]							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SYS2_CHAN_SA[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-66. ADC System Channel 2 Start Address Register (ADC_SYSCHA2_SA)

Table 44-86. ADC_SYSCHA2_SA Field Descriptions

Field	Description
31–23 SYS2_START_TIME	Delay between display vertical synchronization pulse and start time point of system channel 1 window. The delay is defined in pairs of rows. It is used for tearing elimination.
22–16 SYS2_CHAN_SA[22:16]	Channel 2 Start Address. This field contains the start address of channel 2. • In the case of full addressing display, Start address is expressed in bytes. • In the case of XY addressing display, X is located in the bits [log2(DISP#_SL+1)-1 : 0] and Y is located in the bits [22 : log2(DISP#_SL+1)], where display stride line DISP#_SL+1 must be a power of 2.
15–0 SYS2_CHAN_SA[15:0]	

44.3.3.7.4 ADC Preprocessing Channel Start Address Register (ADC_PRPCHAN_SA)

This register contains the start address of PRP channel.

0x53FC_00EC (ADC_PRPCHAN_SA) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	PRP_CHAN_SA[22:16]						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PRP_CHAN_SA[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-67. ADC Preprocessing Channel Start Address Register (ADC_PRP_CHAN_SA)

Table 44-87. ADC_PRP_CHAN_SA Field Descriptions

Field	Description
31–23	Reserved
22–16 PRP_CHAN_SA[22:16]	PRP start address. This field contains the start address of PRP channel. • In the case of full addressing display, start address is expressed in bytes. • In the case of XY addressing display, X is located in the bits [log_2(DISP#_SL+1)-1 : 0] and Y is located in the bits [22 : log_2(DISP#_SL+1)], where display stride line DISP#_SL+1 must be a power of 2.
15–0 PRP_CHAN_SA[15:0]	

44.3.3.7.5 ADC Post-Processing Channel Start Address Register (ADC_PPCHAN_SA)

This register contains the start address of PP channel.

0x53FC_00F0 (ADC_PPCHAN_SA) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PP_START_TIME								PP_CHAN_SA[22:16]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PP_CHAN_SA[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-68. ADC Post-Processing Channel Start Address Register (ADC_PPCHAN_SA)

Table 44-88. ADC_PPCHAN_SA Field Descriptions

Field	Description
31–23 PP_START_TIME	Delay between display vertical synchronization pulse and start time point of system channel 1 window. The delay is defined in pairs of rows. It is used for tearing elimination.
22–16 PP_CHAN_SA[22:16]	PP start address. This field contains the start address of PP channel. • In the case of full addressing display, start address is expressed in bytes. • In the case of XY addressing display, X is located in the bits [$\log_2$(DISP#_SL+1)-1 : 0] and Y is located in the bits [22 : $\log_2$(DISP#_SL+1)], where display stride line DISP#_SL+1 must be a power of 2.
15–0 PP_CHAN_SA[15:0]	

44.3.3.7.6 ADC Display 0 Configuration Register (ADC_DISP0_CONF)

This register contains configuration parameters for display 0.

0x53FC_00F4 (ADC_DISP0_CONF) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MCU_DISP0_DATA_MAP	MCU_DISP0_DATA_WIDTH	DISP0_TYPE		DISP0_SL											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-69. ADC Display 0 Configuration Register (ADC_DISP0_CONF)

Table 44-89. ADC_DISP0_CONF Field Descriptions

Field	Description
31–16	Reserved
15 MCU_DISP0_DATA_MAP	Select data mapping rule used when the MCU accesses the display 0. This field specifies data mapping rule in the DI used when the MCU accesses the display 0. 0 data mapping rule. 1 command mapping rule.
14 MCU_DISP0_DATA_WIDTH	Generic data word width for display 0 access by the MCU. 0 16 bits 1 24 bits (in 32-bit word)
13–12 DISP0_TYPE	Display 0 addressing format. This field specifies display addressing format. Values: 00 Full addressing without byte enable 01 Full addressing with byte enable 10 XY addressing 11 Reserved
11–0 DISP0_SL	Display 0 stride line length minus 1. This field specifies the stride line length expressed in bytes for full addressing displays or in pixels for XY addressing displays.

44.3.3.7.7 ADC Display 0 Read Acknowledge Pattern Register (ADC_DISP0_RD_AP)

This register defines acknowledge word for reading data from the display 0.

0x53FC_00F8 (ADC_DISP0_RD_AP) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	DISP0_ACK_MAP	DISP0_ACK_PTRN[23:16]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DISP0_ACK_PTRN[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-70. ADC Display 0 Read Acknowledge Pattern Register (ADC_DISP0_RD_AP)

Table 44-90. ADC_DISP0_RD_AP Field Descriptions

Field	Description
31–25	Reserved
24 DISP0_ACK_MAP	Select data mapping rule for the acknowledge word. 0 data mapping rule 1 command mapping rule
23–0 DISP0_ACK_PTRN	Acknowledge pattern for the display 0. This word is the acknowledge pattern expected for read operation from the display 0.

44.3.3.7.8 ADC Display 0 Read Mask Register (ADC_DISP0_RDM)

This register contains mask word for masking data that are transferred from display 0.

0x53FC_00FC (ADC_DISP0_RDM) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	DISP0_MASK_ACK_DATA[23:16]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DISP0_MASK_ACK_DATA[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-71. ADC Display 0 Read Mask Register (ADC_DISP0_RDM)

Table 44-91. ADC_PRCHAN_SA Field Descriptions

Field	Description
31–24	Reserved
23–0 DISP0_MASK_ACK_DATA	The mask for display 0 read data.

44.3.3.7.9 ADC Display 0 Screen Size Register (ADC_DISP0_SS)

This register defines display 0 screen size.

0x53FC_0100 (ADC_DISP0_SS) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	SCREEN0_HEIGHT									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	SCREEN0_WIDTH									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-72. ADC Display 0 Screen Size Register (ADC_DISP0_SS)

Table 44-92. ADC_DISP0_SS Field Descriptions

Field	Description
31–26	Reserved
25–16 SCREEN0_HEIGHT	Display 0 screen height minus 1. This field contains total number of display rows (including vertical blanking intervals).
15–10	Reserved
9–0 SCREEN0_WIDTH	Display 0 screen width minus 1. This field contains total number of pixels in a display row (including horizontal blanking intervals).

44.3.3.7.10 ADC Display 1 Configuration Register (ADC_DISP1_CONF)

This register contains configuration parameters for display 1.

0x53FC_0104 (ADC_DISP1_CONF) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MCU_DISP1_DATA_MAP	MCU_DISP1_DATA_WIDTH	DISP1_TYPE		DISP1_SL											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-73. ADC Display 1 Configuration Register (ADC_DISP1_CONF)

Table 44-93. ADC_DISP1_CONF Field Descriptions

Field	Description
31–16	Reserved
15 MCU_DISP1_DATA_MAP	Select data mapping rule used when the MCU accesses the display 1. This field specifies data mapping rule in the DI used when the MCU accesses the display 1. 0 data mapping rule. 1 command mapping rule.
14 MCU_DISP1_DATA_WIDTH	Data word width for display 1 access by the MCU. 0 16 bits 1 24 bits (in 32-bit word)
13–12 DISP1_TYPE	Display 1 addressing format. This field specifies display addressing format. Values: 00 Full addressing without byte enable 01 Full addressing with byte enable 10 XY addressing 11 Reserved
11–0 DISP1_SL	Display 1 stride line length minus 1. This field specifies the stride line length expressed in bytes for full addressing displays or in pixels for XY addressing displays.

44.3.3.7.11 ADC Display 1 Read Acknowledge Pattern Register (ADC_DISP1_RD_AP)

This register defines acknowledge word for reading data from the display 1.

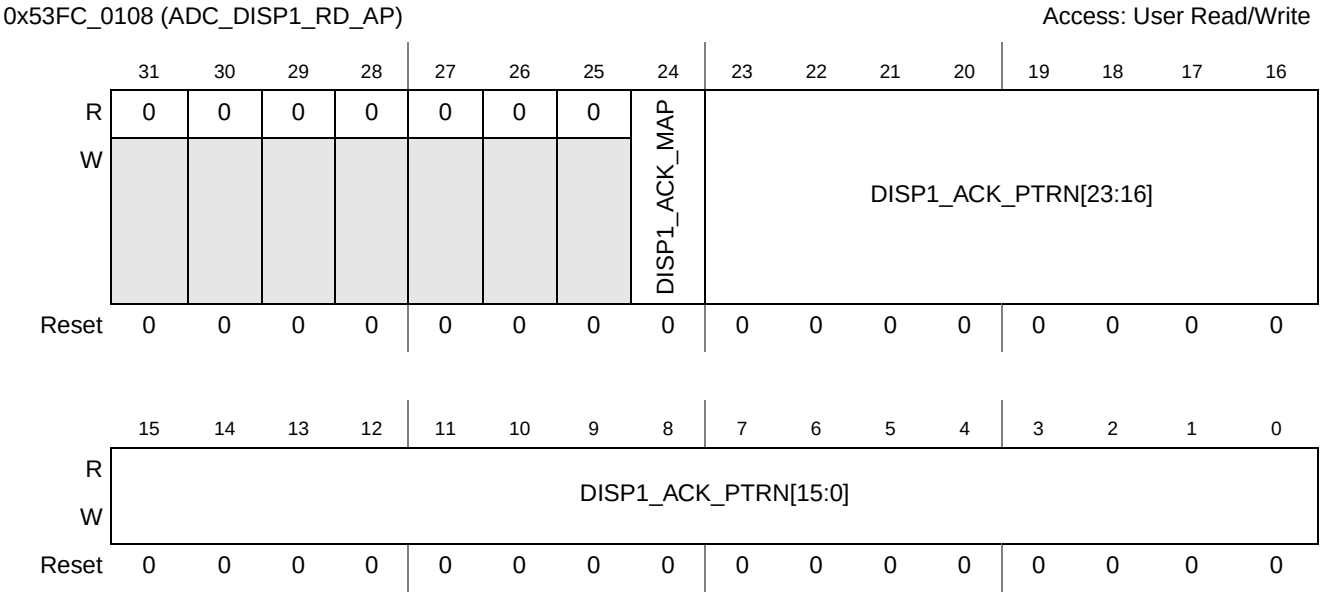


Figure 44-74. ADC Display 1 Read Acknowledge Pattern Register (ADC_DISP1_RD_AP)

Table 44-94. ADC_DISP1_RD_AP Field Descriptions

Field	Description
31–25	Reserved
24 DISP1_ACK_MAP	Select data mapping rule for the acknowledge word. 0 data mapping rule. 1 command mapping rule.
23–0 DISP1_ACK_PTRN	Acknowledge pattern for the display 1. This word is the acknowledge pattern expected for read operation from the display 1.

44.3.3.7.12 ADC Display 1 Read Mask Register (ADC_DISP1_RDM)

This register contains mask word for masking data that are transferred from display 1.

0x53FC_010C (ADC_DISP1_RDM) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	DISP1_MASK_ACK_DATA[23:16]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DISP1_MASK_ACK_DATA[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-75. ADC Display 1 Read Mask Register (ADC_DISP1_RDM)

Table 44-95. ADC_DISP1_RDM Field Descriptions

Field	Description
31–24	Reserved
23–0 DISP1_MASK_ACK_DATA	Mask for display 1 read data.

44.3.3.7.13 ADC Displays 1 or 2 Screen Size Register (ADC_DISP12_SS)

This register defines displays 1 or 2 screen size. It is used for one of displays 1 or 2 when VSYNC synchronization is enabled for this display.

0x53FC_0110 (ADC_DISP12_SS) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	SCREEN12_HEIGHT									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	SCREEN12_WIDTH									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-76. ADC Displays 1 or 2 Screen Size Register (ADC_DISP12_SS)

Table 44-96. ADC_DISP12_SS Field Descriptions

Field	Description
31–26	Reserved
25–16 SCREEN12_HEIGHT	Display 1 or 2 screen height minus 1. This field contains total number of display rows (including vertical blanking intervals).
15–10	Reserved
9–0 SCREEN12_WIDTH	Display 1 or 2 screen width minus 1. This field contains total number of pixels in a display row (including horizontal blanking intervals).

44.3.3.7.14 ADC Display 2 Configuration Register (ADC_DISP2_CONF)

This register contains configuration parameters for display 3.

0x53FC_0114 (ADC_DISP2_CONF) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MCU_DISP2_DATA_MAP	MCU_DISP2_DATA_WIDTH	DISP2_TYPE		DISP2_SL											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-77. ADC Display 2 Configuration Register (ADC_DISP2_CONF)

Table 44-97. ADC_DISP2_CONF Field Descriptions

Field	Description
31–16	Reserved
15 MCU_DISP2_DATA_MAP	Select data mapping rule used when the MCU accesses the display 2. This field specifies data mapping rule in the DI used when the MCU accesses the display 2. 0 data mapping rule 1 command mapping rule
14 MCU_DISP2_DATA_WIDTH	Data word width for display 2 access by the MCU 0 16 bits 1 24 bits (in 32-bit word)
13–12 DISP2_TYPE	Display 2 addressing format. This field specifies display addressing format. Values: 00 Full addressing without byte enable 01 Full addressing with byte enable 10 XY addressing 11 Reserved
11–0 DISP2_SL	Display 2 stride line length minus 1. This field specifies the stride line length expressed in bytes for full addressing displays or in pixels for XY addressing displays.

44.3.3.7.15 ADC Display 2 Read Acknowledge Pattern Register (ADC_DISP2_RD_AP)

This register defines acknowledge word for reading data from the display 2.

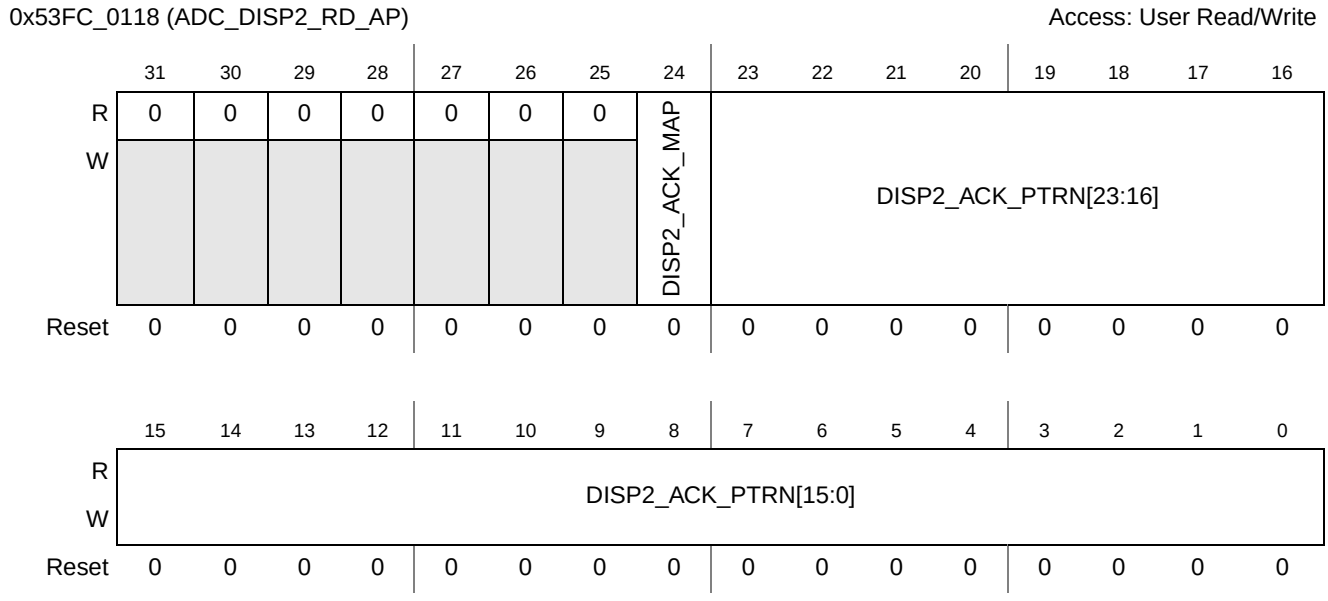


Figure 44-78. ADC Display 2 Read Acknowledge Pattern Register (ADC_DISP2_RD_AP)

Table 44-98. ADC_DISP1_RD_AP Field Descriptions

Field	Description
31–25	Reserved
24 DISP2_ACK_MAP	Select data mapping rule for the acknowledge word. 0 data mapping rule. 1 command mapping rule.
23–0 DISP2_ACK_PTRN	Acknowledge pattern for the display 2. This word is the acknowledge pattern expected for read operation from the display 2.

44.3.3.7.16 ADC Display 2 Read Mask Register (ADC_DISP2_RDM)

0x53FC_011C (ADC_DISP2_RDM) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	DISP2_MASK_ACK_DATA[23:16]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DISP2_MASK_ACK_DATA[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-79. ADC Display 2 Read Mask Register (ADC_DISP2_RDM)

Table 44-99. ADC_DISP2_RDM Field Descriptions

Field	Description
31–24	Reserved
23–0 DISP2_MASK_ACK_DATA	Mask for display 2 read data. This register contains mask word for masking data that are transferred from display 2.

44.3.3.7.17 ADC Displays Vertical Synchronization Register (ADC_DISP_VSYNC)

0x53FC_0120 (ADC_DISP_VSYNC) Access: User Read/Write

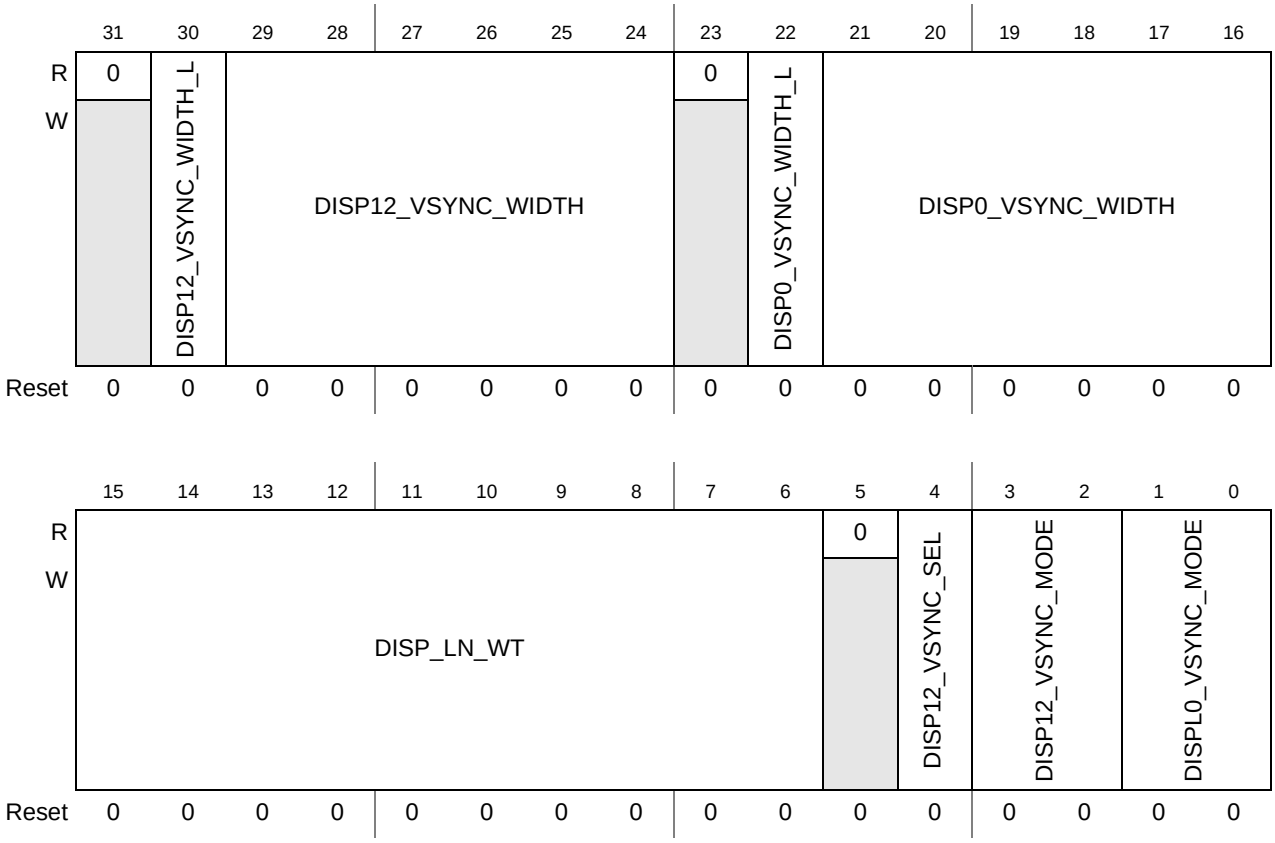


Figure 44-80. ADC Displays Vertical Synchronization Register (ADC_DISP_VSYNC)

Table 44-100. ADC_DISP_VSYNC Field Descriptions

Field	Description
31	Reserved
30 DISP12_VSYNC_WIDTH_L	Resolution of DISP0_VSYNC_WIDTH for the displays 1 or 2. 0 in pixels 1 in lines
29–24 DISP12_VSYNC_WIDTH	VSYNC pulse width minus 1 of the displays 1 or 2. This register contains number of pixels or lines according to DISP0_VSYNC_WIDTH_L definition between pos-edge and neg-edge of VSYNC.
23	Reserved
22 DISP0_VSYNC_WIDTH_L	Resolution of DISP0_VSYNC_WIDTH for the display 0 0 in pixels 1 in lines
21–16 DISP0_VSYNC_WIDTH	VSYNC pulse width minus 1 of the display 0. This register contains number of pixels or lines according to DISP0_VSYNC_WIDTH_L definition between pos-edge and neg-edge of VSYNC.

Table 44-100. ADC_DISP_VSYNC Field Descriptions (continued)

Field	Description
15–6 DISP_LN_WT	Delay between the start point of the first sensor row and the VSYNC pulse of displays 0, 1 or 2 expressed in display rows.
5	Reserved
4 DISP12_VSYNC_SEL	Selection of the displays 1 or 2 to be in vertical synchronization mode with the corresponding parameters from the ADC_DISP12_SS and ADC_DISP_VSYNC Registers 0 select display 1 for vertical synchronization 1 select display 2 for vertical synchronization
3–2 DISP12_VSYNC_MODE	Displays 1 or 2 vertical synchronization mode. 00 Vertical synchronization disabled 01 ADC internal VSYNC generation 10 ADC internal VSYNC generation with synchronization to sensor VSYNC 11 External VSYNC generation (by display)
1–0 DISPL0_VSYNC_MODE	Display 0 vertical synchronization mode. 00 Vertical synchronization disabled 01 ADC internal VSYNC generation 10 ADC internal VSYNC generation with synchronization to sensor VSYNC 11 External VSYNC generation (by display)

44.3.3.8 DI Registers

44.3.3.8.1 DI Display Interface Configuration Register (DI_DISP_IF_CONF)

0x53FC_0124 (DI_DISP_IF_CONF) Access: User Read/Write

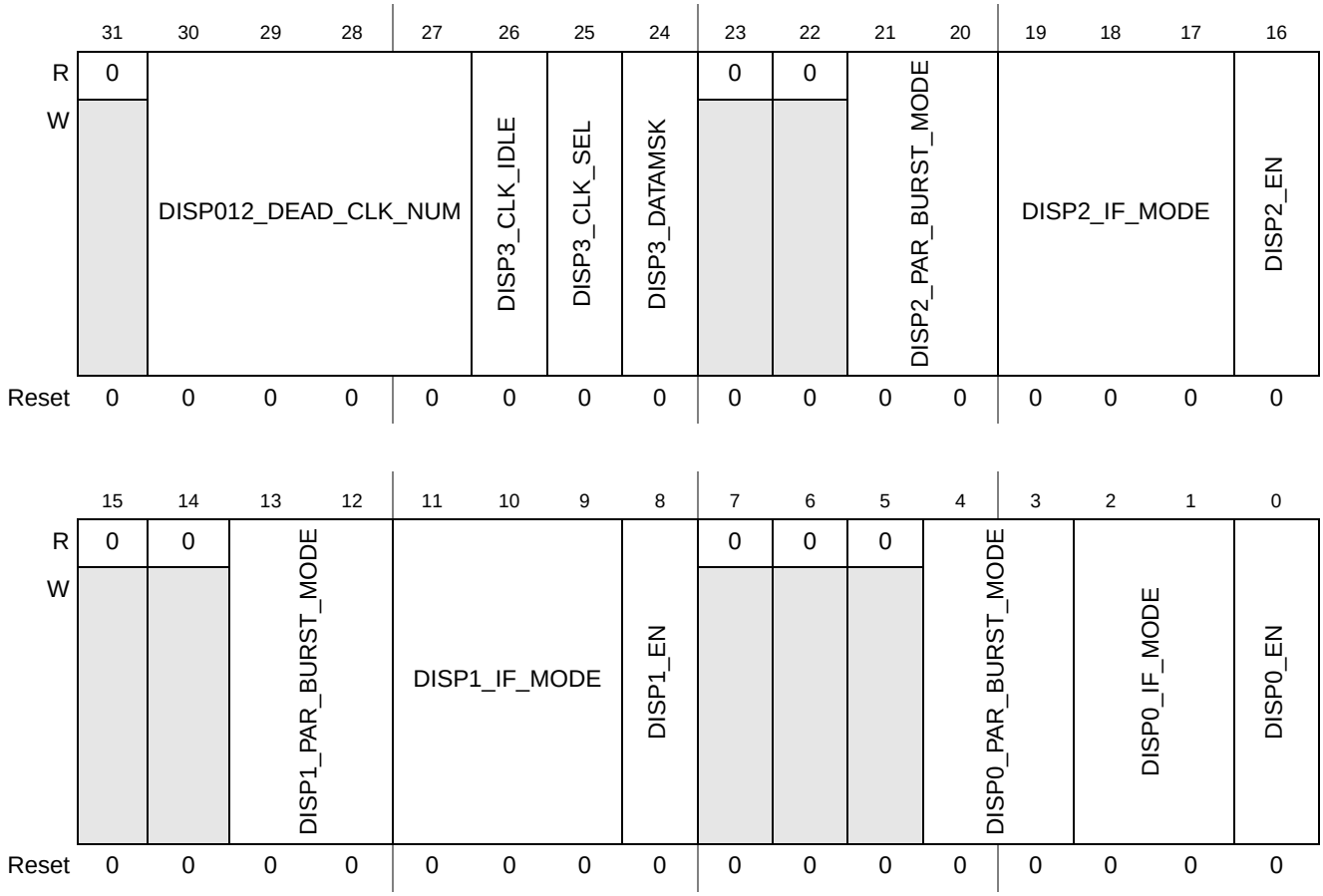


Figure 44-81. DI Display Interface Configuration Register (DI_DISP_IF_CONF)

Table 44-101. DI_DISP_IF_CONF Field Descriptions

Field	Description
31	Reserved
30–27 DISP012_DEAD_CLK_NUM	Number of dead clock cycles. Defines a number of dead cycles of the HSP_CLK clock inserted between accesses of two different displays or between two bursts transferred to/from a single display. 0000 4 dead cycles 0001 5 dead cycles .... 1111 19 dead cycles

Table 44-101. DI_DISP_IF_CONF Field Descriptions (continued)

Field	Description
26 DISP3_CLK_IDLE	Display 3 interface clock idle enable. Enables/disables the display 3 clock when VSYNC is active. 0 Enable clock. 1 Disable clock.
25 DISP3_CLK_SEL	Select display 3 interface clock. Selects whether to enable or disable display clock when there is no data output. 0 Always enable clock even if there is no data output. 1 Disable clock when no data output.
24 DISP3_DATAMSK	Data mask for the display 3. Enables/Disables the data output to zero for the Sharp TFT panel power-off sequence. 0 data is normal. 1 data always equals 0.
23–22	Reserved
21–20 DISP2_PAR_BURST_MODE	Display 2 parallel interface burst mode. Values: 00 Burst access mode with sampling by CS or R/W or ENABLE signals 01 Burst access mode with sampling by separate burst clock (BCLK) 10 Single access mode (all control signals are not active for one display interface clock after each display access) 11 Reserved
19–17 DISP2_IF_MODE	Display 2 interface mode. Values: 000 System 80 (type 1) parallel interface (sampling with CS signal) 001 System 80 (type 2) parallel interface (sampling with R/W signals) 010 System 68k (type 1) parallel interface (sampling with CS signal) 011 System 68k (type 2) parallel interface (sampling with ENABLE signal) 100 Serial 3-wire interface 101 Serial 4-wire interface 110 Serial 5-wire interface, RS is sampled with serial clock 111 Serial 5-wire interface, RS is sampled with CS signal
16 DISP2_EN	Display 2 enable. Enables/disables the display 2. 0 Disable display 2. 1 Enable display 2.
15–14	Reserved
13–12 DISP1_PAR_BURST_MODE	Display 1 parallel interface burst mode. Values: 00 Burst access mode with sampling by CS or R/W or ENABLE signals 01 Burst access mode with sampling by separate burst clock (BCLK) 10 Single access mode (all control signals are not active for one display interface clock after each display access) 11 Reserved

Table 44-101. DI_DISP_IF_CONF Field Descriptions (continued)

Field	Description
11–9 DISP1_IF_MODE	Display 1 interface mode. Values: 000 System 80 (type 1) parallel interface (sampling with CS signal) 001 System 80 (type 2) parallel interface (sampling with R/W signals) 010 System 68k (type 1) parallel interface (sampling with CS signal) 011 System 68k (type 2) parallel interface (sampling with ENABLE signal) 100 Serial 3-wire interface 101 Serial 4-wire interface 110 Serial 5-wire interface, RS is sampled with serial clock 111 Serial 5-wire interface, RS is sampled with CS signal
8 DISP1_EN	Display 1 enable. Enables/disables the display 1. 0 Disable display 1. 1 Enable display 1.
7–5	Reserved
4–3 DISP0_PAR_BURST_MODE	Display 0 parallel interface burst mode. 00 Burst access mode with sampling by CS or R/W or ENABLE signals 01 Burst access mode with sampling by separate burst clock (BCLK) 10 Single access mode (all control signals are not active for one display interface clock after each display access) 11 Reserved
2–1 DISP0_IF_MODE	Display 0 interface mode 00 System 80 (type 1) parallel interface (sampling with CS signal) 01 System 80 (type 2) parallel interface (sampling with R/W signals) 10 System 68k (type 1) parallel interface (sampling with CS signal) 11 System 68k (type 2) parallel interface (sampling with ENABLE signal)
0 DISP0_EN	Display 0 enable. Enables/disables the display 0. 0 Disable display 0. 1 Enable display 0.

44.3.3.8.2 DI Display Signals Polarity Register (DI_DISP_SIG_POL)

0x53FC_0128 (DI_DISP_SIG_POL) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	D2_BCLK_POL	D1_BCLK_POL	D0_BCLK_POL	D3_VSYNC_POL	D3_HSYNC_POL	D3_DRDY_SHARP_POL	D3_CLK_POL	D3_DATA_POL	D2_SER_RS_POL	D2_SD_CLK_POL	D2_SD_D_POL	D2_RD_POL	D2_WR_POL	D2_PAR_RS_POL	D2_CS_POL	D2_DATA_POL
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	D1_SER_RS_POL	D1_SD_CLK_POL	D1_SD_D_POL	D1_RD_POL	D1_WR_POL	D1_PAR_RS_POL	D1_CS_POL	D1_DATA_POL	0	D12_VSYNC_POL	D0_VSYNC_POL	D0_RD_POL	D0_WR_POL	D0_PAR_RS_POL	D0_CS_POL	D0_DATA_POL
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-82. DI Display Signals Polarity Register (DI_DISP_SIG_POL)

Table 44-102. DI_DISP_SIG_POL Field Descriptions

Field	Description
31 D2_BCLK_POL	Display parallel interface burst clock polarity for the display 2. Sets the polarity of the IPP_DO_DISPB_BCLK output pin when accessing display 2. 0 Straight polarity. 1 Inverse polarity.
30 D1_BCLK_POL	Display parallel interface burst clock polarity for the display 1. Sets the polarity of the IPP_DO_DISPB_BCLK output pin when accessing display 1. 0 Straight polarity. 1 Inverse polarity.
29 D0_BCLK_POL	Display parallel interface burst clock polarity for the display 0. Sets the polarity of the IPP_DO_DISPB_BCLK output pin when accessing display 1. 0 Straight polarity. 1 Inverse polarity.
28 D3_VSYNC_POL	Vertical synchronization signal polarity for the display 3. Sets the polarity of the IPP_DO_DISPB_D3_VSYNC output pin. 0 Active low. 1 Active high.

Table 44-102. DI_DISP_SIG_POL Field Descriptions (continued)

Field	Description
27 D3_HSYNC_POL	Horizontal synchronization signal polarity for the display 3. Sets the polarity of the IPP_DO_DISPB_D3_HSYNC output pin. 0 Active low. 1 Active high.
26 D3_DRDY_SHARP_POL	Output enable polarity or Sharp signals polarity (SPL, REV, CLS, PS) for the display 3. Sets the polarity of the IPP_DO_DISPB_D3_DRDY, IPP_DO_DISPB_D3_CLS, IPP_DO_DISPB_D3_REV, IPP_DO_DISPB_D3_PS output pins. The IPP_DO_DISPB_D3_DRDY pin can be used both as the output enable signal or the PS signal. Timing diagrams of Sharp signals shown in the IPU Electrical Spec correspond to D3_DRDY_SHARP_POL = 0. 0 Active low (for output enable signal) or straight polarity for sharp signals. 1 Active high (for output enable signal) or inverse polarity for sharp signals.
25 D3_CLK_POL	Display interface clock polarity for the display 3. Sets the polarity of the IPP_DO_DISPB_D3_CLK output pin. 0 Straight polarity. 1 Inverse polarity.
24 D3_DATA_POL	Data polarity for the display 3. Sets the polarity of the IPP_DO_DISPB_DATA output pin and the IPP_IND_DISPB_DATA input pin when the display 3 is accessed. 0 Straight polarity. 1 Inverse polarity.
23 D2_SER_RS_POL	Address bit polarity for the display 2 serial interface. Sets the polarity of the IPP_DO_DISPB_SER_RS output pin when the display 2 is accessed via serial interface. 0 Straight polarity. 1 Inverse polarity.
22 D2_SD_CLK_POL	Serial interface clock polarity for the display 2. Sets the polarity of the IPP_DO_DISPB_SD_CLK output pin. 0 Straight polarity. 1 Inverse polarity.
21 D2_SD_D_POL	Serial data polarity for the display 2. Sets the polarity of the IPP_DO_DISPB_SD_D output pin and the IPP_IND_DISPB_SD_D input pin when the display 2 is accessed via serial interface. 0 Straight polarity. 1 Inverse polarity.
20 D2_RD_POL	Write signal polarity for the display 2. Sets the polarity of the IPP_DO_DISPB_RD output pin when the display 2 is accessed. 0 Active low. 1 Active high.
19 D2_WR_POL	Write signal polarity for the display 2. Sets the polarity of the IPP_DO_DISPB_WR output pin (for parallel interface) or the polarity of the write control bit in the output stream (for serial interface) when the display 2 is accessed. 0 Active low. 1 Active high.

Table 44-102. DI_DISP_SIG_POL Field Descriptions (continued)

Field	Description
18 D2_PAR_RS_POL	Address bit polarity for the display 2 parallel interface. Sets the polarity of the IPP_DO_DISPB_PAR_RS output pin when the display 2 is accessed. 0 Straight polarity. 1 Inverse polarity.
17 D2_CS_POL	Chip select signal polarity for the display 2. Sets the polarity of the IPP_DO_DISPB_D2_CS output pin. 0 Active low. 1 Active high.
16 D2_DATA_POL	Data polarity for the display 2. Sets the polarity of the IPP_DO_DISPB_DATA output pin and the IPP_IND_DISPB_DATA input pin when the display 2 is accessed. 0 Straight polarity. 1 Inverse polarity.
15 D1_SER_RS_POL	Address bit polarity for the display 1 serial interface. Sets the polarity of the IPP_DO_DISPB_SER_RS output pin when the display 1 is accessed via serial interface. 0 Straight polarity. 1 Inverse polarity.
14 D1_SD_CLK_POL	Serial interface clock polarity for the display 1. Sets the polarity of the IPP_DO_DISPB_SD_CLK output pin. 0 Straight polarity. 1 Inverse polarity.
13 D1_SD_D_POL	Serial data polarity for the display 1. Sets the polarity of the IPP_DO_DISPB_SD_D output pin and the IPP_IND_DISPB_SD_D input pin when the display 1 is accessed via serial interface. 0 Straight polarity. 1 Inverse polarity.
12 D1_RD_POL	Write signal polarity for the display 1. Sets the polarity of the IPP_DO_DISPB_RD output pin when the display 1 is accessed. 0 Active low. 1 Active high.
11 D1_WR_POL	Write signal polarity for the display 1. Sets the polarity of the IPP_DO_DISPB_WR output pin (for parallel interface) or the polarity of the write control bit in the output stream (for serial interface) when the display 1 is accessed. 0 Active low. 1 Active high.
10 D1_PAR_RS_POL	Address bit polarity for the display 1 parallel interface. Sets the polarity of the IPP_DO_DISPB_PAR_RS output pin when the display 1 is accessed. 0 Straight polarity. 1 Inverse polarity.
9 D1_CS_POL	Chip select signal polarity for the display 1. Sets the polarity of the IPP_DO_DISPB_D1_CS output pin. 0 Active low. 1 Active high.

Table 44-102. DI_DISP_SIG_POL Field Descriptions (continued)

Field	Description
8 D1_DATA_POL	Data polarity for the display 1. Sets the polarity of the IPP_DO_DISPB_DATA output pin and the IPP_IND_DISPB_DATA input pin when the display 1 is accessed. 0 Straight polarity. 1 Inverse polarity.
7	Reserved
6 D12_VSYNC_POL	Vertical synchronization signal polarity for the displays 1 and 2. Sets the polarity of the IPP_DO_DISPB_D12_VSYNC output pin and the IPP_IND_DISPB_D12_VSYNC input pin. 0 Active low. 1 Active high.
5 D0_VSYNC_POL	Vertical synchronization signal polarity for the display 0. Sets the polarity of the IPP_DO_DISPB_D0_VSYNC output pin and the IPP_IND_DISPB_D0_VSYNC input pin. 0 Active low. 1 Active high.
4 D0_RD_POL	Write signal polarity for the display 0. Sets the polarity of the IPP_DO_DISPB_RD output pin when the display 0 is accessed. 0 Active low. 1 Active high.
3 D0_WR_POL	Write signal polarity for the display 0. Sets the polarity of the IPP_DO_DISPB_WR output pin when the display 0 is accessed. 0 Active low. 1 Active high.
2 D0_PAR_RS_POL	Address bit polarity for the display 0 parallel interface. Sets the polarity of the IPP_DO_DISPB_PAR_RS output pin when the display 0 is accessed. 0 Straight polarity. 1 Inverse polarity.
1 D0_CS_POL	Chip select signal polarity for the display 0. Sets the polarity of the IPP_DO_DISPB_D0_CS output pin. 0 Active low. 1 Active high.
0 D0_DATA_POL	Data polarity for the display 0. Sets the polarity of the IPP_DO_DISPB_DATA output pin and the IPP_IND_DISPB_DATA input pin when the display 0 is accessed. 0 Straight polarity. 1 Inverse polarity.

44.3.3.8.3 DI Serial Display 1 Configuration Register (DI_SER_DISP1_CONF)

This register defines serial mode for the display 1.

0x53FC_012C (DI_SER_DISP1_CONF) Access: User Read/Write

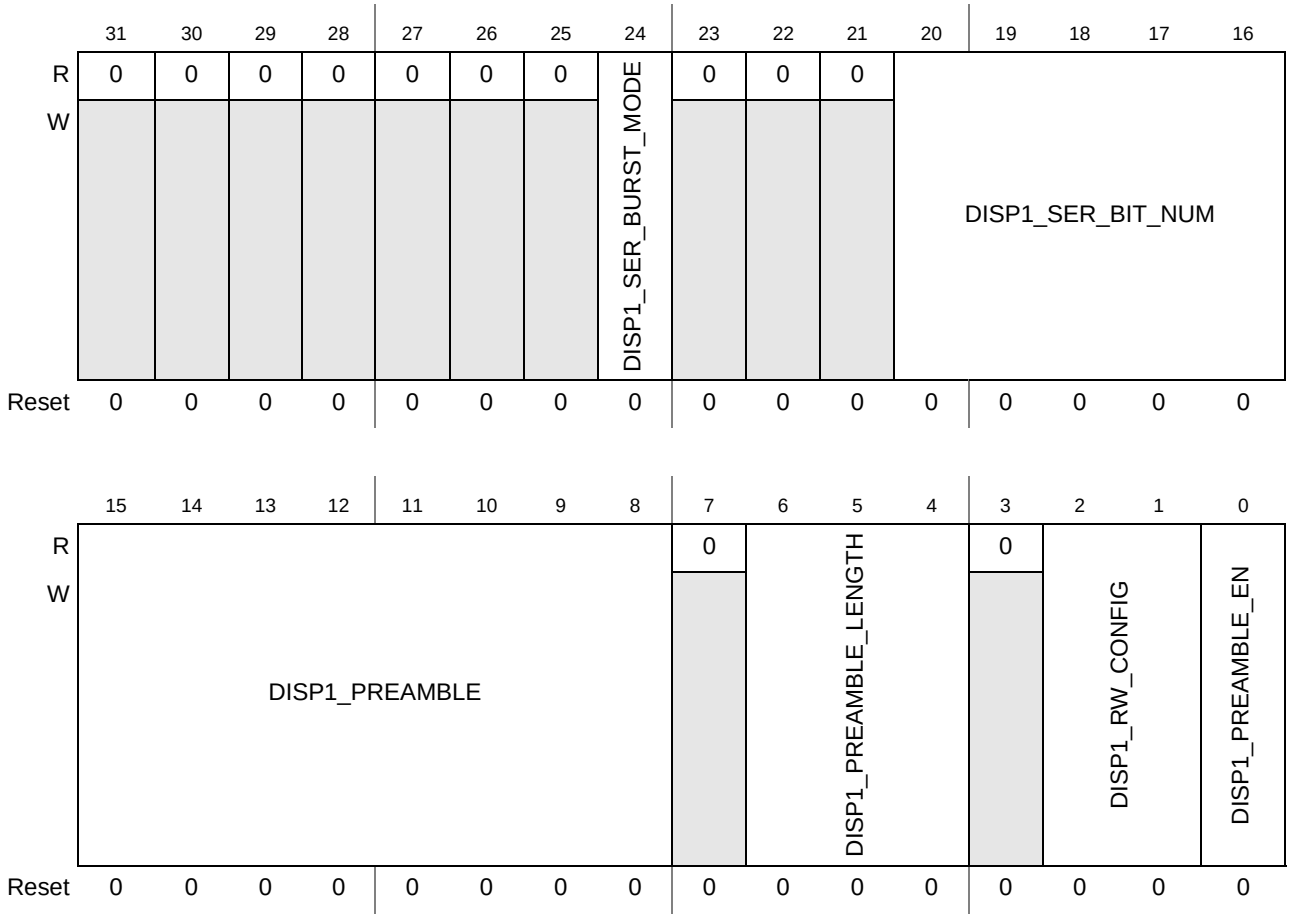


Figure 44-83. DI Serial Display 1 Configuration Register (DI_SER_DISP1_CONF)

Table 44-103. DI_SER_DISP1_CONF Field Descriptions

Field	Description
31–25	Reserved
24 DISP1_SER_BURST_MODE	Burst mode enable. Enables burst mode when the display chip select remains active during whole burst transfer. In single access mode, the display chip select is not active between display accesses. 0 single access mode. 1 burst access mode.
23–21	Reserved
20–16 DISP1_SER_BIT_NUM	Output/input data bit number minus 1. Defines number of data bits per display access. Values: 000001 bit .... 1000118 bits (max)

Table 44-103. DI_SER_DISP1_CONF Field Descriptions (continued)

Field	Description
15–8 DISP1_PREAMBLE	Preamble contents. If DISP1_PREAMBLE_LENGTH is less than 8, only DISP1_PREAMBLE_LENGTH least significant bits of DISP1_PREAMBLE are sent.
7	Reserved
6–4 DISP1_PREAMBLE_LENGTH	Preamble length minus 1 in bits (without RS and RW bits). Values: 000 1 bit .... 101 6 bits 110 7 bits 111 8 bits (max)
3	Reserved
2–1 DISP1_RW_CONFIG	Configuration of read/write (RW) control bit to be sent via serial interface. For 5-wire serial interface (when RS is send via a separate wire), the ‘01’ and ‘10’ values of DISP1_RW_CONFIG have same effect. Values: 00 No RW bit in serial interface output sequence 01 RW bit is sent after preamble and before RS bit 10 RW bit is sent after preamble and after RS bit 11 reserved
0 DISP1_PREAMBLE_EN	Preamble enable. 0 disable preamble 1 enable preamble

44.3.3.8.4 DI Serial Display 2 Configuration Register (DI_SER_DISP2_CONF)

This register defines serial mode for the display 2.

0x53FC_0130 (DI_SER_DISP2_CONF) Access: User Read/Write

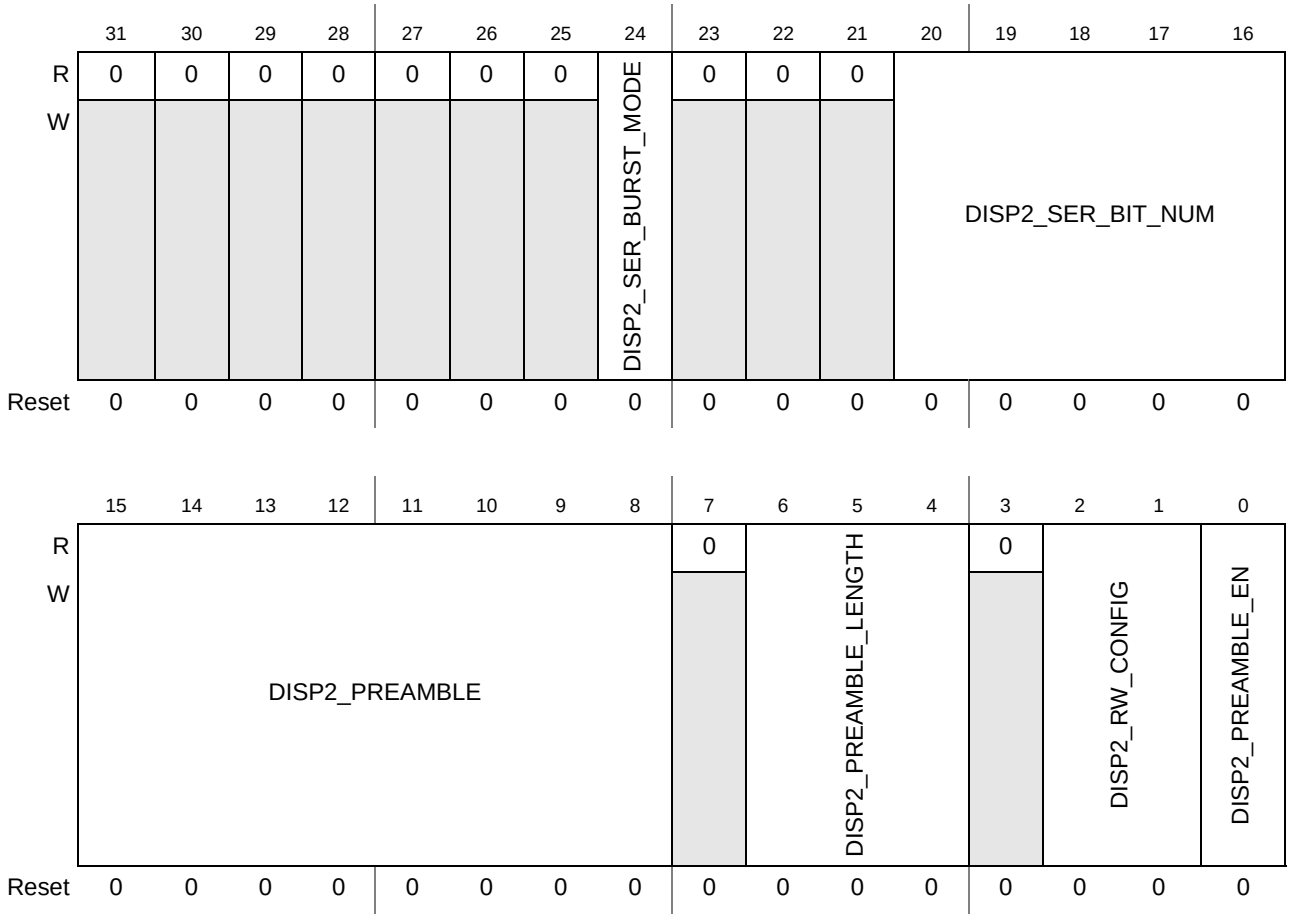


Figure 44-84. DI Serial Display 2 Configuration Register (DI_SER_DISP2_CONF)

Table 44-104. DI_SER_DISP2_CONF Field Descriptions

Field	Description
31–25	Reserved
24 DISP2_SER_BURST_MODE	Burst mode enable. Enables burst mode when the display chip select remains active during whole burst transfer. In single access mode, the display chip select is not active between display accesses. 0 single access mode. 1 burst access mode.
23–21	Reserved
20–16 DISP2_SER_BIT_NUM	Output/input data bit number minus 1. Defines number of data bits per display access. Values: 00000 1 bit .... 10001 18 bits (max)

Table 44-104. DI_SER_DISP2_CONF Field Descriptions (continued)

Field	Description
15–8 DISP2_PREAMBLE	Preamble contents. If DISP2_PREAMBLE_LENGTH is less than 8, only DISP2_PREAMBLE_LENGTH least significant bits of DISP2_PREAMBLE are sent.
7	Reserved
6–4 DISP2_PREAMBLE_LENGTH	Preamble length minus 1 in bits (without RS and RW bits). Values: 000 1 bit .... 101 6 bits 110 7 bits 111 8 bits (max)
3	Reserved
2–1 DISP2_RW_CONFIG	Configuration of read/write (RW) control bit to be sent via serial interface. For 5-wire serial interface (when RS is send via a separate wire), the ‘01’ and ‘10’ values of DISP2_RW_CONFIG have same effect. Values: 00 No RW bit in serial interface output sequence 01 RW bit is sent after preamble and before RS bit 10 RW bit is sent after preamble and after RS bit 11 reserved
0 DISP2_PREAMBLE_EN	Preamble enable. 0 disable preamble 1 enable preamble

44.3.3.8.5 DI HSP_CLK Period Register (DI_HSP_CLK_PER)

This register defines HSP_CLK_PERIOD for the DI. It is used to hold the correct value of all the display clock rates display clock rates after the HSP_CLK rate has been changed on-the-fly. The MCU should program this value before HSP_CLK clock rate change. switch between the HSP_CLK_PERIOD_1 and HSP_CLK_PERIOD_2 values is performed on the HSP_CLK_PER_SEL signal from the Clock Controller.

0x53FC_0134 (DI_HSP_CLK_PER) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	HSP_CLK_PERIOD_2						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	HSP_CLK_PERIOD_1						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-85. DI HSP_CLK Period Register (DI_HSP_CLK_PER)

Table 44-105. DI_HSP_CLK_PER Field Descriptions

Field	Description
31–23	Reserved
22–16 HSP_CLK_PERIOD_2	HSP_CLK period option 2. This parameter contains integer part (bits [6:4]) and fractional part (bits [3:0]).
15–7	Reserved
6–0 HSP_CLK_PERIOD_1	HSP_CLK period option 1. This parameter contains integer part (bits [6:4]) and fractional part (bits [3:0]).

44.3.3.8.6 DI Display 0 Time Configuration Register 1 (DI_DISP0_TIME_CONF_1)

This register contains the first part of timing configuration parameters for the display 0.

0x53FC_0138 (DI_DISP0_TIME_CONF_1) Access: User Read/Write

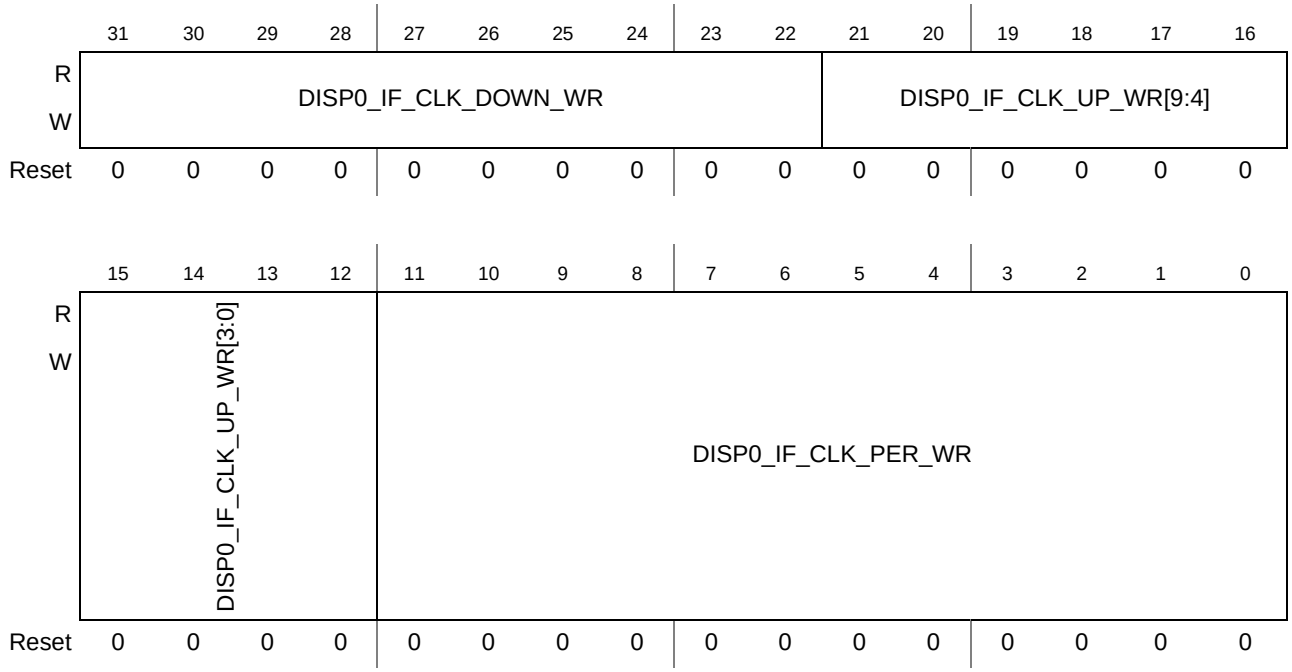


Figure 44-86. DI Display 0 Time Configuration Register 1 (DI_DISP0_TIME_CONF_1)

Table 44-106. DI_DISP0_TIME_CONF_1 Field Descriptions

Field	Description
31–22 DISP0_IF_CLK_DOWN_WR	Display 0 interface clock falling edge position for display write access. This parameter contains an integer part (bits 9:2) and a fractional part (bits 1:0). • The position value is a time interval between display write access start point and display 0 interface clock falling edge. • The actual position value is equal to CEIL[2*DISP0_IF_CLK_DOWN_WR/HSP_CLK_PERIOD_1(2)]/2 (in high speed clock (HSP_CLK) cycles). • The range of DISP0_IF_CLK_DOWN_WR should be from DISP0_IF_CLK_UP_WR+0.5*HSP_CLK_PERIOD_1(2) to DISP0_IF_CLK_PER_WR.
21–16 DISP0_IF_CLK_UP_WR[9:4]	

Table 44-106. DI_DISP0_TIME_CONF_1 Field Descriptions (continued)

Field	Description
15–12 DISP0_IF_CLK_UP_WR[3:0]	Display 0 interface clock rising edge position for display write access. • This parameter contains an integer part (bits 9:2) and a fractional part (bits 1: 0). The position is a time interval between display write access start point and display 0 interface clock rising edge. • The actual position value is equal to CEIL[2*DISP0_IF_CLK_UP_WR/HSP_CLK_PERIOD_1(2)]/2 (in high speed clock (HSP_CLK) cycles). • The range of DISP0_IF_CLK_UP_WR should be from 0 to DISP0_IF_CLK_PER_WR-0.5*HSP_CLK_PERIOD_1(2)
11–0 DISP0_IF_CLK_PER_WR	Display 0 interface clock period for display write access. • This parameter contains an integer part (bits 11:4) and a fractional part (bits 3:0). • The actual value of the interface clock period is equal to CEIL[DISP0_IF_CLK_PER_WR/HSP_CLK_PERIOD_1(2)] (in high speed clock (HSP_CLK) cycles) where CEIL(X) rounds the elements of X to the nearest integers towards infinity.

44.3.3.8.7 DI Display 0 Time Configuration Register 2 (DI_DISP0_TIME_CONF_2)

This register contains the second part of timing configuration parameters for the display 0.

0x53FC_013C (DI_DISP0_TIME_CONF_2) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DISP0_IF_CLK_DOWN_RD										DISP0_IF_CLK_UP_RD[9:4]					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DISP0_IF_CLK_UP_RD[3:0]				DISP0_IF_CLK_PER_RD											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-87. DI Display 0 Time Configuration Register 2 (DI_DISP0_TIME_CONF_2)

Table 44-107. DI_DISP0_TIME_CONF_2 Field Descriptions

Field	Description
31–22 DISP0_IF_CLK_DOWN_RD	Display 0 interface clock falling edge position for display read access. • This parameter contains an integer part (bits 9:2) and a fractional part (bits 1:0). The position value is a time interval between display read access start point and display 0 interface clock falling edge. • The actual position value is equal to CEIL[2*DISP0_IF_CLK_DOWN_RD/HSP_CLK_PERIOD_1(2)]/2 (in high speed clock (HSP_CLK) cycles) where CEIL(X) rounds the elements of X to the nearest integers towards infinity. • The range of DISP0_IF_CLK_DOWN_RD should be from DISP0_IF_CLK_UP_RD+0.5*HSP_CLK_PERIOD_1(2) to DISP0_IF_CLK_PER_RD.
21–16 DISP0_IF_CLK_UP_RD[9:4]	Display 0 interface clock rising edge position for display read access. • This parameter contains an integer part (bits 9:2) and a fractional part (bits 1: 0). • The position is a time interval between display read access start point and display 0 interface clock rising edge. • The actual position value is equal to CEIL[2*DISP0_IF_CLK_UP_RD/HSP_CLK_PERIOD_1(2)]/2 (in high speed clock (HSP_CLK) cycles) where CEIL(X) rounds the elements of X to the nearest integers towards infinity. The range of DISP0_IF_CLK_UP_RD should be from 0 to DISP0_IF_CLK_PER_RD-0.5*HSP_CLK_PERIOD_1(2).
15–12 DISP0_IF_CLK_UP_RD[3:0]	
11–0 DISP0_IF_CLK_PER_RD	Display 0 interface clock period for display read access. • This parameter contains an integer part (bits 11:4) and a fractional part (bits 3:0). • The actual value of the interface clock period is equal to CEIL[DISP0_IF_CLK_PER_RD/HSP_CLK_PERIOD_1(2)] (in high speed clock (HSP_CLK) cycles) where CEIL(X) rounds the elements of X to the nearest integers towards infinity.

44.3.3.8.8 DI Display 0 Time Configuration Register 3 (DI_DISP0_TIME_CONF_3)

This register contains the third part of timing configuration parameters for the display 0.

0x53FC_0140 (DI_DISP0_TIME_CONF_3) Access: User Read/Write

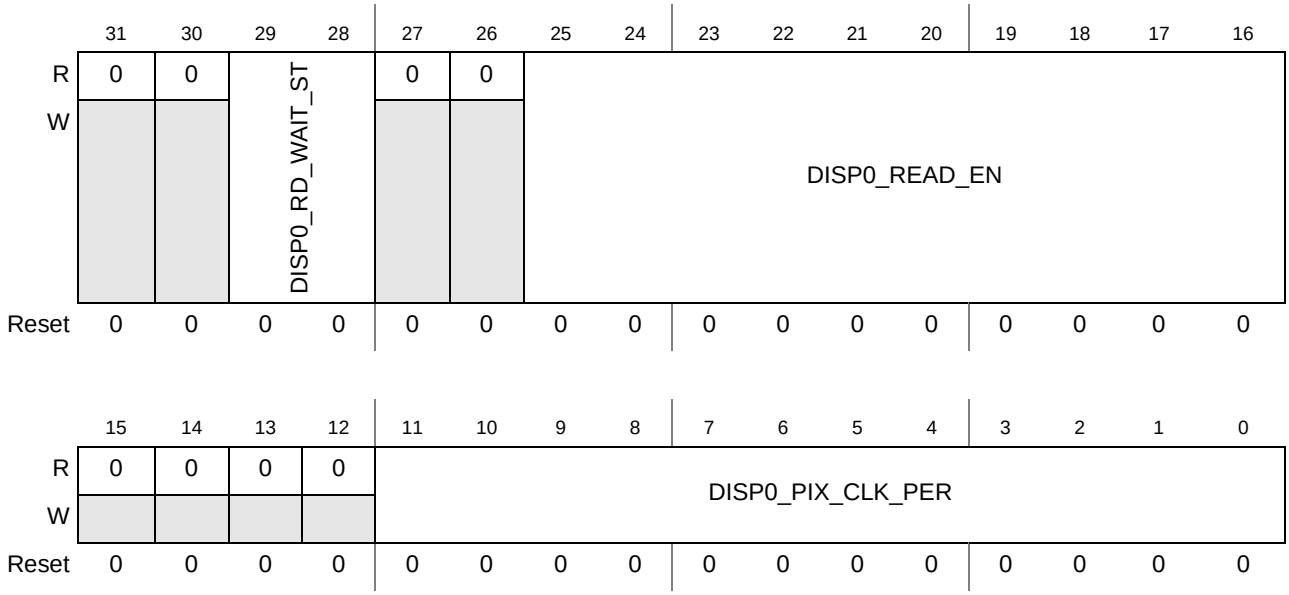


Figure 44-88. DI Display 0 Time Configuration Register 3 (DI_DISP0_TIME_CONF_3)

Table 44-108. DI_DISP0_TIME_CONF_3 Field Descriptions

Field	Description
31–30	Reserved
29–28 DISP0_RD_WAIT_ST	Contains the number of wait states required to read data from the display. Values: 00 0 wait states 01 1 wait state 10 2 wait states 11 3 wait states
25–16 DISP0_READ_EN	Display 0 read point position. • The position is a time interval between display 0 read access start and data read points. This parameter contains an integer part (bits 9:2) and a fractional part (bits 1: 0). • The actual position value is equal to CEIL[DISP0_READ_EN/HSP_CLK_PERIOD_1(2) (in high speed clock (HSP_CLK) periods) where CEIL(X) rounds the elements of X to the nearest integers towards infinity. The range of DISP0_READ_EN is from 0 to DISP0_IF_CLK_PER_RD.
15–12	Reserved
11–0 DISP0_PIX_CLK_PER	Display 0 pixel clock period. • This parameter contains an integer part (bits 11:4) and a fractional part (bits 3:0). It defines fractional division ratio of the HSP_CLK clock for generation of the display 0 pixel clock. • The actual average value of the pixel clock period is equal to DISP0_PIX_CLK_PER/HSP_CLK_PERIOD_1(2) high speed clock (HSP_CLK) periods.

44.3.3.8.9 DI Display 1 Time Configuration Register 1 (DI_DISP1_TIME_CONF_1)

This register contains the first part of timing configuration parameters for the display 1.

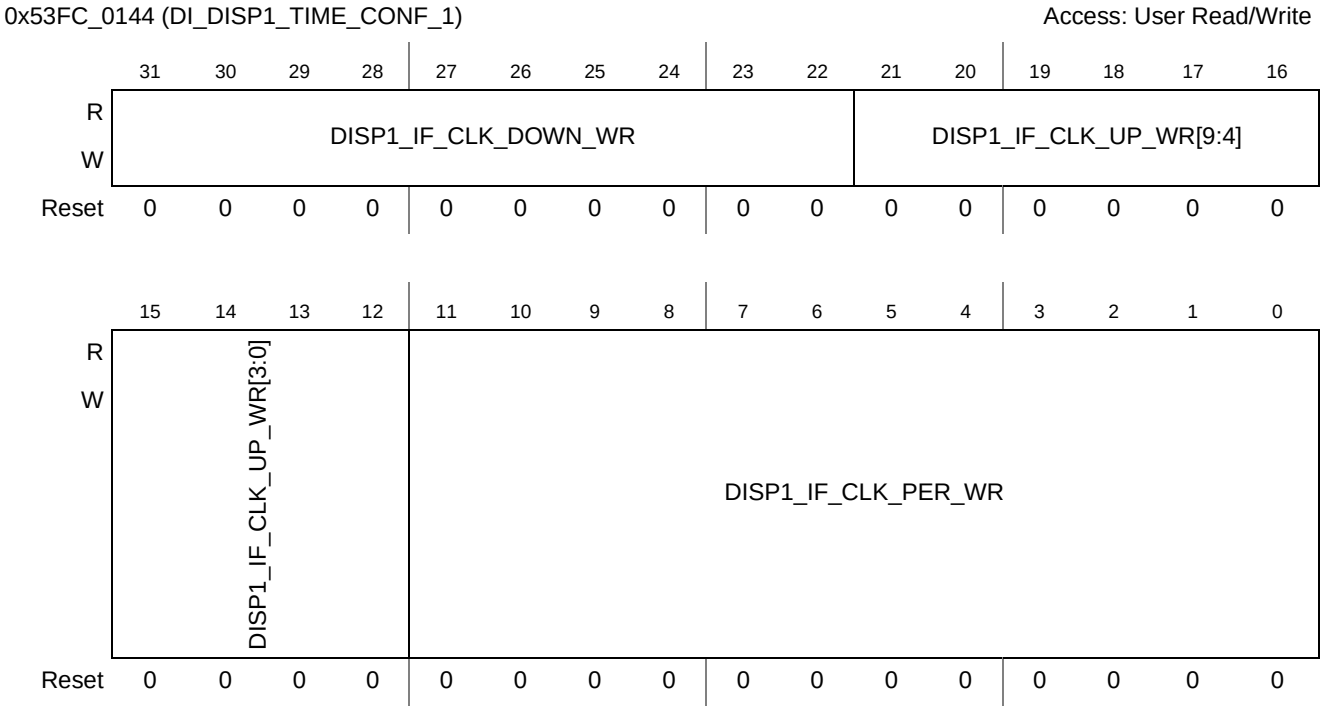


Figure 44-89. DI Display 1 Time Configuration Register 1 (DI_DISP1_TIME_CONF_1)

Table 44-109. DI_DISP1_TIME_CONF_1 Field Descriptions

Field	Description
31–22 DISP1_IF_CLK_DOWN_WR	Display 1 interface clock falling edge position for display write access. • This parameter contains an integer part (bits 9:2) and a fractional part (bits 1:0). • The position value is a time interval between display write access start point and display 1 interface clock falling edge. • The actual position value is equal to CEIL[2*DISP1_IF_CLK_DOWN_WR/HSP_CLK_PERIOD_1(2)]/2 (in high speed clock (HSP_CLK) cycles). The range of DISP1_IF_CLK_DOWN_WR should be from DISP1_IF_CLK_UP_WR+0.5*HSP_CLK_PERIOD_1(2) to DISP1_IF_CLK_PER_WR.

Table 44-109. DI_DISP1_TIME_CONF_1 Field Descriptions (continued)

Field	Description
21–16 DISP1_IF_CLK_UP_WR[9:4]	Display 1 interface clock rising edge position for display write access. • This parameter contains an integer part (bits 9:2) and a fractional part (bits 1: 0). • The position is a time interval between display write access start point and display 1 interface clock rising edge. • The actual position value is equal to CEIL[2*DISP1_IF_CLK_UP_WR/HSP_CLK_PERIOD_1(2)]/2 (in high speed clock (HSP_CLK) cycles). The range of DISP1_IF_CLK_UP_WR should be from 0 to DISP1_IF_CLK_PER_WR-0.5*HSP_CLK_PERIOD_1(2).
15–12 DISP1_IF_CLK_UP_WR[3:0]	
11–0 DISP1_IF_CLK_PER_WR	Display 1 interface clock period for display write access. • This parameter contains an integer part (bits 11:4) and a fractional part (bits 3:0). • The actual value of the interface clock period is equal to CEIL[DISP1_IF_CLK_PER_WR/HSP_CLK_PERIOD_1(2)] (in high speed clock (HSP_CLK) cycles) where CEIL(X) rounds the elements of X to the nearest integers towards infinity.

44.3.3.8.10 DI Display 1 Time Configuration Register 2 (DI_DISP1_TIME_CONF_2)

This register contains the second part of timing configuration parameters for the display 1.

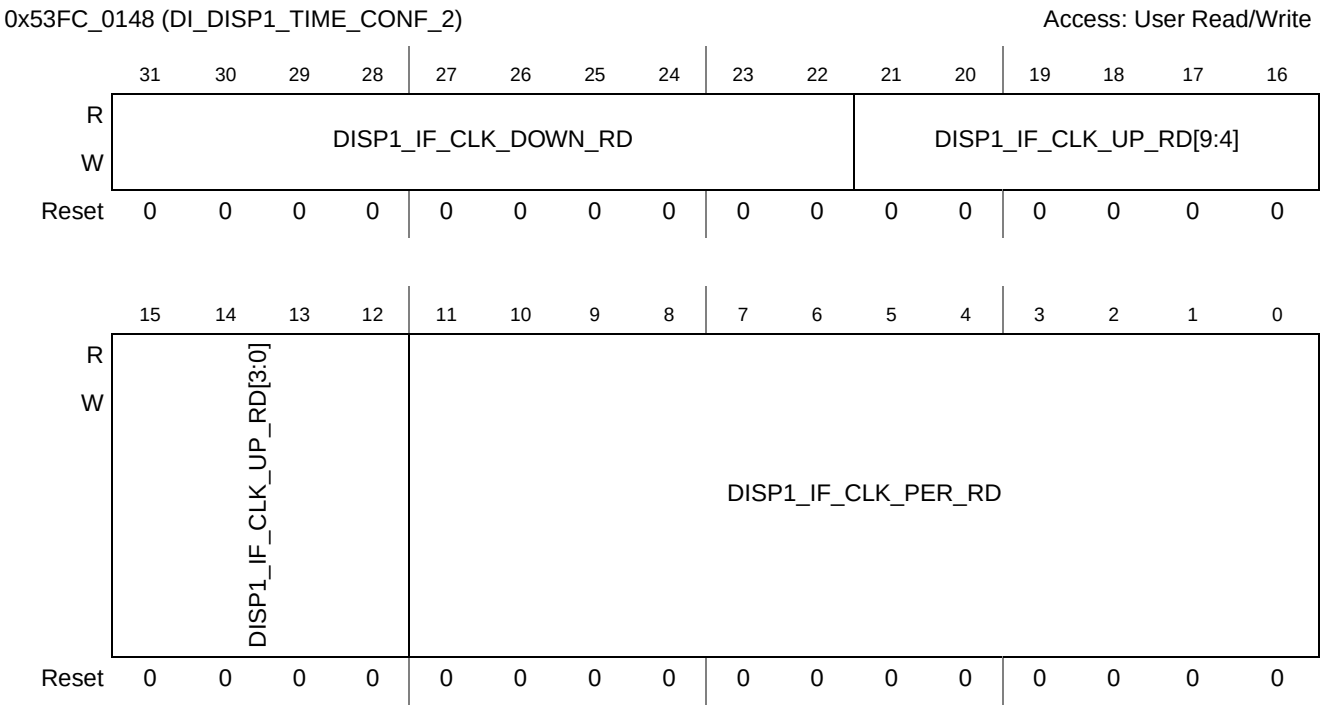


Figure 44-90. DI Display 1 Time Configuration Register 2 (DI_DISP1_TIME_CONF_2)

Table 44-110. DI_DISP1_TIME_CONF_2 Field Descriptions

Field	Description
31–22 DISP1_IF_CLK_DOWN_RD	Display 1 interface clock falling edge position for display read access. • This parameter contains an integer part (bits 9:2) and a fractional part (bits 1:0). • The position value is a time interval between display read access start point and display 1 interface clock falling edge. • The actual position value is equal to CEIL[2*DISP1_IF_CLK_DOWN_RD/HSP_CLK_PERIOD_1(2)]/2 (in high speed clock (HSP_CLK) cycles) where CEIL(X) rounds the elements of X to the nearest integers towards infinity. The range of DISP1_IF_CLK_DOWN_RD should be from DISP1_IF_CLK_UP_RD+0.5*HSP_CLK_PERIOD_1(2) to DISP1_IF_CLK_PER_RD.
21–16 DISP1_IF_CLK_UP_RD[9:4]	Display 1 interface clock rising edge position for display read access. • This parameter contains an integer part (bits 9:2) and a fractional part (bits 1: 0). • The position is a time interval between display read access start point and display 1 interface clock rising edge. • The actual position value is equal to CEIL[2*DISP1_IF_CLK_UP_RD/HSP_CLK_PERIOD_1(2)]/2 (in high speed clock (HSP_CLK) cycles) where CEIL(X) rounds the elements of X to the nearest integers towards infinity. The range of DISP1_IF_CLK_UP_RD should be from 0 to DISP1_IF_CLK_PER_RD-0.5*HSP_CLK_PERIOD_1(2).
15–12 DISP1_IF_CLK_UP_RD[3:0]	
11–0 DISP1_IF_CLK_PER_RD	Display 1 interface clock period for display read access. • This parameter contains an integer part (bits 11:4) and a fractional part (bits 3:0). • The actual value of the interface clock period is equal to CEIL[DISP1_IF_CLK_PER_RD/HSP_CLK_PERIOD_1(2)] (in high speed clock (HSP_CLK) cycles) where CEIL(X) rounds the elements of X to the nearest integers towards infinity.

44.3.3.8.11 DI Display 1 Time Configuration Register 3 (DI_DISP1_TIME_CONF_3)

This register contains the third part of timing configuration parameters for the display 1.

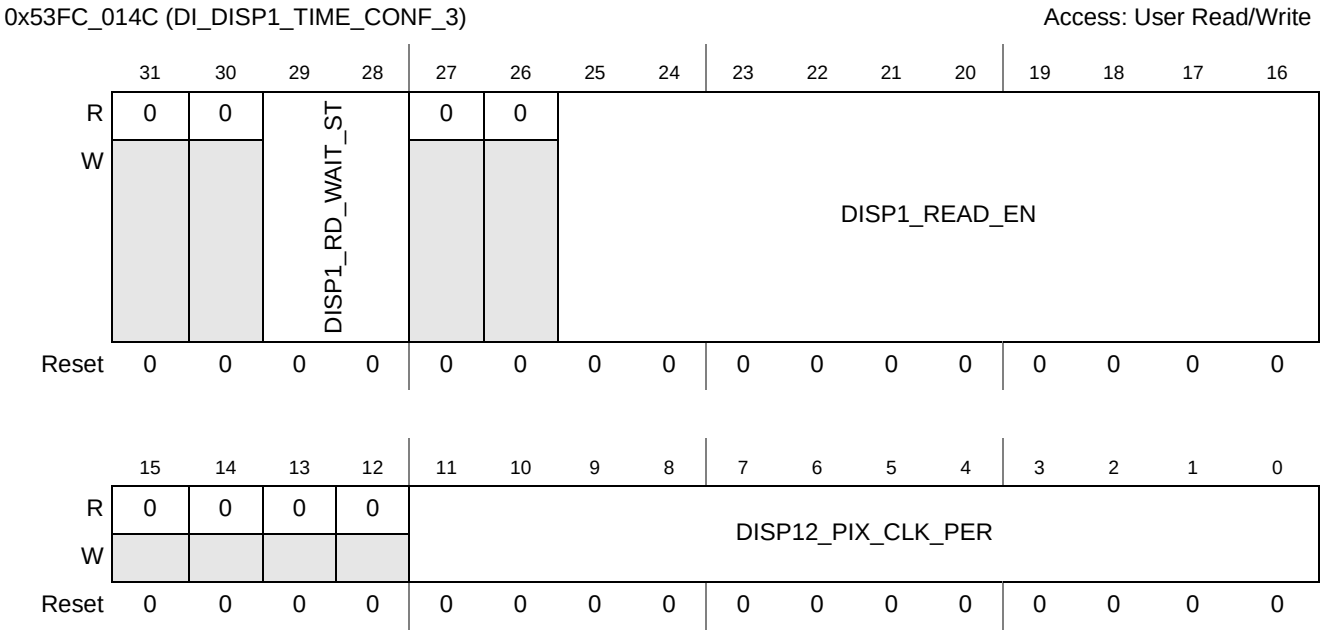


Figure 44-91. DI Display 1 Time Configuration Register 3 (DI_DISP1_TIME_CONF_3)

Table 44-111. DI_DISP1_TIME_CONF_3 Field Descriptions

Field	Description
31–30	Reserved
29–28 DISP1_RD_WAIT_ST	Contains the number of wait states required to read data from the display. Values: 00 0 wait states 01 1 wait state 10 2 wait states 11 3 wait states
25–16 DISP1_READ_EN	Display 1 read point position. • The position is a time interval between display 1 read access start and data read points. This parameter contains an integer part (bits 9:2) and a fractional part (bits 1: 0). • The actual position value is equal to CEIL[DISP1_READ_EN/HSP_CLK_PERIOD_1(2)] (in high speed clock (HSP_CLK) periods) where CEIL(X) rounds the elements of X to the nearest integers towards infinity. The range of DISP1_READ_EN is from 0 to DISP1_IF_CLK_PER_RD.

Table 44-111. DI_DISP1_TIME_CONF_3 Field Descriptions (continued)

Field	Description
15–12	Reserved
11–0 DISP12_PIX_CLK_PER	Displays 1 and 2 pixel clock period. • This parameter contains an integer part (bits 11:4) and a fractional part (bits 3:0). It defines fractional division ratio of the HSP_CLK clock for generation of the displays 1 and 2 pixel clock. • The actual average value of the pixel clock period is equal to DISP12_PIX_CLK_PER/HSP_CLK_PERIOD_1(2) high speed clock (HSP_CLK) periods.

44.3.3.8.12 DI Display 2 Time Configuration Register 1 (DI_DISP2_TIME_CONF_1)

This register contains the first part of timing configuration parameters for the display 2.

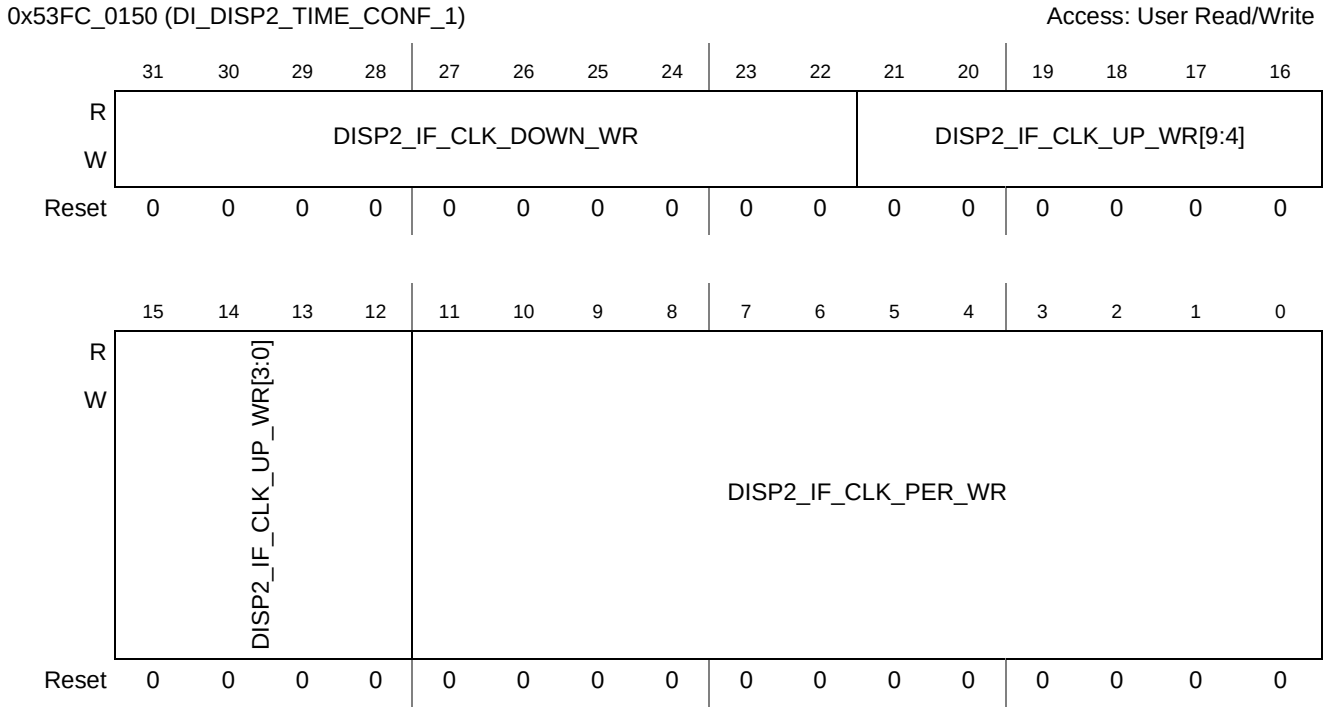


Figure 44-92. DI Display 2 Time Configuration Register 1 (DI_DISP2_TIME_CONF_1)

Table 44-112. DI_DISP2_TIME_CONF_1 Field Descriptions

Field	Description
31–22 DISP2_IF_CLK_DOWN_WR	Display 2 interface clock falling edge position for display write access. • This parameter contains an integer part (bits 9:2) and a fractional part (bits 1:0). • The position value is a time interval between display write access start point and display 2 interface clock falling edge. • The actual position value is equal to CEIL[2*DISP2_IF_CLK_DOWN_WR/HSP_CLK_PERIOD_1(2)]/2 (in high speed clock (HSP_CLK) cycles). The range of DISP2_IF_CLK_DOWN_WR should be from DISP2_IF_CLK_UP_WR+0.5*HSP_CLK_PERIOD_1(2) to DISP2_IF_CLK_PER_WR.
21–16 DISP2_IF_CLK_UP_WR[9:4]	Display 2 interface clock rising edge position for display write access. • This parameter contains an integer part (bits 9:2) and a fractional part (bits 1: 0). • The position is a time interval between display write access start point and display 2 interface clock rising edge. • The actual position value is equal to CEIL[2*DISP2_IF_CLK_UP_WR/HSP_CLK_PERIOD_1(2)]/2 (in high speed clock (HSP_CLK) cycles). The range of DISP2_IF_CLK_UP_WR should be from 0 to DISP2_IF_CLK_PER_WR-0.5*HSP_CLK_PERIOD_1(2).
15–12 DISP2_IF_CLK_UP_WR[3:0]	
11–0 DISP2_IF_CLK_PER_WR	Display 2 interface clock period for display write access. • This parameter contains an integer part (bits 11:4) and a fractional part (bits 3:0). • The actual value of the interface clock period is equal to CEIL[DISP2_IF_CLK_PER_WR/HSP_CLK_PERIOD_1(2)] (in high speed clock (HSP_CLK) cycles) where CEIL(X) rounds the elements of X to the nearest integers towards infinity.

44.3.3.8.13 DI Display 2 Time Configuration Register 2 (DI_DISP2_TIME_CONF_2)

This register contains the second part of timing configuration parameters for the display 2.

0x53FC_0154 (DI_DISP2_TIME_CONF_2) Access: User Read/Write

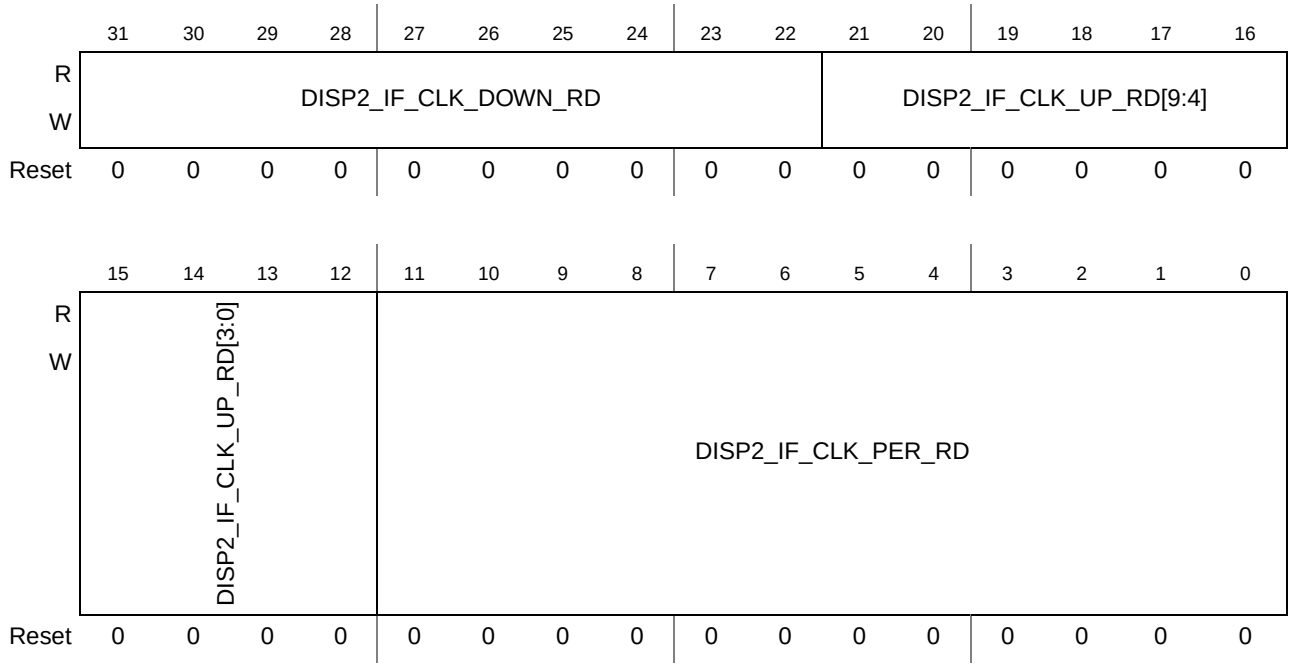


Figure 44-93. DI Display 2 Time Configuration Register 2 (DI_DISP2_TIME_CONF_2)

Table 44-113. DI_DISP2_TIME_CONF_2 Field Descriptions

Field	Description
31–22 DISP2_IF_CLK_DOWN_RD	Display 2 interface clock falling edge position for display read access. • This parameter contains an integer part (bits 9:2) and a fractional part (bits 1:0). • The position value is a time interval between display read access start point and display 2 interface clock falling edge. • The actual position value is equal to CEIL[2*DISP2_IF_CLK_DOWN_RD/HSP_CLK_PERIOD_1(2)]/2 (in high speed clock (HSP_CLK) cycles) where CEIL(X) rounds the elements of X to the nearest integers towards infinity. The range of DISP2_IF_CLK_DOWN_RD should be from DISP2_IF_CLK_UP_RD+0.5*HSP_CLK_PERIOD_1(2) to DISP2_IF_CLK_PER_RD.

Table 44-113. DI_DISP2_TIME_CONF_2 Field Descriptions (continued)

Field	Description
21–16 DISP2_IF_CLK_UP_RD[9:4]	Display 2 interface clock rising edge position for display read access. • This parameter contains an integer part (bits 9:2) and a fractional part (bits 1: 0). • The position is a time interval between display read access start point and display 2 interface clock rising edge. • The actual position value is equal to CEIL[2*DISP2_IF_CLK_UP_RD/HSP_CLK_PERIOD_1(2)]/2 (in high speed clock (HSP_CLK) cycles) where CEIL(X) rounds the elements of X to the nearest integers towards infinity. The range of DISP2_IF_CLK_UP_RD should be from 0 to DISP2_IF_CLK_PER_RD-0.5*HSP_CLK_PERIOD_1(2).
15–12 DISP2_IF_CLK_UP_RD[3:0]	
11–0 DISP2_IF_CLK_PER_RD	Display 2 interface clock period for display read access. • This parameter contains an integer part (bits 11:4) and a fractional part (bits 3:0). • The actual value of the interface clock period is equal to CEIL[DISP2_IF_CLK_PER_RD/HSP_CLK_PERIOD_1(2)] (in high speed clock (HSP_CLK) cycles) where CEIL(X) rounds the elements of X to the nearest integers towards infinity.

44.3.3.8.14 DI Display 2 Time Configuration Register 3 (DI_DISP2_TIME_CONF_3)

This register contains the third part of timing configuration parameters for the display 2.

0x53FC_0158 (DI_DISP2_TIME_CONF_3) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	DISP2_RD_WAIT_ST		0	0	DISP2_READ_EN									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-94. DI Display 2 Time Configuration Register 3 (DI_DISP2_TIME_CONF_3)

Table 44-114. DI_DISP2_TIME_CONF_3 Field Descriptions

Field	Description
31–30	Reserved
29–28 DISP2_RD_WAIT_ST	Contains the number of wait states required to read data from the display. Values: 00 0 wait states 01 1 wait state 10 2 wait states 11 3 wait states
27–26	Reserved
25–16 DISP2_READ_EN	Display 2 read point position. • The position is a time interval between display 2 read access start and data read points. • This parameter contains an integer part (bits 9:2) and a fractional part (bits 1: 0). • The actual position value is equal to CEIL[DISP2_READ_EN/HSP_CLK_PERIOD_1(2)] (in high speed clock (HSP_CLK) periods) where CEIL(X) rounds the elements of X to the nearest integers towards infinity. The range of DISP2_READ_EN is from 0 to DISP2_IF_CLK_PER_RD.
15–0	Reserved

44.3.3.8.15 DI Display 3 Time Configuration Register (DI_DISP3_TIME_CONF)

This register contains the first part of timing configuration parameters for the display 3.

0x53FC_015C (DI_DISP3_TIME_CONF) Access: User Read/Write

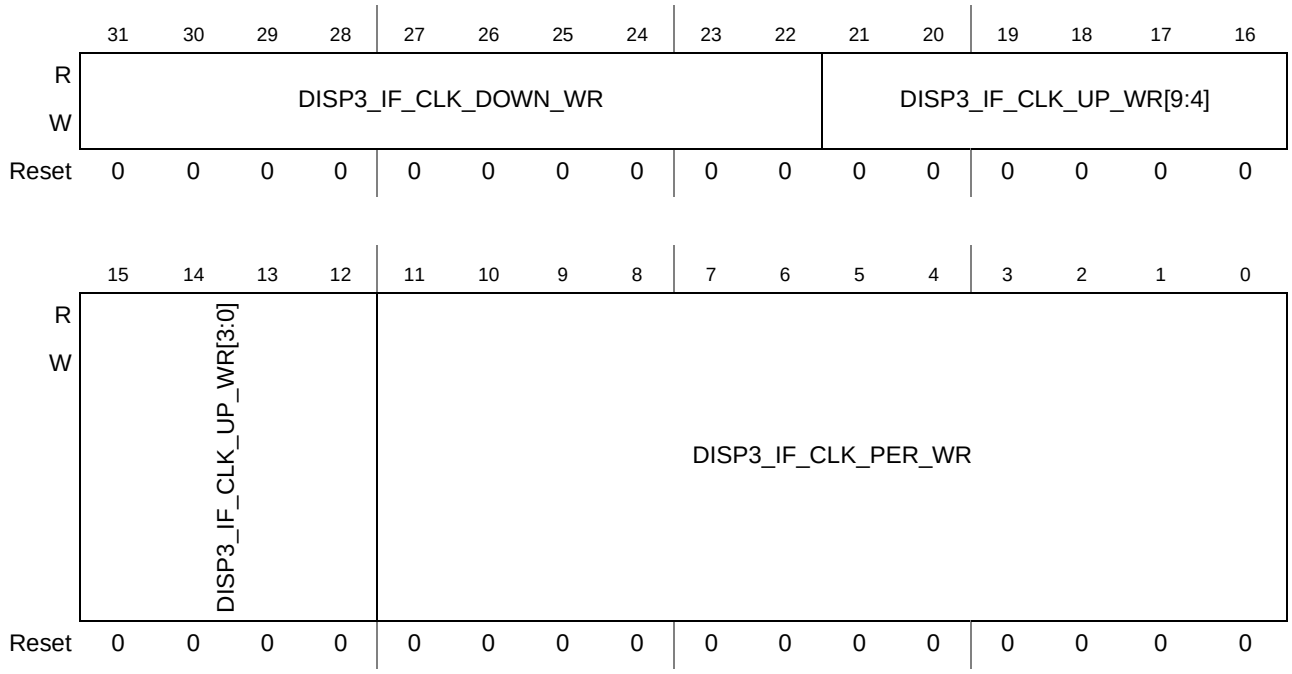


Figure 44-95. DI Display 3 Time Configuration Register (DI_DISP3_TIME_CONF)

Table 44-115. DI_DISP3_TIME_CONF Field Descriptions

Field	Description
31–22 DISP3_IF_CLK_DOWN_WR	Display 3 interface clock falling edge position for display write access. • This parameter contains an integer part (bits 9:2) and a fractional part (bits 1:0). • The position value is a time interval between display write access start point and display 3 interface clock falling edge. • The actual position value is equal to CEIL[2*DISP3_IF_CLK_DOWN_WR/HSP_CLK_PERIOD_1(2)]/2 (in high speed clock (HSP_CLK) cycles) where CEIL(X) rounds the elements of X to the nearest integers towards infinity. The range of DISP3_IF_CLK_DOWN_WR should be from DISP3_IF_CLK_UP_WR+0.5*HSP_CLK_PERIOD_1(2) to DISP3_IF_CLK_PER_WR.

Table 44-115. DI_DISP3_TIME_CONF Field Descriptions (continued)

Field	Description
21–16 DISP3_IF_CLK_UP_WR[9:4]	Display 3 interface clock rising edge position for display write access. • This parameter contains an integer part (bits 9:2) and a fractional part (bits 1: 0). • The position is a time interval between display write access start point and display 3 interface clock rising edge. • The actual position value is equal to CEIL[2*DISP3_IF_CLK_UP_WR/HSP_CLK_PERIOD_1(2)]/2 (in high speed clock (HSP_CLK) cycles) where CEIL(X) rounds the elements of X to the nearest integers towards infinity. The range of DISP3_IF_CLK_UP_WR should be from 0 to DISP3_IF_CLK_PER_WR-0.5*HSP_CLK_PERIOD_1(2).
15–12 DISP3_IF_CLK_UP_WR[3:0]	
11–0 DISP3_IF_CLK_PER_WR	Display 3 interface clock period for display write access. • This parameter contains an integer part (bits 11:4) and a fractional part (bits 3:0). It defines fractional division ratio of the HSP_CLK clock for generation of the displays 3 interface clock. • The actual average value of the interface clock period is equal to DISP3_IF_CLK_PER_WR/HSP_CLK_PERIOD_1(2) (in high speed clock (HSP_CLK) cycles).

44.3.3.8.16 DI Display 0 Data Byte 0 Mapping Register (DI_DISP0_DB0_MAP)

This register defines masks and offsets of data byte 0 (least significant byte) for display 0.

0x53FC_0160 (DI_DISP0_DB0_MAP) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	MD00_OFFS2					MD00_OFFS1						MD00_OFFS0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MD00_M7		MD00_M6		MD00_M5		MD00_M4		MD00_M3		MD00_M2		MD00_M1		MD00_M0	
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 44-96. DI Display 3 Data Byte 0 Mapping Register (DI_DISP0_DB0_MAP)

Table 44-116. DI_DISP0_DB0_MAP Field Descriptions

Field	Description
31	Reserved
30–26 MD00_OFFS2	Offset in third clock cycle. This bit define the position of the byte most significant bit in the third clock cycle.
25–21 MD00_OFFS1	Offset in second clock cycle. This bit define the position of the byte most significant bit in the second clock cycle.

Table 44-116. DI_DISP0_DB0_MAP Field Descriptions (continued)

Field	Description
20–16 MD00_OFFS0	Offset in first clock cycle. This bit defines the position of the byte most significant bit in the first clock cycle.
15–14 MD00_M7	Masks for bit 0 (least significant bit). These fields specify in which clock cycle the bit should be sent to the display. Values: 00 Enable in first clock cycle 01 Enable in second clock cycle 10 Enable in third clock cycle 11 Masked
13–12 MD00_M6	
11–10 MD00_M5	
9–8 MD00_M4	
7–6 MD00_M3	
5–4 MD00_M2	
3–2 MD00_M1	
1–0 MD00_M0	

44.3.3.8.17 DI Display 0 Data Byte 1 Mapping Register (DI_DISP0_DB1_MAP)

This register defines masks and offsets of data byte 1 (middle byte) for display 0.

0x53FC_0164 (DI_DISP0_DB1_MAP) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	MD01_OFFS2					MD01_OFFS1						MD01_OFFS0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MD01_M7		MD01_M6		MD01_M5		MD01_M4		MD01_M3		MD01_M2		MD01_M1		MD01_M0	
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 44-97. DI Display 0 Data Byte 1 Mapping Register (DI_DISP0_DB1_MAP)

Table 44-117. DI_DISP0_DB1_MAP Field Descriptions

Field	Description
31	Reserved
30–26 MD01_OFFS2	Offset in third clock cycle. This bit define the position of the byte most significant bit in the third clock cycle.
25–21 MD01_OFFS1	Offset in second clock cycle. This bit define the position of the byte most significant bit in the second clock cycle.
20–16 MD01_OFFS0	Offset in first clock cycle. This bit define the position of the byte most significant bit in the first clock cycle.
15–14 MD01_M7	Masks for bit 0 (least significant bit). These fields specify in which clock cycle the bit should be sent to the display. Values: 00 Enable in first clock cycle 01 Enable in second clock cycle 10 Enable in third clock cycle 11 Masked
13–12 MD01_M6	
11–10 MD01_M5	
9–8 MD01_M4	
7–6 MD01_M3	
5–4 MD01_M2	
3–2 MD01_M1	
1–0 MD01_M0	

44.3.3.8.18 DI Display 0 Data Byte 2 Mapping Register (DI_DISP0_DB2_MAP)

This register defines masks and offsets of data byte 2 (most significant byte) for display 0.

0x53FC_0168 (DI_DISP0_DB2_MAP) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	MD02_OFFS2					MD02_OFFS1						MD02_OFFS0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MD02_M7		MD02_M6		MD02_M5		MD02_M4		MD02_M3		MD02_M2		MD02_M1		MD02_M0	
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 44-98. DI Display 0 Data Byte 2 Mapping Register (DI_DISP0_DB2_MAP)

Table 44-118. DI_DISP0_DB2_MAP Field Descriptions

Field	Description
31	Reserved
30–26 MD02_OFFS2	Offset in third clock cycle. This bit defines the position of the byte most significant bit in the third clock cycle.
25–21 MD02_OFFS1	Offset in second clock cycle. This bit defines the position of the byte most significant bit in the second clock cycle.
20–16 MD02_OFFS0	Offset in first clock cycle. This bit defines the position of the byte most significant bit in the first clock cycle.
15–14 MD02_M7	Masks for bit 0 (least significant bit). These fields specify in which clock cycle the bit should be sent to the display. Values: 00 Enable in first clock cycle 01 Enable in second clock cycle 10 Enable in third clock cycle 11 Masked
13–12 MD02_M6	
11–10 MD02_M5	
9–8 MD02_M4	
7–6 MD02_M3	
5–4 MD02_M2	
3–2 MD02_M1	
1–0 MD02_M0	

44.3.3.8.19 DI Display 0 Command Byte 0 Mapping Register (DI_DISP0_CB0_MAP)

This register defines masks and offsets of command byte 0 (least significant byte) for display 0.

0x53FC_016C (DI_DISP0_CB0_MAP) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	MC00_OFFS2					MC00_OFFS1						MC00_OFFS0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MC00_M7		MC00_M6		MC00_M5		MC00_M4		MC00_M3		MC00_M2		MC00_M1		MC00_M0	
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 44-99. DI Display 0 Command Byte 0 Mapping Register (DI_DISP0_CB0_MAP)

Table 44-119. DI_DISP0_CB0_MAP Field Descriptions

Field	Description
31	Reserved
30–26 MC00_OFFS2	Offset in third clock cycle. This bit defines the position of the byte most significant bit in the third clock cycle.
25–21 MC00_OFFS1	Offset in second clock cycle. This bit defines the position of the byte most significant bit in the second clock cycle.
20–16 MC00_OFFS0	Offset in first clock cycle. This bit define the position of the byte most significant bit in the first clock cycle.

Table 44-119. DI_DISP0_CB0_MAP Field Descriptions (continued)

Field	Description
15–14 MC00_M7	Masks for bit 0 (least significant bit). These fields specify in which clock cycle the bit should be sent to the display. Values: 00 Enable in first clock cycle 01 Enable in second clock cycle 10 Enable in third clock cycle 11 Masked
13–12 MC00_M6	
11–10 MC00_M5	
9–8 MC00_M4	
7–6 MC00_M3	
5–4 MC00_M2	
3–2 MC00_M1	
1–0 MC00_M0	

44.3.3.8.20 DI Display 0 Command Byte 1 Mapping Register (DI_DISP0_CB1_MAP)

This register defines masks and offsets of command byte 1 (middle byte) for display 0.

0x53FC_0170 (DI_DISP0_CB1_MAP) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	MC01_OFFS2					MC01_OFFS1						MC01_OFFS0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MC01_M7		MC01_M6		MC01_M5		MC01_M4		MC01_M3		MC01_M2		MC01_M1		MC01_M0	
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 44-100. DI Display 0 Command Byte 1 Mapping Register (DI_DISP0_CB1_MAP)

Table 44-120. DI_DISP0_CB1_MAP Field Descriptions

Field	Description
31	Reserved
30–26 MC01_OFFS2	Offset in third clock cycle. This bit defines the position of the byte most significant bit in the third clock cycle.

Table 44-120. DI_DISP0_CB1_MAP Field Descriptions (continued)

Field	Description
25–21 MC01_OFFS1	Offset in second clock cycle. This bit defines the position of the byte most significant bit in the second clock cycle.
20–16 MC01_OFFS0	Offset in first clock cycle. This bit define the position of the byte most significant bit in the first clock cycle.
15–14 MC01_M7	Masks for bit 0 (least significant bit). These fields specify in which clock cycle the bit should be sent to the display. Values: 00 Enable in first clock cycle 01 Enable in second clock cycle 10 Enable in third clock cycle 11 Masked
13–12 MC01_M6	
11–10 MC01_M5	
9–8 MC01_M4	
7–6 MC01_M3	
5–4 MC01_M2	
3–2 MC01_M1	
1–0 MC01_M0	

44.3.3.8.21 DI Display 0 Command Byte 2 Mapping Register (DI_DISP0_CB2_MAP)

This register defines masks and offsets of command byte 2 (most significant byte) for display 0.

0x53FC_0174 (DI_DISP0_CB2_MAP) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	MC02_OFFS2					MC02_OFFS1						MC02_OFFS0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MC02_M7		MC02_M6		MC02_M5		MC02_M4		MC02_M3		MC02_M2		MC02_M1		MC02_M0	
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 44-101. DI Display 0 Command Byte 2 Mapping Register (DI_DISP0_CB2_MAP)

Table 44-121. DI_DISP0_CB2_MAP Field Descriptions

Field	Description
31	Reserved
30–26 MC02_OFFS2	Offset in third clock cycle. This bit defines the position of the byte most significant bit in the third clock cycle.
25–21 MC02_OFFS1	Offset in second clock cycle. This bit defines the position of the byte most significant bit in the second clock cycle.
20–16 MC02_OFFS0	Offset in first clock cycle. This bit define the position of the byte most significant bit in the first clock cycle.
15–14 MC02_M7	Masks for bit 0 (least significant bit). These fields specify in which clock cycle the bit should be sent to the display. Values: 00 Enable in first clock cycle 01 Enable in second clock cycle 10 Enable in third clock cycle 11 Masked
13–12 MC02_M6	
11–10 MC02_M5	
9–8 MC02_M4	
7–6 MC02_M3	
5–4 MC02_M2	
3–2 MC02_M1	
1–0 MC02_M0	

44.3.3.8.22 DI Display 1 Data Byte 0 Mapping Register (DI_DISP1_DB0_MAP)

This register defines masks and offsets of data byte 0 (least significant byte) for display 1.

0x53FC_0178 (DI_DISP1_DB0_MAP) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	MD10_OFFS2					MD10_OFFS1						MD10_OFFS0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MD10_M7		MD10_M6		MD10_M5		MD10_M4		MD10_M3		MD10_M2		MD10_M1		MD10_M0	
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 44-102. DI Display 1 Data Byte 0 Mapping Register (DI_DISP1_DB0_MAP)

Table 44-122. DI_DISP1_DB0_MAP Field Descriptions

Field	Description
31	Reserved
30–26 MD10_OFFS2	Offset in third clock cycle. This bit defines the position of the byte most significant bit in the third clock cycle.
25–21 MD10_OFFS1	Offset in second clock cycle. This bit defines the position of the byte most significant bit in the second clock cycle.
20–16 MD10_OFFS0	Offset in first clock cycle. This bit define the position of the byte most significant bit in the first clock cycle.
15–14 MD10_M7	Masks for bit 0 (least significant bit). These fields specify in which clock cycle the bit should be sent to the display. Values: 00 Enable in first clock cycle 01 Enable in second clock cycle 10 Enable in third clock cycle 11 Masked
13–12 MD10_M6	
11–10 MD10_M5	
9–8 MD10_M4	
7–6 MD10_M3	
5–4 MD10_M2	
3–2 MD10_M1	
1–0 MD10_M0	

44.3.3.8.23 DI Display 1 Data Byte 1 Mapping Register (DI_DISP1_DB1_MAP)

This register defines masks and offsets of data byte 1 (middle byte) for display 1.

0x53FC_017C (DI_DISP1_DB1_MAP) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	MD11_OFFS2					MD11_OFFS1						MD11_OFFS0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MD11_M7		MD11_M6		MD11_M5		MD11_M4		MD11_M3		MD11_M2		MD11_M1		MD11_M0	
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 44-103. DI Display 1 Data Byte 1 Mapping Register (DI_DISP1_DB1_MAP)

Table 44-123. DI_DISP1_DB1_MAP Field Descriptions

Field	Description
31	Reserved
30–26 MD11_OFFS2	Offset in third clock cycle. This bit defines the position of the byte most significant bit in the third clock cycle.
25–21 MD11_OFFS1	Offset in second clock cycle. This bit defines the position of the byte most significant bit in the second clock cycle.
20–16 MD11_OFFS0	Offset in first clock cycle. This bit define the position of the byte most significant bit in the first clock cycle.

Table 44-123. DI_DISP1_DB1_MAP Field Descriptions (continued)

Field	Description
15–14 MD11_M7	Masks for bit 0 (least significant bit). These fields specify in which clock cycle the bit should be sent to the display. Values: 00 Enable in first clock cycle 01 Enable in second clock cycle 10 Enable in third clock cycle 11 Masked
13–12 MD11_M6	
11–10 MD11_M5	
9–8 MD11_M4	
7–6 MD11_M3	
5–4 MD11_M2	
3–2 MD11_M1	
1–0 MD11_M0	

44.3.3.8.24 DI Display 1 Data Byte 2 Mapping Register (DI_DISP1_DB2_MAP)

This register defines masks and offsets of data byte 2 (most significant byte) for display 1.

0x53FC_0180 (DI_DISP1_DB2_MAP) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	MD12_OFFS2					MD12_OFFS1						MD12_OFFS0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MD12_M7		MD12_M6		MD12_M5		MD12_M4		MD12_M3		MD12_M2		MD12_M1		MD12_M0	
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 44-104. DI Display 1 Data Byte 2 Mapping Register (DI_DISP1_DB2_MAP)

Table 44-124. DI_DISP1_DB2_MAP Field Descriptions

Field	Description
31	Reserved
30–26 MD12_OFFS2	Offset in third clock cycle. This bit defines the position of the byte most significant bit in the third clock cycle.

Table 44-124. DI_DISP1_DB2_MAP Field Descriptions (continued)

Field	Description
25–21 MD12_OFFS1	Offset in second clock cycle. This bit defines the position of the byte most significant bit in the second clock cycle.
20–16 MD12_OFFS0	Offset in first clock cycle. This bit define the position of the byte most significant bit in the first clock cycle.
15–14 MD12_M7	Masks for bit 0 (least significant bit). These fields specify in which clock cycle the bit should be sent to the display. Values: 00 Enable in first clock cycle 01 Enable in second clock cycle 10 Enable in third clock cycle 11 Masked
13–12 MD12_M6	
11–10 MD12_M5	
9–8 MD12_M4	
7–6 MD12_M3	
5–4 MD12_M2	
3–2 MD12_M1	
1–0 MD12_M0	

44.3.3.8.25 DI Display 1 Command Byte 0 Mapping Register (DI_DISP1_CB0_MAP)

This register defines masks and offsets of command byte 0 (least significant byte) for display 1.

0x53FC_0184 (DI_DISP1_CB0_MAP) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	MC10_OFFS2					MC10_OFFS1						MC10_OFFS0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MC10_M7		MC10_M6		MC10_M5		MC10_M4		MC10_M3		MC10_M2		MC10_M1		MC10_M0	
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 44-105. DI Display 1 Command Byte 0 Mapping Register (DI_DISP1_CB0_MAP)

Table 44-125. DI_DISP1_CB0_MAP Field Descriptions

Field	Description
31	Reserved
30–26 MC10_OFFS2	Offset in third clock cycle. This bit defines the position of the byte most significant bit in the third clock cycle.
25–21 MC10_OFFS1	Offset in second clock cycle. This bit defines the position of the byte most significant bit in the second clock cycle.
20–16 MC10_OFFS0	Offset in first clock cycle. This bit define the position of the byte most significant bit in the first clock cycle.
15–14 MC10_M7	Masks for bit 0 (least significant bit). These fields specify in which clock cycle the bit should be sent to the display. Values: 00 Enable in first clock cycle 01 Enable in second clock cycle 10 Enable in third clock cycle 11 Masked
13–12 MC10_M6	
11–10 MC10_M5	
9–8 MC10_M4	
7–6 MC10_M3	
5–4 MC10_M2	
3–2 MC10_M1	
1–0 MC10_M0	

44.3.3.8.26 DI Display 1 Command Byte 1 Mapping Register (DI_DISP1_CB1_MAP)

This register defines masks and offsets of command byte 1 (middle byte) for display 1.

0x53FC_0188 (DI_DISP1_CB1_MAP) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	MC11_OFFS2					MC11_OFFS1						MC11_OFFS0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MC11_M7		MC11_M6		MC11_M5		MC11_M4		MC11_M3		MC11_M2		MC11_M1		MC11_M0	
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 44-106. DI Display 1 Command Byte 1 Mapping Register (DI_DISP1_CB1_MAP)

Table 44-126. DI_DISP1_CB1_MAP Field Descriptions

Field	Description
31	Reserved
30–26 MC11_OFFS2	Offset in third clock cycle. This bit defines the position of the byte most significant bit in the third clock cycle.
25–21 MC11_OFFS1	Offset in second clock cycle. This bit defines the position of the byte most significant bit in the second clock cycle.
20–16 MC11_OFFS0	Offset in first clock cycle. This bit define the position of the byte most significant bit in the first clock cycle.
15–14 MC11_M7	Masks for bit 0 (least significant bit). These fields specify in which clock cycle the bit should be sent to the display. Values: 00 Enable in first clock cycle 01 Enable in second clock cycle 10 Enable in third clock cycle 11 Masked
13–12 MC11_M6	
11–10 MC11_M5	
9–8 MC11_M4	
7–6 MC11_M3	
5–4 MC11_M2	
3–2 MC11_M1	
1–0 MC11_M0	

44.3.3.8.27 DI Display 1 Command Byte 2 Mapping Register (DI_DISP1_CB2_MAP)

This register defines masks and offsets of command byte 2 (most significant byte) for display 1.

0x53FC_018C (DI_DISP1_CB2_MAP) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	MC12_OFFS2					MC12_OFFS1						MC12_OFFS0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MC12_M7		MC12_M6		MC12_M5		MC12_M4		MC12_M3		MC12_M2		MC12_M1		MC12_M0	
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 44-107. DI Display 1 Command Byte 2 Mapping Register (DI_DISP1_CB2_MAP)

Table 44-127. DI_DISP1_CB2_MAP Field Descriptions

Field	Description
31	Reserved
30–26 MC12_OFFS2	Offset in third clock cycle. This bit defines the position of the byte most significant bit in the third clock cycle.
25–21 MC12_OFFS1	Offset in second clock cycle. This bit defines the position of the byte most significant bit in the second clock cycle.
20–16 MC12_OFFS0	Offset in first clock cycle. This bit define the position of the byte most significant bit in the first clock cycle.

Table 44-127. DI_DISP1_CB2_MAP Field Descriptions (continued)

Field	Description
15–14 MC12_M7	Masks for bit 0 (least significant bit). These fields specify in which clock cycle the bit should be sent to the display. Values: 00 Enable in first clock cycle 01 Enable in second clock cycle 10 Enable in third clock cycle 11 Masked
13–12 MC12_M6	
11–10 MC12_M5	
9–8 MC12_M4	
7–6 MC12_M3	
5–4 MC12_M2	
3–2 MC12_M1	
1–0 MC12_M0	

44.3.3.8.28 DI Display 2 Data Byte 0 Mapping Register (DI_DISP2_DB0_MAP)

This register defines masks and offsets of data byte 0 (least significant byte) for display 2.

0x53FC_0190 (DI_DISP2_DB0_MAP) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	MD20_OFFS2					MD20_OFFS1						MD20_OFFS0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MD20_M7		MD20_M6		MD20_M5		MD20_M4		MD20_M3		MD20_M2		MD20_M1		MD20_M0	
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 44-108. DI Display 2 Data Byte 0 Mapping Register (DI_DISP2_DB0_MAP)

Table 44-128. DI_DISP2_DB0_MAP Field Descriptions

Field	Description
31	Reserved
30–26 MD20_OFFS2	Offset in third clock cycle. This bit defines the position of the byte most significant bit in the third clock cycle.

Table 44-128. DI_DISP2_DB0_MAP Field Descriptions (continued)

Field	Description
25–21 MD20_OFFS1	Offset in second clock cycle. This bit defines the position of the byte most significant bit in the second clock cycle.
20–16 MD20_OFFS0	Offset in first clock cycle. This bit define the position of the byte most significant bit in the first clock cycle.
15–14 MD20_M7	Masks for bit 0 (least significant bit). These fields specify in which clock cycle the bit should be sent to the display. Values: 00 Enable in first clock cycle 01 Enable in second clock cycle 10 Enable in third clock cycle 11 Masked
13–12 MD20_M6	
11–10 MD20_M5	
9–8 MD20_M4	
7–6 MD20_M3	
5–4 MD20_M2	
3–2 MD20_M1	
1–0 MD20_M0	

44.3.3.8.29 DI Display 2 Data Byte 1 Mapping Register (DI_DISP2_DB1_MAP)

This register defines masks and offsets of data byte 1 (middle byte) for display 2.

0x53FC_0194 (DI_DISP2_DB1_MAP) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	MD21_OFFS2					MD21_OFFS1						MD21_OFFS0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MD21_M7		MD21_M6		MD21_M5		MD21_M4		MD21_M3		MD21_M2		MD21_M1		MD21_M0	
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 44-109. DI Display 2 Data Byte 1 Mapping Register (DI_DISP2_DB1_MAP)

Table 44-129. DI_DISP2_DB1_MAP Field Descriptions

Field	Description
31	Reserved
30–26 MD21_OFFS2	Offset in third clock cycle. This bit defines the position of the byte most significant bit in the third clock cycle.
25–21 MD21_OFFS1	Offset in second clock cycle. This bit defines the position of the byte most significant bit in the second clock cycle.
20–16 MD21_OFFS0	Offset in first clock cycle. This bit define the position of the byte most significant bit in the first clock cycle.
15–14 MD21_M7	Masks for bit 0 (least significant bit). These fields specify in which clock cycle the bit should be sent to the display. Values: 00 Enable in first clock cycle 01 Enable in second clock cycle 10 Enable in third clock cycle 11 Masked
13–12 MD21_M6	
11–10 MD21_M5	
9–8 MD21_M4	
7–6 MD21_M3	
5–4 MD21_M2	
3–2 MD21_M1	
1–0 MD21_M0	

44.3.3.8.30 DI Display 2 Data Byte 2 Mapping Register (DI_DISP2_DB2_MAP)

This register defines masks and offsets of data byte 2 (most significant byte) for display 2.

0x53FC_0198 (DI_DISP2_DB2_MAP) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	MD22_OFFS2					MD22_OFFS1						MD22_OFFS0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MD22_M7		MD22_M6		MD22_M5		MD22_M4		MD22_M3		MD22_M2		MD22_M1		MD22_M0	
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 44-110. DI Display 2 Data Byte 2 Mapping Register (DI_DISP2_DB2_MAP)

Table 44-130. DI_DISP2_DB2_MAP Field Descriptions

Field	Description
31	Reserved
30–26 MD22_OFFS2	Offset in third clock cycle. This bit defines the position of the byte most significant bit in the third clock cycle.
25–21 MD22_OFFS1	Offset in second clock cycle. This bit defines the position of the byte most significant bit in the second clock cycle.
20–16 MD22_OFFS0	Offset in first clock cycle. This bit define the position of the byte most significant bit in the first clock cycle.
15–14 MD22_M7	Masks for bit 0 (least significant bit). These fields specify in which clock cycle the bit should be sent to the display. Values: 00 Enable in first clock cycle 01 Enable in second clock cycle 10 Enable in third clock cycle 11 Masked
13–12 MD22_M6	
11–10 MD22_M5	
9–8 MD22_M4	
7–6 MD22_M3	
5–4 MD22_M2	
3–2 MD22_M1	
1–0 MD22_M0	

44.3.3.8.31 DI Display 2 Command Byte 0 Mapping Register (DI_DISP2_CB0_MAP)

This register defines masks and offsets of command byte 0 (least significant byte) for display 2.

0x53FC_019C (DI_DISP2_CB0_MAP) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	MC20_OFFS2					MC20_OFFS1						MC20_OFFS0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MC20_M7		MC20_M6		MC20_M5		MC20_M4		MC20_M3		MC20_M2		MC20_M1		MC20_M0	
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 44-111. DI Display 2 Command Byte 0 Mapping Register (DI_DISP2_CB0_MAP)

Table 44-131. DI_DISP2_CB0_MAP Field Descriptions

Field	Description
31	Reserved
30–26 MC20_OFFS2	Offset in third clock cycle. This bit defines the position of the byte most significant bit in the third clock cycle.
25–21 MC20_OFFS1	Offset in second clock cycle. This bit defines the position of the byte most significant bit in the second clock cycle.
20–16 MC20_OFFS0	Offset in first clock cycle. This bit define the position of the byte most significant bit in the first clock cycle.

Table 44-131. DI_DISP2_CB0_MAP Field Descriptions (continued)

Field	Description
15–14 MC20_M7	Masks for bit 0 (least significant bit). These fields specify in which clock cycle the bit should be sent to the display. Values: 00 Enable in first clock cycle 01 Enable in second clock cycle 10 Enable in third clock cycle 11 Masked
13–12 MC20_M6	
11–10 MC20_M5	
9–8 MC20_M4	
7–6 MC20_M3	
5–4 MC20_M2	
3–2 MC20_M1	
1–0 MC20_M0	

44.3.3.8.32 DI Display 2 Command Byte 1 Mapping Register (DI_DISP2_CB1_MAP)

This register defines masks and offsets of command byte 1 (middle byte) for display 2.

0x53FC_01A0 (DI_DISP2_CB1_MAP) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	MC21_OFFS2					MC21_OFFS1						MC21_OFFS0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MC21_M7		MC21_M6		MC21_M5		MC21_M4		MC21_M3		MC21_M2		MC21_M1		MC21_M0	
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 44-112. DI Display 2 Command Byte 1 Mapping Register (DI_DISP2_CB1_MAP)

Table 44-132. DI_DISP2_CB1_MAP Field Descriptions

Field	Description
31	Reserved
30–26 MC21_OFFS2	Offset in third clock cycle. This bit defines the position of the byte most significant bit in the third clock cycle.

Table 44-132. DI_DISP2_CB1_MAP Field Descriptions (continued)

Field	Description
25–21 MC21_OFFS1	Offset in second clock cycle. This bit defines the position of the byte most significant bit in the second clock cycle.
20–16 MC21_OFFS0	Offset in first clock cycle. This bit define the position of the byte most significant bit in the first clock cycle.
15–14 MC21_M7	Masks for bit 0 (least significant bit). These fields specify in which clock cycle the bit should be sent to the display. Values: 00 Enable in first clock cycle 01 Enable in second clock cycle 10 Enable in third clock cycle 11 Masked
13–12 MC21_M6	
11–10 MC21_M5	
9–8 MC21_M4	
7–6 MC21_M3	
5–4 MC21_M2	
3–2 MC21_M1	
1–0 MC21_M0	

44.3.3.8.33 DI Display 2 Command Byte 2 Mapping Register (DI_DISP2_CB2_MAP)

This register defines masks and offsets of command byte 2 (most significant byte) for display 2.

0x53FC_01A4 (DI_DISP2_CB2_MAP) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	MC22_OFFS2					MC22_OFFS1						MC22_OFFS0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MC22_M7		MC22_M6		MC22_M5		MC22_M4		MC22_M3		MC22_M2		MC22_M1		MC22_M0	
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 44-113. DI Display 2 Command Byte 2 Mapping Register (DI_DISP2_CB2_MAP)

Table 44-133. DI_DISP2_CB2_MAP Field Descriptions

Field	Description
31	Reserved
30–26 MC22_OFFS2	Offset in third clock cycle. This bit defines the position of the byte most significant bit in the third clock cycle.
25–21 MC22_OFFS1	Offset in second clock cycle. This bit defines the position of the byte most significant bit in the second clock cycle.
20–16 MC22_OFFS0	Offset in first clock cycle. This bit define the position of the byte most significant bit in the first clock cycle.
15–14 MC22_M7	Masks for bit 0 (least significant bit). These fields specify in which clock cycle the bit should be sent to the display. Values: 00 Enable in first clock cycle 01 Enable in second clock cycle 10 Enable in third clock cycle 11 Masked
13–12 MC22_M6	
11–10 MC22_M5	
9–8 MC22_M4	
7–6 MC22_M3	
5–4 MC22_M2	
3–2 MC22_M1	
1–0 MC22_M0	

44.3.3.8.34 MDI Display 3 Byte 0 Mapping Register (DI_DISP3_B0_MAP)

This register defines masks and offsets of data byte 0 (least significant byte) for display 3.

0x53FC_01A8 (DI_DISP3_B0_MAP) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	M30_OFFS2					M30_OFFS1						M30_OFFS0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	M30_M7		M30_M6		M30_M5		M30_M4		M30_M3		M30_M2		M30_M1		M30_M0	
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 44-114. DI Display 3 Byte 0 Mapping Register (DI_DISP3_B0_MAP)

Table 44-134. DI_DISP3_B0_MAP Field Descriptions

Field	Description
31	Reserved
30–26 M30_OFFS2	Offset in third clock cycle. This bit defines the position of the byte most significant bit in the third clock cycle.
25–21 M30_OFFS1	Offset in second clock cycle. This bit defines the position of the byte most significant bit in the second clock cycle.
20–16 M30_OFFS0	Offset in first clock cycle. This bit define the position of the byte most significant bit in the first clock cycle.
15–14 M30_M7	Masks for bit 0 (least significant bit). These fields specify in which clock cycle the bit should be sent to the display. Values: 00 Enable in first clock cycle 01 Enable in second clock cycle 10 Enable in third clock cycle 11 Masked
13–12 M30_M6	
11–10 M30_M5	
9–8 M30_M4	
7–6 M30_M3	
5–4 M30_M2	
3–2 M30_M1	
1–0 M30_M0	

44.3.3.8.35 DI Display 3 Byte 1 Mapping Register (DI_DISP3_B1_MAP)

This register defines masks and offsets of data byte 1 (middle byte) for display 3.

0x53FC_01AC (DI_DISP3_B1_MAP) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	M31_OFFS2					M31_OFFS1						M31_OFFS0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	M31_M7		M31_M6		M31_M5		M31_M4		M31_M3		M31_M2		M31_M1		M31_M0	
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 44-115. DI Display 3 Byte 1 Mapping Register (DI_DISP3_B1_MAP)

Table 44-135. DI_DISP3_B1_MAP Field Descriptions

Field	Description
31	Reserved
30–26 M31_OFFS2	Offset in third clock cycle. This bit defines the position of the byte most significant bit in the third clock cycle.
25–21 M31_OFFS1	Offset in second clock cycle. This bit defines the position of the byte most significant bit in the second clock cycle.
20–16 M31_OFFS0	Offset in first clock cycle. This bit define the position of the byte most significant bit in the first clock cycle.

Table 44-135. DI_DISP3_B1_MAP Field Descriptions (continued)

Field	Description
15–14 M31_M7	Masks for bit 0 (least significant bit). These fields specify in which clock cycle the bit should be sent to the display. Values: 00 Enable in first clock cycle 01 Enable in second clock cycle 10 Enable in third clock cycle 11 Masked
13–12 M31_M6	
11–10 M31_M5	
9–8 M31_M4	
7–6 M31_M3	
5–4 M31_M2	
3–2 M31_M1	
1–0 M31_M0	

44.3.3.8.36 DI Display 3 Byte 2 Mapping Register (DI_DISP3_B2_MAP)

This register defines masks and offsets of data byte 2 (most significant byte) for display 3.

0x53FC_01B0 (DI_DISP3_B2_MAP) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	M32_OFFS2					M32_OFFS1						M32_OFFS0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	M32_M7		M32_M6		M32_M5		M32_M4		M32_M3		M32_M2		M32_M1		M32_M0	
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 44-116. DI Display 3 Byte 2 Mapping Register (DI_DISP3_B2_MAP)

Table 44-136. DI_DISP3_B2_MAP Field Descriptions

Field	Description
31	Reserved
30–26 M32_OFFS2	Offset in third clock cycle. This bit defines the position of the byte most significant bit in the third clock cycle.

Table 44-136. DI_DISP3_B2_MAP Field Descriptions (continued)

Field	Description
25–21 M32_OFFS1	Offset in second clock cycle. This bit defines the position of the byte most significant bit in the second clock cycle.
20–16 M32_OFFS0	Offset in first clock cycle. This bit define the position of the byte most significant bit in the first clock cycle.
15–14 M32_M7	Masks for bit 0 (least significant bit). These fields specify in which clock cycle the bit should be sent to the display. Values: 00 Enable in first clock cycle 01 Enable in second clock cycle 10 Enable in third clock cycle 11 Masked
13–12 M32_M6	
11–10 M32_M5	
9–8 M32_M4	
7–6 M32_M3	
5–4 M32_M2	
3–2 M32_M1	
1–0 M32_M0	

44.3.3.8.37 DI Display Access Cycles Count Register (DI_DISP_ACC_CC)

This register defines how many display clock cycles are required to output/input one pixel.

0x53FC_01B4 (DI_DISP_ACC_CC) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	DISP3_IF_CLK_CNT_D		DISP2_IF_CLK_CNT_C		DISP2_IF_CLK_CNT_D		DISP1_IF_CLK_CNT_C		DISP1_IF_CLK_CNT_D		DISP0_IF_CLK_CNT_C		DISP0_IF_CLK_CNT_D	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-117. DI Display Access Cycles Count Register (DI_DISP_ACC_CC)

Table 44-137. DI_DISP_ACC_CC Field Descriptions

Field	Description
31–14	Reserved
13–12 DISP3_IF_CLK_CNT_D	Display clock cycles number minus 1 for output/input one data word. This bit describe in which clock displayed bit is masked. Values: 00 1 clock cycle 01 2 clock cycles 10 3 clock cycles 11 4 clock cycles
11–10 DISP2_IF_CLK_CNT_C	Display clock cycles number minus 1 for output/input one data word. This bit describe in which clock displayed bit is masked. Values: 00 1 clock cycle 01 2 clock cycles 10 3 clock cycles 11 4 clock cycles
9–8 DISP2_IF_CLK_CNT_D	Display clock cycles number minus 1 for output/input one data word. This bit describe in which clock displayed bit is masked. Values: 00 1 clock cycle 01 2 clock cycles 10 3 clock cycles 11 4 clock cycles

Table 44-137. DI_DISP_ACC_CC Field Descriptions (continued)

Field	Description
7–6 DISP1_IF_CLK_CNT_C	Display clock cycles number minus 1 for output/input one data word. This bit describe in which clock displayed bit is masked. Values: 00 1 clock cycle 01 2 clock cycles 10 3 clock cycles 11 4 clock cycles
5–4 DISP1_IF_CLK_CNT_D	Display clock cycles number minus 1 for output/input one data word. This bit describe in which clock displayed bit is masked. Values: 00 1 clock cycle 01 2 clock cycles 10 3 clock cycles 11 4 clock cycles
3–2 DISP0_IF_CLK_CNT_C	Display clock cycles number minus 1 for output one command word. This bit describe in which clock displayed bit is masked. Values: 00 1 clock cycle 01 2 clock cycles 10 3 clock cycles 11 4 clock cycles
1–0 DISP0_IF_CLK_CNT_D	Display clock cycles number minus 1 for output/input one data word. This bit describe in which clock displayed bit is masked. Values: 00 1 clock cycle 01 2 clock cycles 10 3 clock cycles 11 4 clock cycles

44.3.3.8.38 DI Display Low Level Access Configuration Register (DI_DISP_LLA_CONF)

This register defines properties of MCU low-level access to the smart display.

0x53FC_01B8 (DI_DISP_LLA_CONF) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	DRCT_BE_MODE	DRCT_MAP_DC	DRCT_LOCK	DRCT_DISP_NUM		DRCT_RS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-118. DI Display Low Level Access Configuration Register (DI_DISP_LLA_CONF)

Table 44-138. DI_DISP_LLA_CONF Field Descriptions

Field	Description
31–6	Reserved
5 DRCT_BE_MODE	Set byte enable mode for MCU low level access of the display. This bit sets byte enable mode for MCU low level access of the display. This mode is valid only for parallel display interfaces. In this mode, the byte enable signals of the IP Skyblue Bus which are active while read/write from/to the DI_DISP_LLA_DATA Register are translated to the parallel display interface together with 16-bit least significant data bits. Eight most significant data bits are ignored. 0 switch off byte enable mode 1 switch on byte enable mode
4 DRCT_MAP_DC	Display mapping select. Selects display mapping for data or command. 0 Data 1 Command
3 DRCT_LOCK	Lock bit. When the DRCT_LOCK bit is set, the DI waits for data from the MCU. 0 Unlock 1 Lock

Table 44-138. DI_DISP_LLA_CONF Field Descriptions (continued)

Field	Description
2–1 DRCT_DISP_NUM	The accessed display number. These bits describe which display now active. Values: 00 Display 0 01 Display 1 10 Display 2 11 Reserved
0 DRCT_RS	Command/data address signal to display. 0 RS is 0 1 RS is 1

44.3.3.8.39 DI Display Low Level Access Data Register (DI_DISP_LLA_DATA)

This register contain the data which is to be written to display or being read from display. To read the register, the following settings are required:

1. The HSP_CLK clock must be fed to the IPU.
2. The DI_EN bit in the IPU_CONF register must be set.
3. The DRCT_DISP_NUM parameter in the DI_DISP_LLA_CONF register must be set to select a display number (default =2'b00).
4. The DISP(x)_EN bit in the DI_DISP_IF_CONF register must be set that matches the DRCT_DISP_NUM setting (the other enables are a don't care).
5. All other parameters in the DI registers corresponding to the selected display must be properly configured.

0x53FC_01BC (DI_DISP_LLA_DATA) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	LLA_DATA[23:16]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LLA_DATA[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-119. DI Display Low Level Access Data (DI_DISP_LLA_DATA)

Table 44-139. DI_DISP_LLA_DATA

Field	Description
31–24	Reserved
LLA_DATA[23:16]	Low level access data. This field contain the data of direct access of MCU (read/write).
LLA_DATA[15:0]	

44.4 Functional Description

44.4.1 Camera Sensor Interface (CSI)

44.4.1.1 Block Diagram

The CSI Block Diagram is shown in Figure 44-120.

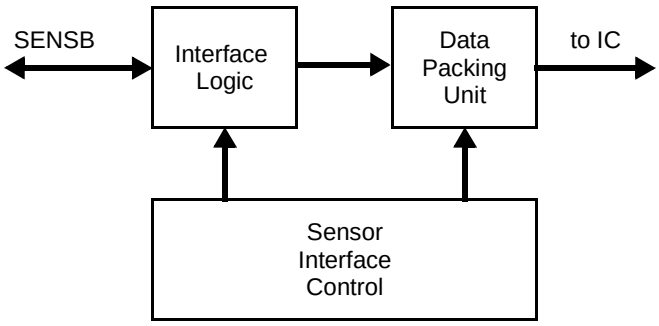


Figure 44-120. CSI Block Diagram

The CSI consists of the Interface Logic, the Data Packing Unit and the Sensor Interface Control. The CSI is controlled via the peripheral bus registers. All programming parameters for the CSI are double buffered with synchronous change at the frame start.

44.4.1.2 Sensor Image Frame Relations

Figure 44-121 illustrates the generalized relations between image frames produced by a sensor and accepted by the CSI. Generally, four frame definitions exist. The virtual frame A starts with the VSYNC signal.

The frame A includes the frame B. The HSYNC signal indicates boundaries of the frame B. The frame B includes both the active sensor frame C and blanking intervals. A size of the blanking intervals depends on sensor type and programming. The CSI selects a window (the frame D) inside the frame C by skipping rows and columns according to parameters defined in the CSI_OUT_FRM_CTRL Register. This scheme may be simpler for the specific sensor type. For example, the frames A and B or B and C can be equal.

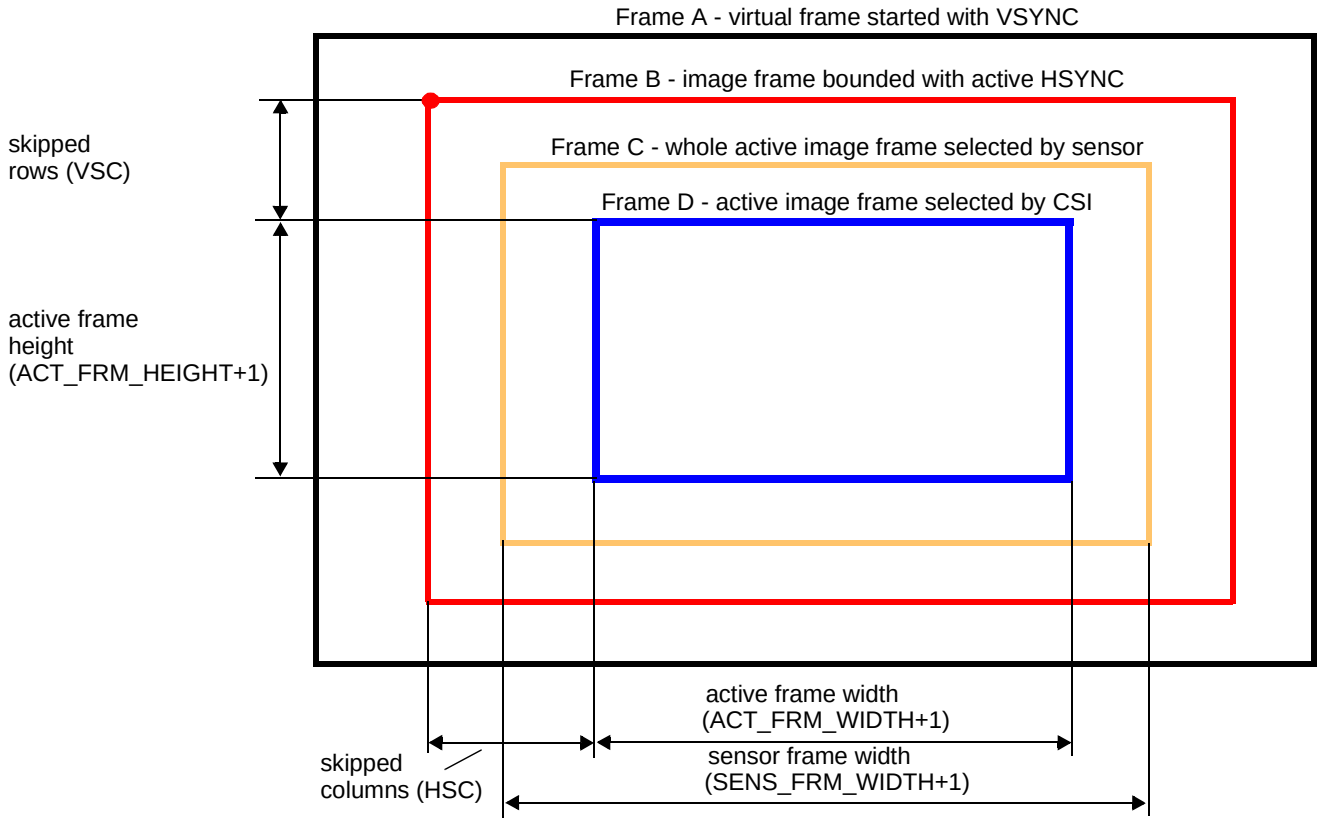


Figure 44-121. Sensor Image Frames

44.4.1.3 Interface Logic

This submodule gets a sensor data from the SENSB bus. The Interface Logic supports required sensor interface protocol according to the CSI_SENS_CONF Register. It can operate in gated clock mode or non-gated clock mode or pseudo BT.656 mode. The master clock provided for sensor can be derived by bypassing the SENSB_SENS_CLK clock or by division of the HSP_CLK clock.

A data stream from the sensor sampled at the sensor clock is split by the CSI into four streams. The sampling rate in each of the streams is a quarter of the sensor clock rate. Data of each stream are synchronized to the high-speed processing clock (HSP_CLK). The synchronized streams are merged by the CSI to a single stream with the sampling rate being equal to the sensor clock rate.

Such solution requires that the maximal sensor clock rate must be less than the HSP_CLK rate. The next bottleneck is an external system memory throughput (including a DMA throughput) which limits sensor image writing speed. This limitation depends on load of the memory by other clients (the MCU, the system DMA controller etc.). The maximal sensor clock limitation has been checked for the following use case:

1. Viewfinder: No writing of whole sensor image to the system memory is performed at this step.

   Use case configuration:

   — Downsizing and color conversion—by the IPU

— QVGA smart display
— 4 Mpixel smart sensor @7.5fps

Video data flow:

— Sensor to IPU: 4 Mpixel,YUV 4:2:2, 7.5 fps (skipping frames)
— IPU to system memory (downsize and color conversion), QVGA, 7.5fps
— Memory to IPU (to display): QVGA, RGB 8:8:8, 7.5 fps
— IPU to display: QVGA,7.5 fps

2. Still image capture: To move the IPU to this step, the MCU has to stop the viewfinder task and to enable sensor data writing to the system memory.

   Use case configuration:

   — JPEG encoding – by the MCU
   — Compressed video stored in SD memory,
   — 4 Mpixel smart sensor

   Video data flow:

   — Sensor to IPU:4 Mpixels,YUV 4:2:2
   — IPU transfers uncompressed image data to memory
   — Memory to MCU for image processing (JPEG)
   — The MCU transfers compressed image data to system memory
   — Compressed image is moved to SD memory for storage

Memory bus load by clients other than the IPU limits the maximal speed sensor image capture at Step 2. If there is no memory accesses by other clients during capture the maximal sensor clock rate is 90 MHz (for the exact sensor size of 2504x1748 pixels). For specific scenario when the MCU or other clients access the system memory during sensor image capture, the maximal sensor rate will be lower than 90 MHz. The actual value of the maximal sensor clock rate depends on the memory bus load. The software should minimize this load as much as possible when data from a large sensor is captured.

44.4.1.4 Data Packing Unit

The pixel formats on the CSI output are YUV 4:4:4, RGB (8 or 10 bits per color component) or generic data (up to 15 bits per word aligned to the most significant bit). The Data Packing Unit collects color components of two adjacent pixels or eight bytes of generic data or four 16-bit words of generic data in a single 64-bit word. Data packing on the output is illustrated by Table 44-168 and Table 44-169. The packing is controlled via the CSI_SENS_CONF Register.

44.4.1.5 Sensor Interface Control

The Sensor Interface Control synchronizes the Interface Logic and the Data Packing Unit. It includes vertical and horizontal counters for support of frame selection function as shown in Figure 44-121. The unit generates the master clock to the sensor. The master clock frequency is programmable via the CSI_SENS_CONF Register.

The unit provides also the frame skipping function according to two separate skipping patterns - one for the encoder flow and the second - for the viewfinder flow. The patterns are programmed by the MCU and cyclically shifted at every frame start point.

In test mode, the Sensor Interface Control generates a test pattern which replaces a real sensor data. The test pattern is a chess field with white and programmable color squares. The color is defined via the CSI_TST_CTRL Register.

44.4.1.6 Non-contiguous Memory Buffers Support

The IPU allows to write large image data from a sensor to a non-contiguous buffer in the external memory. Data transfer to the external memory is performed via the CSI, the IC and the IDMAC. In order to provide non-contiguous buffer mode, the IPU software driver should define appropriate values of frame size in the CSI_ACT_FRM_SIZE Register and the corresponding DMA Channel Parameter Memory (for the DMAIC_7 channel - see Table 44-163). Frame width values must be equal both for the CSI and the IDMAC. Frame height value for the CSI should be a multiple of the frame height for the DMAIC_7 channel. The DMAIC_7 channel must be programmed in double buffer mode via the IPU_CHA_DB_MODE_SEL Register. Writing to the non-contiguous is accomplished as follows:

1. The MCU defines a base address of the buffer 0 for the DMAIC_7 channel (in the Channel Parameter Memory) and enables the channel. The MCU configures and enables the CSI and the IC.
2. The CSI starts to send the data via the IC to the IDMAC at beginning of the next sensor frame.
3. While the IDMAC fills the buffer 0, the MCU defines a base address of the buffer 1 for the DMAIC_7 channel.
4. If the IDMAC has finish to fill the buffer 0 and the sensor frame is not finished yet, the CSI sends a frame signal to the IDMAC to proceed with the buffer 1. Simultaneously, the IDMAC generates the DMAIC_7_EOF interrupt (see the IPU_INT_STAT_1 Register) to the MCU. The steps 4, 5 are repeated.
5. If the sensor frame is finished, the CSI sends the CSI_EOF interrupt (see the IPU_INT_STAT_3 Register) to the MCU.

44.4.1.7 Flash Strobe Generation

The CSI can generate a strobe for the external flash. The strobe parameters are defined in the CSI_FLASH_STROBE_1 and CSI_FLASH_STROBE_2 Registers. The MCU sets strobe start time, duration and polarity. Both parameters are expressed in rows. The start time range is from row 0 to row 8191. The maximal strobe duration is 65536 rows. After that MCU enables strobe generation.

The CSI waits for the start point of the next frame. The start point corresponds to the first HSYNC of the frame (top left corner of red frame in Figure 44-121). This point is defined for all types of sensors. The CSI counts pixels and sensor rows after frame start point. The row width (CSI_SENS_FRM_SIZE) is set in the CSI_SENS_FRM_SIZE Register. The row counter output is used to generate the strobe. To produce an additional strobe, the MCU has to enable it again.

44.4.1.8 Interlaced Sensor Format Support

For sensors produced image data in the interlaced format, the IPU firstly has to save the data to the external memory. There are two modes of sensor data IPU operation in this case:

1. Each field of the frame is stored in a separate frame buffer of the external memory. Conversion from the interlaced format to the progressive scan format should be done by software (if needed).
2. The IPU converts the interlaced format to the progressive scan format during writing to the external memory. This mode is active if the IC_TV_MODE bit in the CSI_OUT_FRM_CTRL Register is asserted.

44.4.2 Image Converter (IC)

44.4.2.1 Block Diagram

The IC Block Diagram is shown in Figure 44-122.

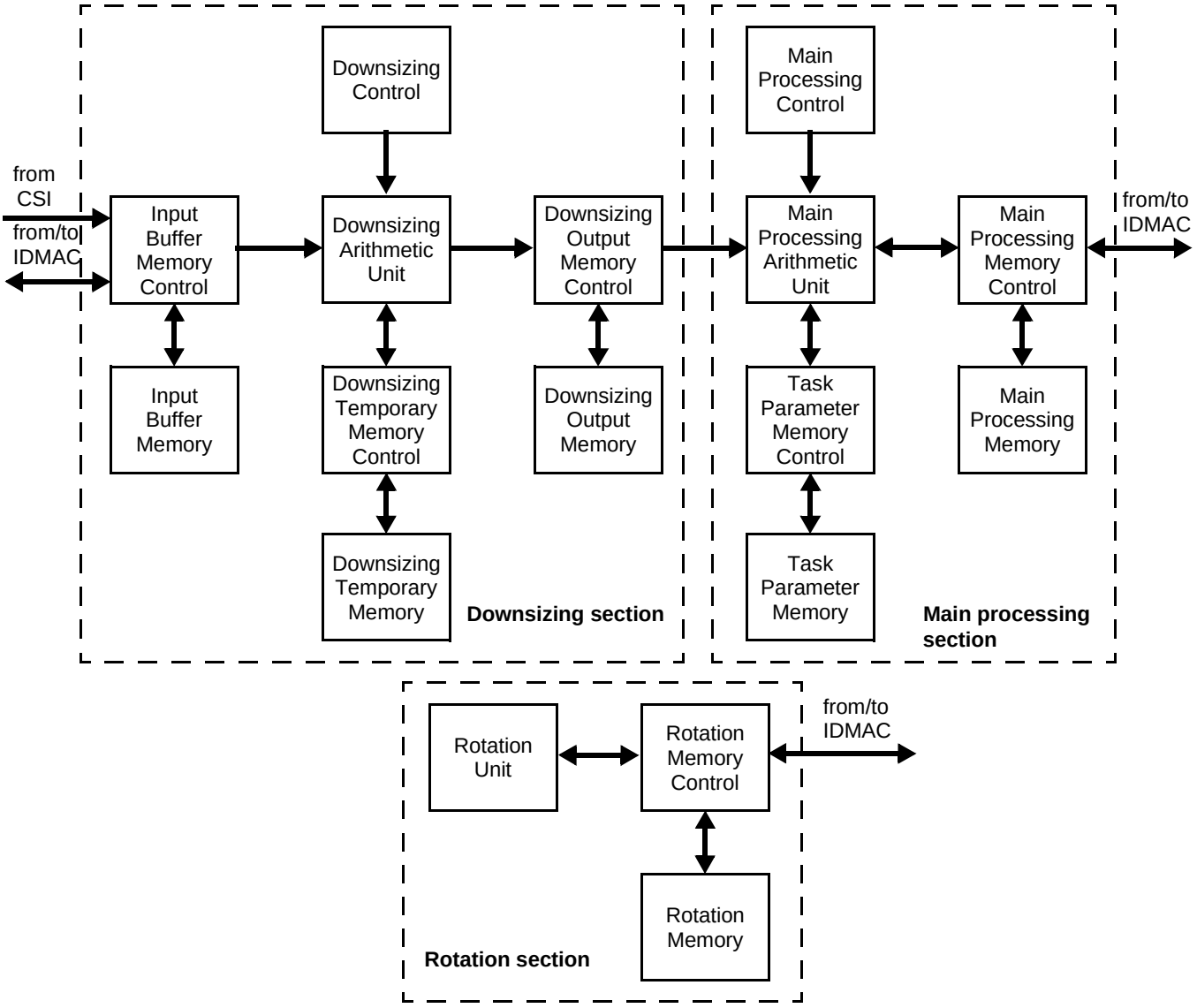


Figure 44-122. IC Block Diagram

The IC contains three processing sections: downsizing, main processing and rotation. The module is controlled via the peripheral bus registers. Some processing parameters should be written by the MCU to the Task Parameter Memory. Writing to the memory is performed via the CM (see Section 44.4.8, “Control Module (CM)”).

44.4.2.2 Processing tasks

Each of three processing section performs up to three processing tasks with time sharing:

1. Preprocessing task for encoding.
2. Preprocessing task for displaying image from sensor (viewfinder).
3. Postprocessing task.

The tasks are performed by single hardware. The MCU configures each task before enabling it. Tasks switching is transparent for the MCU. The time unit for task switching in the downsizing section corresponds to a processing time of one burst of eight pixels, in the main processing section - to a processing time of one image line, in the rotation section - to a processing time of one image frame.

All three tasks includes similar operations controlled by commands. Task configuring consists in definition of commands for each task as described in Table 44-140.

Table 44-140. Task Commands

Command code	Command	Processing unit	Command parameters	Description
EN	Task Enable	DSU, MPU		Task will enabled from next frame. Task 1 is preprocessing for encoding. Task 2 is preprocessing for viewfinder. Task 3 is postprocessing.
	Downsizing	DSU	Downsizing ratio (GCR)	Downsizing ratio 1:1, 2:1, 4:1
	Resizing	MPU	Resizing ratio (GCR)	Resizing ratio from 2:1 to 1:M Resizing ratio = N:M; M =2^13; N = floor(M*(SI-1)/(SO-1)); SI - input size; SO—output size
CSC1	Color space conversion 1	MPU	Color conversion coefficients and offsets (TPM)	Color conversion matrix 1
CSC2	Color space conversion 2	MPU	Color conversion coefficients and offsets (TPM)	Color conversion matrix 2 Used only for Task 2 and Task 3
GLOB_A	Global alpha	MPU		Used only for Task 2 and Task 3
CMB	Combining	MPU		Combining video with graphics. Used only for Task2 and Task3.

The MCU writes the commands to the IC_CONF Register. Because there is no double buffering for all the IC parameters, the MCU must disable a task before changing its parameters. After being disabled, the task is still allowed to complete its current frame execution. At frame finish, the IPU sends an interrupt to the MCU indicating that the MCU can change task parameters. The MCU enables the task again and task execution is resumed from start of the next frame.

44.4.2.3 Downsizing Section

The sensor data from the CSI is written into a FIFO located in the Input Buffer Memory. Depending on programmed processing flow, the FIFO data can be sent to the system memory via the IDMAC or straight forward to the Downsizing Unit. In the first case, the data is processed by the MCU and returned by the IDMAC to another FIFO located in the Input Buffer Memory.

For postprocessing, the IDMAC transfers a data from the system memory to the third FIFO. The data is read by the Downsizing Unit when the postprocessing is performed.

Each or three FIFOs has eight pages (see Table 44-168 and Table 44-169). Each page can store one burst of 4 words which corresponds to 8 pixels. The memory word width is 64 bits. Each memory word contains color components of two adjacent pixels or eight bytes of generic data (e.g. Bayer). Generic data is always fed to preprocessing after transferring via the system memory and preliminary processing by the MCU. Access to the FIFOs is controlled by the Input Buffer Memory Control.

The Downsizing Unit performs averaging and decimation of image pixels both in horizontal and vertical directions according to the following equations:

$$HP_{R,c} = \frac{1}{DS\_R\_H} \sum_{k=0}^{DS\_R\_H-1} IP_{r+k,c}$$

$$VP_{R,C} = \frac{1}{DS\_R\_V} \sum_{l=0}^{DS\_R\_V-1} HP_{R,c+l}$$

where $IP_{r,c}$ - the input pixel, $HP_{R,c}$ - the pixel after horizontal downsizing, $VP_{R,C}$ - the pixel after vertical downsizing, DS_R_H and DS_R_V - the horizontal and vertical downsizing ratios according to the IC_PRP_ENC_RSC, IC_PRP_VF_RSC and IC_PP_RSC Registers. The final calculation result is rounded to 8 bits.

Each of three downsizing tasks processes the data by bursts of 8 pixels. Normally, the current task runs until emptying the corresponding input FIFO. After finishing burst processing, the Downsizing Unit may switch between the current task and another task with higher priority, if the Input Buffer Memory has received a burst for this new task.

Averaging is performed firstly in the horizontal direction. All color components of a pixel are processed in parallel. After horizontal averaging has finished for a single output pixel, the new pixel value is added to the corresponding pixel value of a temporary row derived from previous averaging steps. This provides vertical averaging of the pixels. The temporary row is stored in the Downsizing Temporary Memory. The memory word width matches one accumulated pixel width (36 bits). There are three temporary rows stored in this memory, one per downsizing task. Mapping of the Downsizing Temporary Memory is shown in Table 44-171.

After vertical averaging has been finished, the output row is written to the Downsizing Output Memory. The memory word of 48 bits includes two output pixels. For each task, the memory has a double buffer of one row (see Table 44-172). When the Downsizing Unit fills the foreground part of the double buffer, the Main Processing Unit takes pair or pixels from the background part. After the new downsized row has been ready, the foreground and background memory pointers are swapped.

44.4.2.4 Main Processing Section

The Main Processing Unit reads pairs of pixels from the Downsizing Output Memory background part. It processes the complete pixel row for the current task and after that switches to another task if the input

data for this new task is ready. For each task, the Main Processing Unit is able to perform the following sequence of operations:

1. Horizontal flipping the image (optional) performed with reading from the Downsizing Output Memory. Flipping is enabled via the BAM parameter of the corresponding DMA channels (see Table 44-32 and Table 44-33) responsible for output of the task results. The preprocessing task for encoding uses the BAM parameter from the DMAIC_0 channel, the preprocessing task for the viewfinder (from the DMAIC_1 channel), the postprocessing task (from the DMAIC_2 channel).
2. Horizontal resizing by bilinear interpolation between two adjacent pixels received from the Downsizing Output Memory according to the equation:

$$HP_{R,c} = IP_{r,c} + RS\_C\_H \cdot (IP_{r+1,c} - IP_{r,c})$$

where RS_C_H - the current horizontal resizing coefficient. The calculation result is rounded to 8 bits. The resizing coefficient is calculated as

$$RS\_C\_H = \left(\sum_{k=0}^{R-1} RS\_R\_H \right) \bmod (8196)$$

where RS_R_H - the horizontal resizing ratio from the IC_PRP_ENC_RSC, IC_PRP_VF_RSC and IC_PP_RSC Registers. The RS_R_H parameter is equal to a numerator N of the resizing ratio N:M with M =2^13.

The resulting row of the horizontal resizing is stored in the Task Parameter Memory (see Table 44-173).

3. Vertical resizing by bilinear interpolation between the current and previous results of horizontal resizing. Both current and previous results of horizontal resizing is stored in the Task Parameter Memory. Resizing is accomplished according to the equation:

$$VP_{R,C} = HP_{R,c} + RS\_C\_V \cdot (HP_{R,c+1} - HP_{R,c})$$

where RS_C_V - the current vertical resizing coefficient. The calculation result is rounded to 8 bits. The resizing coefficient is calculated as

$$RS\_C\_V = \left(\sum_{k=0}^{C-1} RS\_R\_V \right) \bmod (8196)$$

where RS_R_V - the horizontal resizing ratio from the IC_PRP_ENC_RSC, IC_PRP_VF_RSC and

IC_PP_RSC Registers. The RS_R_V parameter is equal to a numerator N of the resizing ratio N:M with M =2^13.

At completion of vertical resizing, this row is updated—the current result of horizontal resizing replaces the previous one.

4. First color space conversion YUV to RGB or RGB to YUV with the conversion matrix CSC1. The conversion matrix coefficients are programmable. They are stored in the Task Parameter Memory. The conversion equations are:

$$Z_0 = 2^{SCALE-1} \cdot (X_0 \cdot C_{00} + X_1 \cdot C_{01} + X_2 \cdot C_{02} + A_0)$$
$$Z_1 = 2^{SCALE-1} \cdot (X_0 \cdot C_{10} + X_1 \cdot C_{11} + X_2 \cdot C_{12} + A_1)$$
$$Z_2 = 2^{SCALE-1} \cdot (X_0 \cdot C_{20} + X_1 \cdot C_{21} + X_2 \cdot C_{22} + A_2)$$

where for YUV to RGB: X_0=Y, X_1=U, X_2=V, Z_0=R, Z_1=G, Z_2=B,
for RGB to YUV: X_0=R, X_1=G, X_2=B, Z_0=Y, Z_1=U, Z_2=V.

All the parameters of the conversion matrix are written by the MCU to the Task Parameter Memory as shown in Table 44-28. The final calculation result is limited according to the SAT_MODE parameter and rounded to 8 bits.

5. Combining video with graphics. There are the following combining options:
   — local alpha blending,
   — global alpha blending,
   — use of key color.

   If both alpha blending and color keying are enabled, color keying has higher priority (graphic pixels of the key color are fully transparent independently on the alpha value).
   Combining mode is selected via the IC_CONF Register. The combining equation is:

$$OP = IGP \cdot \alpha + IVP \cdot (1 - \alpha)$$

where IGP - an input graphics pixel, IVP - an input video pixel, α=(A+floor(A/128))/256 - an alpha value, A - a global or local transparency parameter. The global A is written in the IC_CMBP_1 Register, the local A arrives together with the graphics pixel.

A graphics pixel becomes transparent when color keying is enabled and a pixel color matches a key color (independently on an alpha parameter).

The graphics data is read from a FIFO located in the Main Processing Memory. The FIFO contains eight pages of size of eight pixels. The graphics pixel format in the FIFO is RGB or RGBA or YUV 4:4:4 or YUVA. The graphics data is loaded by the IDMAC to the FIFO from the system memory.

6. Second color space conversion YUV to RGB or RGB to YUV with the conversion matrix CSC2. Typically this is color space conversion RGB to YUV. It is not performed if the first color space conversion or combining are disabled. This operation is designed for TV output. The conversion matrix coefficients are programmable with the same parameters and equation as the first color conversion.

All the operation are executed by an unified processing unit sequentially. Steps 1 and 2 cannot be interrupted by another task. All other steps can be interrupted by a task with higher priority if an input row is ready for this task. Preprocessing tasks priority is higher than postprocessing task priority.

The processing unit consists of three identical parts for each color component. All three color components are processed in parallel. Each of the processing operations can be enabled or disabled by an appropriate command according to.

The processing results are written to an output FIFO located in the Main Processing Output Memory row-by-row. The FIFO contains eight pages, each pages can include one pixel burst. The IDMAC transfers the output bursts to the system memory or to the asynchronous display. The Main Processing Memory (Table 44-174) contains three buffers for each tasks: the temporary row buffer, the graphics FIFO and the output FIFO. Each memory word (64 bits) stores two adjacent pixels in formats RGB or RGBA or YUV 4:4:4 or YUVA with 8 bits per color component.

44.4.2.5 Rotation Section

The rotation section includes the Rotation Memory which stores an input rectangular block of 8x8 pixels and an output FIFO containing four pages of 8 pixels each one. The Rotation Memory word width corresponds to two adjacent pixels - 48 bits (see Table 44-175). The input block is loaded to the memory by the IDMAC like to a FIFO.

The Rotation Unit rewrites pixels from the input block to the output FIFO with corresponding relocation of a pixel inside the block. Rotation and/or left/right flipping and/or up/down flipping are enabled separately for each of three tasks. Configuring the rotation and flipping options is performed via the BAM parameters of the corresponding DMA channels (see Table 44-32 and Table 44-33) responsible for task data input. The preprocessing task for encoding uses the BAM parameter from the DMAIC_10 channel, the preprocessing task for viewfinder - from the DMAIC_11 channel, the postprocessing task - from the DMAIC_13 channel.

Rotation and flip options are shown in Table 44-141.

Table 44-141. Rotation and Flip Options

ROT	FLR	FUD	Image
0	0	0	
0	0	1	

Table 44-141. Rotation and Flip Options (continued)

ROT	FLR	FUD	Image
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

After finishing the rotation task, the IDMAC returns the output FIFO content to the system memory. When writing to the system memory, the IDMAC changes a location of the block relative to an input block location in order to provide proper rotation of the whole frame. Rotation tasks switching is performed after completion of rotation of the whole frame.

44.4.3 Post-Filter (PF)

44.4.3.1 Block Diagram

The PF Block Diagram is shown in Figure 44-123. The PF contains two execution units: the Mode Decision Unit and the Filter Arithmetic Unit. Both units share the Postfilter Memory. The Postfilter Flow Control synchronizes filter operations. The PF is controlled via the PF_CONF Register.

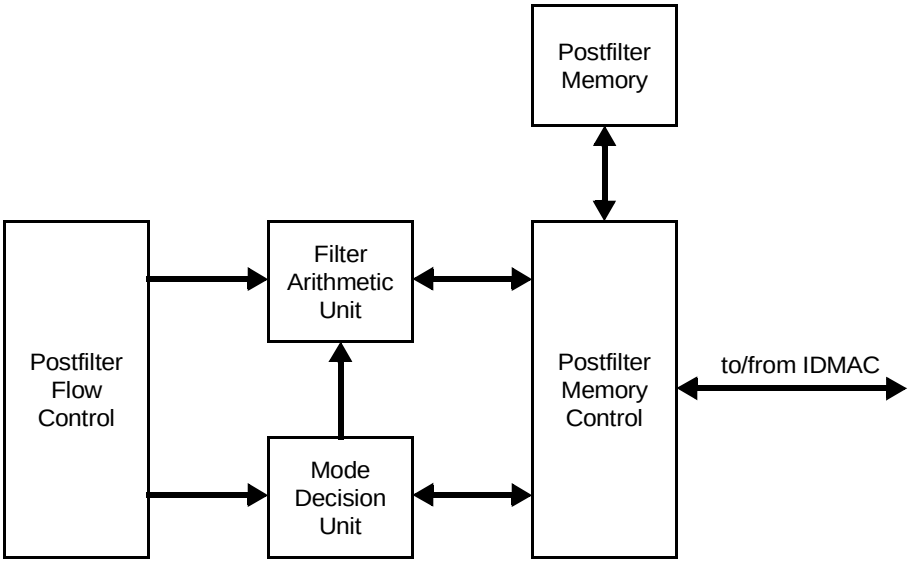
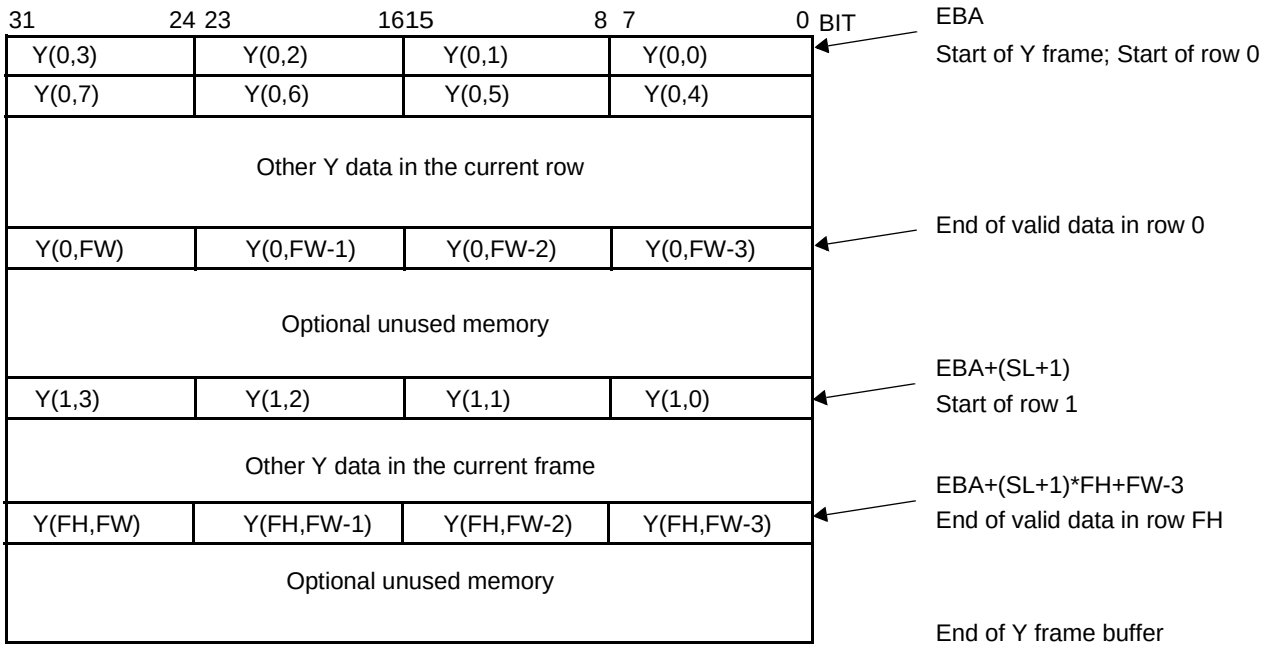


Figure 44-123. PF Block Diagram

44.4.3.2 Filter Algorithms

The PF performs postfiltering for the MPEG-4 (deblocking and deringing) or H.264 (deblocking) video compression standards. The PF has two corresponding operation modes. Mode is selected in the PF_CONF Register. Only one of the modes can be used at the moment. Mode can be changed after completion of frame processing. In each mode, the PF executes a sequence of tasks.

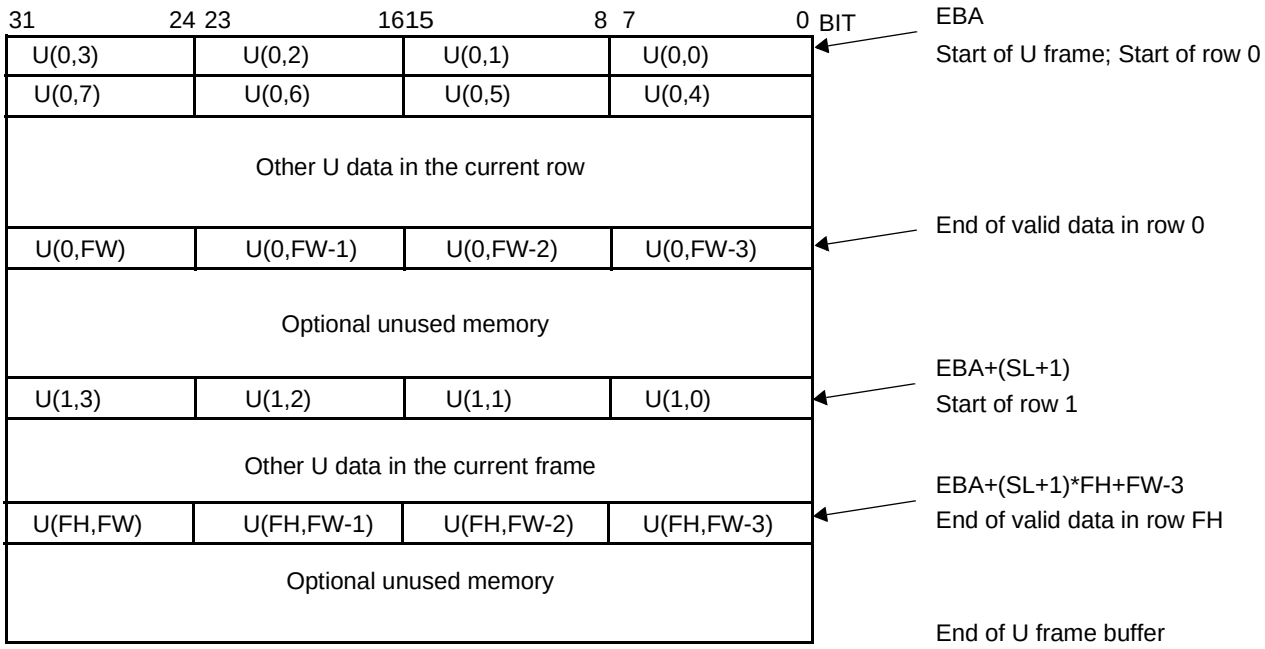
Filter input and output data has YUV 4:2:0 non-interleaved format. Every color component occupies one byte. Data allocation in the external memory is shown in Figure 44-124, Figure 44-125, and Figure 44-126.



Parameters for DMAPF_3 and DMAPF_6 channels:

EBA - External Base Address
SL - Stride Line - 1
FW - Y Frame Width - 1
FH - Y Frame Height - 1

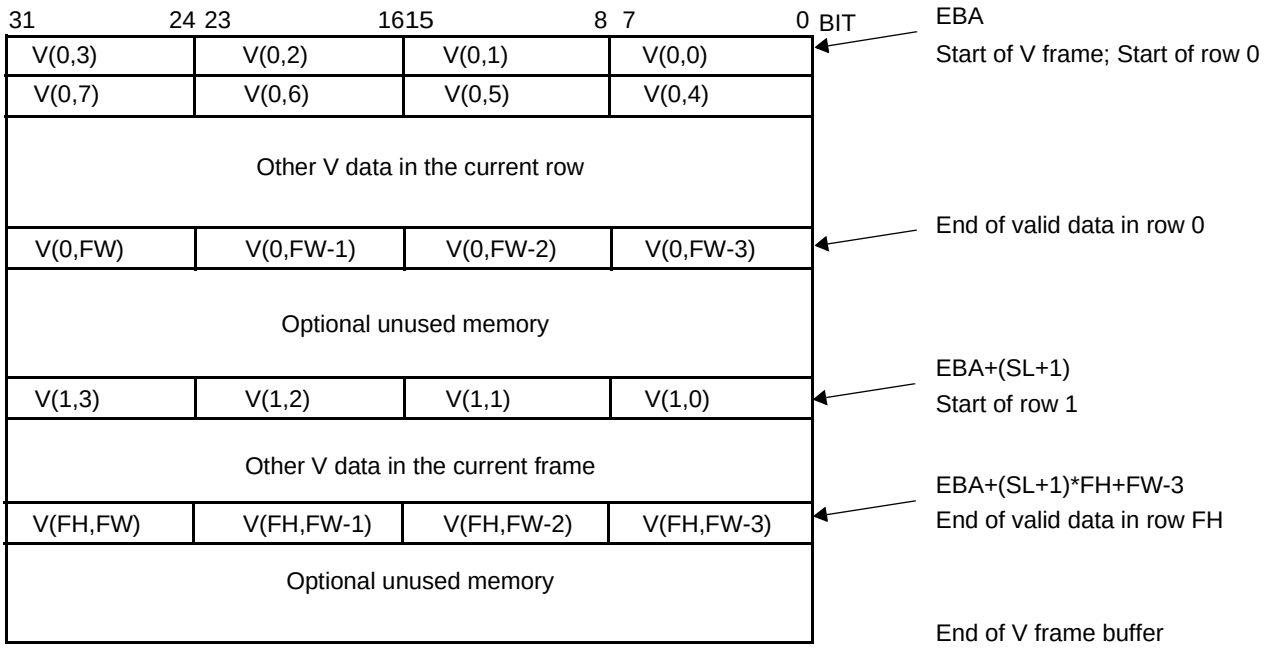
Figure 44-124. Allocation of Y Input and Output Frame Buffer in Memory (Little Endian)



Parameters for DMAPF_4 and DMAPF_7 channels:

EBA - External Base Address
SL - Stride Line - 1
FW - Y Frame Width/2 - 1
FH - Y Frame Height/2 - 1

Figure 44-125. Allocation of U Input and Output Frame Buffer in Memory (Little Endian)



Parameters for DMAPF_5 and DMAPF_8 channels:

EBA - External Base Address
SL - Stride Line - 1
FW - Y Frame Width/2 - 1
FH - Y Frame Height/2 - 1

Figure 44-126. Allocation of V Input and Output Frame Buffer in Memory (Little Endian)

The IDMAC transfer separately each color component to the PF without any format conversion. Seven DMA channels service the PF in MPEG-4 mode: three input data channels for each color component, three output data channels for each color component, one parameter channel for MPEG-4 or two parameter channels for H.264. Filter input and output data is transferred to and from the PF without interleaving because the PF processes separately each color component frame in the Y->U->V order.

44.4.3.2.1 MPEG-4 Mode

In MPEG-4 the operation sequence is:

1. Deblocking of a Y component macroblock.
2. Deringing of the Y component macroblock.
3. Repeating steps 1, 2 until end of frame.
4. Deblocking of a U component macroblock.
5. Repeating step 4 until end of frame.
6. Deblocking of a V component macroblock.
7. Repeating step 6 until end of frame.

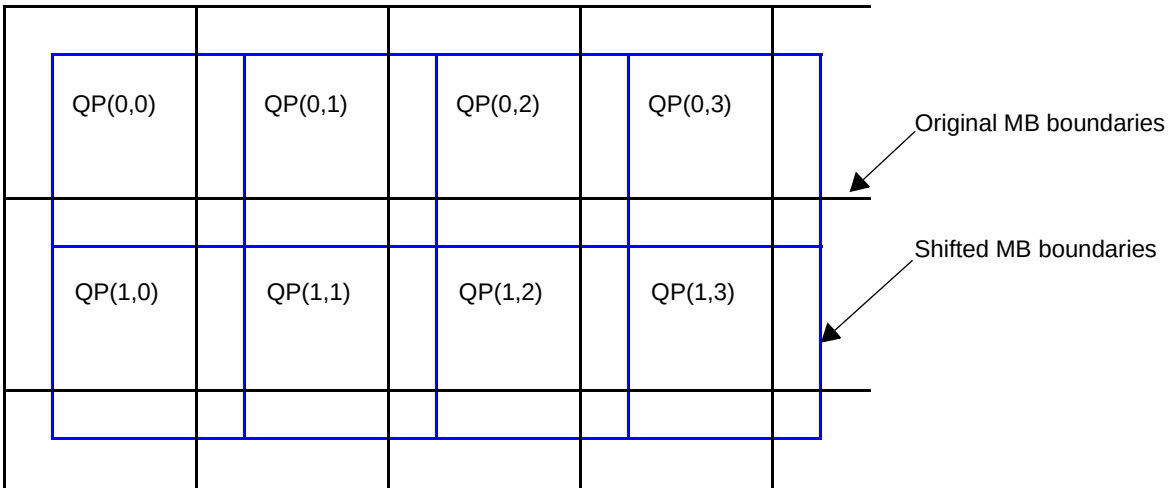
Both input, output and temporary data for the PF calculation is stored in the Postfilter Memory. The IDMAC accesses the memory to supply the input macroblock from the system memory or to put back the processed macroblock to the system memory. The Postfilter Memory word width is 64 bits. The memory stores one of Y or U or V pixel components at the moment (eight pixels per memory word). Memory mapping depends on operation mode.

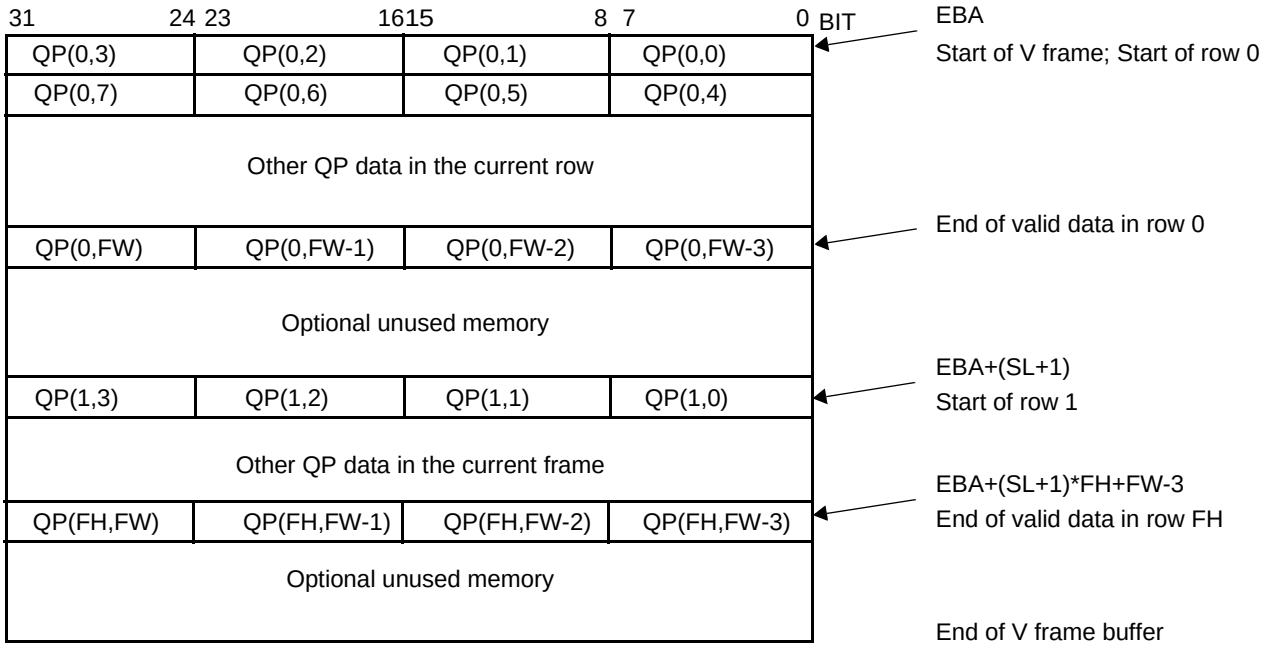
For MPEG-4, there is five buffers in the Postfilter Memory (see Table 44-176): A, B, C, D and E. The buffer A is an input buffer with a maximum used size of 64x21 pixel. It consists of two pages. The buffers B and C are temporary buffers of a size of 18x16 pixels each one. The buffer D is a 32x4-pixels output buffer. The buffer E stores QP parameters.

The following processing steps are performed for Y color component (Figure 44-128 and Figure 44-129):

1. Initial step before processing a page: the first page of the buffer A contains two input non-shifted macroblocks, the buffer B, C and D contents are not specified.
2. Deblocking columns of the shifted macroblock 1:
   a) the macroblock 1 is read from the buffer A in the column order for mode decision and filtering,
   b) filter results are written to the buffer B in the column order,
3. Deblocking columns of the shifted macroblock 2:
   a) the macroblock 2 is read from the buffer A in the column order for mode decision and filtering,
   b) filter results are written to the buffer C in the column order.
4. Deblocking rows of the macroblock 1:
   a) the macroblock 1 is read from the buffer A in the row order for mode decision,
   b) the macroblock 1 is read from the buffer B in the row order for filtering,
   c) filter results are written to the buffer A in the row order,
5. Deblocking columns of the shifted macroblock 3:
   a) the macroblock 3 is read from the buffer A in the column order for mode decision and filtering,
   b) filter results are written to the buffer B in the column order,
6. Deblocking rows of the macroblock 2:
   c) the macroblock 2 is read from buffer A in the row order for mode decision,
   d) the macroblock 2 is read from buffer C in the row order for filtering,
   e) filter results are written to the buffer A in the row order,
7. Deringing:
   a) the first page of the buffer A contents are read in the row order for mode decision and filtering,
   b) filter results are written to the buffer D.
   c) the buffer D contents are moved to the system memory by the IDMAC.
   d) a new data is loaded to the first page of the buffer A instead of processed rows.
8. Toggle a page pointer in buffer A and return to the step 2.

Macroblock QP parameters are calculated by the MCU. They should be stored in the external memory in the same order as pixel data (see Figure 44-127). Each QP parameters occupies 6 LSB bits in a byte (see Table 44-142).





Parameters for DMAPF_0 channel:

EBA - External Base Address
SL - Stride Line - 1
FW - Y Frame Width/16 - 1
FH - Y Frame Height/16 - 1

Figure 44-127. QP Parameters Allocation in the System Memory

Table 44-142. QP Parameter Format

bit	7							0
contents	X	QP						

For the U and V color components, the same algorithm is used excluding deringing step. Additionally, each deblocking step described above means simultaneous processing four neighboring shifted macroblocks of the U or V component.

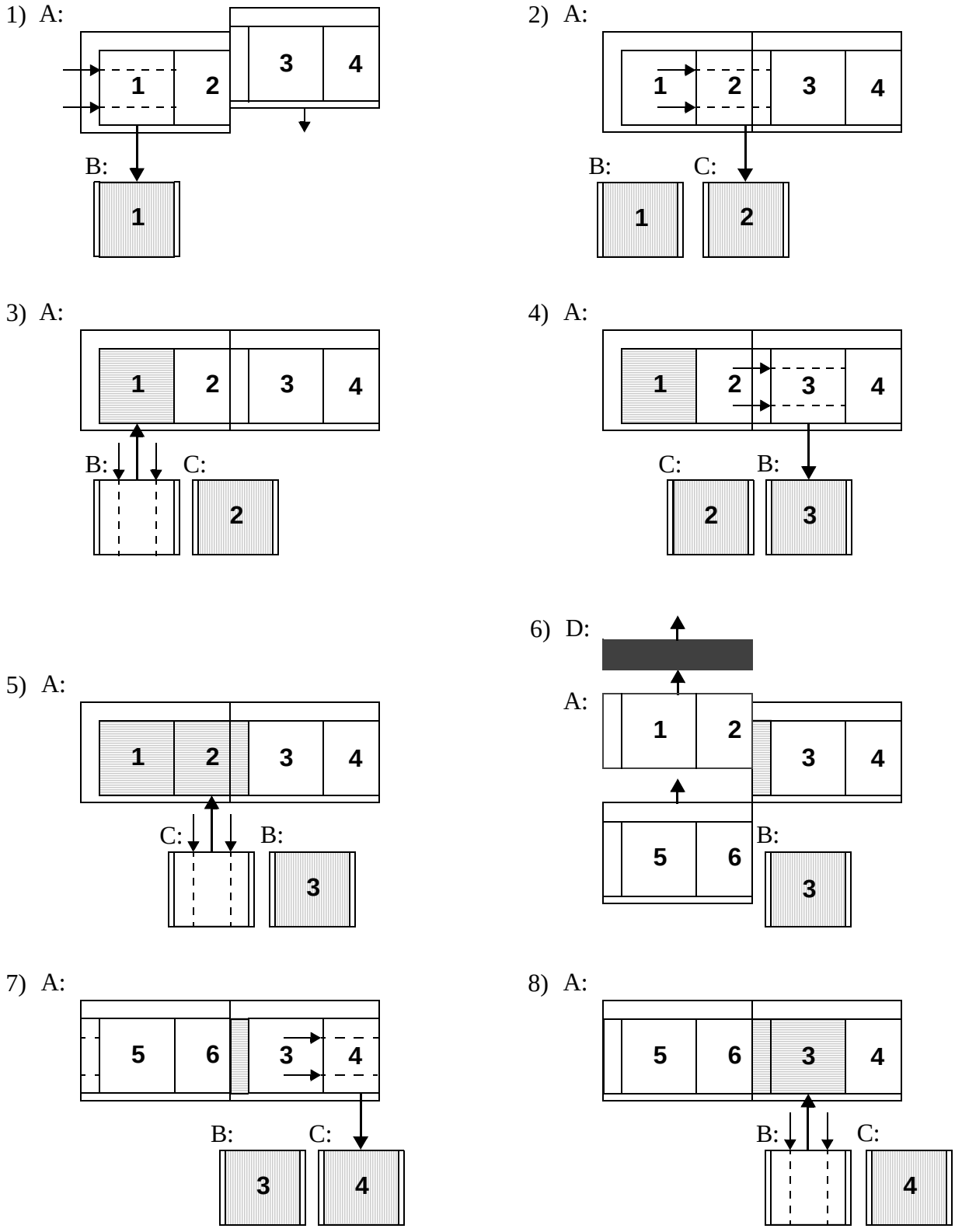


Figure 44-128. Processing Flow for Y Component in MPEG-4 Mode (Part 1)

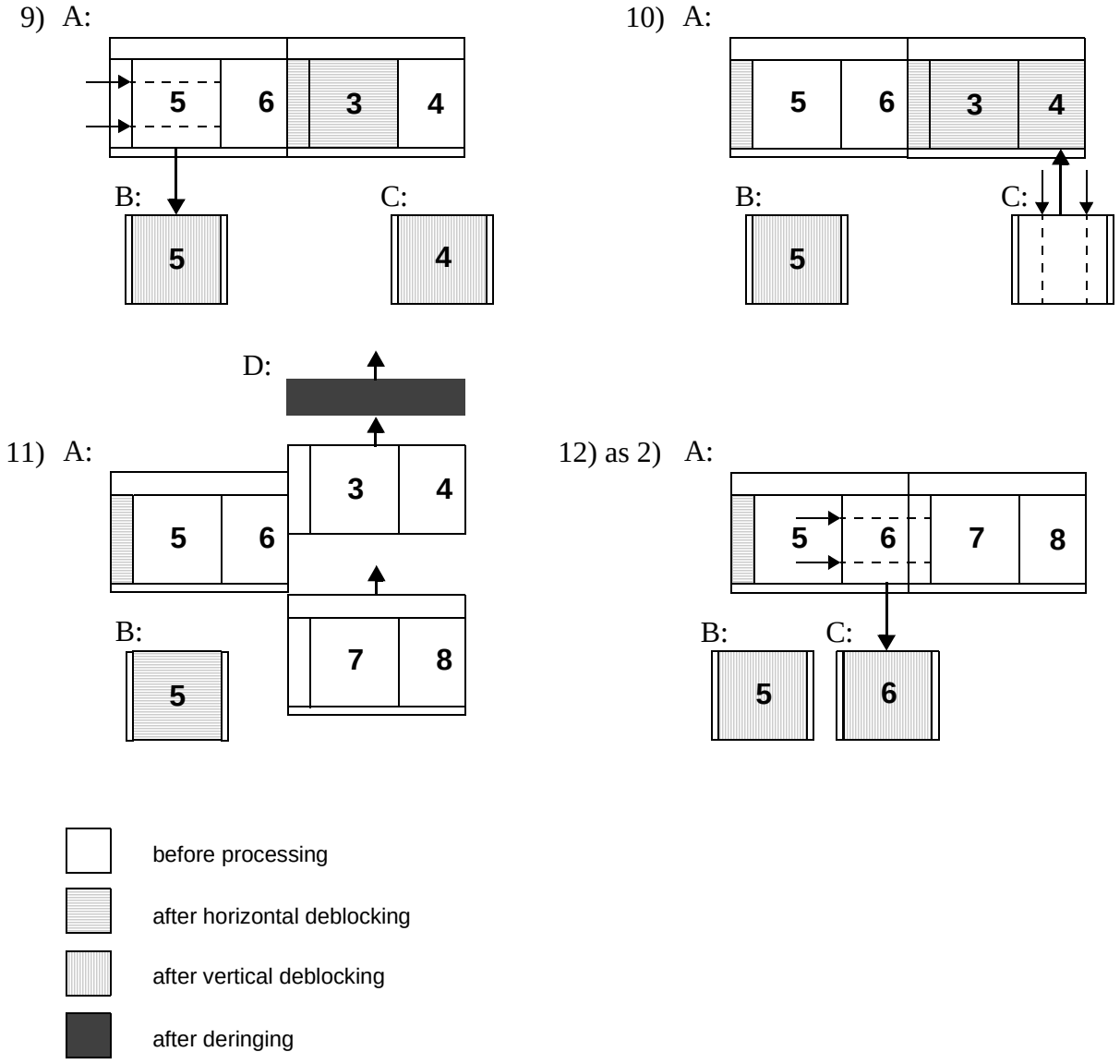


Figure 44-129. Processing Flow for Y Component in MPEG-4 Mode (Part 2)

44.4.3.2.2 H.264 Mode

For H.264 mode, the operation sequence does not include deringing task:

1. Deblocking of a Y component macroblock.
2. Repeating steps 1 until end of frame.
3. Deblocking of a U component macroblock.
4. Repeating step 3 until end of frame.
5. Deblocking of a V component macroblock.

6. Repeating step 5 until end of frame.

There is three buffers in the Postfilter Memory: A, B and C. The buffer A is an input buffer with a maximal used size of 64x20 pixel. It consists of two pages. The buffer B stores BS parameters. The buffer C stores QP parameters and filter offsets.

The following processing steps are performed (Figure 44-133 and Figure 44-134):

1. Initial step before processing a page: the first page of the buffer A contains two input non-shifted macroblocks.
2. Deblocking rows of the macroblock 1:
   a) the macroblock 1 is read from the buffer A in the row order for mode decision and filtering,
   b) filter results are written to the buffer A in the row order,
3. Deblocking columns of the macroblock 1:
   a) the macroblock 1 is read from the buffer A in the column order for mode decision and filtering,
   b) filter results are written to the buffer A in the column order.
4. Deblocking rows of the macroblock 2:
   a) the macroblock 2 is read from the buffer A in the row order for mode decision and filtering,
   b) filter results are written to the buffer A in the row order,
5. Deblocking columns of the macroblock 2:
   a) the macroblock 2 is read from the buffer A in the column order for mode decision and filtering,
   b) filter results are written to the buffer A in the column order.
6. Deblocking rows of the macroblock 3:
   c) the macroblock 3 is read from buffer A in the row order for mode decision and filtering,
   d) filter results are written to the buffer A in the row order,
   e) the first page of the buffer A contents are moved to the system memory by the IDMAC.
7. Toggle a page pointer in buffer A and return to the step 2.

Eight DMA channels service the PF in H.264 mode: three input data channels for each color component, three output data channels for each color component and two parameter channel. The DMAPF_0 parameter channel supplies the Quantization Parameter for luma (QP), Quantization Parameter for Chroma (QPC), Filter Offset A (FOA) and Filter Offset B (FOB) parameters. The DMAPF_1 parameter channel supplies BS parameters for each macroblock.

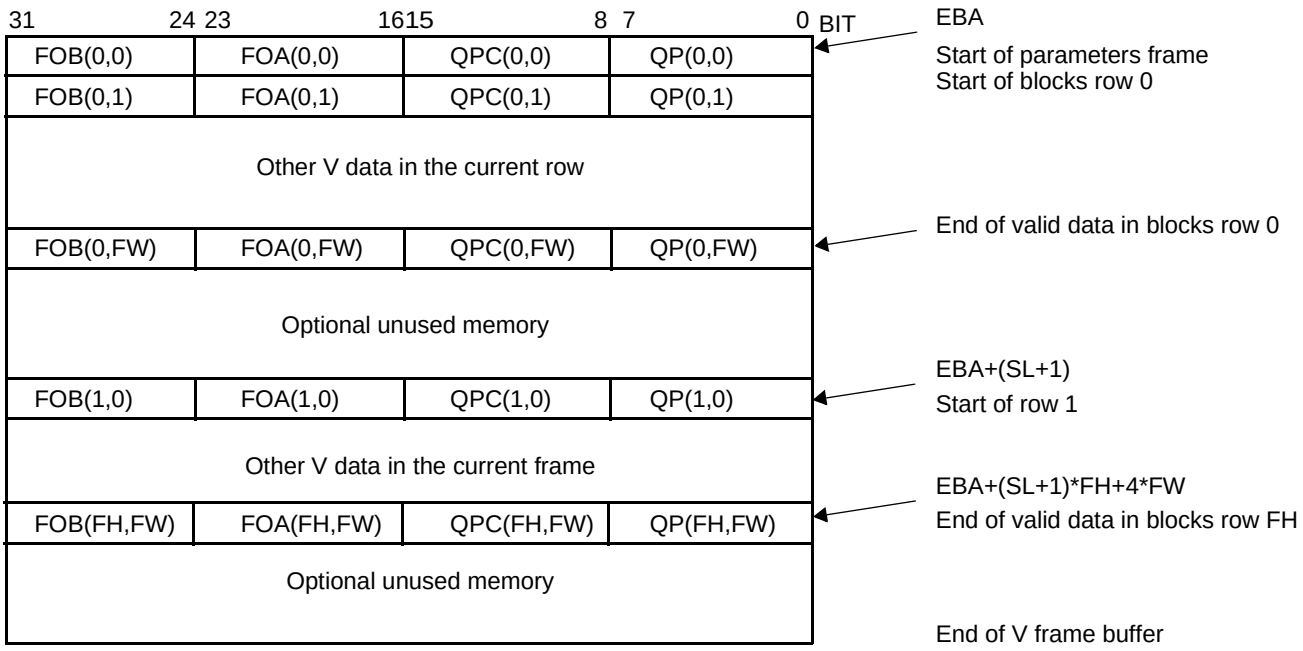
The QP, QPC, FOA, FOB parameters are packed into a 32-bit word as shown in Table 44-143. The DMAPF_0 channel should read these parameters with BPP corresponding to 32.

Table 44-143. QP, QPC, FOA, FOB Parameters Format in the External Memory (Little Endian)

bit	31	28 … 24	23	20 … 16	15	13 … 8	7	5 … 0
contents	X	FOB	X	FOA	X	QPC	X	QP

The QP parameter is an unsigned integer, assuming values from 0 to 51 (6 bits). The QPC parameter is an unsigned integer, assuming values from 0 to 39 (6 bits). The FOA and FOB parameters are signed integers,

assuming values from -12 to 12 (5 bits) in 2's complement representation. The parameters has to be written to the external memory as shown in Figure 44-130.



Parameters for DMAPF_0 channel:

EBA - External Base Address
SL - Stride Line - 1
FW - Y Frame Width/16 - 1
FH - Y Frame Height/16 - 1

Figure 44-130. QP, QPC, FOA, FOB Parameters Allocation in the System Memory (Little Endian)

The BS parameters are calculated by the MCU. There are two BS parameters for each 4x4 pixel block - for left block edge (BSL) and for upper block edge (BSU) (see Figure 44-131). Both parameters are stored as a single BSB parameter occupied one byte in the memory (see Table 44-142). The BSB parameters should be stored in the external memory in the same order as pixel data (see Figure 44-132).

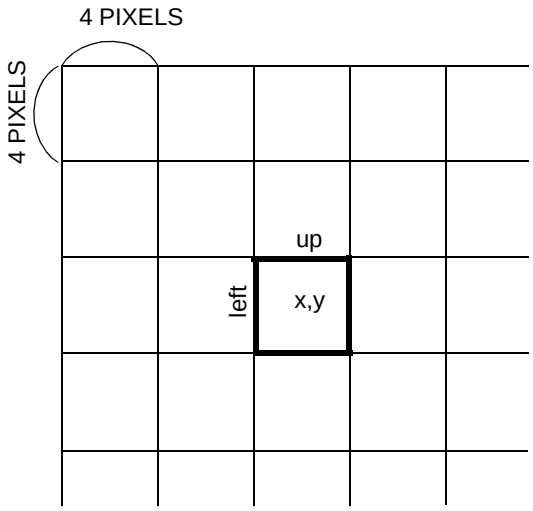
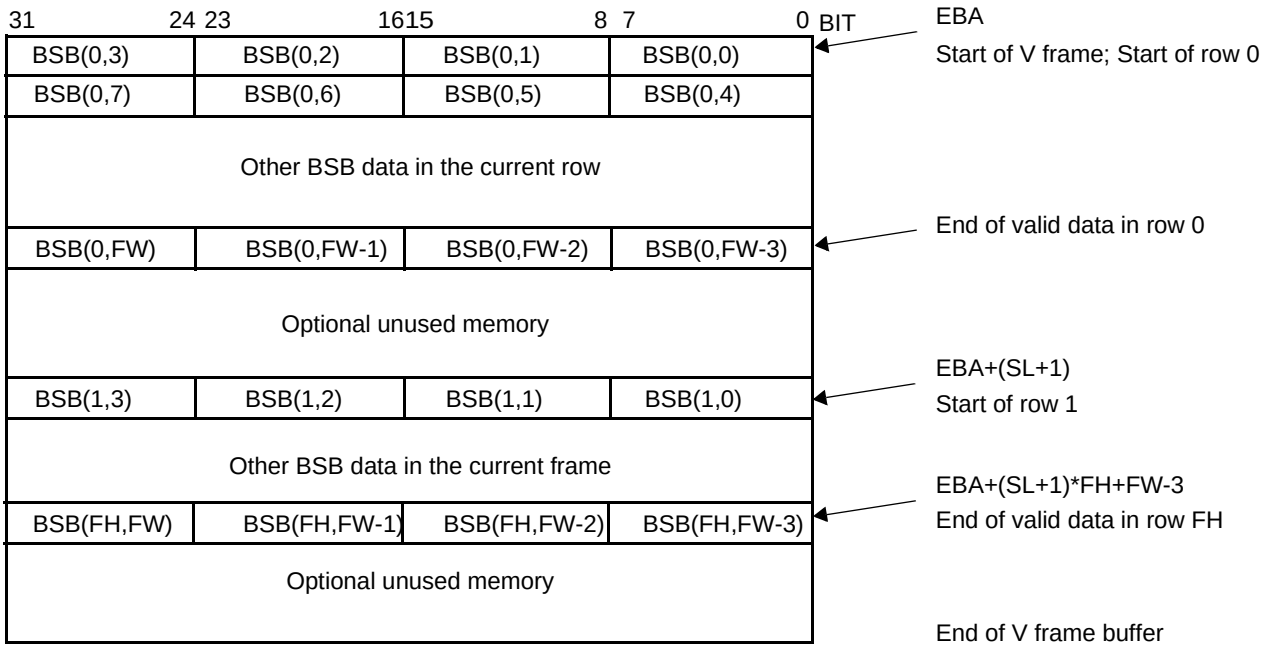


Figure 44-131. BSB Parameter Definition

Table 44-144. BSB Parameter Format

bit	7			0
contents	X	BSU	X	BSL



Parameters for DMAPF_1 channel:

EBA - External Base Address
SL - Stride Line - 1
FW - Y Frame Width/4 - 1
FH - Y Frame Height/4 - 1

Figure 44-132. BSB Parameters Allocation in the System Memory (Little Endian)

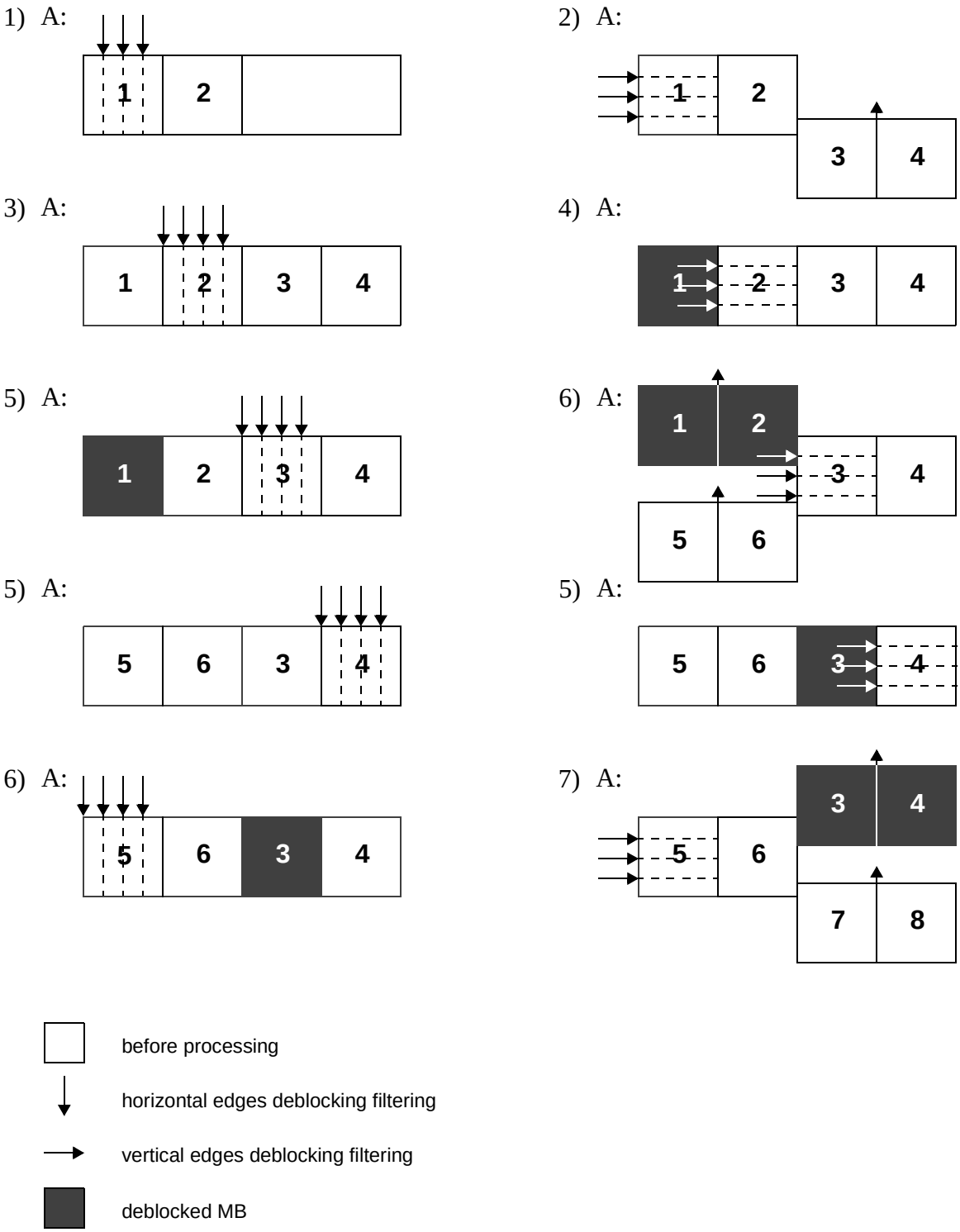


Figure 44-133. Processing Flow for H.264 (First Line of Frame)

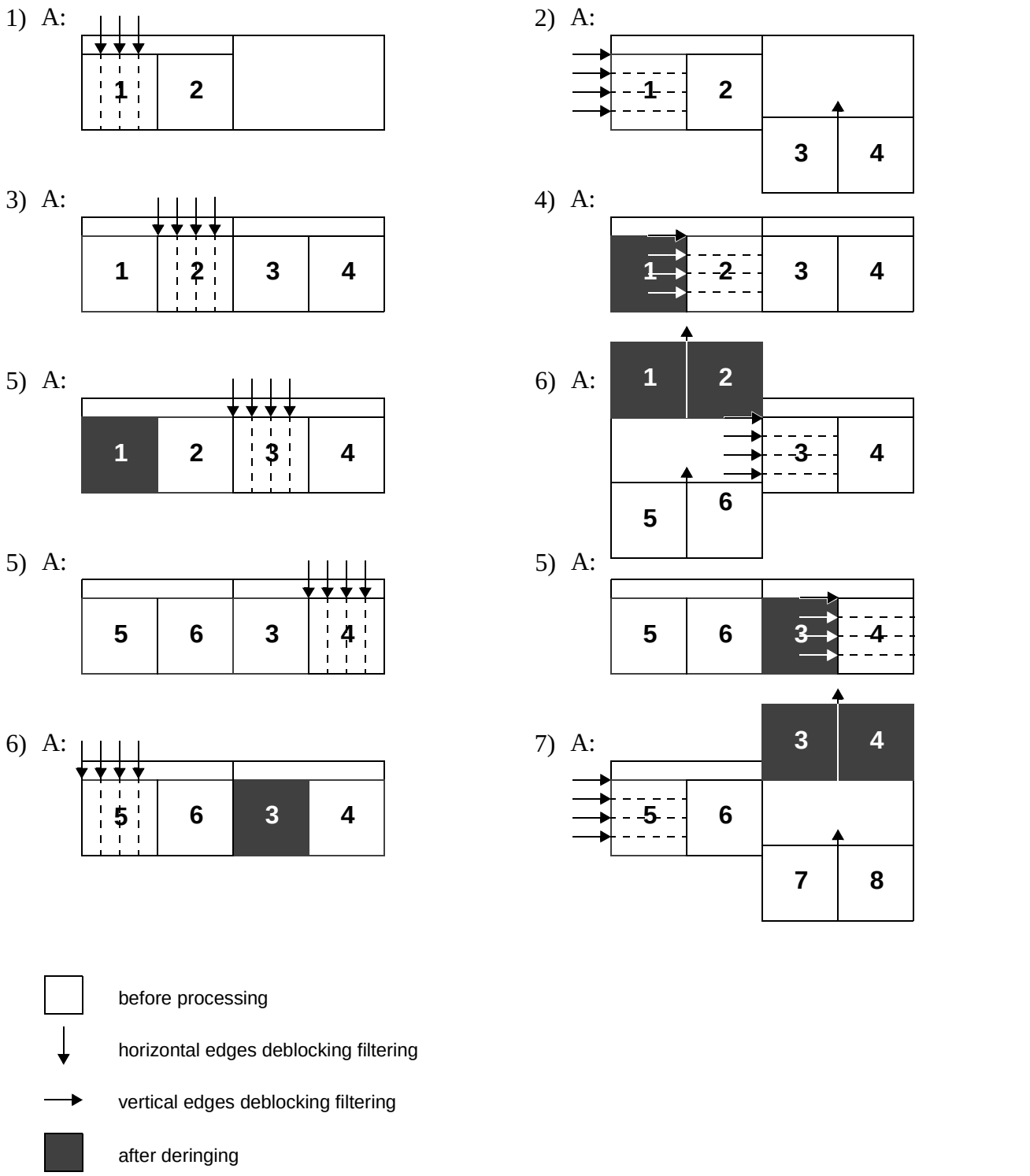


Figure 44-134. Processing Flow for H.264 (All Lines of the Frame Excluding the First)

44.4.3.3 Mode Decision Unit

The Mode Decision Unit is responsible for selection of a filter type to be used for calculation of the given output pixel. The unit reads and analyzes a row or a column of macroblock pixels. The analysis includes a number of steps (from 1 to 3 steps for MPEG-4 deblocking, from 1 to 4 steps for MPEG-4 deringing and for H.264 deblocking). At the finish step, the type of the filter to be applied is selected. The most of data path submodules in the Mode Decision Unit are reused for all tasks and operation modes. The analysis is controlled by tables defining a sequence of steps. For each algorithm, a specific control table is used.

44.4.3.4 Filter Arithmetic Unit

The derived filter number is fed to the Filter Arithmetic Unit which performs calculation of a single filter output in one clock cycle for MPEG-4 mode and in one or two clock cycles in H.264 mode. The unit The same filter data path are used for all operation modes. The difference between modes is reflected by separate control which defines a filter length and filter coefficients according to the selected filter type.

44.4.3.5 Postfilter Flow Control

The Postfilter Flow Control contains a set of finite state machines for each of performed tasks. An appropriate task state machine is invoked by the task control finite state machine according to the selected operation mode (MPEG-4 or H.264) and the current processing stage (deblocking or deringing, color component). The operation mode is selected via the PF_CONF Register.

In MPEG-4 mode, the software driver can request execution of only deblocking or only deringing or both of them. If just deringing is performed, only Y color component is processed.

In H.264 mode, the PF can operate with pauses. The MCU performs decoding in software. Decoder intermediate results should be processed by the PF. The PF output is sent back to the MCU to proceed decoding of the next frame. PF operation pauses allow parallel work of the MCU and the PF.

In this case, the MCU processes a portion of the frame (for example, a half). After that, it defines the number of rows to be processed by the PF (via programming the H264_Y_PAUSE_ROW parameter in the PF_CONF Register) and enables the postfilter task The PF performs processing of the Y component frame until the desired row and stops after that. The MCU updates the H264_Y_PAUSE_ROW parameter on readiness of the next frame portion. The PF resumes operation. After finishing the whole Y component frame, the PF processes the U and V component frames without pauses.

44.4.4 Synchronous Display Controller (SDC)

44.4.4.1 Block Diagram

The SDC Block Diagram is shown in Figure 44-135. The SDC data path includes the Synchronous Display Buffer Memory, the Combining Unit, the Cursor Generator and the Synchronous Display Adapter.

The Parameter Buffer is used for double buffering of all the parameters received from the configuration registers and synchronous their change on frame boundaries. The Synchronous Display Adapter provides the required timing control and display signal interface for VSYNC, HSYNC, etc.

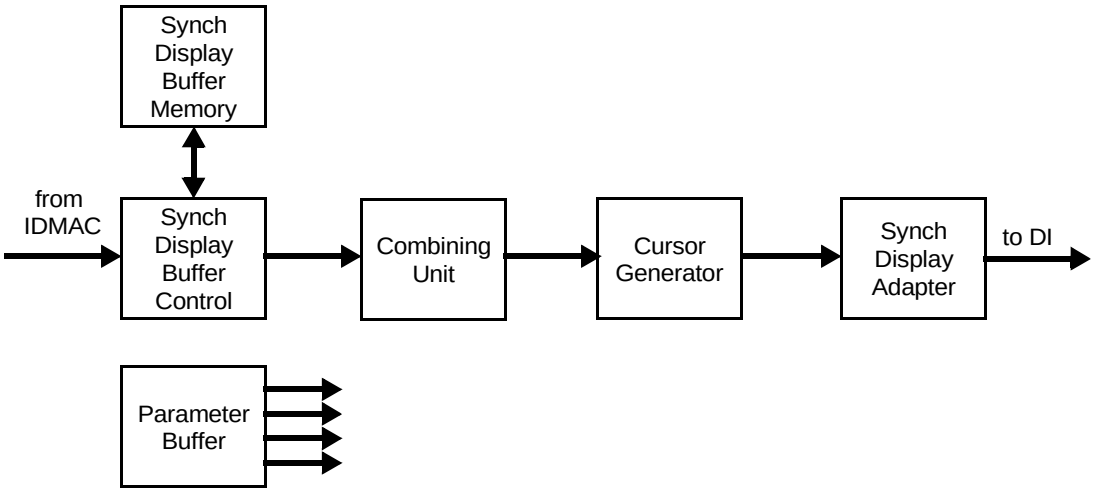


Figure 44-135. SDC Block Diagram

44.4.4.2 Input Data Formats

The pixel data to be displayed is written to the Synchronous Display Buffer Memory. There are two FIFO in the memory - for background and foreground planes (see Table 44-178).

Each FIFO contains eight pages, the page size is eight 64-bit words. Each word contains two color pixels in the RGBA/YUVA formats or eight 8-bits monochrome pixels or four 8-bits monochrome graphics pixels with four 8-bits alpha parameters. Table 44-145 shows used pixel formats and burst sizes.

Table 44-145. Supported Pixel Formats and Bursts

Pixel Format		Number of Pixels in 8-Word AHB Burst	Internal IPU 32-Bits Word Format	AHB Burst Parameters		Internal IPU Burst Parameters		Max Page Size in Pixels
Type	Bits Per Pixel			Burst length, 32-bits words	Number of Bursts Per Page	Burst Length, 32-Bits Words	Number of Bursts Per Page	
Mono-chrome video	8	32	$D_3D_2D_1D_0$(8888)	8-9	1	8	1	32
Mono-chrome graphics	16	16	$D_1A_1D_0A_0$(8888)	8-9	1	8	1	16
Color video or graphics	4	64	RGBA(8888)	1-2	1	8	1	8
	8	32	RGBA(8888)	2-3	1	8	1	8
	16	16	RGBA(8888)	4-5	1	8	1	8
	32	8	RGBA(8888)	8-9	2	8	1	8

Additionally, one 64-bits mask word is stored in a register. The mask is used for disabling some regions in the display window. This provides windowing function for smart displays with a synchronous interface.

44.4.4.3 Displayed Planes

Figure 44-136 shows the planes displayed on a synchronous display. There are foreground and background planes. A background frame position (BGXP and BGYP) defined relative to the VSYNC start point and a whole screen size (SCREEN_WIDTH and SCREEN_HEIGHT) are set via the SDC_BG_POS, SDC_HOR_CONF and SDC_VER_CONF Registers correspondingly. A foreground frame position (BGXP and BGYP) defined relative to the VSYNC start point is set via the SDC_FG_POS Register. Sizes of both planes are set by programming the corresponding IDMAC channels (see Table 44-29 through Table 44-33).

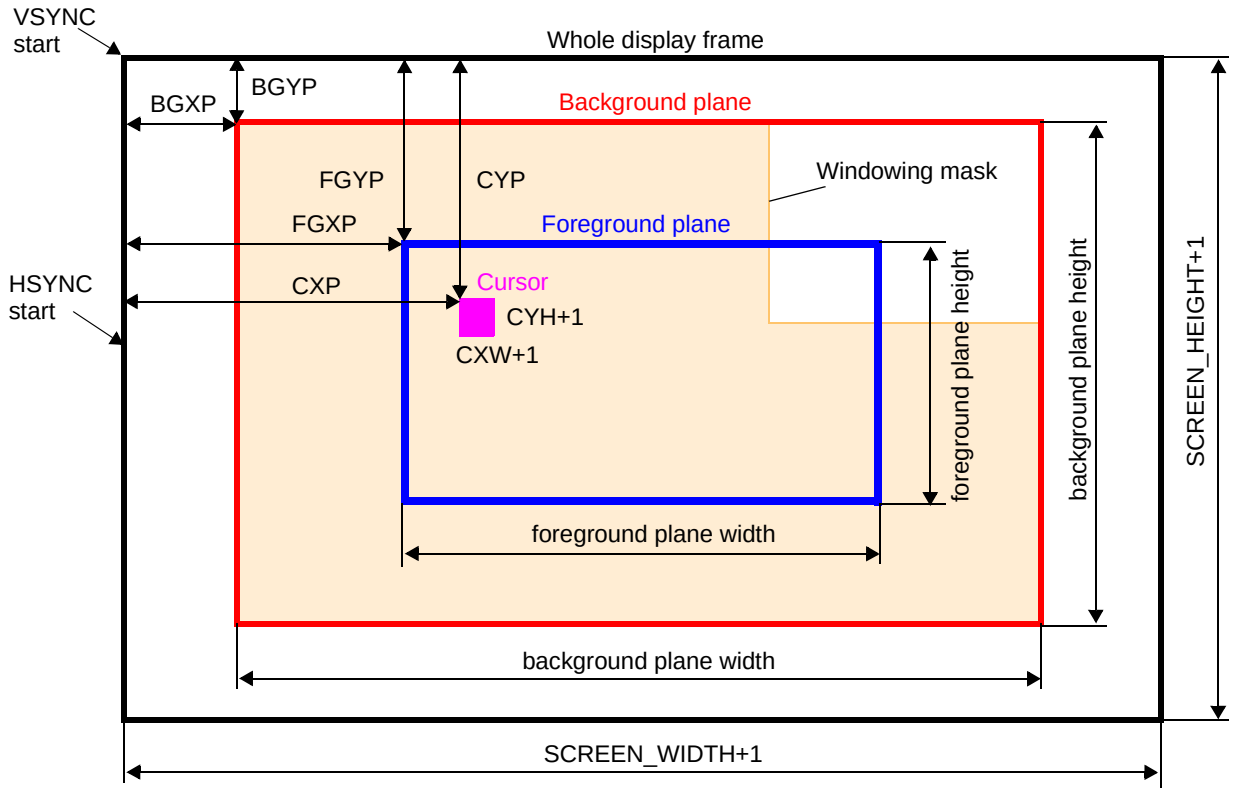


Figure 44-136. Displayed Planes

One of the planes can be graphics, another plane displays video. Selection of the graphics plane can be done via the SDC_COM_CONF Register.

Cursor position and parameters are set in the SDC_CUR_POS, SDC_CUR_BLINK_PWM_CTRL and SDC_CUR_MAP Registers.

The SDC is able to provide the windowing function on displays that have data enable control. This is achieved by masking of some screen regions according to a 1-bit/pixel mask prepared by MCU software. When the mask value is zero, the corresponding both background, foreground and cursor pixels are not displayed. This feature can be used for dual-port smart displays.

For memory-less displays, both foreground and background planes are sent to the display every refresh cycle. For dual-interface smart displays, data transfer depends on changes in plane and cursor positions, arriving of a new data, combining parameters and cursor parameters. If no data or parameters are changed, transfer to the display is not performed at the current refresh cycle. If only the foreground plane data and/or combining parameters have been changed, the foreground data and underlying background plane part are transferred to the display at the current refresh cycle. In all other cases, both foreground and background planes are send to the display.

44.4.4.4 Combining Unit

The Combining Unit performs combining foreground and background planes. The background plane may be a video image while the foreground plane is a graphics image or vice versa.

There are the following combining options:

— local alpha blending,
— global alpha blending,
— use of key color.

Combining mode is selected via the SDC_COM_CONF Register. The combining equation is:

$$OP = IGP \cdot \alpha + IVP \cdot (1 - \alpha)$$

where IGP - an input graphics pixel, IVP - an input video pixel, α=(A+floor(A/128))/256 - an alpha value, A - a global or local transparency parameter. The global A is written in the SDC_GRAPH_WIND_CTRL Register, the local A arrives together with the graphics pixel.

A graphics pixel becomes transparent when color keying is enabled and a pixel color matches a key color (independently on an alpha parameter).

Combining takes 3 cycles per color pixel or 1 cycle per monochrome pixel. The Combining Unit outputs 24-bit words in the RGB/YUV format for color pixels or 8-bit words for monochrome pixels.

44.4.4.5 Cursor Generator

The Combining Unit output is passes through the Cursor Generator. Cursor size and position are set via the SDC_CUR_POS Register, a cursor color - via the SDC_CUR_MAP Register. Different logic functions of combining the cursor with the image are supported as defined by the SDC_COM_CONF Register. The cursor can be blinking as defined by the SDC_CUR_BLINK_PWM_CTRL Register.

44.4.4.6 Display Refresh Rate

The SDC_HOR_CONF and SDC_VER_CONF Registers define screen width and screen height in pixels, the DI_DISP3_TIME_CONF registers defines the synchronous display (the display 3) clock period, the DI_DISP_ACC_CC Register defines the number of display clock cycles in one pixel clock cycle.

The display refresh rate is as follows:

$F_{REF} = F_{HSP}$ / ((SCREEN_WIDTH+1) * (SCREEN_HEIGHT+1) * (DISP3_IF_CLK_PER_WR / HSP_CLK_PERIOD1,2) * (DISP3_ IF_ CLK_ CNT_ D+1))

44.4.5 Asynchronous Display Controller (ADC)

44.4.5.1 Block Diagram

The ADC Block Diagram is shown in Figure 44-137.

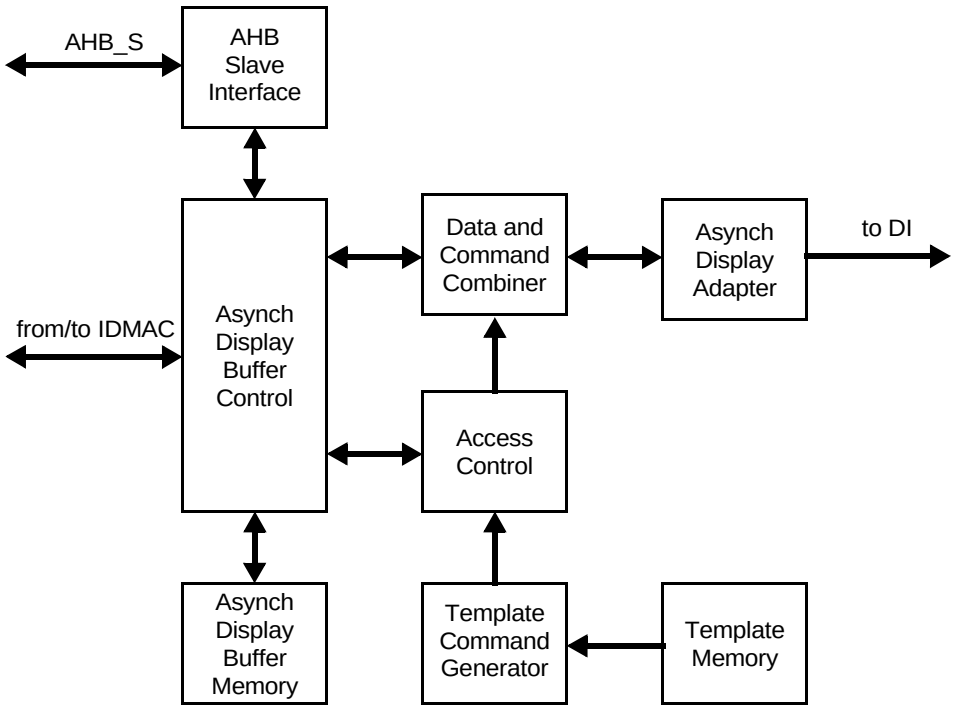


Figure 44-137. ADC Block Diagram

Data to be written/read to/from a smart display or a graphics controller arrives from/to the IDMAC or directly from/to the MCU. The data is temporarily stored in the Asynchronous Display Buffer Memory. The Access Control manages the generation of display commands and the combining of command and data is done by the Data and Command Combiner. The Asynchronous Display Adapter controls display frame timings and provides an interface to the DI.

The ADC is able to support up to three smart displays simultaneously with time multiplexed access.

All ADC control registers are not double buffered, meaning that for reprogramming the access channels or the display parameters, the MCU should wait for the DMA channel to complete its current frame transfer.

44.4.5.2 Display Access Modes

There are two display types to be supported. For the first type, display addressing is done by pointing the corresponding X, Y coordinates. For the second type, the full linear address is sent to the display. The display type is selected via the DISP#_TYPE parameter in the ADC_DISP0_CONF - ADC_DISP2_CONF Registers.

The data to be sent to the display can be treated by the IPU as pixels or a generic data with a word width of 16 or 24 bits (packed in a 32-bit word).

There are three possible sources of display data. the system memory, internal IPU processing parts (preprocessing and postprocessing) and the MCU (direct access to the display).

Five access channels allow to support up to five windows on different displays:

1. System memory channel 1 which can be configured for write or read access
2. System memory channel 2 which can be configured for write or read access
3. Preprocessing channel for write access (internal transfer of data for viewfinder application)
4. Postprocessing channel for write access (internal transfer of data with bypass of the system memory)
5. MCU direct access channel

Each of five windows may be displayed on one of three displays.

The ADC supports the display access modes listed in Table 44-146.

Table 44-146. Asynchronous Display Access Modes

Display Type	Source	Command Generation Method	Access Type	IDMAC/MCU Interface Format	Display Interface Format	Programming ADC/IDMAC Parameters
X,Y addressing, 6/8/9/12/16/18-bit interface	System memory	command buffer	write	32[1]-bit pixel data	up to 24-bit pixel data, data/command mapping format, 1/2/3/4-cycles transfer	DISP#_TYPE = 10, SYS1/2_MODE = 101 or 110 or 111, SYS1/2_ADDR_INC unused, PFS (DMAADC_2/3) = 000 or 100, BPP (DMAADC_2/3) - according to external pixel size
				16-bit generic data	16-bit generic data, data/command mapping format, 1/2/3/4-cycles transfer	DISP#_TYPE = 10, SYS1/2_MODE = 101 or 110 or 111, SYS1/2_ADDR_INC unused, PFS (DMAADC_2/3) = 111, BPP (DMAADC_2/3) = 010
				32[2]-bit generic data	24-bit generic data, data/command mapping format, 1/2/3/4-cycles transfer	DISP#_TYPE = 10, SYS1/2_MODE = 101 or 110 or 111, SYS1/2_ADDR_INC unused, PFS (DMAADC_2/3) = 111, BPP (DMAADC_2/3) = 000
				16-bit command (in 32-bit word)	up to 16-bit command, command mapping, 1/2/3/4-cycles transfer	DISP#_TYPE = 10, SYS1/2_MODE = 111, SYS1/2_ADDR_INC unused, PFS (DMAADC_2/3) = 111, BPP (DMAADC_2/3) = 010, even NPB+1
				32-bit command	up to 24-bit command, command mapping, 1/2/3/4-cycles transfer	DISP#_TYPE = 10, SYS1/2_MODE = 111, SYS1/2_ADDR_INC unused, PFS (DMAADC_2/3) = 111, BPP (DMAADC_2/3) = 000

Table 44-146. Asynchronous Display Access Modes (continued)

Display Type	Source	Command Generation Method	Access Type	IDMAC/MCU Interface Format	Display Interface Format	Programming ADC/IDMAC Parameters
X,Y addressing, 6/8/9/12/16/18-bit interface	System memory	template	write, read	32-bit pixel data	up to 24-bit pixel data, data/command mapping, 1/2/3/4-cycles transfer	DISP#_TYPE = 10, SYS1/2_MODE = 001 or 010 or 011 or 100, SYS1/2_ADDR_INC unused, PFS (DMAADC_2/3/6/7) = 000 or 100, BPP (DMAADC_2/3/6/7) - according to external pixel size
				16-bit generic data	16-bit generic data, data/command mapping, 1/2/3/4-cycles transfer	DISP#_TYPE = 10, SYS1/2_MODE = 001 or 010 or 011 or 100, SYS1/2_ADDR_INC unused, PFS (DMAADC_2/3/6/7) = 111, BPP (DMAADC_2/3/6/7) = 010
				32-bit generic data	24-bit generic data, data/command mapping, 1/2/3/4-cycles transfer	DISP#_TYPE = 10, SYS1/2_MODE = 001 or 010 or 011 or 100, SYS1/2_ADDR_INC unused, PFS (DMAADC_2/3/6/7) = 111, BPP (DMAADC_2/3/6/7) = 000
	IC (pre-processing)	template	write	32-bit pixel data	up to 24-bit pixel data, data/command mapping, 1/2/3/4-cycles transfer	DISP#_TYPE = 10, PRP_ADDR_INC unused, PFS (DMAADC_0) = 100, BPP (DMAADC_0) = 000
	IC (post-processing)	template	write	32-bit pixel data	up to 24-bit pixel data, data/command mapping, 1/2/3/4-cycles transfer	DISP#_TYPE = 10, PP_ADDR_INC unused, PFS (DMAADC_1) = 100, BPP (DMAADC_1) = 000
	MCU	template	write, read	16-bit generic data	16-bit generic data, data/command mapping, 1/2/3/4-cycles transfer	DISP#_TYPE = 10, MCU_DISP#_DATA_WIDTH = 0
				32-bit generic data	24-bit generic data, data/command mapping, 1/2/3/4-cycles transfer	DISP#_TYPE = 10, MCU_DISP#_DATA_WIDTH = 1

Table 44-146. Asynchronous Display Access Modes (continued)

Display Type	Source	Command Generation Method	Access Type	IDMAC/MCU Interface Format	Display Interface Format	Programming ADC/IDMAC Parameters
Full addressing, 16-bit interface with byte enable	System memory	command buffer	write	32-bit pixel data	16-bit pixel data, data/command mapping, single 1-cycle full 16-bit transfer	DISP#_TYPE = 01, SYS1/2_MODE = 101 or 110 or 111, SYS1/2_ADDR_INC unused, PFS (DMAADC_2/3) = 000 or 100, BPP (DMAADC_2/3) - according to external pixel size
					up to 24-bit pixel data, data/command mapping, single 2-cycle full 32-bit transfer	DISP#_TYPE = 01, SYS1/2_MODE = 101 or 110 or 111, SYS1/2_ADDR_INC unused, PFS (DMAADC_2/3) = 000 or 100, BPP (DMAADC_2/3) - according to external pixel size
				16-bit generic data	16-bit generic data, data/command mapping, single 1-cycle full 16-bit transfer	DISP#_TYPE = 01, SYS1/2_MODE = 101 or 110 or 111, SYS1/2_ADDR_INC unused, PFS (DMAADC_2/3) = 111, BPP (DMAADC_2/3) = 010
				32-bit generic data	24-bit generic data, data/command mapping, single 2-cycles full 32-bit transfer	DISP#_TYPE = 01, SYS1/2_MODE = 101 or 110 or 111, SYS1/2_ADDR_INC unused, PFS (DMAADC_2/3) = 111, BPP (DMAADC_2/3) = 000
				16-bit command (in 32-bit word)	16-bit command, command mapping, single 1-cycle 16-bit transfer with byte enable bits defined in input command word	DISP#_TYPE = 01, SYS1/2_MODE = 111, SYS1/2_ADDR_INC unused, PFS (DMAADC_4/5) = 111, BPP (DMAADC_4/5) = 010, even NPB+1

Table 44-146. Asynchronous Display Access Modes (continued)

Display Type	Source	Command Generation Method	Access Type	IDMAC/MCU Interface Format	Display Interface Format	Programming ADC/IDMAC Parameters
Full addressing, 16-bit interface with byte enable	System memory	template	write	32-bit pixel data	8-bit pixel data, data/command mapping, 1-cycle[3] 16-bit transfer with byte enable bits ‘01’ or ‘10’ matching pixel address	DISP#_TYPE = 01, SYS1/2_MODE = 001 or 011, SYS1/2_ADDR_INC = 00, PFS (DMAADC_2/3) = 000 or 100, BPP (DMAADC_2/3) - according to external pixel size
					16-bit pixel data, data/command mapping, single 1-cycle full 16-bit transfer	DISP#_TYPE = 01, SYS1/2_MODE = 001 or 011, SYS1/2_ADDR_INC = 01, PFS (DMAADC_2/3) = 000 or 100, BPP (DMAADC_2/3) - according to external pixel size
					upto 24-bit pixel data, data/command mapping, single 2-cycle full 32-bit transfer	DISP#_TYPE = 01, SYS1/2_MODE = 001 or 011, SYS1/2_ADDR_INC = 11, PFS (DMAADC_2/3) = 000 or 100, BPP (DMAADC_2/3) - according to external pixel size
				8-bit generic data	8-bit generic data, data/command mapping, single 1-cycle 16-bit transfer with byte enable bits which are 11 in row middle and may be ‘10’ or ‘11’ at row beginning and ‘01’ or ‘11’ at row end	DISP#_TYPE = 01, SYS1/2_MODE = 001 or 011, SYS1/2_ADDR_INC = 01, PFS (DMAADC_2/3) = 111, BPP (DMAADC_2/3) = 011
				16-bit generic data	16-bit generic data, data/command mapping, single 1-cycle full 16-bit transfer	DISP#_TYPE = 01, SYS1/2_MODE = 001 or 011, SYS1/2_ADDR_INC = 01, PFS (DMAADC_2/3) = 111, BPP (DMAADC_2/3) = 010

Table 44-146. Asynchronous Display Access Modes (continued)

Display Type	Source	Command Generation Method	Access Type	IDMAC/MCU Interface Format	Display Interface Format	Programming ADC/IDMAC Parameters
Full addressing, 16-bit interface with byte enable	System memory	template	write	32-bit generic data	24-bit generic data, data/command mapping, single 2-cycle full 32-bit transfer	DISP#_TYPE = 01, SYS1/2_MODE = 001 or 011, SYS1/2_ADDR_INC = 11, PFS (DMAADC_2/3) = 111, BPP (DMAADC_2/3) = 000
			read	32-bit pixel data	8-bit pixel data, data/command mapping, single 1-cycle 16-bit transfer with byte enable bits ‘01’ or ‘10’ matching pixel address	DISP#_TYPE = 01, SYS1/2_MODE = 010 or 100, SYS1/2_ADDR_INC = 00, PFS (DMAADC_6/7) = 100, BPP (DMAADC_6/7) - according to external pixel size
					16-bit pixel data, data/command mapping, 1-cycle full 16-bit transfer	DISP#_TYPE = 01, SYS1/2_MODE = 010 or 100, SYS1/2_ADDR_INC = 01, PFS (DMAADC_6/7) = 100, BPP (DMAADC_6/7) - according to external pixel size
					upto 24-bit pixel data, data/command mapping, single 2-cycle full 32-bit transfer	DISP#_TYPE = 01, SYS1/2_MODE = 010 or 100, SYS1/2_ADDR_INC = 11, PFS (DMAADC_6/7) = 100, BPP (DMAADC_6/7) - according to external pixel size
				16-bit generic data	16-bit generic data, data/command mapping, single 1-cycle full 16-bit transfer	DISP#_TYPE = 01, SYS1/2_MODE = 010 or 100, SYS1/2_ADDR_INC = 01, PFS (DMAADC_6/7) = 111, BPP (DMAADC_6/7) = 010
				32-bit generic data	24-bit generic data, data/command mapping, single 2-cycle full 32-bit transfer	DISP#_TYPE = 01, SYS1/2_MODE = 010 or 100, SYS1/2_ADDR_INC = 11, PFS (DMAADC_6/7) = 111, BPP (DMAADC_6/7) = 000

Table 44-146. Asynchronous Display Access Modes (continued)

Display Type	Source	Command Generation Method	Access Type	IDMAC/MCU Interface Format	Display Interface Format	Programming ADC/IDMAC Parameters
Full addressing, 16-bit interface with byte enable	IC (pre-processing)	template	write	32-bit pixel data	8-bit pixel data, data/command mapping, single 1-cycle 16-bit transfer with byte enable bits ‘01’ or ‘10’ matching pixel address	DISP#_TYPE = 01, PRP_ADDR_INC = 00, PFS (DMAADC_0) = 100, BPP (DMAADC_0) = 000
					16-bit pixel data, data/command mapping, 1-cycle full 16-bit transfer	DISP#_TYPE = 01, PRP_ADDR_INC = 01, PFS (DMAADC_0) = 100, BPP (DMAADC_0) = 000
					upto 24-bit pixel data, data/command mapping, single 2-cycle full 32-bit transfer	DISP#_TYPE = 01, PRP_ADDR_INC = 11, PFS (DMAADC_0) = 100, BPP (DMAADC_0) = 000
	IC (post-processing)	template	write	32-bit pixel data	8-bit pixel data, data/command mapping, single 1-cycle 16-bit transfer with byte enable bits ‘01’ or ‘10’ matching pixel address	DISP#_TYPE = 01, PP_ADDR_INC = 00, PFS (DMAADC_1) = 100, BPP (DMAADC_1) = 000
					16-bit pixel data, data/command mapping, 1-cycle full 16-bit transfer	DISP#_TYPE = 01, PP_ADDR_INC = 01, PFS (DMAADC_1) = 100, BPP (DMAADC_1) = 000
					upto 24-bit pixel data, data/command mapping, single 2-cycle full 32-bit transfer	DISP#_TYPE = 01, PP_ADDR_INC = 11, PFS (DMAADC_1) = 100, BPP (DMAADC_1) = 000

Table 44-146. Asynchronous Display Access Modes (continued)

Display Type	Source	Command Generation Method	Access Type	IDMAC/MCU Interface Format	Display Interface Format	Programming ADC/IDMAC Parameters
Full addressing, 16-bit interface with byte enable	MCU	template	write, read	8-bit generic data	16-bit generic data, data/command mapping, single 1-cycle 16-bit transfer with byte enable bits ‘01’ or ‘10’ matching AHB byte enable bits	DISP#_TYPE = 01, MCU_DISP#_DATA_WIDTH = 0
				16-bit generic data	16-bit generic data, data/command mapping, single 1-cycle full 16-bit transfer	DISP#_TYPE = 01, MCU_DISP#_DATA_WIDTH = 0
				32-bit generic data	two 16-bit generic data words, data/command mapping, two 1-cycle 16-bit transfers with byte enable bits ‘01’ or ‘10’ in each cycle matching AHB byte enable bits	DISP#_TYPE = 01, MCU_DISP#_DATA_WIDTH = 0
					24-bit generic data, data/command mapping, single 2-cycle full 32-bit transfer	DISP#_TYPE = 01, MCU_DISP#_DATA_WIDTH = 1

Table 44-146. Asynchronous Display Access Modes (continued)

Display Type	Source	Command Generation Method	Access Type	IDMAC/MCU Interface Format	Display Interface Format	Programming ADC/IDMAC Parameters
Full addressing, 8/16-bit interface without byte enable	System memory	command buffer	write	32-bit pixel data	8-bit pixel data, data/command mapping, single 1-cycle 8-bit transfer (only for 8-bit interface)	DISP#_TYPE = 00, SYS1/2_MODE = 101 or 110 or 111, SYS1/2_ADDR_INC unused, PFS (DMAADC_2/3) = 000 or 100, BPP (DMAADC_2/3) - according to external pixel size
					16-bit pixel data, data/command mapping, single 1/2-cycle 16-bit transfer	DISP#_TYPE = 00, SYS1/2_MODE = 101 or 110 or 111, SYS1/2_ADDR_INC unused, PFS (DMAADC_2/3) = 000 or 100, BPP (DMAADC_2/3) - according to external pixel size
					upto 24-bit pixel data, data/command mapping, single 2/4-cycle 32-bit transfer	DISP#_TYPE = 00, SYS1/2_MODE = 101 or 110 or 111, SYS1/2_ADDR_INC unused, PFS (DMAADC_2/3) = 000 or 100, BPP (DMAADC_2/3) - according to external pixel size
				16-bit generic data	16-bit generic data, data/command mapping, single 1/2-cycle 16-bit transfer	DISP#_TYPE = 00, SYS1/2_MODE = 101 or 110 or 111, SYS1/2_ADDR_INC unused, PFS (DMAADC_2/3) = 111, BPP (DMAADC_2/3) = 010
				32-bit generic data	24-bit generic data, data/command mapping, single 2/4-cycles 32-bit transfer	DISP#_TYPE = 00, SYS1/2_MODE = 101 or 110 or 111, SYS1/2_ADDR_INC unused, PFS (DMAADC_2/3) = 111, BPP (DMAADC_2/3) = 000

Table 44-146. Asynchronous Display Access Modes (continued)

Display Type	Source	Command Generation Method	Access Type	IDMAC/MCU Interface Format	Display Interface Format	Programming ADC/IDMAC Parameters
Full addressing, 8/16-bit interface without byte enable	System memory	command buffer	write	16-bit command (in 32-bit word)	8-bit command, command mapping, single 1-cycle 8-bit transfer (only for 8-bit interface with truncation of source data bits)	DISP#_TYPE = 00, SYS1/2_MODE = 111, SYS1/2_ADDR_INC unused, PFS (DMAADC_4/5) = 111, BPP (DMAADC_4/5) = 010, even NPB+1
					16-bit command, command mapping, single 1/2-cycle 16-bit transfer	DISP#_TYPE = 00, SYS1/2_MODE = 111, SYS1/2_ADDR_INC unused, PFS (DMAADC_4/5) = 111, BPP (DMAADC_4/5) = 010, even NPB+1
		template	write	32-bit pixel data	8-bit pixel data, data/command mapping, single 1-cycle 8-bit transfer (only for 8-bit interface)	DISP#_TYPE = 00, SYS1/2_MODE = 001 or 011, SYS1/2_ADDR_INC = 00, PFS (DMAADC_2/3) = 000 or 100, BPP (DMAADC_2/3) - according to external pixel size
					16-bit pixel data, data/command mapping, single 1/2-cycle 16-bit transfer	DISP#_TYPE = 00, SYS1/2_MODE = 001 or 011, SYS1/2_ADDR_INC = 01, PFS (DMAADC_2/3) = 000 or 100, BPP (DMAADC_2/3) - according to external pixel size
					upto 24-bit pixel data, data/command mapping, single 2/4-cycle 32-bit transfer	DISP#_TYPE = 00, SYS1/2_MODE = 001 or 011, SYS1/2_ADDR_INC = 11, PFS (DMAADC_2/3) = 000 or 100, BPP (DMAADC_2/3) - according to external pixel size
				16-bit generic data	16-bit generic data, data/command mapping, single 1/2-cycle 16-bit transfer	DISP#_TYPE = 00, SYS1/2_MODE = 001 or 011, SYS1/2_ADDR_INC = 01, PFS (DMAADC_2/3) = 111, BPP (DMAADC_2/3) = 010

Table 44-146. Asynchronous Display Access Modes (continued)

Display Type	Source	Command Generation Method	Access Type	IDMAC/MCU Interface Format	Display Interface Format	Programming ADC/IDMAC Parameters
Full addressing, 8/16-bit interface without byte enable	System memory	template	write	32-bit generic data	24-bit generic data, data/command mapping, single 2/4-cycles 32-bit transfer	DISP#_TYPE = 00, SYS1/2_MODE = 001 or 011, SYS1/2_ADDR_INC = 11, PFS (DMAADC_2/3) = 111, BPP (DMAADC_2/3) = 000
			read	32-bit pixel data	8-bit pixel data, data/command mapping, single 1-cycle 8-bit transfer (only for 8-bit interface)	DISP#_TYPE = 00, SYS1/2_MODE = 010 or 100, SYS1/2_ADDR_INC = 00, PFS (DMAADC_6/7) = 100, BPP (DMAADC_6/7) - according to external pixel size
					16-bit pixel data, data/command mapping, single 1/2-cycle 16-bit transfer	DISP#_TYPE = 00, SYS1/2_MODE = 010 or 100, SYS1/2_ADDR_INC = 01, PFS (DMAADC_6/7) = 100, BPP (DMAADC_6/7) - according to external pixel size
					upto 24-bit pixel data, data/command mapping, single 2/4-cycle 32-bit transfer	DISP#_TYPE = 00, SYS1/2_MODE = 010 or 100, SYS1/2_ADDR_INC = 11, PFS (DMAADC_6/7) = 100, BPP (DMAADC_6/7) - according to external pixel size
				16-bit generic data	16-bit generic data, data/command mapping, single 1/2-cycle 16-bit transfer	DISP#_TYPE = 00, SYS1/2_MODE = 010 or 100, SYS1/2_ADDR_INC = 01, PFS (DMAADC_6/7) = 111, BPP (DMAADC_6/7) = 010
				32-bit generic data	24-bit generic data, data/command mapping, single 2/4-cycles 32-bit transfer	DISP#_TYPE = 00, SYS1/2_MODE = 010 or 100, SYS1/2_ADDR_INC = 11, PFS (DMAADC_6/7) = 111, BPP (DMAADC_6/7) = 000

Table 44-146. Asynchronous Display Access Modes (continued)

Display Type	Source	Command Generation Method	Access Type	IDMAC/MCU Interface Format	Display Interface Format	Programming ADC/IDMAC Parameters
Full addressing, 8/16-bit interface without byte enable	IC (pre-processing)	template	write	32-bit pixel data	8-bit pixel data, data/command mapping, single 1-cycle 8-bit transfer (only for 8-bit interface)	DISP#_TYPE = 00, PRP_ADDR_INC = 00, PFS (DMAADC_0) = 100, BPP (DMAADC_0) = 000
					16-bit pixel data, data/command mapping, single 1/2-cycle 16-bit transfer	DISP#_TYPE = 00, PRP_ADDR_INC = 01, PFS (DMAADC_0) = 100, BPP (DMAADC_0) = 000
					upto 24-bit pixel data, data/command mapping, single 2/4-cycle 32-bit transfer	DISP#_TYPE = 00, PRP_ADDR_INC = 11, PFS (DMAADC_0) = 100, BPP (DMAADC_0) = 000
	IC (post-processing)	template	write	32-bit pixel data	8-bit pixel data, data/command mapping, single 1-cycle 8-bit transfer (only for 8-bit interface)	DISP#_TYPE = 00, PP_ADDR_INC = 00, PFS (DMAADC_1) = 100, BPP (DMAADC_1) = 000
					16-bit pixel data, data/command mapping, single 1/2-cycle 16-bit transfer	DISP#_TYPE = 00, PP_ADDR_INC = 01, PFS (DMAADC_1) = 100, BPP (DMAADC_1) = 000
					upto 24-bit pixel data, data/command mapping, single 2/4-cycle 32-bit transfer	DISP#_TYPE = 00, PP_ADDR_INC = 11, PFS (DMAADC_1) = 100, BPP (DMAADC_1) = 000
	MCU	template	write, read	16-bit generic data	16-bit generic data, data/command mapping, single 1/2-cycle 16-bit transfer	DISP#_TYPE = 00, MCU_DISP#_DATA_WIDTH = 0
				32-bit generic data	24-bit generic data, data/command mapping, single 2/4-cycle 32-bit transfer	DISP#_TYPE = 00, MCU_DISP#_DATA_WIDTH = 1

1 32-bit pixel data format on IDMAC internal interface assumes that only 24 MSB bit are occupied by pixel data and 8 LSB bits are not used

2 32-bit generic data format on IDMAC internal interface of MCU interface assumes that only 24 LSB bit are occupied by data and 8 MSB bits are not used

3 Single transfer means that the ADC sends a whole 16- or 24-bit word to the DI, two transfers mean that the ADC splits a 32-bit word to two 16-bit words

44.4.5.3 Control Sequence Generation Modes

Two modes are exploited for display control sequence generation: command buffer mode and template mode.

For the first mode, the generic data flow from the system memory is interleaved with commands taken from a command buffer. This mode can be used only for the system memory channels 1 and 2. The mode is selected via the ADC_CONF Register.

The command buffer consists of 32-bit words. Each of the words contains 16- or 24-bits display command and a RS parameter. For 16-bit command, there are two byte enable (BE) bits - low and high. The command formats are shown in Table 44-147 and Table 44-148. The command buffer is prepared by the MCU in the system memory as shown in Figure 44-138.

Table 44-147. 16-bits Display Command Format

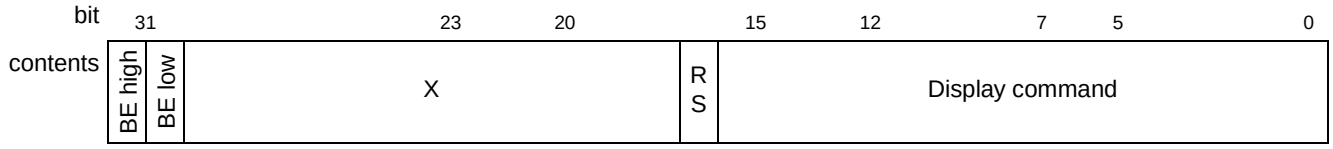
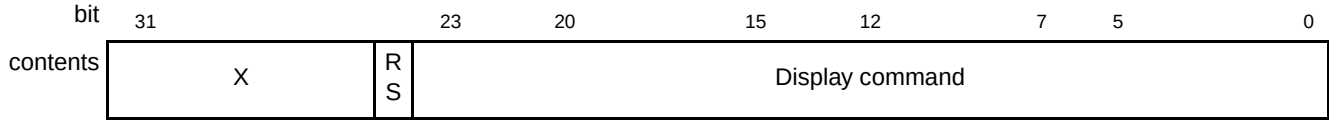


Table 44-148. 24-bits Display Command Format



For the template mode, the command sequence is generated automatically according to command templates prepared by the MCU. There are different templates for read and write operations and for each of displays. The templates are explained in detail in Section 44.4.5.10, “Template Command Generator.”

Selection of command generation mode is done via the ADC_CONF Register.

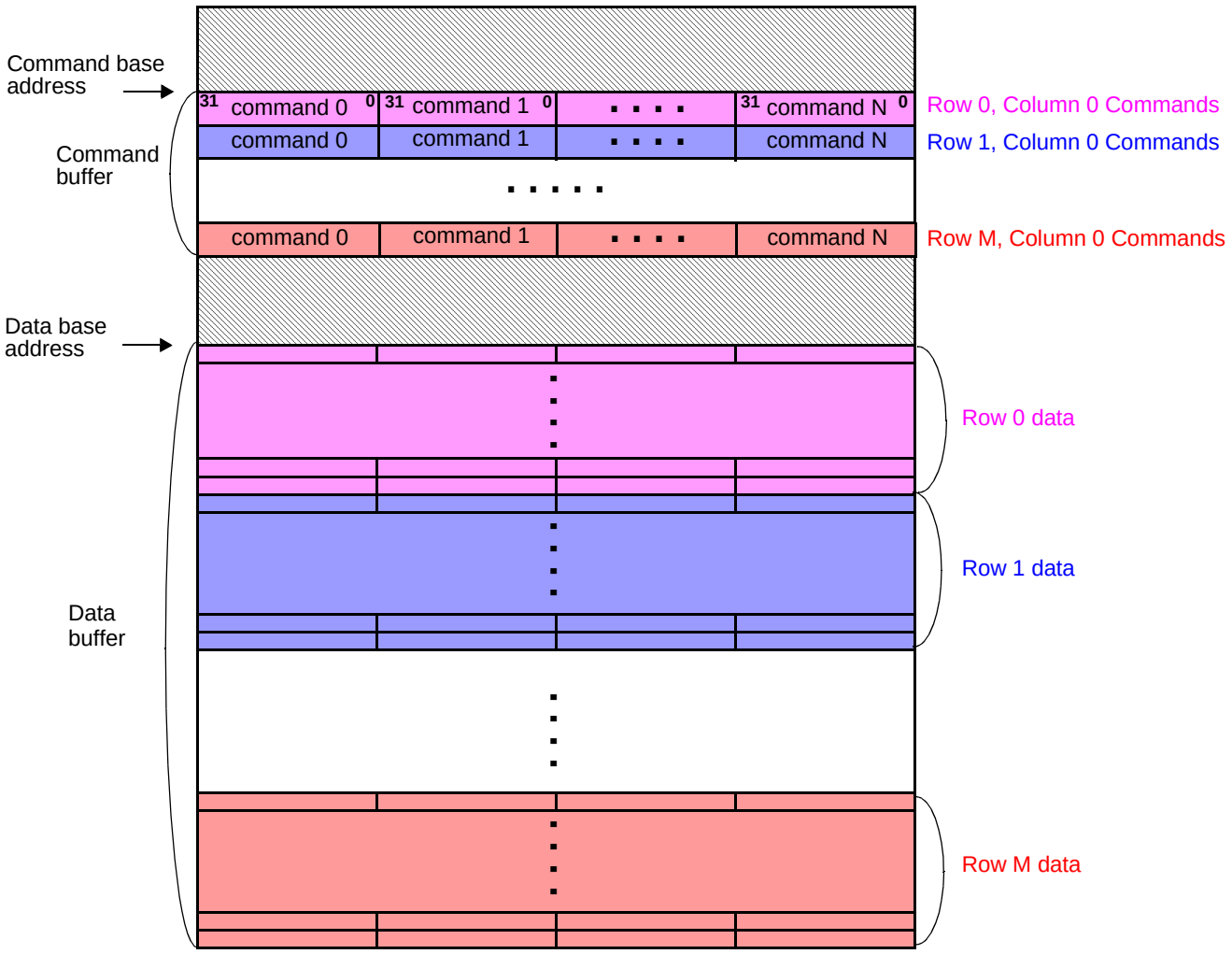


Figure 44-138. Data and Command Buffers in the System Memory

44.4.5.4 Byte Enable Support

The ADC supports byte enable feature used by some display controllers and graphic accelerators. The interface width of such device must be 16 bits. There are the following conditions of byte enable support:

1. If the MCU is an initiator of display access via the slave AHB bus then
   a) Allowed data type is generic data,
   b) AHB single access widths are 8 (16 bit with byte enable), 16 and 32 bits. According to the ARM specification, byte enable options (on the slave AHB bus) are:
   — for 8-bit AHB access - 0001, 0010, 0100, 1000
   — for 16-bit AHB access - 0011, 1100
   — for 32-bit AHB access - 0111, 0110, 1110, 1111

c) AHB burst access width is only 32 bits. Only aligned accesses are allowed (these are ARM limitations for burst access to AHB bus). Byte enable bits are always 1111.

d) Little End Big Endian AHB modes are supported.

2. If the IDMAC is an initiator of display access (system, pre- or postprocessing ADC channels) then

a) Allowed data types are pixels or generic data.

b) Data width may be 8 (16 bit with byte enable), 16 or 32 bits.

c) Generic data is transferred from a two dimensional buffer in the system memory, the buffer row length must be a multiple of 16 bits (for 8- and 16-bit data) or of 32-bits (for 32-bit data) and start from an even address (for 8- and 16-bit data) or an address which is a multiple of 4 (for 32-bit data). This system memory buffer exactly matches the display memory buffer.

d) For non-aligned 8-bit write access (when only 8 bits should be written to the first word in a row of the display memory), an odd display start address has to be set in the corresponding IPU register (ADC_SYSCHA1_SA, ADC_SYSCHA2_SA, ADC_PRPCHAN_SA or ADC_PPCHAN_SA). For aligned 8-bit access (when whole 16 bits should be written to the first word in a row of the display memory), an even display start address has to be set in this register. Read operation is allowed only for 16-bit access (the display start address must be even).

e) Pixel data width in display memory may be 8, 16 or 24 bits.

The ADC translates slave AHB or IDMAC accesses to 16-bit display accesses with the corresponding byte enable bits. The display bus has only Little Endian mode. If sequential writes or reads to/from the display are performed at incremental addresses, the ADC does not send an address to the display (it chains accesses). Otherwise the address is sent before data every time when address incremental order is violated.

The ADC assumes that the display increments the address by 2 after write/read of full 16 word or most significant byte of the word. The address has no change after write/read of least significant byte of the word.

Writing addresses to the display memory and registers are performed by 16-bit access with byte enable bits ‘11’. For command buffer mode, byte enable bits can be used only with 16-bit command. They are located in the 32-bit command word (see Table 44-147).

44.4.5.5 Windows Displayed on a Smart Display

Figure 44-139 illustrates how windows are displayed on an asynchronous display. Each of windows corresponding to the system memory, pre- and post processing access channels have the start address defined in the ADC_SYSCHA1_SA, ADC_SYSCHA2_SA, ADC_PRPCHAN_SA, ADC_PPCHAN_SA Registers. The start address format matches to the display addressing type. The same registers control a display number for the specific window. The start address refers to the VSYNC start point. Sizes of the windows are set by programming the corresponding IDMAC channels (see Table 44-29 through Table 44-33). A whole screen size (SCREEN#_WIDTH and SCREEN#_HEIGHT) are set via the corresponding registers. Sizes of the windows are set by programming the corresponding IDMAC channels (see Table 44-29 and Table 44-33).

The total number of windows which can be displayed on all the displays is five. The maximum number of windows which can be displayed on the same display is up to five, when the template mode of operation

is used. The number of windows on the specific display is restricted to one when command buffer mode of operation is used.

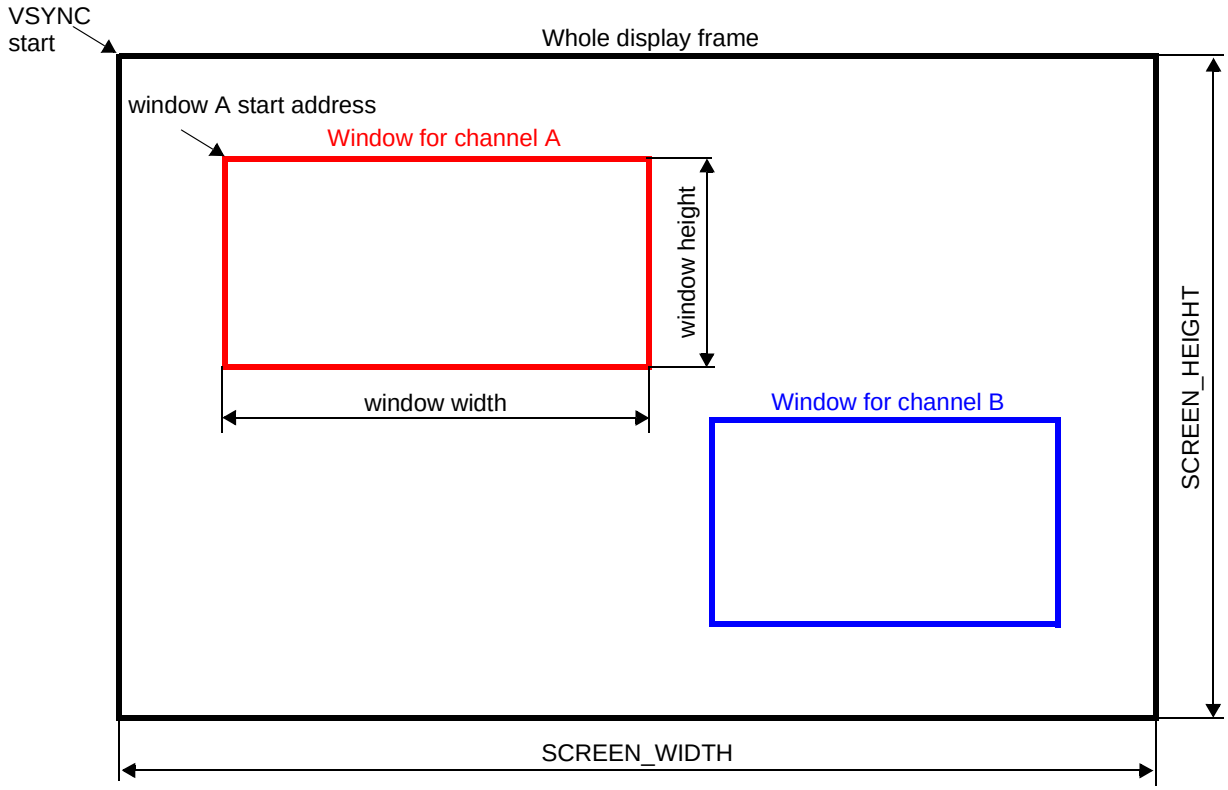


Figure 44-139. Windows on a Smart Display

44.4.5.6 AHB Slave Interface

The AHB Slave Interface is responsible for data transfer from/to the MCU when the MCU accesses a smart display directly. The Interface supports 8 (with byte enable), 16- or 32-bits single accesses or 32-bit burst accesses. For burst access, only aligned addressing is allowed. All Endianness modes (LE, BE32) are supported. The AHB clock rate may be equal or lower than the HSP_CLK clock rate.

44.4.5.7 Asynchronous Display Buffer Memory

Data is written to or read from the Asynchronous Display Buffer Memory. Mapping of the Asynchronous Display Buffer Memory is shown in Table 44-179. The Memory includes 7 FIFOs: two system memory data FIFOs, two command FIFOs, one preprocessing data FIFO, one postprocessing data FIFO and one MCU write data FIFO. Each FIFO has eight pages of eight 64-bit words.

Two data transfer channels associated with the transfer from/to the system memory are bidirectional and can be used both for read and write operations according to the ADC_CONF Register settings. The MCU FIFO is used only for write operations. MCU reading is enabled after finish of previously requested MCU write operations (releasing the MCU FIFO).

44.4.5.8 Access Control

44.4.5.8.1 Access Priorities

The Access Control arbitrates data sources/destinations channels. According to an access address and a source type, the display number and access attributes are selected. The priorities order is as follows:

1. Preprocessing and MCU access channels (the highest priority). Selection between these channels is done according to the round-robin algorithm.
2. Postprocessing channel.
3. System memory channel 1.
4. System memory channel 2 (the lowest priority).

44.4.5.8.2 Tearing Elimination

The ADC has capability to avoid image tearing. If the source is the system memory or postprocessing, the Access Control monitors position of a display refresh pointer. Writing to the display is started only after crossing the window start point by the display refresh pointer. After that, writing to the display is allowed only when a write pointer does not advance beyond the refresh pointer. To provide anti-tearing mode, a window start time (in rows) must be defined in the ADC_SYSCHA1_SA, ADC_SYSCHA2_SA, ADC_PPCHAN_SA Registers for the corresponding channels.

Anti-tearing mode requires that the IPU controls display refresh timing by generation of the VSYNC signal and a display clock (for dual-port displays). There is a separate VSYNC generator for display 0. For displays 1 and 2, the VSYNC generator is shared. Therefore only one of the displays 1 or 2 can be in vertical synchronization mode according to the DISP12_VSYNC_SEL bit in the ADC_DISP_VSYNC Register.

In the case when tearing cannot be avoided (when the refresh rate is too high and the refresh pointer overtakes the write pointer after full refresh cycle), an error interrupt is generated. Tearing elimination mode can be disabled via the ADC_CONF Register.

When the data arrives directly from preprocessing, tearing can be avoided only for the 2:1 ratio between a sensor scan rate and a display refresh rate. The additional conditions for tearing elimination in this case are to synchronize the sensor and display VSYNCs (by programming the appropriate bits in the ADC_DISP_VSYNC Registers) and full-screen image from the preprocessing channel.

44.4.5.8.3 Snooping

This function allows to reduce a rate of write accesses from the system memory to a smart display. When snooping is enabled by configuration of the ADC2_SRC_SEL and ADC3_SRC_SEL fields in the IPU_FS_DISP_FLOW Register, writing to the display is performed only if the corresponding system memory buffers has been changed. Any change of data in the system memory buffers is registered by the External Memory Controller of the chip. When writing to the system memory is performed, the External Memory Controller asserts the SYSCH_SNOOP pin of the IPU. After receiving these signals, the CM initiates transfer of the corresponding buffer to a smart display window.

44.4.5.8.4 Automatic Window Refresh

By programming the IPU_FS_DISP_FLOW Register, MCU software can enable automatic refresh of a window on the smart display. The refresh period is defined via the AUTO_REF_PER field (the time unit is 2^{17} periods of the HSP_CLK clock). The actual value of refresh period is equal to T_{HSP}*217 *(AUTO_REF_PER+1). Auto-refresh can be conditional. In this case, it is accomplished only if the SYSCH_SNOOP pin is asserted.

44.4.5.8.5 Sequential Addressing Access

The Access Controller includes a mechanism allowing to skip transfer of addresses and commands before a data when access addresses are sequential. This mechanism works only in template mode.

First of all, addresses and commands are skipped within a burst sent or received by the IDMAC or by the MCU. Each burst occupies one page in the corresponding FIFO of the Buffer Memory. At start of the burst that should be sent/received to/from a display, the Access Controller compares a new burst start address and channel number with a last address and channel number for previous access to this display. If the channel number has not changed and the new start address follows the previous last address, the Access Controller does not send the new address and commands to the display. Otherwise both the address and the commands are sent.

44.4.5.8.6 MCU Direct Access to Display/Graphic Accelerator Memory

The MCU can directly access display or graphic accelerator memories of all three smart displays through the slave AHB bus. The display memories are mapped to the MCU memory space. The display memory start address in the MCU memory space is as follows:

DISPLAY_MEM_START_ADDRESS = SLAVE_AHB_BASE_ADDRESS + DISPLAY_MEM_START_ADDRESS _OFFSET

where

SLAVE_AHB_BASE_ADDRESS is a 32-bit slave AHB base address defined as {AHB_S_BA[6:0], 25'b0} (binary quotation). The AHB_S_BA[6:0] value is programmed by VIA.

DISPLAY_MEM_START_ADDRESS_OFFSET is 25-bit offset which is a constant for each display. It is 0 for the display 0, 0x0800000 for the display 1, 0x1000000 for the display 2. so each display memory can occupy up to 0x0800000 bytes.

The ADC translates the slave AHB address to the display address. For displays with full addressing (like ATI W2300) the address transferred to the displays is

DISPLAY_ADDRESS = SLAVE_AHB_ADDRESS - DISPLAY_MEM_START_ADDRESS

When sending to the display, DISPLAY_ADDRESS can be split to two or more portions by the template WR_XADDR and WR_YADDR commands. For displays with XY- addressing (like Epson L2F50032T00), the IPU firstly calculates the pixel number in the frame:

```
PIXEL_NUMBER = (SLAVE_AHB_ADDRESS - DISPLAY_MEM_START_ADDRESS) / BYTES_PER_PIXEL,
```

where

BYTES_PER_PIXEL=2 if the MCU_DISP#_DATA_WIDTH parameter in the ADC_DISP#_CONF Register is 0,

BYTES_PER_PIXEL=4 if the MCU_DISP#_DATA_WIDTH parameter in the ADC_DISP#_CONF Register is 1,

After that, the IPU calculates X and Y addresses using the template WR_XADDR and WR_YADDR commands. The WR_XADDR command defines what are low significant bits of PIXEL_NUMBER to be sent as the X address. The WR_YADDR command defines what are most significant bits of PIXEL_NUMBER to be sent as the Y address. The templates should be downloaded to the internal memory before enabling the MCU channel. The display should be configured via the low level access mechanism using the DI_DISP_LLA_CONF and DI_DISP_LLA_DATA Registers. This mechanism allows to access both the display memory and control registers.

44.4.5.9 Data and Command Combiner

The Access Control manages the Data and Command Combiner according to the access method. The Data and Command Combiner contains muxes and aligners. Its output may be 16/24 bits data or 16/24 bits address or 16/24 bits command.

In command buffer mode, it toggles between command and data rows on the end-of-line signal. In template mode, it invokes the Template Command Generator to produce a command sequence and to interleave the commands with the data.

In template mode, commands are taken from the Template Command Generator.

44.4.5.10 Template Command Generator

44.4.5.10.1 Template Functions and Structure

A template is a microprogram executed every time when a data burst should be written to or read from a smart display. A typical structure of the template include the following parts:

1. Sending a pre-commands sequence to the display.
2. Sending an address to the display.
3. Waiting for display acknowledge or for given number of display clocks. Waiting is needed for read access only.
4. Sending a data sequence to the display or receiving a data sequence from the display. This step is repeated while addressing is sequential (see Section 44.4.5.8.5, “Sequential Addressing Access”).
5. Sending a post-commands sequence to the display.

Depending on access type (read or write) and access sequentially the template structure can be varied. For the sequential access (see Section 44.4.5.8.5, “Sequential Addressing Access”) neither commands and not address are sent to the display. Therefore, for the sequential access the templates are not used. Two templates (write and read) exist for each of three displays. The maximal template length is 32 commands.

The Template Command Generator receives a template command sequence from the Template Memory (Table 44-149). Each command consists of the following fields:

1. Display register address RS (1 bit) selects the index or data register.
2. Template flow control (2 bits) modifies order of template commands (step-by-step, pause, stop). Flow control commands are shown in Table 44-149.

Table 44-149. Template Flow Control

Code	Flow control command	Description
00	Step-by-step	Sequential template execution
01	Pause	Flag for breaking and resuming template execution. Used for displays required post-commands. The template part preceding this flag (pre-command and address) is performed before write/read a data sequence. The template part after this flag (post-command) is performed after the data sequence. Has to be located in the command preceding the post-command part of the current template.
10	Stop	Template last command. Has to be located on the last command
11	-	Reserved

3. Opcode (3 bits) controls an action according to Table 44-150.

Table 44-150. Command Opcode

Opcode	Command	Description
000	RD_DATA	Read data from the display
001	RD_ACK	Wait for display acknowledge
010	RD_WAIT	Wait for given number of display clock cycles
011	WR_XADDR	Send X address or LSB of full address to the display
100	WR_YADDR	Send Y address or MSB of full address to the display
101	WR_ADDR	Send whole address to the display
110	WR_CMND	Send command field to the display
111	WR_DATA	Send data from the Buffer Memory to the display

4. Command/data (24 bits) to be sent to the display.

The templates have to be loaded to the Template Memory before enabling any of the ADC channels used templates.

44.4.5.10.2 Format of Template Commands

RD_DATA

Description: Read data from the display. This command is repeated while there is read data with sequential addresses. The RS value is hold all this time.

Table 44-151. RD_DATA Command Format

	29	28 27	26 25 24	23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Field	RS	Flow control	Opcode	Data
Contents	RS	FC	000	not used

RD_ACK

Description: Wait for display acknowledge. The ADC receives data from the display, performs bitwise AND with the mask and compares the result with the acknowledge pattern. Acknowledge patterns and masks are defined in the ADC_DISP0_RD_AP, ADC_DISP0_RDM, ADC_DISP1_RD_AP, ADC_DISP1_RDM, ADC_DISP2_RD_AP, ADC_DISP2_RDM Registers,

Table 44-152. RD_ACK Command Format

	29	28 27	26 25 24	23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	
Field	RS	Flow control	Opcode	Data	
Contents	RS	FC	001	not used	TimeOut

TimeOut The maximal number minus 1 of display read access cycles to wait before beginning read. If TimeOut access cycles occur, data read is started independently on display acknowledge.

RD_WAIT

Wait for given number of display interface clock cycles.

Table 44-153. RD_WAIT Command Format

	29	28 27	26 25 24	23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	
Field	RS	Flow control	Opcode	Data	
Contents	RS	FC	010	not used	Wait Tme

WaitTime The number minus 1 of display interface clock cycles to wait.

WR_XADDR

Send the X address or LSB of full address to the display. This command is repeated while there is write data with sequential addresses. The RS value is hold all this time.

Table 44-154. WR_XADDR Command Format

	29	28 27	26 25 24	23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	
Field	RS	Flow control	Opcode	Data	
Contents	RS	FC	011	Mask	XCodedWidth

XCodedWidth Coded width of LSB address bits to be send to the display:

Table 44-155. XCodedWidth Values

Value	Meaning
0000	send address bits [6:0]
0001	send address bits [7:0]
0010	send address bits [8:0]
0011	send address bits [9:0]
0100	send address bits [10:0]
0101	send address bits [11:0]
0110	send address bits [12:0]
0111	send address bits [13:0]
1000	send address bits [14:0]
1001	send address bits [15:0]
1010	send address bits [16:0]
1011	send address bits [17:0]
1100	send address bits [18:0]
1101	send address bits [19:0]
1110	send address bits [20:0]
1111	send address bits [21:0]

Mask MSB bits of the mask word which is ORed with the address before sending to the display.

The resulting address word to be sent to the display is calculated by the following operations:

1. Select LSB address bits according to the XCodedWidth value.
2. Complement the derived word to 24 bits by adding zero MSB bits.
3. Complement the *Mask* value to 24 bits by adding 4 zero LSB bits
4. Perform OR operation between results of the steps 2 and 3.

The mask can be used for mixing command and address in a single 24-bit word.

WR_YADDR

Description: Send Y address or MSB of full address to the display.

Table 44-156. WR_YADDR Command Format

	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	RS	Flow control		Opcode			Data																							
Contents	RS	FC		100			Mask																						YCodedWidth	

YCodedWidth Coded width of MSB address bits to be send to the display:

Table 44-157. YCodedWidth Values

Value	Meaning
0000	send address bits [22:7]
0001	send address bits [22:8]
0010	send address bits [22:9]
0011	send address bits [22:10]
0100	send address bits [22:11]
0101	send address bits [22:12]
0110	send address bits [22:13]
0111	send address bits [22:14]
1000	send address bits [22:15]
1001	send address bits [22:16]
1010	send address bits [22:17]
1011	send address bits [22:18]
1100	send address bits [22:19]
1101	send address bits [22:20]
1110	send address bits [22:21]
1111	send address bits [22:22]

Mask MSB bits of the mask word which is ORed with the address before sending to the display.

The resulting address word to be sent to the display is calculated by the following operations:

1. Select MSB address bits according to the *YCodedWidth* value and align them to the right (LSB) side.
2. Complement the derived word to 24 bits by adding zero MSB bits.
3. Complement the *Mask* value to 24 bits by adding 4 zero LSB bits
4. Perform OR operation between results of the steps 2 and 3.

The mask can be used for mixing command and address in a single 24-bit word.

WR_ADDR

Description: Send the whole address to the display.

Table 44-158. WR_ADDR Command Format

	29	28 27	26 25 24	23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Field	RS	Flow control	Opcode	Data
Contents	RS	FC	101	Mask (not used)

Mask MSB bits of the mask word which is ORed with the address before sending to the display.

The resulting address word to be sent to the display is calculated by the following operations:

1. Complement the address to 24 bits by adding one zero MSB bit.
2. Complement the *Mask* value to 24 bits by adding 4 zero LSB bits
3. Perform OR operation between results of the steps 2 and 3.

The mask can be used for mixing command and address in a single 24-bit word.

WR_CMND

Description: Send the command field to the display.

Table 44-159. WR_CMND Command Format A

	29	28 27	26 25 24	23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Field	RS	Flow control	Opcode	Data
Contents	RS	FC	110	Command

Table 44-160. WR_CMND Command Format B

	29	28 27	26 25 24	23	22	21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Field	RS	Flow control	Opcode	Data		
Contents	RS	FC	110	BE H	BE L	Command

The format B should be used for displays supporting the byte enable feature. *BEL* and *BEH* are the command byte enable bits.

WR_DATA

Description: Send data from the Buffer Memory to the display.

Table 44-161. WR_DATA Command Format

	29	28 27	26 25 24	23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Field	RS	Flow control	Opcode	Data
Contents	RS	FC	111	not used

44.4.5.11 Asynchronous Display Adapter

The Asynchronous Display Adapter contains a set of counters for generation of the VSYNC signal for the display 0 and the display 1 or 2. The counters may be synchronized to the sensor VSYNC signal with a programmable delay or to the display VSYNC signal if display operates in self-refresh mode. Vertical synchronization is controlled by the ADC_DISP_VSYNC Register. The Asynchronous Display Adapter provides an interface to the DI.

44.4.6 Display Interface (DI)

44.4.6.1 Block Diagram

The DI Block Diagram is shown in Figure 44-140.

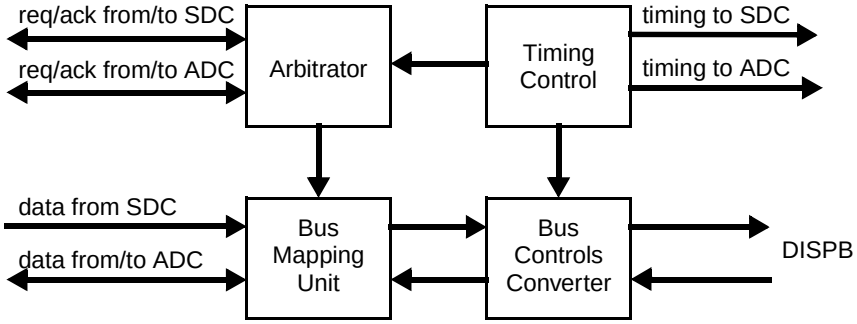


Figure 44-140. DI Block Diagram

The DI provides arbitrates access to up to four displays with time multiplexing. It converts a data from the SDC, the ADC or the MCU via the IP Skyblue Bus (low level access) to a format suitable for the specific display interface. The DI generates display clocks and other display control signals with programmable timings. The DI outputs data to or inputs from parallel and/or serial interfaces.

The DI is controlled via the peripheral bus registers. All the registers are not double buffered. Therefore they should be programmed before use of the corresponding display.

44.4.6.2 Supported Display Interface Types

The DI supports a simultaneously synchronous interface to the display 3 and asynchronous interfaces to the displays 0, 1, 2. The display 0 interface is only parallel, the displays 1, 2 interfaces can be parallel or serial. Common configuration of the interfaces is done via the DI_DISP_IF_CONF Register. All display interfaces have programmable timing.

44.4.6.2.1 Synchronous Interfaces

The DI supports the following synchronous display interfaces:

1. Synchronous generic interfaces to TFT dumb displays or RGB interfaces of smart displays
2. Synchronous interfaces to Sharp displays

3. Synchronous interfaces to TV encoders:
   a) PAL
   b) NTSC

   TV interfaces can operate in progressive or interlaced modes.

44.4.6.2.2 Asynchronous Parallel Interfaces

The DI supports the following asynchronous parallel interfaces:

1. System 80 interface
   a) Type 1 (sampling with the chip select signal) with and without byte enable signals.
   b) Type 2 (sampling with the read and write signals) with and without byte enable signals.
2. System 68k interface
   a) Type 1 (sampling with the chip select signal) with or without byte enable signals.
   b) Type 2 (sampling with the read and write signals) with or without byte enable signals.

For each of four system interfaces there are three burst modes:

1. Burst mode without a separate clock. The burst length is defined by the corresponding parameters of the IDMAC (when data is transferred from the system memory) of by the HBURST signal (when the MCU directly accesses the display via the slave AHB bus). For system 80 and system 68k type 1 interfaces, data is sampled by the CS signal and other control signals changes only when transfer direction is changed during the burst. For type 2 interfaces, data is sampled by the WR/RD signals (system 80) or by the ENABLE signal (system 68k) and the CS signal stays active during the whole burst.
2. Burst mode with the separate clock DISPB_BCLK. In this mode, data is sampled with the DISPB_BCLK clock. The CS signal stays active during whole burst transfer. Other controls are changed simultaneously with data when the bus state (read, write or wait) is altered. The CS signals and other controls move to non-active state after burst has been completed.
3. Single access mode. In this mode, slave AHB and DMA burst are broken to single accesses. The data is sampled with CS or other controls according the interface type as described above. All controls (including CS) become non-active for one display interface clock after each access. This mode corresponds to the ATI single access mode.

Both system 80 and system 68k interfaces are supported for all described modes.

Additionally, the DI allows a programmable pause between two burst. The pause is defined in the HSP_CLK cycles. It allows to avoid timing violation between two sequential bursts or two accesses to different displays. The range of this pause is from 4 to 19 HSP_CLK cycles.

44.4.6.2.3 Asynchronous Serial Interfaces

The DI supports the following asynchronous serial interfaces:

1. 3-wire (with bidirectional data line).
2. 4-wire (with separate data input and output lines).
3. 5-wire type 1 (with sampling RS by the serial clock).

4. 5-wire type 2 (with sampling RS by the chip select signal).

For serial interfaces, data and preamble lengths and other parameters are controlled via the DI_SER_DISP1_CONF and DI_SER_DISP2_CONF Registers.

44.4.6.3 Display Access Priorities

The display 0 is the primary display. It corresponds to the asynchronous (smart) type with parallel interface. It can be used as the asynchronous part of a dual-port smart display which has both synchronous and asynchronous interfaces. The displays 1 and 2 are secondary displays. Both of them can be accessed via parallel or serial interfaces. Assignment of the display to one of the interface types via the DI_DISP_IF_CONF Register. The display 3 is related to the synchronous interface. It can be a dumb display or TV or synchronous part of a dual-port smart display.

Table 44-162 describes the display interfaces and access priorities supported by the DI.

Table 44-162. Display Interfaces and Access Priorities

Access Initiator	Priority	Access Type	Information Sent To Display	Data/Command Packing/ Unpacking	Display Number	Interface Type	Comments
SDC	high	write	data	yes	3	parallel	
MCU via IP Skyblue Bus (low level access)	medium	write and read	data and commands	yes	0, 1, 2	parallel	
				no	1, 2	serial	access can be performed in parallel with access to parallel interface by other sources
ADC	low	write and read	data and commands	yes	0, 1, 2	parallel	
					1, 2	serial	

According to the priorities, the Arbitrator decides which of initiators request display access will be granted. The Arbitrator takes into account two types of interface resources:

1. Parallel or serial interface required data packing/unpacking (See Table 44-162.)
2. Serial interface without data packing/unpacking (This resource is used only by MCU low level access.)

When the SDC or the ADC communicates with one of four displays via the parallel interface, the MCU is allowed to access another display via the serial interface simultaneously.

After grant reception, the initiator asserts a lock signal until access completion. The Arbitrator postpones the next arbitration waiting for negation of the lock signal. Arbitration is postponed for all displays excluding the MCU access case described above. When the lock signal is negated, arbitration is repeated.

The SDC sends access request just (one row) before start of the active synchronous display frame. It asserts lock signal until end of the active frame. Thus, other initiators can access a display during vertical blanking intervals of the synchronous display or when the SDC skips frames (for dual-port displays, see Section 44.4.4, “Synchronous Display Controller (SDC)”).

44.4.6.4 Bus Mapping Unit

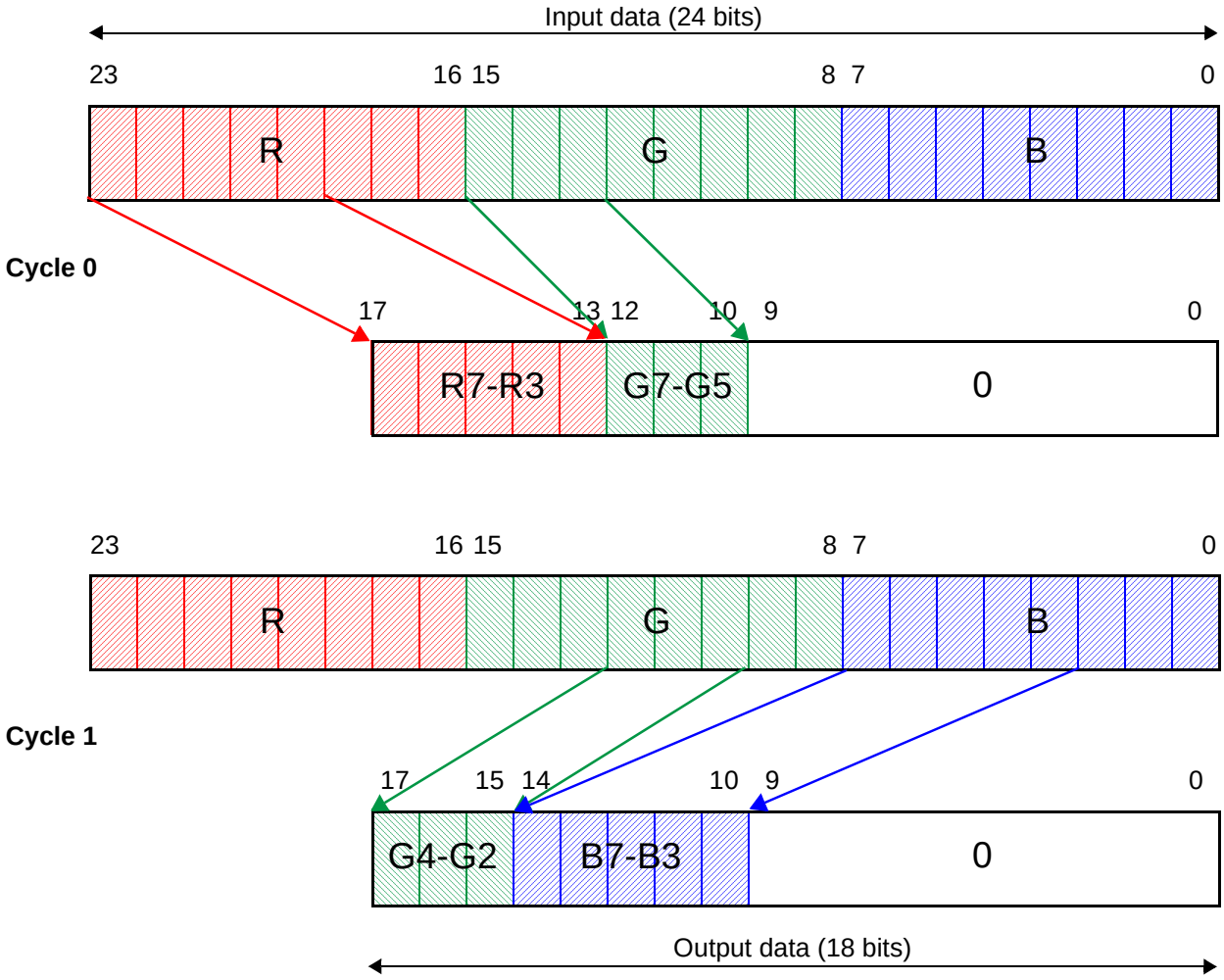
The Bus Mapping Unit is responsible for programmable mapping of the input data and commands to the display interface format and vice versa. The internal DI format for data and commands is a 24-bits word divided into three byte components (eight zeroes are added to MSB for 16-bits words from the ADC). This word can be output or input in one, two, three or four cycles of the display clock.

There are 21 Registers for programming bus mapping: DI_DISP0_DB0_MAP - DI_DISP2_DB2_MAP, DI_DISP0_CB0_MAP - DI_DISP2_CB2_MAP, DI_DISP3_B0_MAP - DI_DISP3_B2_MAP. The registers have identical format. Each of the registers specifies an output/input rule for a certain display (‘DISP0’ - ‘DISP3’ indexes in the Register name) and a certain byte component (‘B0’ - ‘B2’ indexes in the Register name). For all displays excluding the display 3 with synchronous interface, there are two different rules: one for a data word and the second for the command word. The corresponding Registers are marked by the ‘_D’ and ‘_C’ suffixes.

The mapping rule written in each of the Registers defines two types of parameters for the specific byte component and display:

1. Offsets of the byte component MSB relative to the output word LSB. Because the offsets can change dynamically, they are defined separately for the display clock cycles zero, one and two.
2. Numbers of the display clock cycles at which every bit of the byte component should be valid on the display bus. There are eight such 2-bit numbers in the Register.

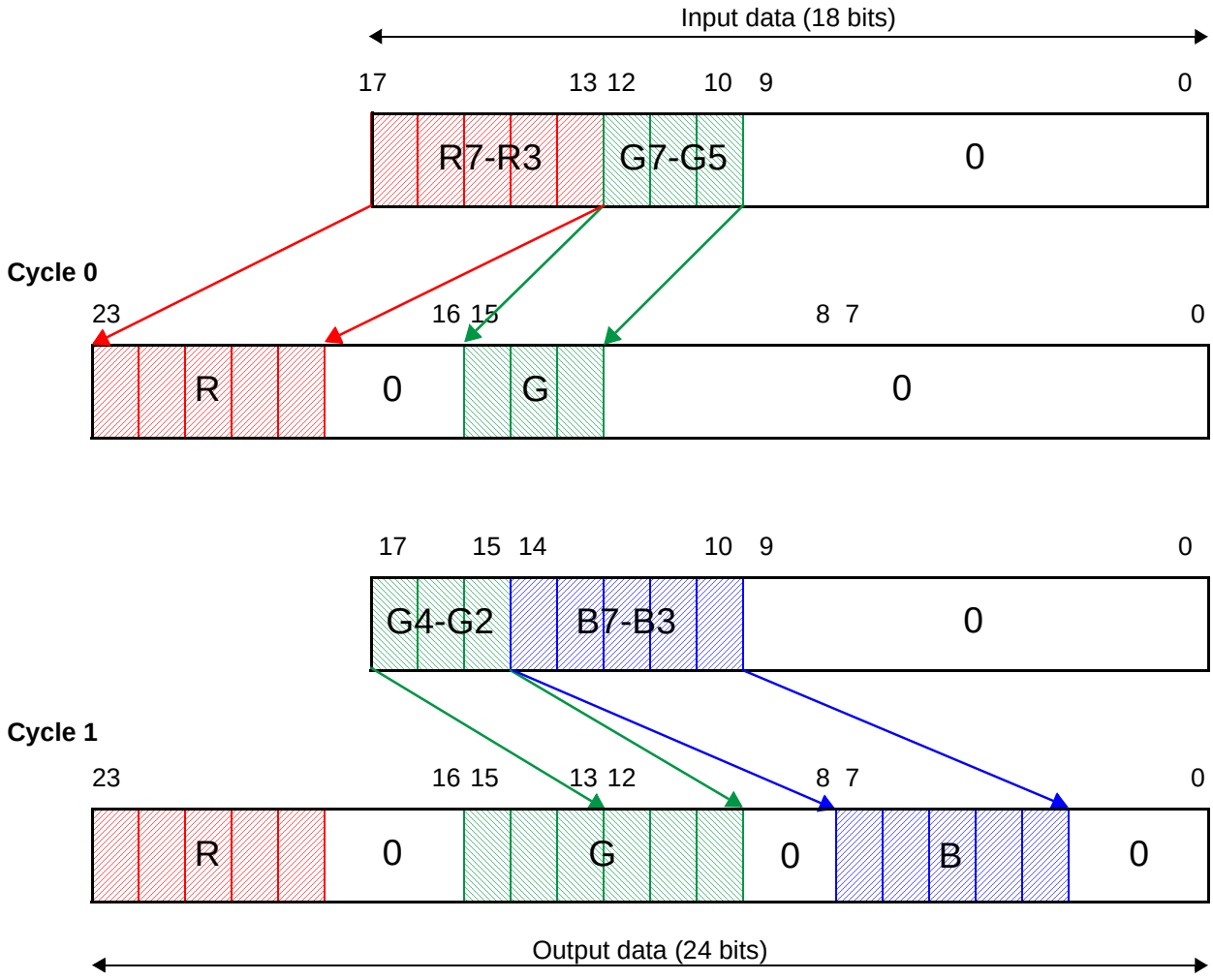
Figure 44-141 presents an example of programming data packing for one of displays.



Byte 2 (red)	OFFS0=17	OFFS1=0	OFFS2=0	M0=11	M1=11	M2=11	M3=00	M4=00	M5=00	M6=00	M7=00
Byte 1 (green)	OFFS0=12	OFFS1=20	OFFS2=0	M0=11	M1=11	M2=01	M3=01	M4=01	M5=00	M6=00	M7=00
Byte 0 (blue)	OFFS0=0	OFFS1=14	OFFS2=0	M0=11	M1=11	M2=11	M3=01	M4=01	M5=01	M6=01	M7=01

Figure 44-141. Example of Data Packing for Writing Data to the Display

Figure 44-142 presents an example of programming data packing for one of displays.



Byte 2 (red)	OFFS0=17	OFFS1=0	OFFS2=0	M0=11	M1=11	M2=11	M3=00	M4=00	M5=00	M6=00	M7=00
Byte 1 (green)	OFFS0=12	OFFS1=20	OFFS2=0	M0=11	M1=11	M2=01	M3=01	M4=01	M5=00	M6=00	M7=00
Byte 0 (blue)	OFFS0=0	OFFS1=14	OFFS2=0	M0=11	M1=11	M2=11	M3=01	M4=01	M5=01	M6=01	M7=01

Figure 44-142. Example of Data Unpacking for Reading Data from the Display

The same packing/unpacking registers are used for parallel and serial interface.

44.4.6.5 Timing Control

The Timing Control provides clocking and other timing signals for each of displays. The following timing parameters are programmable for each display via the DI_HSP_CLK_PER, DI_DISP0_TIME_CONF_1 - DI_DISP2_TIME_CONF_3, DI_DISP3_TIME_CONF Registers:

1. Period of the HSP_CLK clock. It is required for holding unchanged display timing when the HSP_CLK rate is changed on-the-fly (see Section 44.1.1.4.2, “Changing Clock Rates and Disabling Clocks”). The period has an integer part (3 bits) and a fractional part (4 bits).
2. Period of the display interface clock (DISP#_IF_CLK) for display write access. It has an integer part (8 bits) and a fractional part (4 bits). The fractional part allows to define an average period of the clock with high resolution. An current value of the period has the resolution of one HSP_CLK period.
3. Positions of the display interface clock positive and negative edges relative to data timing for display write access. The positions include an integer part (9 bits) and a fractional part (1 bits). The resolution of the positions is a half of the HSP_CLK period.
4. Period of the display interface clock (DISP#_IF_CLK) for display read access (excluding the display 3). It has an integer part (8 bits) and a fractional part (4 bits). The fractional part allows to define an average period of the clock with high resolution. An current value of the period has the resolution of one HSP_CLK period.
5. Positions of the display interface clock positive and negative edges relative to data/address timing for display read access (excluding the display 3). The positions include an integer part (9 bits) and a fractional part (1 bits). The resolution of the positions is a half of the HSP_CLK period.
6. Period of the display pixel clock (DISP#_PIX_CLK) for display read access. For the display 3, this parameter is derived from the display interface clock period and the number of clocks required to output one pixel (see the DI_DISP_ACC_CC Register). It has an integer part (8 bits) and a fractional part (4 bits). The fractional part allows to define an average period of the clock with high resolution. An current value of the period has the resolution of one HSP_CLK period.

   The display clock period is required for generation of the VSYNC and HSYNC signals in the SDC and the ADC and fulfilling the tearing elimination function in the ADC.
7. Position of data sampling point relative to address timing (excluding the display 3) for display read access. This timing includes two parts: a position within the display interface clock period and a number of display interface clock cycles before reading (the number of wait states). The position within the display interface clock period includes an integer part (9 bits) and a fractional part (1 bits). The resolution of the current position value is one HSP_CLK period.

Clock rates and phases relative to data are programmable. The Timing Control allows on-the-fly change of the reference clock frequency as described in Section 44.1.1.4.2, “Changing Clock Rates and Disabling Clocks.”

44.4.6.6 Bus Controls Converter

This unit provides control over interface signals polarity according to the settings of the DI_DISP_SIG_POL Register. All display bus outputs have identical delays relative to the HSP_CLK clock positive or negative edge (depending on which edge the signal is generated).

44.4.7 Image DMA Controller (IDMAC)

44.4.7.1 Block Diagram

The IDMAC Block Diagram is shown in Figure 44-143.

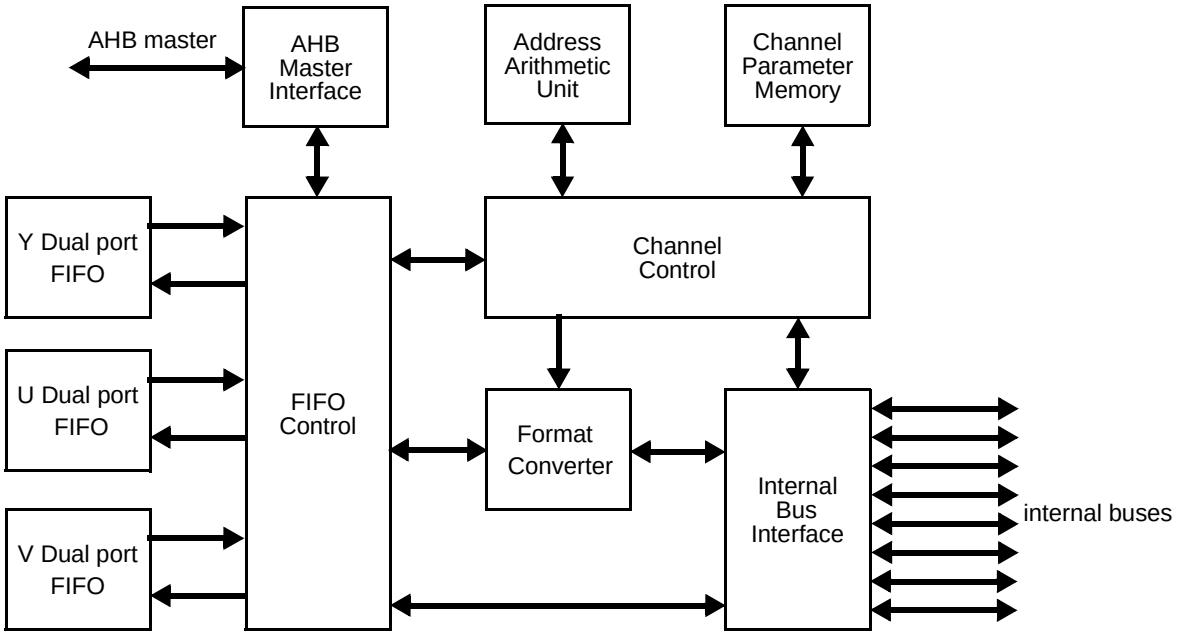


Figure 44-143. IDMAC Block Diagram

The IDMAC consists of three dual port FIFOs, the FIFO Control, the Format Converter, the Address Arithmetic Unit, the Channel Parameter Memory, the Channel Control, the Internal Bus Interface, the AHB Master Interface.

The IDMAC is controlled via the peripheral bus registers and the Channel Parameter Memory.

44.4.7.2 DMA channels

There are 32 DMA channels. Table 44-163 contains description of all the channels.

Table 44-163. IDMAC Channels

Number	Name	Source	Destination	Processing Flow Purpose
0	DMAIC_0	IC	Memory	Preprocessing data from IC (encoding task) to memory
1.A[1]	DMAIC_1	IC	Memory	Preprocessing data from IC (viewfinder task) to memory
1.B	DMAADC_0	IC	ADC	Preprocessing data from IC (viewfinder task) to smart display
2.A[2]	DMAIC_2	IC	Memory	Postprocessing data from IC to memory
2.B	DMAADC_1	IC	ADC	Postprocessing data from IC to smart display
3	DMAIC_3	Memory	IC	Graphics data for combining (viewfinder task)

Table 44-163. IDMAC Channels (continued)

Number	Name	Source	Destination	Processing Flow Purpose
4	DMAIC_4	Memory	IC	Graphics data for combining (post-processing task)
5	DMAIC_5	Memory	IC	Postprocessing data from memory
6	DMAIC_6	Memory	IC	Preprocessing data from sensor stored in memory (for example Bayer)
7	DMAIC_7	IC	Memory	Direct data from IC (sensor data) to memory
8	DMAIC_8	IC	Memory	Preprocessing data after rotation (encoding task)
9	DMAIC_9	IC	Memory	Preprocessing data after rotation (viewfinder task)
10	DMAIC_10	Memory	IC	Preprocessing data for rotation (encoding task)
11	DMAIC_11	Memory	IC	Preprocessing data for rotation (viewfinder task)
12	DMAIC_12	IC	Memory	Postprocessing data after rotation
13	DMAIC_13	Memory	IC	Postprocessing data for rotation
14	DMASDC_0	Memory	SDC	Background data (full refresh)
15	DMASDC_1	Memory	SDC	Foreground data
16	DMASDC_2	Memory	SDC	Mask data
17	DMASDC_3	Memory	SDC	Background data (partial refresh)
18	DMAADC_2	Memory	ADC	System channel 1 write data
19	DMAADC_3	Memory	ADC	System channel 2 write data
20	DMAADC_4	Memory	ADC	Commands stream for system channel 1
21	DMAADC_5	Memory	ADC	Commands stream for system channel 2
22	DMAADC_6	ADC	Memory	System channel 1 read data
23	DMAADC_7	ADC	Memory	System channel 2 read data
24	DMAPF_0	Memory	PF	PF parameters (quantization parameters for MPEG-4 and H.264 and filter offsets for H.264)
25	DMAPF_1	Memory	PF	PF parameters (boundary strength for H.264)
26	DMAPF_2	Memory	PF	Y input data
27	DMAPF_3	Memory	PF	U input data
28	DMAPF_4	Memory	PF	V input data
29	DMAPF_5	PF	Memory	Y output data
30	DMAPF_6	PF	Memory	U output data
31	DMAPF_7	PF	Memory	V output data

1 Channels 1.A and 1.B, 2.A and 2.B do not work simultaneously. Therefore they use the same channel parameters.

2 Channels 2.A and 2.B, 2.A and 2.B do not work simultaneously. Therefore they use the same channel parameters.

The channel parameters are stored in the Channel Parameter Memory. For each channel, two 132-bits parameter words are used. The parameters are explained in Table 44-29 through Table 44-33. There are two types of the parameters: constant and variable. Most of the constant parameters should be written by the MCU before enabling the channel. Only the base address for non-active data buffer allowed to be changed during channel operation. The variable parameters are written by the IDMAC during channel operation. The MCU is not allowed to change the variable parameters in this time interval but it should clear them before channel enabling. The MCU can access the Channel Parameter Memory via the IPU_IMA_ADDR and IPU_IMA_DATA Registers.

According to the channel type, two parameter formats can be used: for channels transferred YUV non-interleaved data with interleaving/de interleaving and for channels transferred YUV or RGB interleaved data.

44.4.7.3 Address Calculation

The following main addressing parameters are used:

- XB—Horizontal pixel position in frame
- YB—Vertical pixel position in frame
- SL—Stride line minus 1 (gap in bytes between two pixels in the same column in two consecutive rows).
- SX—Horizontal pixel scrolling offset
- SY—Vertical pixel scrolling offset
- EBA—Frame buffer base address in bytes (there are two such parameters to support double buffering)
- BPP—Bits per pixel
- FW—Frame width minus 1
- FH—Frame height minus 1

Relations between the addressing parameters and image frame are shown in Figure 44-144.

The system memory address in bytes is calculated as:

$$ADDR = EBA + (XB + SX) \cdot BPP + (YB + SY) \cdot (SL + 1)$$

with 0 < XB =< FW and 0 < YB =< FH.

When double buffering is used, the EBA0 is the base address of the buffer 0 and the EBA1 is the base address of the buffer 1. The IPU_CHA_CUR_BUF Register is a status register. It contains 1-bit pointers to the current working buffers for all IPU DMA channels. The IPU automatically toggles a pointer after completion of the current buffer processing. If the MCU is a data source for specific double-buffered channel, it should check this status bit in order to know what is the IPU current buffer. The MCU is allowed to write to the buffer only when a working DMA channel does not use it. After the MCU has been fill the buffer, it has to set the corresponding bit in the IPU_CHA_BUF0_RDY and IPU_CHA_BUF1_RDY Registers. If needed, the MCU can only clear the pointer by writing 1 but not set it.

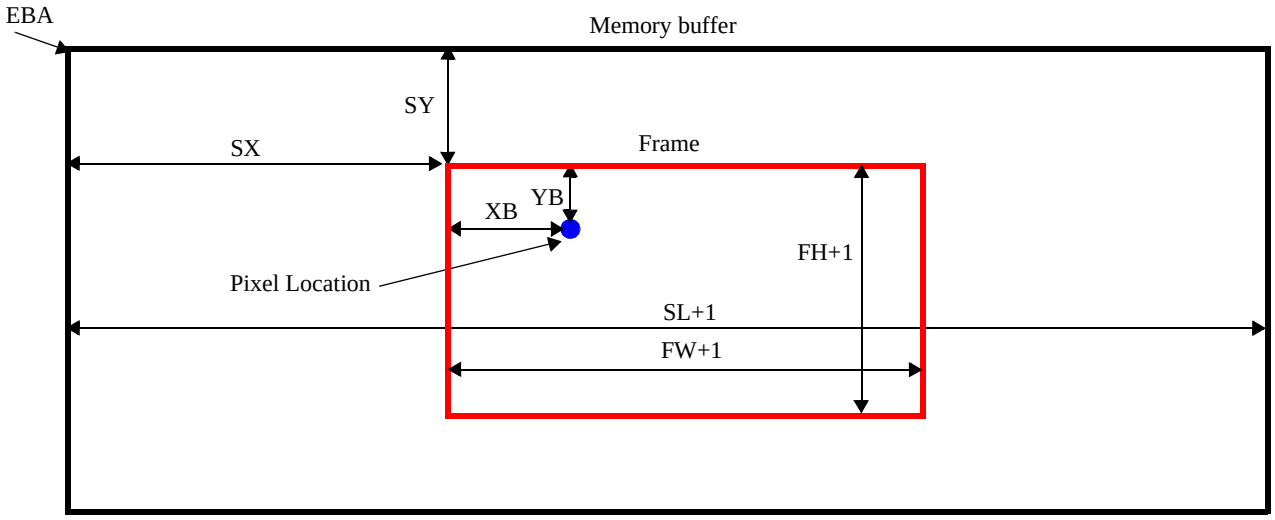


Figure 44-144. Addressing Parameters and Image Frame

The XB and YB coordinates are calculated according to addressing mode. There are two addressing modes:

- 2D mode
- Block mode

In 2D mode the pixel data is transferred to the memory row-by-row. There are two ways to use 2D mode: start from YB = 0 and finish at YB =< FH (YB is incremented) or start from YB = FH and finish at YB =< 0 (YB is decremented). The second option provides vertical flip of the image.

In block mode the frame is divided into blocks. This is needed for rotation or post-filtering, where the order used for data transfers is block-by-block. The order of the block transfer is according to the BAM bits in the IDMAC Channel Parameter Memory. The order within the block is row-by-row where the block size is limited by the block width (BW) and block height (BH) parameters. The BW and BH parameters are set by the PF or the IC rotation section and cannot be configured through the Channel Parameter Memory.

The Channel Control is responsible for the address calculation flow. It takes channel parameters from the Channel Parameter Memory, updates them and controls the Address Arithmetic Unit.

44.4.7.4 Format Converter

The Format Converter performs packing/unpacking the pixels with programmable position and width of color components, decoding 4- or 8-bits coded pixels according to a loaded look-up table, panning of an image read from the system memory according to a panning offset (start pixel address). The Format Converter supports formats with a pixel width of 4, 8, 16, 24 or 32 bits. Formatting parameters are written in the Channel Parameter Memory (see Table 44-29 through Table 44-33). The following parameters are used:

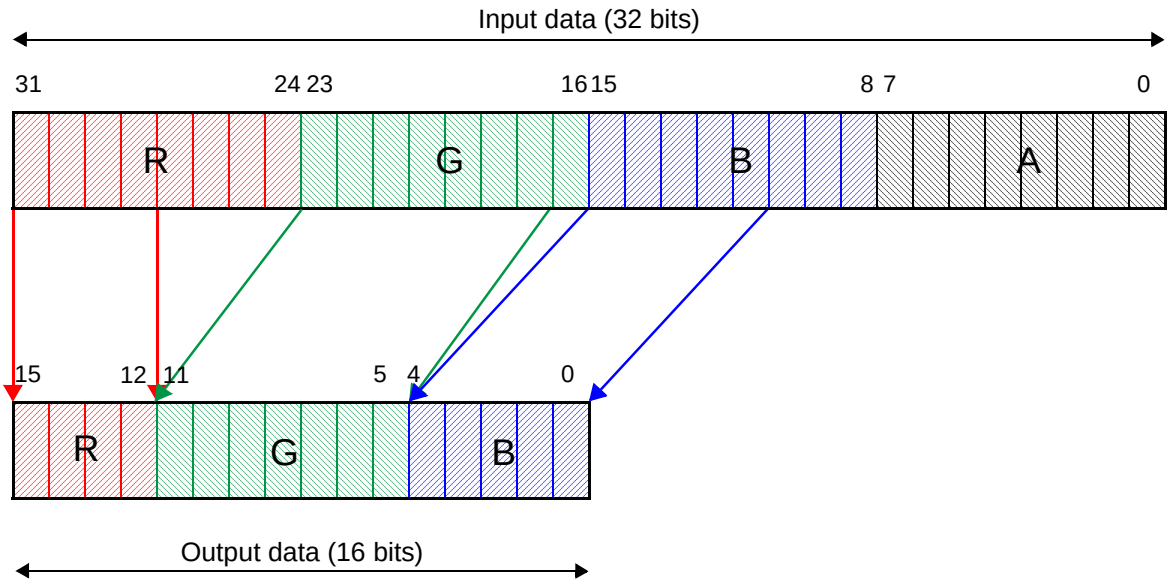
- Offset OFS0 between MSB position of the color component 0 and MSB position of packed pixel. The color component 0 occupies the most significant bits of the unpacked pixel (mostly this is the

R component). For read specified DMA channels, the OFS0 range is from 0 to 31. For write specified DMA channels, the OFS0 range is from 0 to 23.

- Color component 0 width WID0 minus 1.
- Offset OFS1 between MSB position of the color component 1 and MSB position of packed pixel. The color component 1 occupies the middle left bits of the unpacked pixel (mostly this is the G component). For read specified DMA channels, the OFS1 range is from 0 to 31. For write specified DMA channels, the OFS1 range is from 0 to 23.
- Color component 1 width WID1 minus 1.
- Offset OFS2 between MSB position of the color component 2 and MSB position of packed pixel. The color component 2 occupies the middle right bits of the unpacked pixel (mostly this is the B component). For read specified DMA channels, the OFS2 range is from 0 to 31. For write specified DMA channels, the OFS2 range is from 0 to 23.
- Color component 2 width WID2 minus 1.
- Offset OFS3 between MSB position of the color component 3 and MSB position of packed pixel. The color component 3 occupies the least significant bits of the unpacked pixel (mostly this is the A component). For read specified DMA channels, the OFS3 range is from 0 to 31. For write specified DMA channels, the OFS3 value is ignored and the real offset is set to 24 bits. The A value is undefined in these cases.
- Color component 3 width WID3 minus 1. For write specified DMA channels, the OFS3 value is ignored and the real offset is set to 24 bits.

The Format Converter is bypassed for internal IPU data transfers, for monochrome data transfer to the SDC (but panning is still performed), for generic data transfers and for data transfers to/from the PF.

Figure 44-145 and Figure 44-146 show examples of data packing and unpacking.



Byte 0 (red) **OFS0 = 0** **WID0 = 3**
Byte 1 (green) **OFS1 = 4** **WID1 = 6**
Byte 2 (blue) **OFS2 = 11** **WID2 = 4**

Figure 44-145. Example of Packing

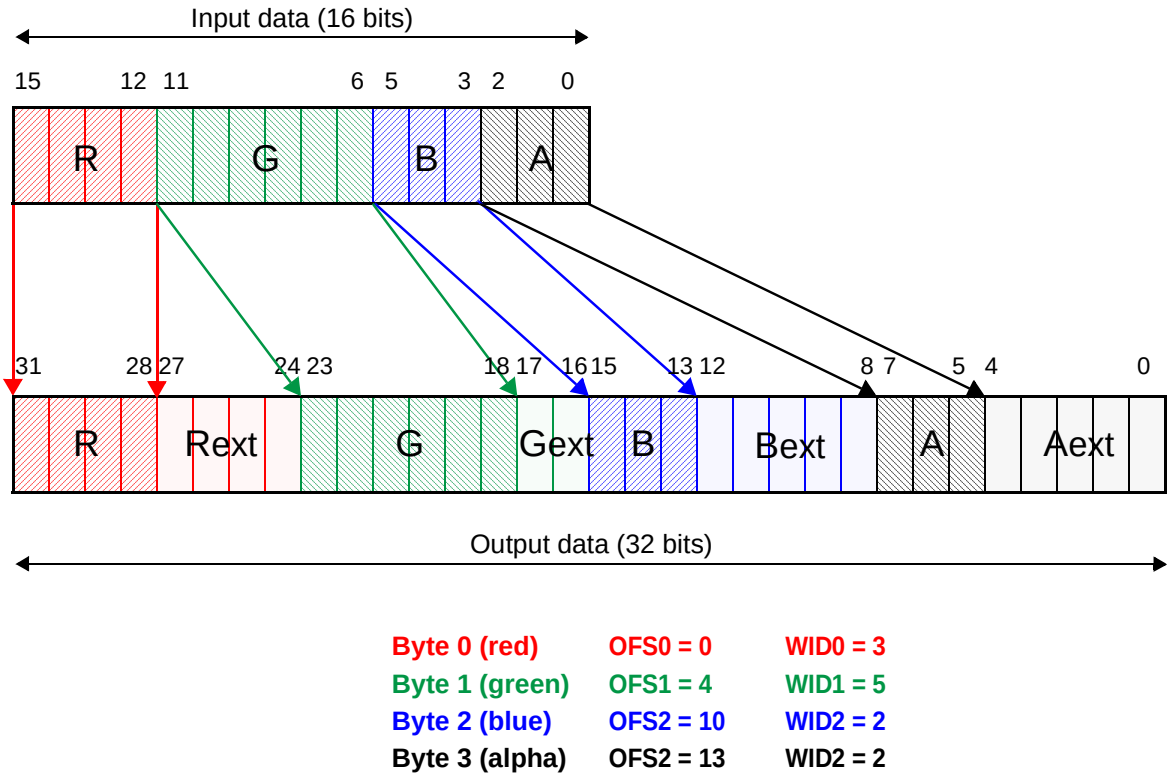


Figure 44-146. Example of Unpacking

If read data has the coded format, it is decoded via the look-up table. The Look-Up Table Memory (see IDMAC Look-Up Table Memory) must be loaded at the IDMAC initialization step. The LUT output format must match the IPU internal format RGBA 8888 where R is placed in msb and A is placed in lsb. The A field is used only for graphics data.

44.4.7.5 Internal Bus Interface

The Internal Bus Interface provides arbitration of internal DMA requests. There are two groups of DMA priorities - high and low. Each channel can get one of two priorities according to the IDMAC_CHA_PRI Register. Within a group, channel priorities are changed cyclically (round-robin method). If a channel has been granted for data transfer, it can transfer up to 4 bursts sequentially.

The Internal Bus Interface supports internal DMA buses protocol. If a DMA channel is enabled it can start data transfer after receiving both the new frame signal and the transfer request from an internal module. Transfer continues every time when the request is active. At the end of frame, data transfer for the channel is paused until the next new frame is arrived.

44.4.7.6 Dual Port FIFOs

The Dual Port FIFOs are used only for access to the external memory. They store different color components in the case of non-interleaved pixel formats. For interleaved pixel formats, only the Y Dual Port is used. Each FIFO contains two pages (see memory mapping in Table 44-182 and Table 44-183).

The FIFO Control provides read/write access to the FIFOs and supports pixel components interleaving/deinterleaving.

44.4.7.7 AHB Master Interface

The AHB Master Interface is responsible for data transfer from/to the system memory. The Interface supports only 32-bits burst accesses. For burst access, non-aligned addressing with 8-, 16- or 24-bits offsets is allowed. Two byte Endianness modes (LE, BE32) and two pixel Endianness modes are supported. The Interface swaps bytes and pixels according to the byte Endianness mode signal, the pixel Endianness parameter from the IPU_CONF Register and the BPP and BEM channel parameters. The AHB clock rate may be equal or lower than the HSP_CLK clock rate.

44.4.8 Control Module (CM)

44.4.8.1 Block Diagram

The CM Block Diagram is shown in Figure 44-147.

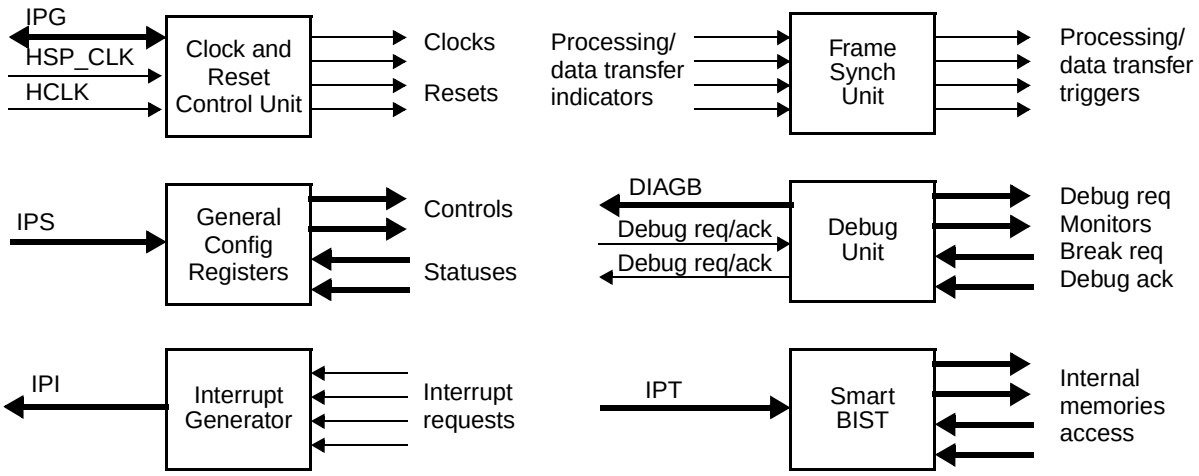


Figure 44-147. CM Block Diagram

The CM consists of the Frame Synchronization Unit (FSU), the Interrupt Generator (IG), the Debug Unit (DU), the General Configuration Registers (GCR), the Clock and Reset Control Unit (CRCU) and the Smart BIST (SBIST).

44.4.8.2 Frame Synchronization Unit

44.4.8.2.1 General Description

The FSU provides synchronization of tasks performed by different IPU submodules and MCU tasks. This allows to build complex processing flows which are performed automatically (without MCU involvement in synchronization IPU tasks). The FSU supports double buffering of image frames stored in the external memory and allows chaining IPU processing flows in automatic mode. Table 44-164 describes the most important use cases of chaining the IPU tasks.

Table 44-164. Use Cases of Chaining the IPU Tasks

Flow	Tasks chain	DMA Channels		
		Video Input	Other Input	Output
Capturing image from sensor and storing it in the memory without processing[1]	CSI --> IC --> MEM	---	---	DMAIC_7
Preprocessing raw image from dumb sensor for encoding	CSI --> IC --> MEM	---	---	DMAIC_7
	MEM or CSI[2] --> IC (PRP ENC) --> MEM	DMAIC_6	---	DMAIC_0
Preprocessing image from smart sensor for encoding	CSI --> IC (PRP ENC) --> MEM	---	---	DMAIC_0
Preprocessing and rotation of raw image from dumb sensor for encoding	CSI --> IC --> MEM	---	---	DMAIC_7
	MEM or CSI[2] --> IC (PRP ENC) --> MEM	DMAIC_6	---	DMAIC_0
	MEM --> IC (ROT ENC) --> MEM	DMAIC_10	---	DMAIC_8
Rotation and preprocessing of raw image from dumb sensor for encoding	CSI --> IC --> MEM	---	---	DMAIC_7
	MEM --> IC (ROT ENC) --> MEM	DMAIC_10	---	DMAIC_8
	MEM --> IC (PRP ENC) --> MEM	DMAIC_6	---	DMAIC_0
Preprocessing and rotation of image from smart sensor for encoding	CSI --> IC (PRP ENC) --> MEM	---	---	DMAIC_0
	MEM --> IC (ROT ENC) --> MEM	DMAIC_10	---	DMAIC_8
Preprocessing raw image from dumb sensor for viewfinder and displaying it on synchronous display	CSI --> IC --> MEM	---	---	DMAIC_7
	MEM or CSI[2] --> IC (PRP VF) --> MEM	DMAIC_6	DMAIC_3	DMAIC_1
	MEM --> SDC (BG/FG)	DMASDC_0/1	DMASDC_2	---

Table 44-164. Use Cases of Chaining the IPU Tasks (continued)

Flow	Tasks chain	DMA Channels		
		Video Input	Other Input	Output
Preprocessing and rotation of raw image from dumb sensor for viewfinder and displaying it on synchronous display	CSI --> IC --> MEM	---	---	DMAIC_7
	MEM or CSI[2] --> IC (PRP VF) --> MEM	DMAIC_6	DMAIC_3	DMAIC_1
	MEM --> IC (ROT VF) --> MEM	DMAIC_11	---	DMAIC_9
	MEM --> SDC (BG/FG)	DMASDC_0/1	DMASDC_2	---
Rotation and preprocessing of raw image from dumb sensor for viewfinder and displaying it on synchronous display	CSI --> IC --> MEM	---	---	DMAIC_7
	MEM --> IC (ROT VF) --> MEM	DMAIC_11	---	DMAIC_9
	MEM --> IC (PRP VF) --> MEM	DMAIC_6	DMAIC_3	DMAIC_1
	MEM --> SDC (BG/FG)	DMASDC_0/1	DMASDC_2	---
Preprocessing raw image from dumb sensor for viewfinder and displaying it on asynchronous display in command buffer mode	CSI --> IC --> MEM	---	---	DMAIC_7
	MEM or CSI[2] --> IC (PRP VF) --> MEM	DMAIC_6	DMAIC_3	DMAIC_1
	MEM --> ADC (SYS1/2)	DMAADC_2/3	DMAADC_4/5	---
Preprocessing and rotation of raw image from dumb sensor for viewfinder and displaying it on asynchronous display in command buffer mode	CSI --> IC --> MEM	---	---	DMAIC_7
	MEM or CSI[2] --> IC (PRP VF) --> MEM	DMAIC_6	DMAIC_3	DMAIC_1
	MEM --> IC (ROT VF) --> MEM	DMAIC_11	---	DMAIC_9
	MEM --> ADC (SYS1/2)	DMAADC_2/3	DMAADC_4/5	---
Rotation and preprocessing of raw image from dumb sensor for viewfinder and displaying it on asynchronous display in command buffer mode	CSI --> IC --> MEM	---	---	DMAIC_7
	MEM --> IC (ROT VF) --> MEM	DMAIC_11	---	DMAIC_9
	MEM --> IC (PRP VF) --> MEM	DMAIC_6	DMAIC_3	DMAIC_1
	MEM --> ADC (SYS1/2)	DMAADC_2/3	DMAADC_4/5	---
Preprocessing raw image from dumb sensor for viewfinder and direct displaying it on asynchronous display	CSI --> IC --> MEM	---	---	DMAIC_7
	MEM or CSI[2] --> IC (PRP VF) --> ADC (PRP)	DMAIC_6	DMAIC_3	DMAIC_1
Preprocessing image from smart sensor and displaying it on synchronous display	CSI --> IC (PRP VF) --> MEM	---	DMAIC_3	DMAIC_1
	MEM --> SDC (BG/FG)	DMASDC_0/1	DMASDC_2	---
Preprocessing and rotation of image from smart sensor and displaying it on synchronous display	CSI --> IC (PRP VF) --> MEM	---	DMAIC_3	DMAIC_1
	MEM --> IC (ROT VF) --> MEM	DMAIC_11	---	DMAIC_9
	MEM --> SDC (BG/FG)	DMASDC_0/1	DMASDC_2	---

Table 44-164. Use Cases of Chaining the IPU Tasks (continued)

Flow	Tasks chain	DMA Channels		
		Video Input	Other Input	Output
Preprocessing image from smart sensor and displaying it on asynchronous display in command buffer mode	CSI --> IC (PRP VF) --> MEM	---	DMAIC_3	DMAIC_1
	MEM --> ADC (SYS1/2)	DMAADC_2/3	DMAADC_4/5	---
Preprocessing and rotation of image from smart sensor and displaying it on asynchronous display in command buffer mode	CSI --> IC (PRP VF) --> MEM	---	DMAIC_3	DMAIC_1
	MEM --> IC (ROT VF) --> MEM	DMAIC_11	---	DMAIC_9
	MEM --> ADC (SYS1/2)	DMAADC_2/3	DMAADC_4/5	---
Preprocessing image from smart sensor for viewfinder and direct displaying it on asynchronous display	CSI --> IC (PRP VF) --> ADC (PRP)	---	DMAIC_3	DMAIC_1
Postprocessing image	MEM --> IC (PP) --> MEM	DMAIC_5	DMAIC_4	DMAIC_2
Postprocessing image and displaying it on synchronous display	MEM --> IC (PP) --> MEM	DMAIC_5	DMAIC_3	DMAIC_2
	MEM --> SDC (BG/FG)	DMASDC_0/1	DMASDC_2	---
Postprocessing and rotation of image and displaying it on synchronous display	MEM --> IC (PP) --> MEM	DMAIC_5	DMAIC_3	DMAIC_2
	MEM --> IC (ROT PP) -->MEM	DMAIC_13	---	DMAIC_12
	MEM --> SDC (BG/FG)	DMASDC_0/1	DMASDC_2	---
Rotation and postprocessing of image and displaying it on synchronous display	MEM --> IC (ROT PP) -->MEM	DMAIC_13	---	DMAIC_12
	MEM --> IC (PP) --> MEM	DMAIC_5	DMAIC_3	DMAIC_2
	MEM --> SDC (BG/FG)	DMASDC_0/1	DMASDC_2	---
Postprocessing image and displaying it on asynchronous display in command buffer mode	MEM --> IC (PP) --> MEM	DMAIC_5	DMAIC_3	DMAIC_2
	MEM --> ADC (SYS1/2)	DMAADC_2/3	DMAADC_4/5	---
Rotation and postprocessing of image and displaying it on asynchronous display in command buffer mode	MEM --> IC (ROT PP) -->MEM	DMAIC_13	---	DMAIC_12
	MEM --> IC (PP) --> MEM	DMAIC_5	DMAIC_3	DMAIC_2
	MEM --> ADC (SYS1/2)	DMAADC_2/3	DMAADC_4/5	---
Rotation and postprocessing of image and direct displaying it on asynchronous display	MEM --> IC (ROT PP) -->MEM	DMAIC_13	---	DMAIC_12
	MEM --> IC (PP) --> ADC (PP)	DMAIC_5	DMAIC_3	DMAIC_2
Postprocessing of image and direct displaying it on asynchronous display	MEM --> IC (PP) --> ADC (PP)	DMAIC_5	DMAIC_3	DMAIC_2

Table 44-164. Use Cases of Chaining the IPU Tasks (continued)

Flow	Tasks chain	DMA Channels		
		Video Input	Other Input	Output
Postfiltering (optional before any PP path)	MEM --> PF --> MEM	DMAPF_0/1/2/3/4	---	DMAPF_5/6/7
Synchronous display refresh	MEM --> SDC (BG/FG)	DMASDC_0/1	DMASDC_2	---
Asynchronous display refresh	MEM --> ADC (SYS1/2)	DMAADC_2/3	DMAADC_4/5	---
Reading data from asynchronous display	ADC (SYS1/2) --> MEM	---	---	DMAADC_6/7

1 In order to provide this flow, the RWS_EN bit should be 1, and the DMAIC_ 6_ BUF0_ RDY and DMAIC_ 6_ BUF1_ RDY bits for the DMAIC_6 channel should not be set.

2 If RWS_EN=0, the source is the CSI, else the source is the external memory

The FSU provides two mode of tasks synchronization: manual (when the MCU starts the task) and automatic (when the task is started automatically after finishing a previous chained task). Programming the IPU flows and tasks via the FSU is described by Table 44-165.

Table 44-165. Programming IPU Flows and Tasks

Tasks Chain	Frame Synchro-nization Mode	Control Bits Configuration	Description
Video coming from sensor directly to the system memory: (CSI --> MEM)	Manual on output (destination - the MCU)	Task is enabled by at least one of the following conditions: a) CSI_MEM_WR_EN = 0 and RWS_EN = 1 and at least one of PRPENC_EN/PRPVF_EN is 1 or b) CSI_MEM_WR_EN = 1 and RWS_EN = 0 Task is triggered for option (a) by internal signal which comes from CSI (CSI_IC_NF) if DMAIC_7_BUFn_RDY is set. For option (b), task is enabled asynchronously to the sensor frame boundaries. The CSI_MEM_WR_EN bit is a statthe IC mode control. It should be set/cleared either when the CSI is disabled or during the ICal blanking interval.	For option (a), the FSU waits for the trigger after the task has been enabled. If the trigger arrives and the output buffer in the system memory is ready, then the FSU signals the IC to start data transferring from the CSI to the system memory. If the trigger arrives and output buffer in the system memory is not ready then the FSU signals the IC to stop working and frame is dropped (the BAYER_FRM_LOST_ERR interrupt is set).
Video coming from the system memory to the IC for preprocessing for encoding and sent back to the system memory. (MEM --> IC (PRP ENC) --> MEM)	Manual on input (source - the MCU), manual on output (destination - the MCU)	Choose manual-out flow by setting PRPENC_DEST_SEL to 0 for the MCU. Task is enabled by RWS_EN = 1 and PRPENC_EN = 1. Task is triggered when both input and output buffers are ready. Input buffer ready is indicated by setting DMAIC_6_BUFn_RDY. ENC Input buffer ready is valid by setting ENC_IN_VALID. ENC output buffer ready is indicated by setting DMAIC_0_BUFn_RDY by the MCU.	After the task is enabled and If input buffer is ready and valid there are two cases: a) the VF task is enabled and its input buffer valid. b) the VF task is not enabled or its input buffer is not valid. In the first case the FSU will signals the IC to start working on encoding task only if both output buffers of VF and ENC are ready. In the second case the FSU will signals the IC to start working on encoding task when ENC output buffer is ready.
	Manual on input (source - the MCU), automatic on output (destination - the IC (ROT ENC))	Choose auto-out flow by setting PRPENC_DEST_SEL to1 for ROTENC. Task is enabled by RWS_EN = 1 and PRPENC_EN = 1. Task is triggered when both input and output buffers are ready. Input buffer ready is indicated by setting DMAIC_6_BUFn_RDY. ENC input buffer ready is valid by setting ENC_IN_VALID. ENC output buffer ready is indicated set by EOF of rotation for encoding channel (DMAIC_10_LBF).	

Table 44-165. Programming IPU Flows and Tasks (continued)

Tasks Chain	Frame Synchro-nization Mode	Control Bits Configuration	Description
Video coming from the system memory to the IC for pre-processing for view-finder and back to the system memory or directly to the ADC. (MEM --> IC (PRP VF) --> MEM)	Manual on input (source - the MCU), manual on output (destination - the MCU)	Choose manual-out flow by setting PRPVF_DEST_SEL to 0. Task is enabled by RWS_EN = 1 and PRPVF_EN = 1. Task is triggered when both input and output buffers are ready. Input buffer ready is indicated by setting DMAIC_6_BUFn_RDY. VF input buffer ready is valid by setting VF_IN_VALID. VF output buffer ready is indicated by setting DMAIC_1_BUFn_RDY by the MCU.	After the task is enabled and If input buffer is ready and valid there are two cases: a) the ENC task is enabled and its input buffer valid. b) the ENC task is not enabled or its input buffer is not valid. In the first case the FSU will signals the IC to start working on encoding task only if both output buffers of VF and ENC are ready. In the second case the FSU will signals the IC to start working on encoding task when VF output buffer is ready.
	Manual on input (source - the MCU), automatic on output (destination - the IC (ROT VF) or the SDC (BG/FG) or the ADC (SYS1/2))	Choose auto-out flow by setting PRPENC_DEST_SEL to: - 1 for ROTVF - 2 for the ADCSYS1 - 3 for the ADCSYS2 - 4 for the SDCBG - 5 for the SDCFG - 6 for the ADC direct. Task is enabled by RWS_EN = 1 and PRPVF_EN = 1. Task is triggered when both input and output buffers are ready. Input buffer ready is indicated by setting DMAIC_6_BUFn_RDY. VF input buffer ready is valid by setting VF_IN_VALID. VF output buffer ready is indicated by EOF of the DMA channel which is defined by PRPENC_DEST_SEL. For SDCBG, SDCFG, ADC direct output buffer is always ready.	

Table 44-165. Programming IPU Flows and Tasks (continued)

Tasks Chain	Frame Synchro-nization Mode	Control Bits Configuration	Description
Video coming from sensor to the IC for pre-processing for encoding and back to the system memory. (CSI --> IC (PRP ENC) --> MEM)	Manual on output (destination - the MCU)	Choose manual-out flow by setting PRPENC_DEST_SEL to 0 for the MCU. Task is enabled by RWS_EN = 0 and PRPENC_EN = 1. Task is triggered by CSI internal output (CSI_IC_NF) if output buffer is ready. ENC input is valid by SKIP_ENC_FRM = 0. ENC output buffer ready is indicated by setting DMAIC_0_BUFn_RDY by the MCU.	After the task is enabled, the FSU waits for its trigger. If the trigger arrives and the output buffer in the system memory is ready, then the FSU signals the IC to start processing the data from the CSI and send it to the system memory after processing. If the trigger arrives and output buffer in the system memory is not ready then the FSU signals the IC to stop working and frame is dropped (ENC_FRM_LOST_ERR interrupt is set).
	Automatic on output (destination - the IC (ROT ENC))	Choose auto-out flow by setting PRPENC_DEST_SEL to 1for the MCU. Task is enabled by RWS_EN = 0 and PRPENC_EN = 1. Task is triggered by CSI internal output (CSI_IC_NF) if output buffer is ready. ENC input is valid by SKIP_ENC_FRM = 0. ENC output buffer ready is indicated by EOF of rotation for encoding channel (DMAIC_10_LBF).	
Video coming from sensor to the IC for pre-processing for view-finder and back to the system memory. (CSI --> IC (PRP VF) --> MEM), CSI --> IC (PRP VF) --> ADC (PRP))	Manual on output (destination - the MCU)	Choose manual-out flow by setting PRPVF_DEST_SEL to 0 for the MCU. Task is enabled by RWS_EN = 0 and PRPVF_EN = 1. Task is triggered by CSI internal output (CSI_IC_NF) if output buffer is ready. VF input is valid by SKIP_VF_FRM = 0. ENC output buffer ready is indicated by setting DMAIC_1_BUFn_RDY by the MCU.	After the task is enabled, the FSU wait for its trigger. If the trigger arrives and the output buffer in the system memory is ready, then the FSU signals the IC to start working - process the data from CSI and send it to the system memory after processing. If the trigger arrives and output buffer in the system memory is not ready then the FSU signals the IC to stop working and frame is dropped (VF_FRM_LOST_ERR interrupt is set).
	Automatic on output (destination - the IC (ROT VF) or the SDC or the ADC)	Choose auto-out flow by setting PRPVF_DEST_SEL to: - 1 for ROTVF. - 2 for the ADCSYS1. - 3 for the ADCSYS2. - 4 for the SDCBG. - 5 for the SDCFG. - 6 for the ADC direct. Task is enabled by RWS_EN = 0 and PRPVF_EN = 1. Task is triggered by CSI internal output (CSI_IC_NF) if output buffer is ready. VF input is valid by SKIP_VF_FRM = 0. VF output buffer ready is indicated by EOF of the DMA channel which is defined by PRPVF_DEST_SEL.	

Table 44-165. Programming IPU Flows and Tasks (continued)

Tasks Chain	Frame Synchro-nization Mode	Control Bits Configuration	Description
Video coming from the system memory to the IC for rotation for encoding and back to the system memory. (MEM --> IC (ROT ENC) --> MEM)	Manual on input (source - the MCU), manual on output (destination - the MCU)	Choose manual-in flow by setting PRPENC_ROT_SRC_SEL to 0 for the MCU. Task is enabled by PRPENC_ROT = 1. Task is triggered if both input and output buffers are ready. Input buffer ready is indicated by setting DMAIC_10_BUFn_RDY by the MCU. Output buffer ready is indicated by setting DMAIC_8_BUFn_RDY by the MCU.	After the task is enabled and If input buffer and output buffer are ready, the FSU will signal rotation unit to start working on rotation for encoding task.
	Automatic on input (source - the IC (PRP ENC), manual on output (destination - the MCU)	Choose auto-in flow by setting PRPENC_ROT_SRC_SEL to 1 for ENC. Task is enabled by PPRPENC_ROT = 1. Task is triggered if both input and output buffers are ready. Input buffer ready is indicated by output EOF of encoding task (DMAIC_0_LBF). Output buffer ready is indicated by setting DMAIC_8_BUFn_RDY by the MCU.	

Table 44-165. Programming IPU Flows and Tasks (continued)

Tasks Chain	Frame Synchro-nization Mode	Control Bits Configuration	Description
Video coming from the system memory to the IC for rotation for view-finder and back to the system memory. (MEM --> IC (ROT VF) --> MEM)	Manual on input (source - the MCU), manual on output (destination - the MCU)	Choose manual-in flow by setting PRPVF_ROT_SRC_SEL to 0 for the MCU. Choose manual-out flow by setting PRPVF_ROT_DEST_SEL to 0 for the MCU. Task is enabled by PRPVF_ROT = 1. Task is triggered if both input and output buffers are ready. input buffer ready is indicated by setting DMAIC_11_BUFn_RDY by the MCU. Output buffer ready is indicated by setting DMAIC_9_BUFn_RDY by the MCU.	After the task is enabled and If input buffer and output buffer are ready, the FSU will signal rotation unit to start working on rotation for view-finder task.
	Automatic on input (source - the IC (PRP VF)), manual on output (destination - the MCU)	Choose auto-in flow by setting PRPVF_ROT_SRC_SEL to 1 for VF. Choose manual-out flow by setting PRPVF_ROT_DEST_SEL to 0 for the MCU. Task is enabled by PRPVF_ROT = 1. Task is triggered if both input and output buffers are ready. Input buffer ready is indicated by output EOF of encoding task (DMAIC_1_LBF). Output buffer ready is indicated by setting DMAIC_9_BUFn_RDY by the MCU.	
	Manual on input (source - the MCU), automatic on output (destination - the SDC or the ADC)	Choose manual-in flow by setting PRPVF_ROT_SRC_SEL to 0 for the MCU. Choose auto-out flow by setting PRPVF_ROT_DEST_SEL to: - 2 for the ADCSYS1. - 3 for the ADCSYS2. - 4 for the SDCBG. - 5 for the SDCFG. Task is enabled by PRPVF_ROT = 1. Task is triggered if both input and output buffers are ready. Input buffer ready is indicated by setting DMAIC_11_BUFn_RDY by the MCU. Output buffer ready is indicated by EOF of the DMA channel which is defined by PRPVF_ROT_DEST_SEL.	

Table 44-165. Programming IPU Flows and Tasks (continued)

Tasks Chain	Frame Synchro-nization Mode	Control Bits Configuration	Description
Video coming from the system memory to the IC for rotation for view-finder and back to the system memory. (MEM --> IC (ROT VF) --> MEM)	Automatic on input (source - the IC (PRP VF)), automatic on output (destination - the SDC or the ADC)	Choose manual-in flow by setting PRPVF_ROT_SRC_SEL to 1 for VF. Choose auto-out flow by setting PRPVF_ROT_DEST_SEL to: - 2 for the ADCSYS1. - 3 for the ADCSYS2. - 4 for the SDCBG. - 5 for the SDCFG. Task is enabled by PRPVF_ROT = 1. Task is triggered if both input and output buffers are ready. Input buffer ready is indicated by output EOF of encoding task (DMAIC_1_LBF). Output buffer ready is indicated by EOF of the DMA channel which is defined by PRPVF_ROT_DEST_SEL.	
Video coming from the system memory to the IC for post-processing and back to the system memory or to the ADC. (MEM --> IC (PP) --> MEM MEM --> IC (PP) --> ADC (PP))	Manual on input (source - the MCU), manual on output (destination - the MCU)	Choose manual-in flow by setting PP_SRC_SEL to 0 for the MCU. Choose auto-out flow by setting PP_DEST_SEL to 0 for the MCU. Task is enabled by PP_EN = 1. Task is triggered if both input and output buffers are ready. Input buffer ready is indicated by setting DMAIC_5_BUFn_RDY by the MCU. Output buffer ready is indicated by setting DMAIC_2_BUFn_RDY by the MCU.	After the task is enabled and If input buffer and output buffers are ready then the FSU will signal the IC to start working on post-processing (PP) task. From now on, the FSU will check if input and output buffers are ready after completion frame processing. If they are ready, it will signal the IC again to start working, else it will wait until the buffer are ready or until task is disabled.
	Manual on input (source - the MCU), automatic on output (destination - the IC (ROT PP) or the SDC or the ADC)	Choose manual-in flow by setting PP_SRC_SEL to 0 for the MCU. Choose auto-out flow by setting PP_DEST_SEL to: 1 for ROTPP. 2 for the ADCSYS1. 3 for the ADCSYS2. 4 for the SDCBG. 5 for the SDCFG. 6 for the ADC direct. Task is enabled by PP_EN = 1. Task is triggered if both input and output buffers are ready. Input buffer ready is indicated by setting DMAIC_5_BUFn_RDY by the MCU. Output buffer ready is indicated by EOF of the DMA channel which is defined by PP_DEST_SEL.	

Table 44-165. Programming IPU Flows and Tasks (continued)

Tasks Chain	Frame Synchro-nization Mode	Control Bits Configuration	Description
Video coming from the system memory to the IC for post-processing and back to the system memory or to the ADC. (MEM --> IC (PP) --> MEM MEM --> IC (PP) --> ADC (PP))	Automatic on input (source - the PF or ROTPP), manual on output (destination - the MCU)	Choose auto-in flow by setting PP_SRC_SEL to: 1 for the PF. 2 for ROTPP. Choose manual-out flow by setting PP_DEST_SEL to 0 for the MCU. Task is enabled by PP_EN = 1. Task is triggered if both input and output buffers are ready. Input buffer ready is indicated by output EOF of the DMA channel which is defined by PP_SRC_SEL. Output buffer ready is indicated by setting DMAIC_2_BUFn_RDY by the MCU.	
	Automatic on input (source - the PF or ROT), automatic on output (destination - ROTPP or the SDC or the ADC)	Choose auto-in flow by setting PP_SRC_SEL to: 1 for the PF for ROTPP. Choose auto-out flow by setting PP_DEST_SEL to: - 1 for ROTPP. - 2 for the ADCSYS1. - 3 for the ADCSYS2. - 4 for the SDCBG. - 5 for the SDCFG. - 6 for the ADC direct. Task is enabled by PP_EN = 1. Task is triggered by EOF of this task (DMAIC_5_LBF). Input buffer ready is indicated by output EOF of the DMA channel which is defined by PP_SRC_SEL. Output buffer ready is indicated by EOF of the DMA channel which is defined by PP_DEST_SEL.	

Table 44-165. Programming IPU Flows and Tasks (continued)

Tasks Chain	Frame Synchro-nization Mode	Control Bits Configuration	Description
Video coming from the system memory to the IC for rotation for post-processing and back to the system memory. (MEM --> IC (ROT PP) --> MEM)	Manual on input (source - the MCU), manual on output (destination - the MCU)	Choose manual-in flow by setting PP_ROT_SRC_SEL to 0 for the MCU. Choose auto-out flow by setting PP_ROT_DEST_SEL to 0 for the MCU. Task is enabled by PP_ROT = 1. Task is triggered if both input and output buffers are ready. Input buffer ready is indicated by setting DMAIC_13_BUFn_RDY by the MCU. Output buffer ready is indicated by setting DMAIC_12_BUFn_RDY by the MCU.	After the task is enabled and If input and output buffers are ready, the FSU will signal the rotation unit to start working on rotation for PP task. From now, the FSU will check if input and output buffers are ready after completion frame processing. If they are ready, it will signal the rotation unit again to start working, else it will wait until the buffer are ready or until task is disabled.
	Manual on input (source - the MCU), automatic on output (destination - the IC (PP) or the SDC or the ADC)	Choose manual-in flow by setting PP_ROT_SRC_SEL = to 0 for the MCU. Choose auto-out flow by setting PP_ROT_DEST_SEL to: - 1 for PP. - 2 for the ADCSYS1. - 3 for the ADCSYS2. - 4 for the SDCBG. - 5 for the SDCFG. Task is enabled by PP_ROT = 1. Task is triggered if both input and output buffers are ready. Input buffer ready is indicated by setting DMAIC_13_BUFn_RDY by the MCU. Output buffer ready is indicated by EOF of the DMA channel which is defined by PP_ROT_DEST_SEL.	
	Automatic on input (source - the IC (PP) or the PF), manual on output (destination - the MCU)	Choose auto-in flow by setting PP_ROT_SRC_SEL to: - 1 for PP. - 2 for the PF. Choose manual-out flow by setting PP_DEST_SEL to 0 for the MCU. Task is enabled by PP_ROT = 1. Task is triggered if both input and output buffers are ready. Input buffer ready is indicated by output EOF of the DMA channel which is defined by PP_ROT_SRC_SEL. Output buffer ready is indicated by setting DMAIC_12_BUFn_RDY by the MCU.	

Table 44-165. Programming IPU Flows and Tasks (continued)

Tasks Chain	Frame Synchro-nization Mode	Control Bits Configuration	Description
Video coming from the system memory to the IC for rotation for post-processing and back to the system memory. (MEM --> IC (ROT PP) --> MEM)	Automatic on input (source - the IC (PP) or the PF), automatic on output (destination - the IC (PP) or the SDC or the ADC)	Choose auto-in flow by setting PP_ROT_SRC_SEL to: 1 for PP. 2 for the PF. Choose auto-out flow by setting PP_ROT_DEST_SEL to: - 1 for PP. - 2 for the ADCSYS1. - 3 for the ADCSYS2. - 4 for SDCBG. - 5 for the SDCFG. Task is enabled by PP_ROT = 1. Task is triggered if both input and output buffers are ready. Input buffer ready is indicated by output EOF of the DMA channel which is defined by PP_ROT SRC_SEL. Output buffer ready is indicated by EOF of the DMA channel which is defined by PP_ROT_DEST_SEL.	
Video coming from the system memory for post-filtering and back to the system memory. (MEM --> PF --> MEM)	Manual on input (source - the MCU), manual on output (destination - the MCU)	Choose manual-out flow by setting the PF_DEST_SEL to 0 for the MCU. Task is enabled by the PF_TYPE != “00”. Task is triggered if both input and output buffers are ready. Input buffer ready is indicated by setting DMAPF_0/1/2/3/4_BUFn_RDY by the MCU. Output buffer ready is indicated by setting DMAIC_5/6/7_BUFn_RDY by the MCU.	After the task is enabled and If input buffer and output buffers are ready, the FSU will signal the PF unit to start working on the post-filtering task. From now on, the FSU will check if input and output buffers are ready after completion current frame processing. If they are ready it will signal the PF unit again to start working, else it will wait until the buffer are ready or until task is disabled.
	Manual on input (source - the MCU), automatic on output (destination - the IC (ROT PP) or the IC (PP))	Choose manual-out flow by setting the PF_DEST_SEL to: 1 for PP. 2 for PP_ROT. Task is enabled by the PF_TYPE != “00”. Task is triggered if both input and output buffers are ready. Input buffer ready is indicated by setting DMAPF_0/1/2/3/4_BUFn_RDY by the MCU. Output buffer ready is indicated by EOF of the DMA channel which is defined by the PF_DEST_SEL.	

Table 44-165. Programming IPU Flows and Tasks (continued)

Tasks Chain	Frame Synchro-nization Mode	Control Bits Configuration	Description
Video or graphics coming from the system memory for display by the SDC as foreground. (MEM --> SDC (FG))	Manual on input (source - the MCU)	Choose manual-out flow by setting the SDC0_SRC_SEL to 0 for the MCU. Task is enabled by the SDC_EN = 1. Task is triggered by the FG_EN = 1. Input buffer ready is indicated by setting DMASDC_0_BUFn_RDY.	Task is enabled by the SDC_EN. The SDC sends a request to the FSU for asking if there is a new foreground buffer. The FSU acknowledges if the foreground input buffer is ready. For dumb displays, the SDC sends DMA request for refreshing the foreground plane independently on the FSU acknowledge.
	Automatic on input (the IC (PRP VF) or the IC (PP) or the IC (ROT VF) or the IC (ROT PP))	Choose auto-out flow by setting the SDC0_SRC_SEL to: - 1 for ROTVF. - 2 for ROTPP. - 3 for VF. - 4 for PP. Task is enabled by the SDC_EN = 1. Task is triggered by the FG_EN = 1. Input buffer ready is indicated by EOF of the DMA channel which is defined by the SDC0_SRC_SEL.	
Video or graphics coming from the system memory for display by the SDC as background. (MEM --> SDC (BG))	Manual on input (source - the MCU)	Choose manual-out flow by setting the SDC0_SRC_SEL to 0 for the MCU. Task is enabled by the SDC_EN = 1. Task is triggered by the BG_EN = 1. Input buffer ready is indicated by setting DMASDC_0_BUFn_RDY.	Task is enabled by the SDC_EN. The SDC sends a request to the FSU for asking if there is a new background buffer. The FSU acknowledges if the foreground input buffer is ready. For dumb displays, the SDC sends DMA request for refreshing the background plane independently on the FSU acknowledge.
	Automatic on input (the IC (PRP VF) or the IC (PP) or the IC (ROT VF) or the IC (ROT PP))	Choose auto-out flow by setting the SDC1_SRC_SEL to: - 1 for ROTVF. - 2 for ROTPP. - 3 for VF. - 4 for PP. Task is enabled by the SDC_EN = 1. Task is triggered by the BG_EN = 1. Input buffer ready is indicated by EOF of the DMA channel which is defined by the SDC1_SRC_SEL.	

Table 44-165. Programming IPU Flows and Tasks (continued)

Tasks Chain	Frame Synchro-nization Mode	Control Bits Configuration	Description
Video or graphics coming from the system memory for display by the ADC channel 1. (MEM --> ADC (SYS1))	Manual on input (source - the MCU)	Choose manual-out flow by setting the ADC2_SRC_SEL to 0 for the MCU. Task is enabled when SYS1_MODE = 1 or 3 or 5 or 6 or 7 (write to display). Task is triggered if the input buffer is ready. Input buffer ready is indicated by setting DMAADC_2_BUFn_RDY by the MCU. If SYS1_MODE = 7 (command mode), setting DMAADC_4_BUFn_RDY by the MCU is also required.	After the task is enabled and If input buffer is ready, the FSU will signal the ADC to start transferring data from memory to display on the system channel 1. From now on, the FSU will check if input and buffer is ready after current frame transfer. If buffer is ready it will signal the ADC again to start working, else it will wait until the buffer is ready or until task is disabled.
	Automatic on input (source - the IC (PRP VF) or the IC (PP) or the IC (ROT VF) or the IC (ROT PP))	Choose manual-out flow by setting the ADC2_SRC_SEL to: - 1 for ROTVF. - 2 for ROTPP. - 3 for VF. - 4 for PP. Task is enabled when SYS1_MODE = 1 or 3 or 5 or 6 or 7 (write to display). Task is triggered if the input buffer is ready. Input buffer ready is indicated by EOF of the DMA channel which is defined by the ADC2_SRC_SEL. If SYS1_MODE = 7 (command mode), setting DMAADC_4_BUFn_RDY by the MCU is also required.	
	Snooping mode	Choose snoop mode by setting the ADC2_SRC_SEL to 5. Task is enabled when SYS1_MODE = 1 or 3 or 5 or 6 or 7 (write to display). Task is triggered if the input buffer is ready. Input buffer ready is indicated when SNOOP signal for channel 1 comes from the EMI towards the IPU.	
	ADC autorefresh	Choose ADC refresh by setting the ADC2_SRC_SEL to 6. Task is enabled when SYS1_MODE = 1 or 3 or 5 or 6 or 7 (write to display). Configure AUTO_REF_PER to the wanted autorefresh period (see the IPU_FS_DISP_FLOW register). Task is triggered if the input buffer is ready. Input buffer ready is indicated when the ADC_REFRESH signal for channel 1 is asserted.	

Table 44-165. Programming IPU Flows and Tasks (continued)

Tasks Chain	Frame Synchro-nization Mode	Control Bits Configuration	Description
Video or graphics coming from the system memory for display by the ADC channel 1. (MEM --> ADC (SYS1))	ADC autorefresh with snooping	Choose ADC refresh with snooping by setting the ADC2_SRC_SEL to 7. Task is enabled when SYS1_MODE = 1 or 3 or 5 or 6 or 7 (write to display). Configure AUTO_REF_PER to the wanted period of refresh time (see IPU_FS_DISP_FLOW register). Task is triggered if the input buffer is ready. Input buffer ready is indicated when the ADC_REFRESH signal for channel 1 is asserted after or with snoop signal.	

Table 44-165. Programming IPU Flows and Tasks (continued)

Tasks Chain	Frame Synchro-nization Mode	Control Bits Configuration	Description
Video or graphics coming from the system memory for display by the ADC channel 2. (MEM --> ADC (SYS2))	Manual on input (source - the MCU)	Choose manual-out flow by setting the ADC3_SRC_SEL to 0 for the MCU. Task is enabled when SYS2_MODE = 1 or 3 or 5 or 6 or 7 (write to display). Task is triggered if the input buffer is ready. Input buffer ready is indicated by setting DMAADC_3_BUFn_RDY by the MCU. If SYS2_MODE = 7 (command mode), setting DMAADC_5_BUFn_RDY by the MCU is also required.	After the task is enabled and If input buffer is ready, the FSU will signal the ADC to start transferring data from memory to display on the system channel 2. From now on, the FSU will check if input and buffer is ready after current frame transfer. If buffer is ready it will signal the ADC again to start working, else it will wait until the buffer is ready or until task is disabled.
	Automatic on input (source - the IC (PRP VF) or the IC (PP) or the IC (ROT VF) or the IC (ROT PP))	Choose manual-out flow by setting the ADC3_SRC_SEL to: - 1 for ROTVF. - 2 for ROTPP. - 3 for VF. - 4 for PP. Task is enabled when SYS2_MODE = 1 or 3 or 5 or 6 or 7 (write to display). Task is triggered if the input buffer is ready. Input buffer ready is indicated by EOF of the DMA channel which is defined by the ADC3_SRC_SEL. If SYS2_MODE = 7 (command mode), setting DMAADC_5_BUFn_RDY by the MCU is also required.	
	Snooping mode	Choose snoop mode by setting the ADC3_SRC_SEL to 5. Task is enabled when SYS2_MODE = 1 or 3 or 5 or 6 or 7 (write to display). Task is triggered if the input buffer is ready. Input buffer ready is indicated when SNOOP signal for channel 2 comes from the EMI towards the IPU.	
	ADC autorefresh	Choose ADC refresh by setting the ADC3_SRC_SEL to 6. Task is enabled when SYS2_MODE = 1 or 3 or 5 or 6 or 7 (write to display). Configure AUTO_REF_PER to the wanted autorefresh period (see the IPU_FS_DISP_FLOW register). Task is triggered if the input buffer is ready. Input buffer ready is indicated when the ADC_REFRESH signal for channel 2 is asserted.	

Table 44-165. Programming IPU Flows and Tasks (continued)

Tasks Chain	Frame Synchro-nization Mode	Control Bits Configuration	Description
Video or graphics coming from the system memory for display by the ADC channel 2. (MEM --> ADC (SYS2))	ADC autorefresh with snooping	Choose ADC refresh with snooping by setting the ADC3_SRC_SEL to 7. Task is enabled when SYS2_MODE = 1 or 3 or 5 or 6 or 7 (write to display). Configure AUTO_REF_PER to the wanted period of refresh time (see IPU_FS_DISP_FLOW register). Task is triggered if the input buffer is ready. Input buffer ready is indicated when the ADC_REFRESH signal for channel 2 is asserted after or with snoop signal.	

44.4.8.2.2 Frame Synchronization Flow

1. Initialization

   The MCU initializes all parameters for a task by writing to the GCR and to the parameters memories which reside in IPU submodules. The initialization must occur before the MCU enables the task.

2. Enabling

   After the initialization step has been completed, the MCU enables the task by setting its enable bit in a appropriate register.

3. Triggering

   After the task is enabled, the FSU waits for triggering signal. The triggering signals is a combination of the enable bit and the buffer ready signal which can be driven by the MCU (DMA<BL>_<#>_BUF_RDY) and/or by the preceding task (DMA<BL>_<#>_EOF).

   The trigger causes the FSU to invoke the relevant unit to start by assertion of the <TASK>_NEW_FRM_RDY signal. In some cases triggering occurs at the enabling step (when the enable bit is the trigger for the task).

4. Operating

   The triggering step cause the task to move to active mode, this is the operating step. In this step, the FSU monitors the synchronization signals from MCU, IDMAC and the corresponding processing units, and controls the units operation. The FSU also controls the IDMAC buffer toggling when double buffer page flipping is used.

   The FSU checks at end of each frame if the next frame can be served. If the answer is yes, the FSU stays in active mode with re-sending the <TASK>_NEW_FRM_RDY signal and updating the relevant flags (e.g. DMA<BL>_<#>_CUR_BUF and DMA<BL>_<#>_BUF_RDY). If the answer is no, the FSU moves to pause mode and pauses the task waiting until the next frame can be served.

5. Disabling

   When the task is disabled by the MCU (by negating the enable bit), it moves back to non-active mode.

44.4.8.2.3 Frame Synchronization Example

The following example describes the flow of the pre-processing task in the IC. The example will be explained step by step.

1. Initialization

   The MCU should initialize all relevant parameters:

   — Initialize the GCR parameters which relevant for this task.

   — Initialize the Task Parameter Memory of the IC with the relevant parameters for this task.

   — Initialize the Channel Parameter Memory of IDMAC: post-processing video input channel (DMAIC_CB5), post-processing graphics input channel (DMAIC_CB4).post-processing video output channel (DMAIC_CB3).

   — Initialize the first input buffer of video and graphics and allocate the buffer for output video, and then set the “BUF0/1_RDY” bits for each ready channel.

2. Enabling

   The MCU should set the proper bit in GCR for enabling the task. In this example the bit is PP_EN in IC_CONF Register (Figure 44-148).

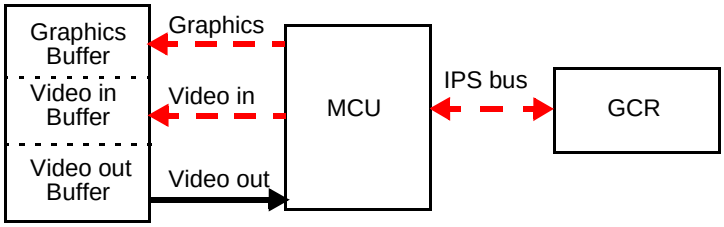


Figure 44-148. Initialization and Enabling Steps

3. Triggering

   In this case, trigger condition is when buffers of input and output video are ready (DMAIC_5(2)_BUF0_RDY is set) and task is enabled (PP_EN are set). Then the FSU sends the PP_NEW_FRM_RDY signal to the IC (Figure 44-149).

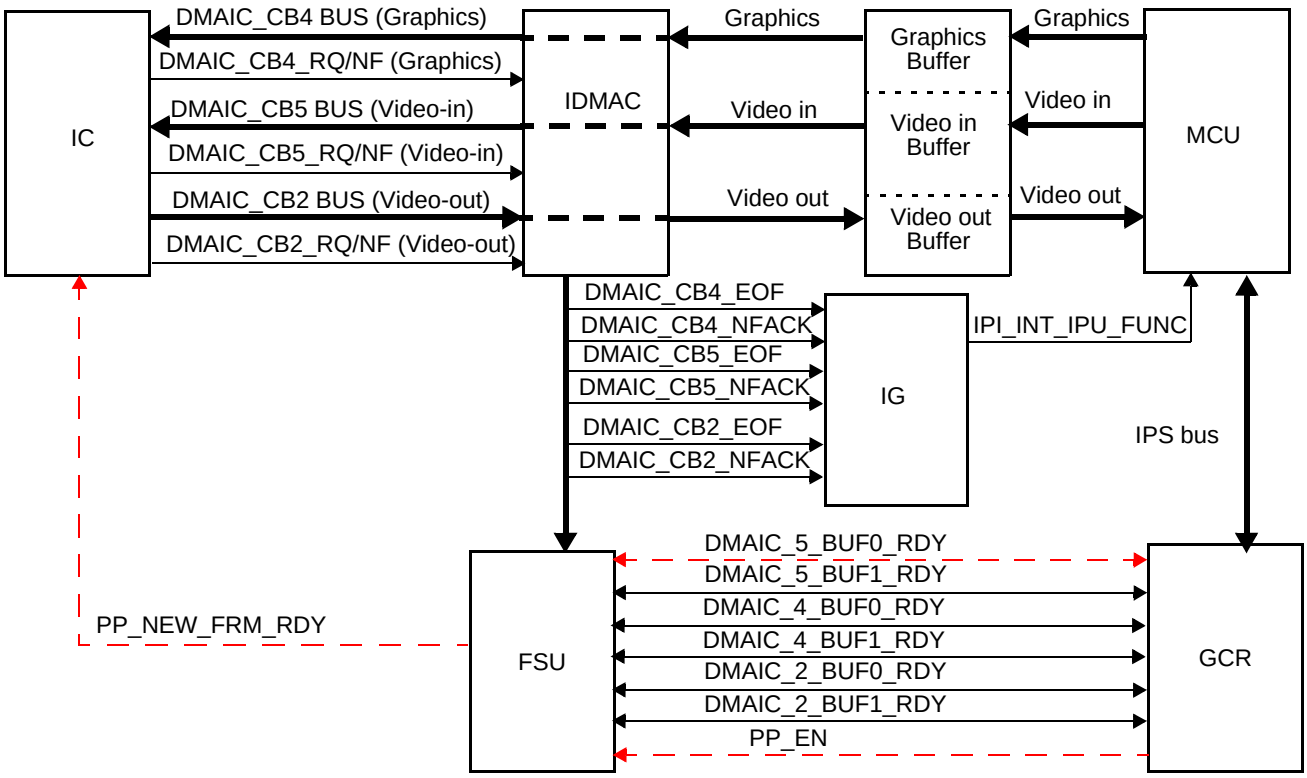


Figure 44-149. Triggering Step

4. Operating

   After the IC has got PP_NEW_FRM_RDY, it sends the DMAIC_CB5_RQ request to the IDMAC for fetching input video from the system memory. (In this case, the IC also sends DMAIC_CB5_NF indicating the first request for the frame).

   The IDMAC starts fetching data from the input video buffer and send it to the IC using DMA internal protocol ((see Figure 44-150). When IDMAC sends acknowledge to IC (DMAIC_CB5_ACK), it also generates the DMAIC_CB5_NFACK signal which indicate that the current request was accompanied with the new frame signal from the IC. This signal is sent to the FSU for buffer toggling when double buffer mode is set and also as an interrupt to the MCU.

   The FSU resets DMAIC_5_BUF0_RDY to indicate the MCU that this buffer is read. Meanwhile the MCU can prepare buffer 1 (in double buffer mode) or wait for end of frame (in single buffer mode).

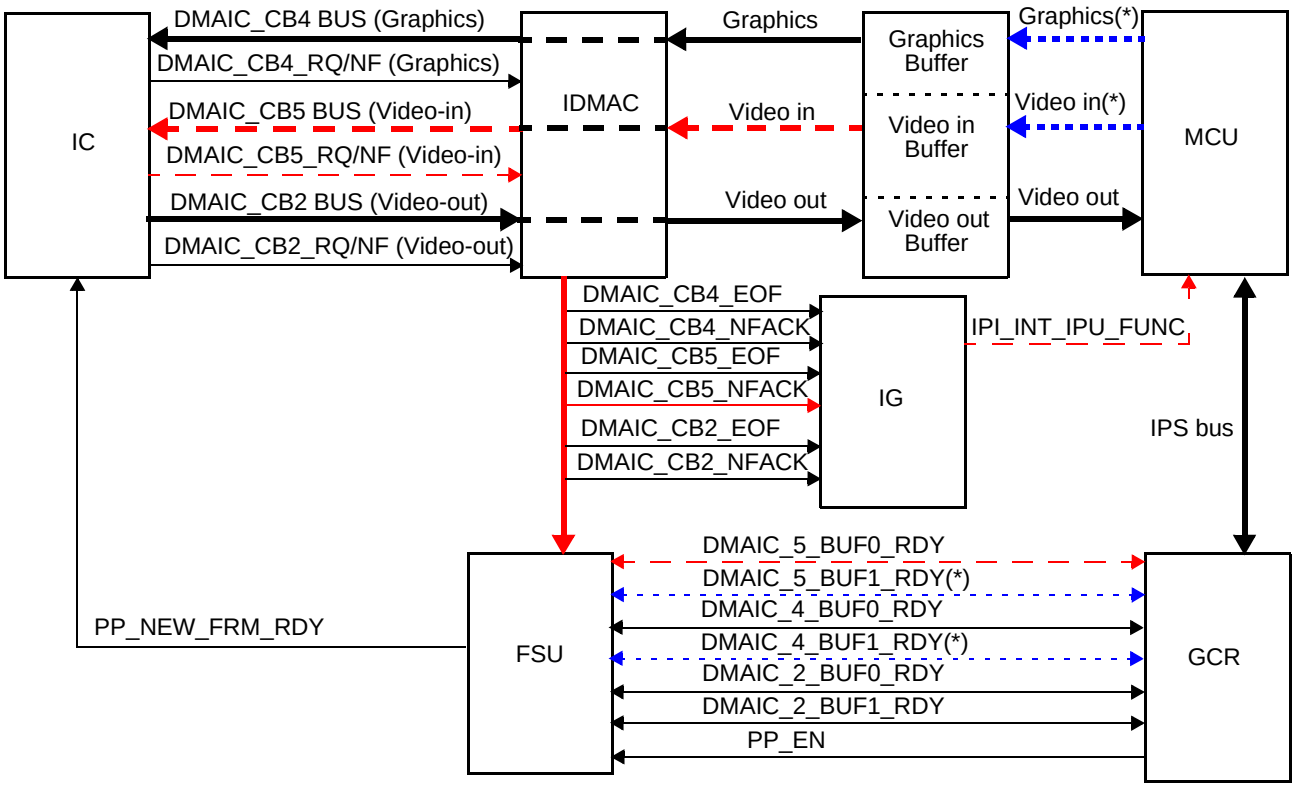


Figure 44-150. Operating Step—First Input Data Fetching

When the IC begins combining (if needs) it should receive graphics data from the DMA channel 4 (see Figure 44-151). The IC sends the request and the new frame signals. The IDMAC fetches graphics data from the input graphics buffer and send it to the IC. The IDMAC also sends DMAIC_CB4_NFACK which causes toggling in the FSU in double buffer mode. The FSU resets DMAIC_4_BUF0_RDY to indicate that the graphics buffer is read.

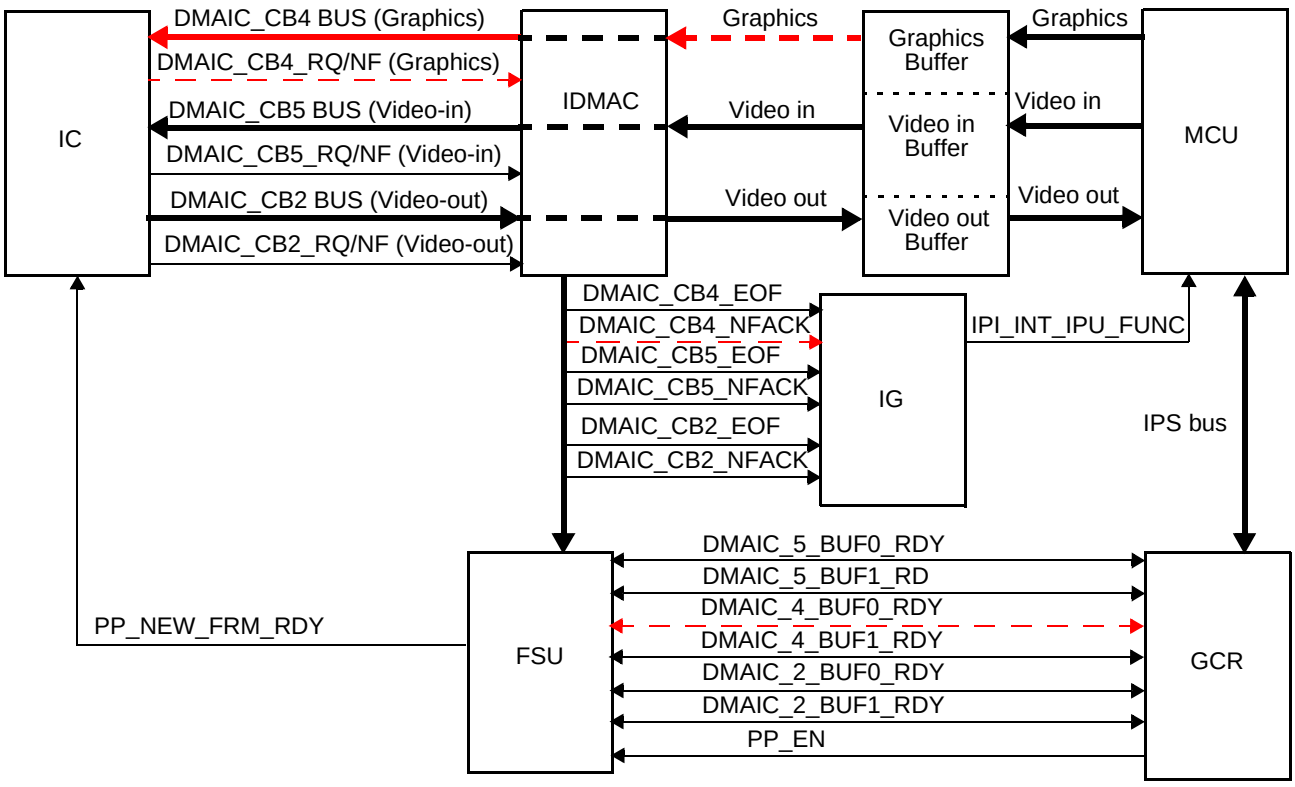


Figure 44-151. Operating Step - First Input Graphics Fetching

When the IC finishes processing the first burst (video and graphics) it send to the IDMAC the request and the new frame signal for output of the video buffer (see Figure 44-152). The IDMAC should transfer the burst to the video output buffer in the system memory. It sends DMAIC_CB2_NFACK to the FSU which causes toggling in the FSU in double buffer mode. The FSU resets DMAIC_2_BUF0_RDY to indicate that the output buffer is being written. The MCU should wait for finish of writing into output buffer (DMAIC_CB2_EOF) before reading it (*).

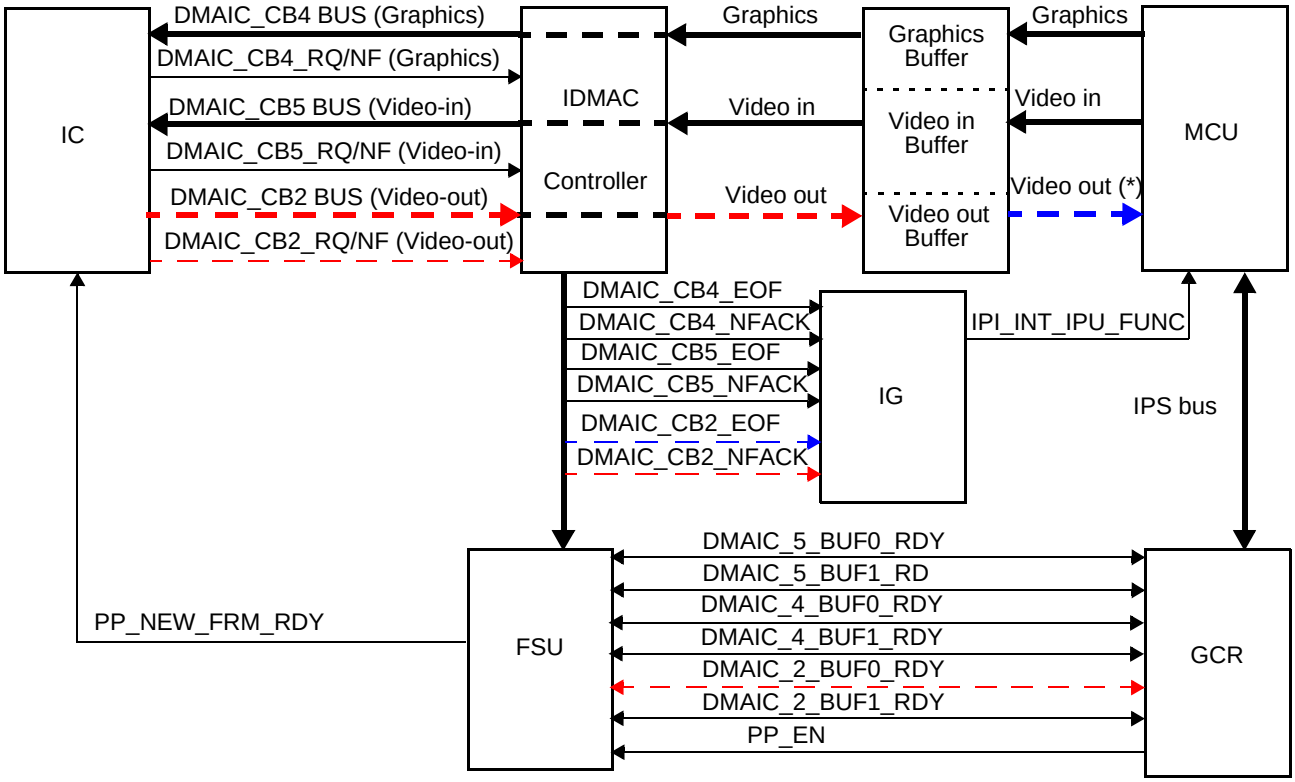


Figure 44-152. Operating Step—First Output Data Writing to the System Memory

When the IC continues processing the current frame, the last three steps are repeated without asserting the new frame signal and the FSU is not involved in the process.

When the whole input frame is read, the IDMAC sends the DMAIC_CB5_EOF signal and masks this channel. The same happens for other channels (graphics channel is masked after all graphics was read and output channel is masked after all output was written to the memory). The FSU monitors only the video input channel end of frame signal (DMAIC_CB5_EOF). When it arrives (also interrupt to the MCU is sent), the FSU checks if task is still enabled and if the next buffer is ready (BUF1 in double buffer mode and BUF0 in single buffer mode).

The FSU waits for readiness of the buffers. If the buffer is ready, the FSU sends PP_NEW_FRM_RDY to the IC and the whole process starts again. If task is not enabled, then the FSU returns to the idle state.

During current frame processing, the MCU should fill the next input graphics and video buffers, read the previous output buffer and then set the relevant frame ready bits for each channel. The MCU can re-configure parameters in the GCR and in parameter memory but only when the task is not active.

44.4.8.3 Interrupt Generator (IG)

The interrupt generator (IG) produces two interrupts to the MCU—the functional interrupt and the error interrupt. All the interrupts are maskable.

Table 44-166 and Table 44-167 describe both the functional interrupts and the error interrupts.

Table 44-166. Functional Interrupts Summary

Location	Submodule	Interrupt Status Bit Name	Description
IPU_INT_STAT_1	IDMAC	DMAIC_0_EOF	End of frame for IC DMA channel 0
		DMAIC_1_EOF	End of frame for IC DMA channel 1
		DMAIC_2_EOF	End of frame for IC DMA channel 2
		DMAIC_3_EOF	End of frame for IC DMA channel 3
		DMAIC_4_EOF	End of frame for IC DMA channel 4
		DMAIC_5_EOF	End of frame for IC DMA channel 5
		DMAIC_6_EOF	End of frame for IC DMA channel 6
		DMAIC_7_EOF	End of frame for IC DMA channel 7
		DMAIC_8_EOF	End of frame for IC DMA channel 8
		DMAIC_9_EOF	End of frame for IC DMA channel 9
		DMAIC_10_EOF	End of frame for IC DMA channel 10
		DMAIC_11_EOF	End of frame for IC DMA channel 11
		DMAIC_12_EOF	End of frame for IC DMA channel 12
		DMAIC_13_EOF	End of frame for IC DMA channel 13
		DMASDC_0_EOF	End of frame for SDC DMA channel 0
		DMASDC_1_EOF	End of frame for SDC DMA channel 1
		DMASDC_2_EOF	End of frame for SDC DMA channel 2
		DMASDC_3_EOF	End of frame for SDC DMA channel 3
		DMAADC_2_EOF	End of frame for ADC DMA channel 2
		DMAADC_3_EOF	End of frame for ADC DMA channel 3
		DMAADC_4_EOF	End of frame for ADC DMA channel 4
		DMAADC_5_EOF	End of frame for ADC DMA channel 5
		DMAADC_6_EOF	End of frame for ADC DMA channel 6
		DMAADC_7_EOF	End of frame for ADC DMA channel 7
		DMAPF_0_EOF	End of frame for PF DMA channel 0
		DMAPF_1_EOF	End of frame for PF DMA channel 1
		DMAPF_2_EOF	End of frame for PF DMA channel 2
		DMAPF_3_EOF	End of frame for PF DMA channel 3
		DMAPF_4_EOF	End of frame for PF DMA channel 4
		DMAPF_5_EOF	End of frame for PF DMA channel 5
		DMAPF_6_EOF	End of frame for PF DMA channel 6
		DMAPF_7_EOF	End of frame for PF DMA channel 7

Table 44-166. Functional Interrupts Summary (continued)

Location	Submodule	Interrupt Status Bit Name	Description
IPU_INT_STAT_2	IDMAC	DMAIC_0_NFACK	New frame for IC DMA channel 0
		DMAIC_1_NFACK	New frame for IC DMA channel 1
		DMAIC_2_NFACK	New frame for IC DMA channel 2
		DMAIC_3_NFACK	New frame for IC DMA channel 3
		DMAIC_4_NFACK	New frame for IC DMA channel 4
		DMAIC_5_NFACK	New frame for IC DMA channel 5
		DMAIC_6_NFACK	New frame for IC DMA channel 6
		DMAIC_7_NFACK	New frame for IC DMA channel 7
		DMAIC_8_NFACK	New frame for IC DMA channel 8
		DMAIC_9_NFACK	New frame for IC DMA channel 9
		DMAIC_10_NFACK	New frame for IC DMA channel 10
		DMAIC_11_NFACK	New frame for IC DMA channel 11
		DMAIC_12_NFACK	New frame for IC DMA channel 12
		DMAIC_13_NFACK	New frame for IC DMA channel 13
		DMASDC_0_NFACK	New frame for SDC DMA channel 0
		DMASDC_1_NFACK	New frame for SDC DMA channel 1
		DMASDC_2_NFACK	New frame for SDC DMA channel 2
		DMASDC_3_NFACK	New frame for SDC DMA channel 3
		DMAADC_2_NFACK	New frame for ADC DMA channel 2
		DMAADC_3_NFACK	New frame for ADC DMA channel 3
		DMAADC_4_NFACK	New frame for ADC DMA channel 4
		DMAADC_5_NFACK	New frame for ADC DMA channel 5
		DMAADC_6_NFACK	New frame for ADC DMA channel 6
		DMAADC_7_NFACK	New frame for ADC DMA channel 7
		DMAPF_0_NFACK	New frame for PF DMA channel 0
		DMAPF_1_NFACK	New frame for PF DMA channel 1
		DMAPF_2_NFACK	New frame for PF DMA channel 2
		DMAPF_3_NFACK	New frame for PF DMA channel 3
		DMAPF_4_NFACK	New frame for PF DMA channel 4
		DMAPF_5_NFACK	New frame for PF DMA channel 5
		DMAPF_6_NFACK	New frame for PF DMA channel 6
		DMAPF_7_NFACK	New frame for PF DMA channel 7

Table 44-166. Functional Interrupts Summary (continued)

Location	Submodule	Interrupt Status Bit Name	Description
IPU_INT_STAT_3	CM	BRK_RQ_STAT	Break request
		STOP_MODE_ACK_EN	Stop mode acknowledge
	SDC	SDC_BG_EOF	End of background frame on SDC output
		SDC_FG_EOF	End of foreground frame on SDC output
		SDC_MSK_EOF	End of mask frame on SDC output
		SDC_DISP3_VSYNC	VSYNC for display 3
	DI	SERIAL_DATA_FINISH	Data transfer finished in DI for low level access
	CSI	CSI_NF	New frame for CSI
		CSI_EOF	End of frame for CSI
	IDMAC	DMAIC_3_SBUF_END	Scroll buffer end for IC DMA channel 3
		DMAIC_4_SBUF_END	Scroll buffer end for IC DMA channel 4
		DMAIC_5_SBUF_END	Scroll buffer end for IC DMA channel 5
		DMAIC_6_SBUF_END	Scroll buffer end for IC DMA channel 6
		DMASDC_0_SBUF_END	Scroll buffer end for SDC DMA channel 0
		DMASDC_1_SBUF_END	Scroll buffer end for SDC DMA channel 1
		DMASDC_2_SBUF_END	Scroll buffer end for SDC DMA channel 2
		DMAADC_2_SBUF_END	Scroll buffer end for ADC DMA channel 2
		DMAADC_3_SBUF_END	Scroll buffer end for ADC DMA channel 3
	ADC	ADC_DISP0_VSYNC	VSYNC for display 0
		ADC_DISP12_VSYNC	VSYNC for displays 1 or 2
		ADC_PRP_EOF	End of frame for ADC PRP channel
		ADC_PP_EOF	End of frame for ADC PP channel
		ADC_SYS1_EOF	End of frame for ADC system 1 channel
		ADC_SYS2_EOF	End of frame for ADC system 2 channel

Table 44-167. Error Interrupts Summary

Location	Submodule	Interrupt Status Name	Description
IPU_INT_STAT_4	IDMAC	DMAIC_0_NFB4EOF_ERR	New frame before end of frame for IC DMA channel 0
		DMAIC_1_NFB4EOF_ERR	New frame before end of frame for IC DMA channel 1
		DMAIC_2_NFB4EOF_ERR	New frame before end of frame for IC DMA channel 2
		DMAIC_3_NFB4EOF_ERR	New frame before end of frame for IC DMA channel 3
		DMAIC_4_NFB4EOF_ERR	New frame before end of frame for IC DMA channel 4
		DMAIC_5_NFB4EOF_ERR	New frame before end of frame for IC DMA channel 5
		DMAIC_6_NFB4EOF_ERR	New frame before end of frame for IC DMA channel 6
		DMAIC_7_NFB4EOF_ERR	New frame before end of frame for IC DMA channel 7
		DMAIC_8_NFB4EOF_ERR	New frame before end of frame for IC DMA channel 8
		DMAIC_9_NFB4EOF_ERR	New frame before end of frame for IC DMA channel 9
		DMAIC_10_NFB4EOF_ERR	New frame before end of frame for IC DMA channel 10
		DMAIC_11_NFB4EOF_ERR	New frame before end of frame for IC DMA channel 11
		DMAIC_12_NFB4EOF_ERR	New frame before end of frame for IC DMA channel 12
		DMAIC_13_NFB4EOF_ERR	New frame before end of frame for IC DMA channel 13
		DMASDC_0_NFB4EOF_ERR	New frame before end of frame for SDC DMA channel 0
		DMASDC_1_NFB4EOF_ERR	New frame before end of frame for SDC DMA channel 1
		DMASDC_2_NFB4EOF_ERR	New frame before end of frame for SDC DMA channel 2
		DMASDC_3_NFB4EOF_ERR	New frame before end of frame for SDC DMA channel 3
		DMAADC_2_NFB4EOF_ERR	New frame before end of frame for ADC DMA channel 2
		DMAADC_3_NFB4EOF_ERR	New frame before end of frame for ADC DMA channel 3
		DMAADC_4_NFB4EOF_ERR	New frame before end of frame for ADC DMA channel 4
		DMAADC_5_NFB4EOF_ERR	New frame before end of frame for ADC DMA channel 5
		DMAADC_6_NFB4EOF_ERR	New frame before end of frame for ADC DMA channel 6
		DMAADC_7_NFB4EOF_ERR	New frame before end of frame for ADC DMA channel 7
		DMAPF_0_NFB4EOF_ERR	New frame before end of frame for PF DMA channel 0
		DMAPF_1_NFB4EOF_ERR	New frame before end of frame for PF DMA channel 1
		DMAPF_2_NFB4EOF_ERR	New frame before end of frame for PF DMA channel 2
		DMAPF_3_NFB4EOF_ERR	New frame before end of frame for PF DMA channel 3
		DMAPF_4_NFB4EOF_ERR	New frame before end of frame for PF DMA channel 4
		DMAPF_5_NFB4EOF_ERR	New frame before end of frame for PF DMA channel 5
		DMAPF_6_NFB4EOF_ERR	New frame before end of frame for PF DMA channel 6
		DMAPF_7_NFB4EOF_ERR	New frame before end of frame for PF DMA channel 7

Table 44-167. Error Interrupts Summary (continued)

Location	Submodule	Interrupt Status Name	Description
IPU_INT_STAT_5	IC	BAYER_BUF_OVF_ERR	Bayer buffer overflow
		ENC_BUF_OVF_ERR	Encoding buffer overflow
		VF_BUF_OVF_ERR	Viewfinder buffer overflow
	ADC	ADC_PP_TEARING_ERR	Tearing in ADC postprocessing channel
		ADC_SYS1_TEARING_ERR	Tearing in ADC system 1 channel
		ADC_SYS2_TEARING_ERR	Tearing in ADC system 2 channel
		SAHB_ADDR_ERR_EN	Slave AHB base address error
	SDC	SDC_BGD_ERR	Background data not arrived on time
		SDC_FGD_ERR	Foreground data not arrived on time
		SDC_MSKD_ERR	Mask data not arrived on time
	CM	BAYER_FRM_LOST_ERR	Bayer frame lost
		ENC_FRM_LOST_ERR	Encoding frame lost
		VF_FRM_LOST_ERR	View-finder frame lost
	DI	DI_ADC_LOCK_ERR	Display interface locked for ADC access while SDC access should begin
		DI_LLA_LOCK_ERR	Display interface locked for low level access while SDC access should begin
	IDMAC	AHB_M1_ERR	AHB master 1 error response
		AHB_M2_ERR	AHB master 2 error response

44.4.8.4 Debug Unit

The DU is able to break IPU data/control flows on a MCU request on an internal event. The DU is controlled by the IPU_BRK_CTRL_1, IPU_BRK_CTRL_2 and IPU_DIAGB_CTRL Registers. Debug status is reflected by the IPU_BRK_STAT Register.

The internal break event may correspond to a selected point in one of processed data frames or an internal signal or simultaneous fulfillment of both conditions. The DU contains an event counter that allows to perform break after given repetition number of the event.

The break data point is selected by definition of the following conditions:

- DMA channel number or CSI output.
- Frame row number in the IDMAC or in the CSI.
- Frame column number in the IDMAC or in the CSI.

Each of the break data point conditions can be masked independently on other conditions. The masked condition is considered as fulfilled and does not affect break.

Internal break signals are composed in five groups corresponding to IPU interrupts.

There are the following modes of entering debug:

1. IPU internal break forces the MCU to enter debug. MCU entering debug forces the IPU to enter debug.
2. IPU internal break forces the MCU to enter debug. MCU entering debug does not force the IPU to enter debug.
3. IPU internal break does not force the MCU to enter debug. MCU entering debug forces the IPU to enter debug.
4. IPU internal break does not force the MCU to enter debug. MCU entering debug does not force the IPU to enter debug.

Exiting debug mode can be done by setting the DBG_EXIT bit.

The IPU has two debug modes:

- DMA transfer stop without freezing of the HSP_CLK clock
- Freeze of the HSP_CLK clock.

In the first mode, all requests to the IDMAC are masked at the break event. The IDMAC completes current data transfers and stops. Other IPU submodules can continue operation for some time. They stop after filling or emptying the corresponding data FIFOs. The MCU can check IPU data buffers in the external memory for debug purposes. Proper IPU operation can be resumed by assertion the DBG_EXIT bit in the IPU_BRK_CTRL_1 Register.

In the second mode, all IPU clocks excluding clocks used for IP Skyblue bus registers, memories and the DU are stopped immediately. the MCU can read all internal IPU memories or monitor the DIAGB bus. Proper IPU operation cannot be resumed on exiting freeze mode.

Additionally, the DU provides real-time monitoring of IPU internal signals via the DIAGB bus, which is output to chip external pins. DIAGB signals are arranged in 19 groups (bus configurations). The MCU change selection of bus configuration during IPU operation.

44.4.8.5 General Configuration Registers

The GCR contains a set of control/status/data registers. It provides an IPU interface to the IPS bus. The IPG_CLK rate can be equal to or lower than the HSP_CLK rate. The detail description of the registers is found in Section 44.3, “Memory Map and Register Definition.”

44.4.8.6 Smart BIST

The SBIST performs memory BIST functions. Additionally, the SBIST supports MCU access to the IPU internal memories (see description of the IPU_IMA_ADDR and IPU_IMA_DATA registers in Section 44.3.3, “Register Descriptions”).

44.5 Initialization/Application Information

The following sections describe application information for managing and working with IPU.

44.5.1 IPU Management Flow

Figure 44-153 shows the typical IPU management flow. The main steps are described below in detail.

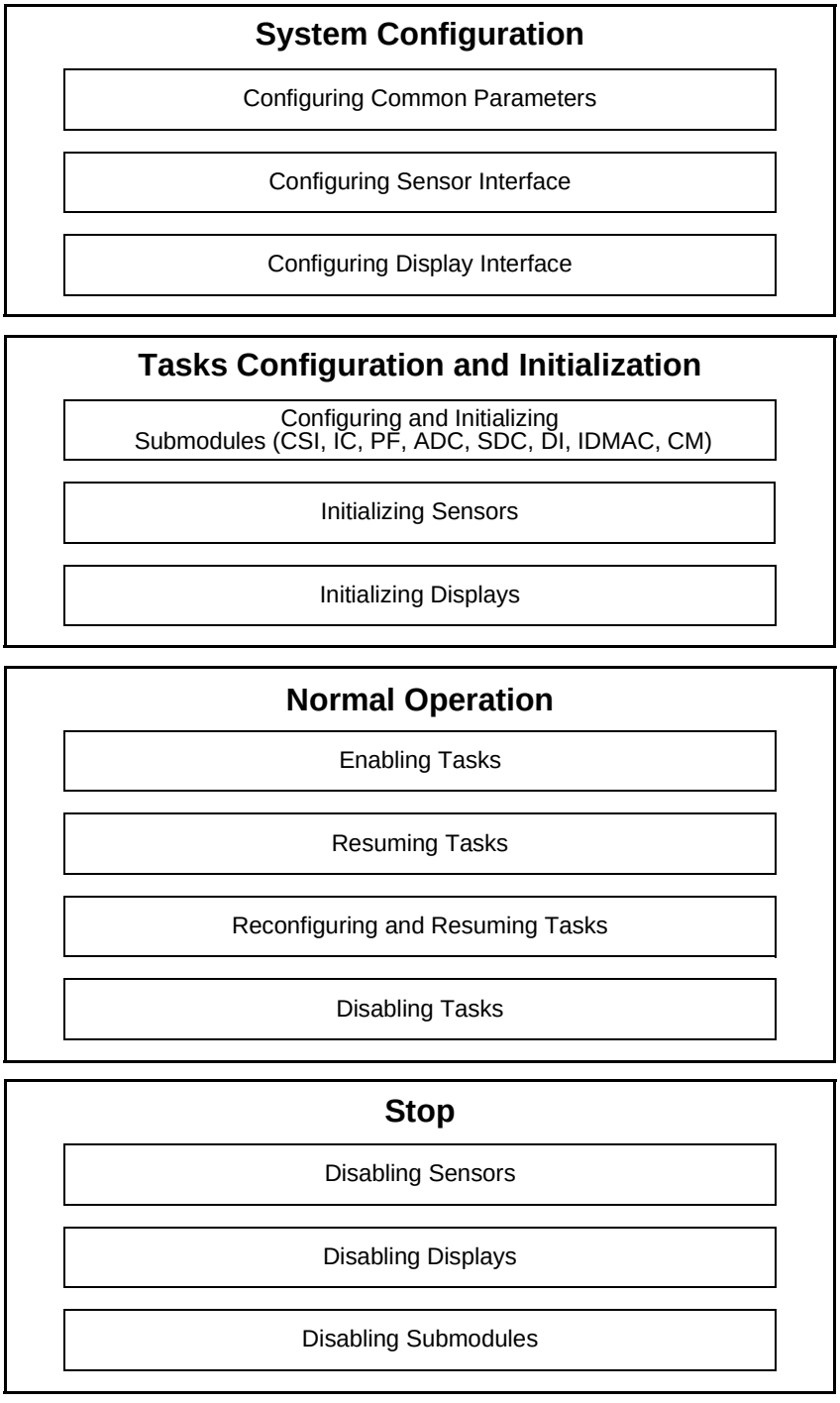


Figure 44-153. IPU Management Flow

44.5.2 System Configuration

This step is dedicated for static configuration of system parameters and interfaces. The IPU should be configured according to physical parameters of the connected devices such as display type, bus width, timings, etc. Typically, these configurations do not depend on the specific IPU tasks and applications.

44.5.3 Configuring Common Parameters

The following operations should be performed:

1. Select pixel Endianness in the IPU_CONF Register.
2. Select single or dual AHB mode in the IDMAC_CONF Register.

44.5.4 Configuring Sensor Interface

At this step, the CSI Registers are configured according to the parameters of the used camera sensor. The following operations should be performed:

1. Set the sensor configuration in the CSI_SENS_CONF Register.
2. Define the sensor frame size in the CSI_SENS_FRM_SIZE Registers.
3. Set BT.565 codes for the corresponding sensor type in the CSI_CCIR_CODE_1, CSI_CCIR_CODE_2 and CSI_CCIR_CODE_3 Registers.
4. Specify flash strobe parameters in the CSI_FLASH_STROBE_1 and CSI_FLASH_STROBE_2 Registers.

44.5.5 Configuring Display Interface

At this step, the corresponding ADC, SDC and DI Registers are programmed with the information that describes physical display characteristics. The following operations should be performed:

1. Define display interface modes in the DI_DISP_IF_CONF Register.
2. Configure display interface polarities in the DI_DISP_SIG_POL Register.
3. Configure serial interfaces in the DI_SER_DISP1_CONF and DI_SER_DISP2_CONF Register.
4. Configure display timings in the DI_DISP0_TIME_CONF_1 – DI_DISP0_TIME_CONF_3, DI_DISP1_TIME_CONF_1 – DI_DISP1_TIME_CONF_3, DI_DISP2_TIME_CONF_1 – DI_DISP2_TIME_CONF_3, and DI_DISP3_TIME_CONF Registers.
5. Configure display data bus mapping for the available displays in the DI_DISP#_DB#_MAP and DI_DISP#_CB#_MAP Register (see Section 44.4.6.4, “Bus Mapping Unit”).
6. Define the count of display clock cycles required to output/input one pixel for the available displays the DI_DISP_ACC_CC Register.
7. Configure the available display parameters in the ADC_DISP0_CONF, ADC_DISP1_CONF and ADC_DISP2_CONF Registers.
8. Define read patterns for the available displays in the ADC_DISP0_RD_AP, ADC_DISP0_RDM, ADC_DISP1_RD_AP, ADC_DISP1_RDM, and ADC_DISP2_RD_AP, ADC_DISP2_RDM Registers.

9. Set screen sizes of the available displays in the ADC_DISP0_SS, ADC_DISP12_SS Registers.
10. Configure vertical synchronization in the ADC_DISP_VSYNC Register.
11. Load the appropriate ADC templates according to Table 44-35. Loading is performed via the IPU_IMA_ADDR and IPU_IMA_DATA Registers.
12. Define the SDC parameters related to display static configuration in the SDC_COM_CONF Register.
13. Define horizontal and vertical dimensions of the synchronous display in the SDC_HOR_CONF and SDC_VER_CONF Registers.
14. Configure parameters of the Sharp display (if needed) in the SDC_SHARP_CONF_1 and SDC_SHARP_CONF_2 Registers.

44.5.6 Tasks Configuration and Initialization

Configuration and initialization of the IPU tasks includes programming dynamic parameters that are specific for the current use case. This step comprises also display and sensor initialization procedures (power up, initial programming etc.).

44.5.7 Configuring and Initializing Submodules

At this step, all IPU submodules (the CSI, the IC, the PF, the ADC, the SDC, the DI, the IDMAC and the common IPU Registers) are configured for the desired use case.

44.5.7.1 CSI Configuring

The following operations should be performed:

1. Define actual frame size in the CSI Actual Frame Size Register (CSI_ACT_FRM_SIZE).
2. Define output data properties in the CSI Output Frame Control Register (CSI_OUT_FRM_CTRL).

44.5.7.2 IC Configuring

The following operations should be performed:

1. Enable appropriate task options in the IC_CONF Register.
2. Define resizing ratios for each task in the IC_PRP_ENC_RSC, IC Preprocessing View-Finder Resizing Coefficients Register (IC_PRP_VF_RSC) and IC_PP_RSC Registers.
3. Define global alpha and key color values in the IC_CMBP_1 and IC_CMBP_2 Registers.
4. Load the appropriate IC task parameters according to Table 44-28. Loading is performed via the IPU_IMA_ADDR and IPU_IMA_DATA Registers.

44.5.7.3 PF Configuring

The following operations should be performed:

1. Set the postfilter type in the PF_CONF Registers.
2. For H.264 standard, enable pause and define pause row (if needed).

44.5.7.4 ADC Configuring

The following operations should be performed:

1. Define basic ADC channel parameters in the ADC_CONF Register.
2. Set start address and start time values for all used asynchronous display access channels in the ADC_SYSCHA1_SA, ADC_SYSCHA2_SA, ADC_PRPCHAN_SA and ADC_PPCHAN_SA Registers.
3. Configure the MCU access parameters in the ADC_DISP0_CONF, ADC_DISP1_CONF and ADC_DISP2_CONF Registers.

44.5.7.5 SDC Configuring

The following operations should be performed:

1. Define basic SDC parameters related to input data format and combining in the SDC_COM_CONF Register.
2. Define the global alpha and the key color in the SDC_GRAPH_WIND_CTRL Register.
3. Define foreground and background windows position in the SDC_FG_POS and SDC_BG_POS Registers.
4. Define cursor position, cursor blinking, cursor color and PWM parameters in the SDC_CUR_POS, SDC_CUR_BLINK_PWM_CTRL, and SDC_CUR_MAP Registers.

44.5.7.6 DI Configuring

For dynamic DI configuration, the initial HSP_CLK period should be written to the DI_HSP_CLK_PER Register. For typical use cases, all other configuration parameters are static They have to be programmed at the display interface configuration step (see Section 44.5.5, “Configuring Display Interface”).

44.5.7.7 IDMAC Configuring

The following operations should be performed:

1. Set a channel arbitration option in the IDMAC_CONF Register.
2. Set the DMA channel priorities in the IDMAC_CHA_EN Register.
3. Load the appropriate IDMAC channel parameters according to Table 44-29 through Table 44-33. Loading is performed via the IPU_IMA_ADDR and IPU_IMA_DATA Registers.
4. Load the appropriate IDMAC decoding look-up-table according to Table 44-34. Loading is performed via the IPU_IMA_ADDR and IPU_IMA_DATA Registers.

44.5.7.8 CM Configuring

At this step, frame synchronization chains and interrupts are programmed. The following operations should be performed:

1. Select single or double buffer mode for each used DMA channel in the IPU_CHA_DB_MODE_SEL Register.

2. Clear current buffer number bits for each used DMA channel in the IPU_CHA_CUR_BUF Register (by writing one to the corresponding bits).
3. Configure tasks chaining in the IPU_FS_PROC_FLOW and IPU_FS_DISP_FLOW Registers according to Table 44-164 and Table 44-165 (see Section 44.4.8.2, “Frame Synchronization Unit”).
4. Enable appropriate interrupts in the IPU_INT_CTRL_1, IPU_INT_CTRL_2, IPU_INT_CTRL_3, IPU_INT_CTRL_4, and IPU_INT_CTRL_5 Registers.

44.5.8 Initializing Sensors

This step includes powering up and programming the camera sensors connected to the IPU. The IPU does not control the sensors. The I^2C interface of the chip is used for this purpose.

44.5.9 Initializing Displays

This step includes powering up and programming the displays connected to the IPU. The IPU cannot manage power-up sequence of the displays. This operation should be done by MCU software via GPIOs of the chip or by an external power management chip. The IPU provides configuring and initializing smart displays via the asynchronous interface. The following operations should be performed to initialize a smart display:

1. Enable the DI in the IPU_CONF Register.
2. Program the display using the low-level access DI_DISP_LLA_CONF and DI_DISP_LLA_DATA Registers.

Programming the display should be synchronized with control of the display via the GPIOs.

44.5.10 Normal Operation

At this step, the IPU executes the programmed tasks.

44.5.11 Enabling Tasks

The following operations should be performed (the order of the operations must be kept)

1. Enable the corresponding submodules in the IPU_CONF Register.
2. Enable the required IDMAC channels in the IDMAC_CHA_EN Register.
3. Enable the required IC tasks in the IC_CONF Register.
4. Fill video and graphics IPU input buffers 0 in the system memory and set the corresponding buffer ready bits in the IPU_CHA_BUF0_RDY Register. The enabled tasks start (see Section 44.4.8.2.2, “Frame Synchronization Flow”).
5. Enable the background and (optionally) foreground plane in the SDC_COM_CONF Register.

44.5.12 Resuming Tasks

Some IPU tasks are resumed automatically (without MCU software involvement). For example, the viewfinder task is resumed whenever a new sensor frame arrives. For other tasks (for example, postprocessing), the MCU has to resume the task on completion of previous frame processing. The following operations should be performed in this case:

1. On reception of the end of frame interrupt from one of running tasks, read output data from the corresponding buffer and set the corresponding buffer ready bits in the IPU_CHA_BUF0_RDY or IPU_CHA_BUF1_RDY Registers. The IPU_INT_STAT_1 - IPU_INT_STAT_4 Registers indicate what is the interrupt source.
2. If required, enable flash strobe generation at the desired sensor frame using the end of frame interrupt of the previous frame.
3. Fill video and graphics IPU input buffers which new data and set the corresponding buffer ready bits in the IPU_CHA_BUF0_RDY or IPU_CHA_BUF1_RDY Registers.

For more detailed description of MCU and IPU interaction, see Section 44.4.8.2.2, “Frame Synchronization Flow” and Section 44.4.8.2.3, “Frame Synchronization Example.”

44.5.13 Reconfiguring and Resuming Tasks

To reconfigure the task, the task has to be firstly disabled. After the corresponding busy bits in the IDMAC_CHA_BUSY Register will be no active, the task or the channel can be re-programmed. For DMA channels, re-programming base address of non-active memory buffer is allowed at any time. For the SDC, position of the planes and cursor parameters can be also re-programmed at any time. The functions for task re-configuring are similar to those described in Section 44.5.7, “Configuring and Initializing Submodules.” The MCU can re-program one of the HSP_CLK period values (unused in this time) in the DI_HSP_CLK_PER Register.

44.5.14 Disabling Tasks

The following operations should be performed:

1. Disable the required IC tasks in the IC_CONF Register. The tasks will stop on completion the current frame.
2. Do not set the corresponding buffer ready bits in the IPU_CHA_BUF0_RDY Register. The enabled tasks start will stop on completion of processing the current frame.
3. Disable the background and (optionally) foreground plane in the SDC_COM_CONF Register.
4. Disable the corresponding IDMAC channels in the IDMAC_CHA_EN Register.

44.5.15 Disabling Sensors

This step includes powering down the camera sensors connected to the IPU. The IPU does not control the sensors. The I^2C interface of the chip is used for this purpose.

44.5.16 Disabling Displays

This step includes disabling and powering down displays connected to the IPU. The IPU cannot manage power-down sequence of the displays. This operation should be done by MCU software via GPIOs of the chip or by an external power management chip. The IPU can only provide disabling smart displays via the asynchronous interface using the low-level access DI_DISP_LLA_CONF and DI_DISP_LLA_DATA Registers. This operation should be synchronized with control of the display via the GPIOs.

44.5.17 Disabling Submodules

To disable the IPU submodules, the corresponding bits should be cleared in the IPU_CONF Register.

44.6 Internal Memories Mapping

44.6.1 IC Memories

44.6.1.1 Input Buffer Memory

Table 44-168, Table 44-169, and Table 44-170 show the mapping of the Input Buffer Memory.

Table 44-168. Input Buffer Memory Mapping for 8-Bit YUV or Generic Sensor Output

Start Address				63				32	31			0
x0	CSI output FIFO	Page 0	word 0	Y_1/D_7	U_1/D_6	V_1/D_5	X/D_4		Y_0/D_3	U_0/D_2	V_0/D_1	X/D_0
			word 1									
			word 2									
			word 3									
		Page 1										
		. . .						. . . .				
		Page 7										

Table 44-168. Input Buffer Memory Mapping for 8-Bit YUV or Generic Sensor Output (continued)

Start Address				63				32	31			0
x20	Pre-processs-sing IDMAC output FIFO	Page 0	word 0	Y_1	U_1	V_1	X		Y_0	U_0	V_0	X
			word 1									
			word 2									
			word 3									
		Page 1										
		. . .		. . . .								
		Page 7										
x40	Post-processs-sing IDMAC output FIFO	Page 0	word 0	Y_1	U_1	V_1	X		Y_0	U_0	V_0	X
			word 1									
			word 2									
			word 3									
		Page 1										
		. . .		. . . .								
		Page 7										

Table 44-169. Input Buffer Memory Mapping for 16-Bit Generic Sensor Output

Start Address				63				30	29			0
x0	CSI output FIFO	Page 0	word 0	D_3				D_2	D_1			D_0
			word 1									
			word 2									
			word 3									
		Page 1										
		. . .						. . . .				
		Page 3										
x20	PrP IDMAC output FIFO	Page 0	word 0	Y_1	U_1	V_1	X	Y_0	U_0	V_0	X	
			word 1									
			word 2									
			word 3									
		Page 1										
		. . .						. . . .				
		Page 7										
x40	PP IDMAC output FIFO	Page 0	word 0	Y_1	U_1	V_1	X	Y_0	U_0	V_0	X	
			word 1									
			word 2									
			word 3									
		Page 1										
		. . .						. . . .				
		Page 7										

Table 44-170. Input Buffer Memory Mapping for 10-bit RGB Sensor Output

Start address				63							30	29							0
x0	CSI output FIFO	Page 0	word 0	R_1			G_1			B_1	X	R_0			G_0			B_0	X
			word 1																
			word 2																
			word 3																
		Page 1																	
		. . .		. . . .															
		Page 3																	
x20	PrP IDMAC output FIFO	Page 0	word 0	Y_1			U_1			V_1	X	Y_0			U_0			V_0	X
			word 1																
			word 2																
			word 3																
		Page 1																	
		. . .		. . . .															
		Page 7																	
x40	PP IDMAC output FIFO	Page 0	word 0	Y_1			U_1			V_1	X	Y_0			U_0			V_0	X
			word 1																
			word 2																
			word 3																
		Page 1																	
		. . .		. . . .															
		Page 7																	

44.6.1.2 Downsizing Temporary Memory

Table 44-171 shows mapping of the Downsizing Temporary Memory.

Table 44-171. Downsizing Temporary Memory Mapping

Start Address				35		0
x0	Task 1 temporary buffer	Page 0	word 0	Y_0	U_0	V_0
			... ...	... ...		
			word 719			
x2D0	Task 2 temporary buffer	Page 0	word 0	Y_0	U_0	V_0
			... ...	... ...		
			word 719			
x5A0	Task 3 temporary buffer	Page 0	word 0	Y_0	U_0	V_0
			... ...	... ...		
			word 719			

44.6.1.3 Downsizing Output Memory

Table 44-172 shows mapping of the Downsizing Output Memory.

Table 44-172. Downsizing Output Memory Mapping

Start Address				47			24	23		0
x0	Task 1 output buffer	Row 0	word 0	Y_1	U_1	V_1	Y_0	U_0	V_0	
			... ...	... ...						
			word 359							
		Row 1								

Table 44-172. Downsizing Output Memory Mapping (continued)

Start Address				47			24	23		0
x2D0	Task 2 output buffer	Row 0	word 0	Y_1	U_1	V_1	Y_0	U_0	V_0	
			... ...	... ...						
			word 359							
		Row 1								
x5A0	Task 3 output buffer	Row 0	word 0	Y_1	U_1	V_1	Y_0	U_0	V_0	
			... ...	... ...						
			word 359							
		Row 1								

44.6.1.4 Task Parameter Memory Mapping

Table 44-173 shows mapping of the Task Parameter Memory.

Table 44-173. Task Parameter Memory Mapping

Start Address				47			24	23		0
x0	Task 1 temporary data	Row 0	word 0	Y_1	U_1	V_1	Y_0	U_0	V_0	
			... ...	... ...						
			word 359							
		Row 1								
x2D0	Task 1 temp param	word 0		X			FRH	RSC		
x2D1	Task 1 color conversion matrix 1	word 0		X	M	S	A_0	C_{00}	C_{11}	C_{22}
		word 1		X			A_1	C_{01}	C_{10}	C_{20}
		word 2		X			A_2	C_{02}	C_{12}	C_{21}

Table 44-173. Task Parameter Memory Mapping (continued)

Start Address				47				24	23		0
x2D4	Task 2 temporary data	Row 0	word 0	Y_1	U_1	V_1	Y_0	U_0	V_0		
			... ...	... ...							
			word 359								
		Row 1									
x5A4	Task 2 temp param	word 0		X					FRH	RSC	
x5A5	Task 2 color conversion matrix 1	word 0		X	M	S	A_0		C_{00}	C_{11}	C_{22}
		word 1		X			A_1		C_{01}	C_{10}	C_{20}
		word 2		X			A_2		C_{02}	C_{12}	C_{21}
x5A8	Task 2 color conversion matrix 2:	word 0		X	M	S	A_0		C_{00}	C_{11}	C_{22}
		word 1		X			A_1		C_{01}	C_{10}	C_{20}
		word 2		X			A_2		C_{02}	C_{12}	C_{21}
x5AB	Task 3 temporary data	Row 0	word 0	Y_1	U_1	V_1	Y_0	U_0	V_0		
			... ...	... ...							
			word 359								
		Row 1									
x87B	Task 3 temp param	word 0		X					FRH	RSC	
x87C	Task 3 color conversion matrix 1	word 0		X	M	S	A_0		C_{00}	C_{11}	C_{22}
		word 1		X			A_1		C_{01}	C_{10}	C_{20}
		word 2		X			A_2		C_{02}	C_{12}	C_{21}
x87F	Task 3 color conversion matrix 2	word 0		X	M	S	A_0		C_{00}	C_{11}	C_{22}
		word 1		X			A_1		C_{01}	C_{10}	C_{20}
		word 2		X			A_2		C_{02}	C_{12}	C_{21}

44.6.1.5 Main Processing Memory

Table 44-174 shows mapping of the Main Processing Memory.

Table 44-174. Main Processing Memory Mapping

Start Address				63				32	31			0
x0	Task 1 output FIFO	Page 0	word 0	Y_1/R_1	U_1/G_1	V_1/B_1	X	Y_0/R_0	U_0/G_0	V_0/B_0	X	
			word 1									
			word 2									
			word 3									
		Page 1										
		. . .		. . . .								
		Page 7										
x20	Task 2 output FIFO	Page 0	word 0	Y_1/R_1	U_1/G_1	V_1/B_1	X	Y_0/R_0	U_0/G_0	V_0/B_0	X	
			word 1									
			word 2									
			word 3									
		Page 1										
		. . .		. . . .								
		Page 7										
x40	Task 2 graphics Input FIFO	Page 0	word 0	R_1	G_1	B_1	A_1	R_0	G_0	B_0	A_0	
			word 1									
			word 2									
			word 3									
		Page 1										
		. . .		. . . .								
		Page 7										

Table 44-174. Main Processing Memory Mapping (continued)

Start Address				63				32 31			0
x60	Task 3 output FIFO	Page 0	word 0	Y_1/R_1	U_1/G_1	V_1/B_1	X	Y_0/R_0	U_0/G_0	V_0/B_0	X
			word 1								
			word 2								
			word 3								
		Page 1									
		. . .		. . . .							
		Page 7									
x80	Task 3 graphics Input FIFO	Page 0	word 0	R_1	G_1	B_1	A_1	R_0	G_0	B_0	A_0
			word 1								
			word 2								
			word 3								
		Page 1									
		. . .		. . . .							
		Page 7									

44.6.1.6 Rotation Memory

Table 44-175 shows mapping of the Rotation Memory.

Table 44-175. Rotation Memory Mapping

Start Address			63				32 31			0
x0	input FIFO	Page 0	word 0	Y_{01}/R_{01}	U_{01}/G_{01}	V_{01}/B_{01}	X	Y_{00}/R_{00}	U_{00}/G_{00}	V_{00}/B_{00}	X
			... ...	... ...							
			word 31								

Table 44-175. Rotation Memory Mapping (continued)

Start Address				63				32	31			0
x20	output FIFO	Page 0	word 0	Y_1/R_1	U_1/G_1	V_1/B_1	X		Y_0/R_0	U_0/G_0	V_0/B_0	X
			word 1									
			word 2									
			word 3									
		Page 1										
		Page 2										
		Page 3										

44.6.2 Post Filter (PF) Memories

44.6.2.1 Memory for MPEG-4

Table 44-176 shows mapping of the post-filter Memory for MPEG-4.

Table 44-176. Post-filter Memory Mapping for MPEG-4

Start Address					63							32	31						0
x0	Buffer A	Page 0	Row 0	word 0	$Y_7/U_7/V_7$	$Y_6/U_6/V_6$	$Y_5/U_5/V_5$	$Y_4/U_4/V_4$	$Y_3/U_3/V_3$	$Y_2/U_2/V_2$	$Y_1/U_1/V_1$	$Y_0/U_0/V_0$							
				word 1															
				word 2															
				word 3															
			Row 1																
			. . .																
			Row 20																
		Page 1																	

Table 44-176. Post-filter Memory Mapping for MPEG-4 (continued)

Start Address				63							32	31				0
xA8	Buffer B	Page 0		word 0	$Y_7/U_7/V_7$	$Y_6/U_6/V_6$	$Y_5/U_5/V_5$	$Y_4/U_4/V_4$	$Y_3/U_3/V_3$	$Y_2/U_2/V_2$	$Y_1/U_1/V_1$	$Y_0/U_0/V_0$				
				word 1												
				⋮												
				word 35												
xCC	Buffer C	Page 0		word 0	$Y_7/U_7/V_7$	$Y_6/U_6/V_6$	$Y_5/U_5/V_5$	$Y_4/U_4/V_4$	$Y_3/U_3/V_3$	$Y_2/U_2/V_2$	$Y_1/U_1/V_1$	$Y_0/U_0/V_0$				
				word 1												
				⋮												
				word 35												
xF0	Buffer D	Page 0	Row 0	word 0	$Y_7/U_7/V_7$	$Y_6/U_6/V_6$	$Y_5/U_5/V_5$	$Y_4/U_4/V_4$	$Y_3/U_3/V_3$	$Y_2/U_2/V_2$	$Y_1/U_1/V_1$	$Y_0/U_0/V_0$				
				word 1												
				word 2												
				word 3												
			Row 1													
			Row 2													
			Row 3													
x100	Buffer E	Page 0		word 0	X								X	QP_3	QP_2	QP_1	QP_0
				word 1												
				word 2												
				word 3												

44.6.2.2 Memory for H264

Table 44-177 shows mapping of the post-filter memory for H264.

Table 44-177. Post-filter Memory Mapping for H264

Start Address					63						32	31						0
x0	Buffer A	Page 0	Row 0	word 0	$Y_7/U_7/V_7$		$Y_6/U_6/V_6$		$Y_5/U_5/V_5$		$Y_4/U_4/V_4$	$Y_3/U_3/V_3$		$Y_2/U_2/V_2$		$Y_1/U_1/V_1$		$Y_0/U_0/V_0$
				word 1														
				word 2														
				word 3														
			Row 1															
			. . .															
			Row 19															
		Page 1																
xA0	Buffer B	Page 0		word 0	X	FOB_1	X	FOA_1	X	QPC_1	X	QP_1	X	FOB_0	X	FOA_0	X	QPC_0	X	QP_0
				word 1														
				word 2														
				word 3														
		Page 1																
xA8	Buffer C	Page 0		word 0	BS_7		BS_6		BS_5		BS_4	BS_3		BS_2		BS_1		BS_0
				word 7														
		Page 1																

44.6.3 Synchronous Display Controller (SDC) Memories

Table 44-178 shows mapping of the SDC Buffer Memory.

Table 44-178. SDC Buffer Memory Mapping

Start Address				63							32	31							0
x0	Fore-ground FIFO	Page 0	word 0	$Y_1/R_1/D_7$		$U_1/G_1/D_6$		$V_1/B_1/D_5$		$A_1/A_1/D_4$		$Y_0/R_0/D_3$		$U_0/G_0/D_2$		$V_0/B_0/D_1$		$A_0/A_0/D_0$	
			word 1																
			⋮																⋮
			word 7																
		. . .									. . . .								
		Page 7																	
x40	Back-ground FIFO	Page 0	word 0	$Y_1/R_1/D_7$		$U_1/G_1/D_6$		$V_1/B_1/D_5$		$A_1/A_1/D_4$		$Y_0/R_0/D_3$		$U_0/G_0/D_2$		$V_0/B_0/D_1$		$A_0/A_0/D_0$	
			word 1																
			⋮																⋮
			word 7																
		. . .									. . . .								
		Page 7																	

44.6.4 Asynchronous Display Controller (ADC) Memories

44.6.4.1 Buffer Memory

Table 44-179 shows mapping of the ADC Buffer Memory.

Table 44-179. ADC Buffer Memory Mapping

Start Address				63				32	31				0
x0	System FIFO 1	Page 0	word 0	$R_1/D_{31}/X$	$G_1/D_{30}/D_{1_2}$	$B_1/D_{21}/D_{1_1}$	$X/D_{20}/D_{10}$		$R_0/D_{11}/X$	$G_0/D_{10}/D_{0_2}$	$B_0/D_{01}/D_{0_1}$	$X/D_{00}/D_{00}$	
			word 1										
			⋮										⋮
			word 7										
		Page 1											
		. . .						. . . .					
		Page 7											
x40	System FIFO 2	Page 0	word 0	$R_1/D_{31}/X$	$G_1/D_{30}/D_{1_2}$	$B_1/D_{21}/D_{1_1}$	$X/D_{20}/D_{10}$		$R_0/D_{11}/X$	$G_0/D_{10}/D_{0_2}$	$B_0/D_{01}/D_{0_1}$	$X/D_{00}/D_{00}$	
			word 1										
			⋮										⋮
			word 7										
		Page 1											
		. . .						. . . .					
		Page 7											

Table 44-179. ADC Buffer Memory Mapping (continued)

Start Address				63 — 32	31 — 0
x80	Command FIFO 1	Page 0	word 0	C_3/C_{11}	C_2/C_{10}	C_1/C_{01}	C_0/C_{00}
			word 1		
			⋮		⋮
			word 7		
		Page 1			
		. . .		. . . .	
		Page 7			
xC0	Command FIFO 2	Page 0	word 0	C_3/C_{11}	C_2/C_{10}	C_1/C_{01}	C_0/C_{00}
			word 1		
			⋮		⋮
			word 7		
		Page 1			
		. . .		. . . .	
		Page 7			

Table 44-179. ADC Buffer Memory Mapping (continued)

Start Address				63				32	31			0
x100	PrP FIFO	Page 0	word 0	R_1	G_1	B_1	X		R_0	G_0	B_0	X
			word 1									
			⋮									⋮
			word 7									
		Page 1										
		. . .						. . . .				
		Page 7										
x140	PP FIFO	Page 0	word 0	R_1	G_1	B_1	X		R_0	G_0	B_0	X
			word 1									
			⋮									⋮
			word 7									
		Page 1										
		. . .										
		Page 7										

Table 44-179. ADC Buffer Memory Mapping (continued)

Start Address				63			32	31			0
x180	MCU FIFO	Page 0	word 0	$R_1/D_{31}/X$	$G_1/D_{30}/D_{12}$	$B_1/D_{21}/D_{11}$	$X/D_{20}/D_{10}$	$R_0/D_{11}/X$	$G_0/D_{10}/D_{02}$	$B_0/D_{01}/D_{01}$	$X/D_{00}/D_{00}$
			word 1								
			⋮								⋮
			word 7								
		Page 1									
		. . .					. . . .				
		Page 7									

44.6.4.2 Template Memory

Table 44-180 shows mapping of the ADC Template Memory.

Table 44-180. ADC Template Memory Mapping

Start Address			29				0
x0	Write template for display 0	word 0	RS	FC	OC	D	
		word 1					
		⋮					
		word 31					
x20	Read template for display 0	word 0	RS	FC	OC	D	
		word 1					
		⋮					
		word 31					

Table 44-180. ADC Template Memory Mapping (continued)

Start Address			29				0
x40	Write template for display 1	word 0	RS	FC	OC	D	
		word 1					
		⋮					
		word 31					
x60	Read template for display 1	word 0	RS	FC	OC	C	
		word 1					
		⋮					
		word 31					
x80	Write template for display 2	word 0	RS	FC	OC	C	
		word 1					
		⋮					
		word 31					
xA0	Read template for display 2	word 0	RS	FC	OC	C	
		word 1					
		⋮					
		word 31					

44.6.5 Image DMA Controller (IDMAC) Memories

44.6.5.1 IDMAC Channel Parameter Memory

Table 44-182 shows mapping of the IDMAC Channel Parameter Memory.

Table 44-181. Channel Parameter Memory Mapping

Start Address	131		0
x0	Channel 0	word 0	P_1
		word 1	P_2
	Channel 1		
	⋮		⋮
	Channel 31		

44.6.5.2 IDMAC Y Dual-Port FIFO

Table 44-182 shows mapping of the Y Dual-Port FIFO.

Table 44-182. Y Dual-Port FIFO Mapping

Start address		31		0
x0	Data FIFO	Page 0	word 0	
			word 1	
			⋮	⋮
			word 8	
		Page 1		

44.6.5.3 IDMAC U/V Dual-Port FIFO

Table 44-183 shows mapping of the U/V Dual -Port FIFO.

Table 44-183. U/V Dual Port FIFO Mapping

Start Address		31		0
x0	Data FIFO	Page 0	word 0	
			word 1	
			word 2	
			word 3	
		Page 1		

44.6.5.4 IDCMA Decoding Look-Up Table Memory

Table 44-184 shows mapping of the Decoding Look-Up-Table Memory.

Table 44-184. Decoding Look-Up-Table Memory Mapping

Start Address		31					0
x0	Look-up-table	Page 0	word 0	R_0	G_0	B_0	A_0
			word 1				
			⋮				
			word 255				

Chapter 45
Synchronous Serial Interface (SSI)

The Synchronous Serial Interface (SSI) is a full-duplex serial port that enables the chip to communicate with a variety of serial devices. These serial devices can be standard coder-decoder (CODECs), Digital Signal Processors (DSPs), microprocessors, peripherals, and popular industry audio CODECs that implement the inter-IC sound bus standard (I^2S) and Intel AC97 standard.

Figure 45-1 shows the block diagram for the SSI. It consists of control registers to set up the port, a status register, separate transmit and receive circuits with FIFO registers, and a separate serial clock and frame sync generation for the transmit and receive sections. The second set of transmit and receive FIFOs replicate the logic used for the first set of FIFOs.

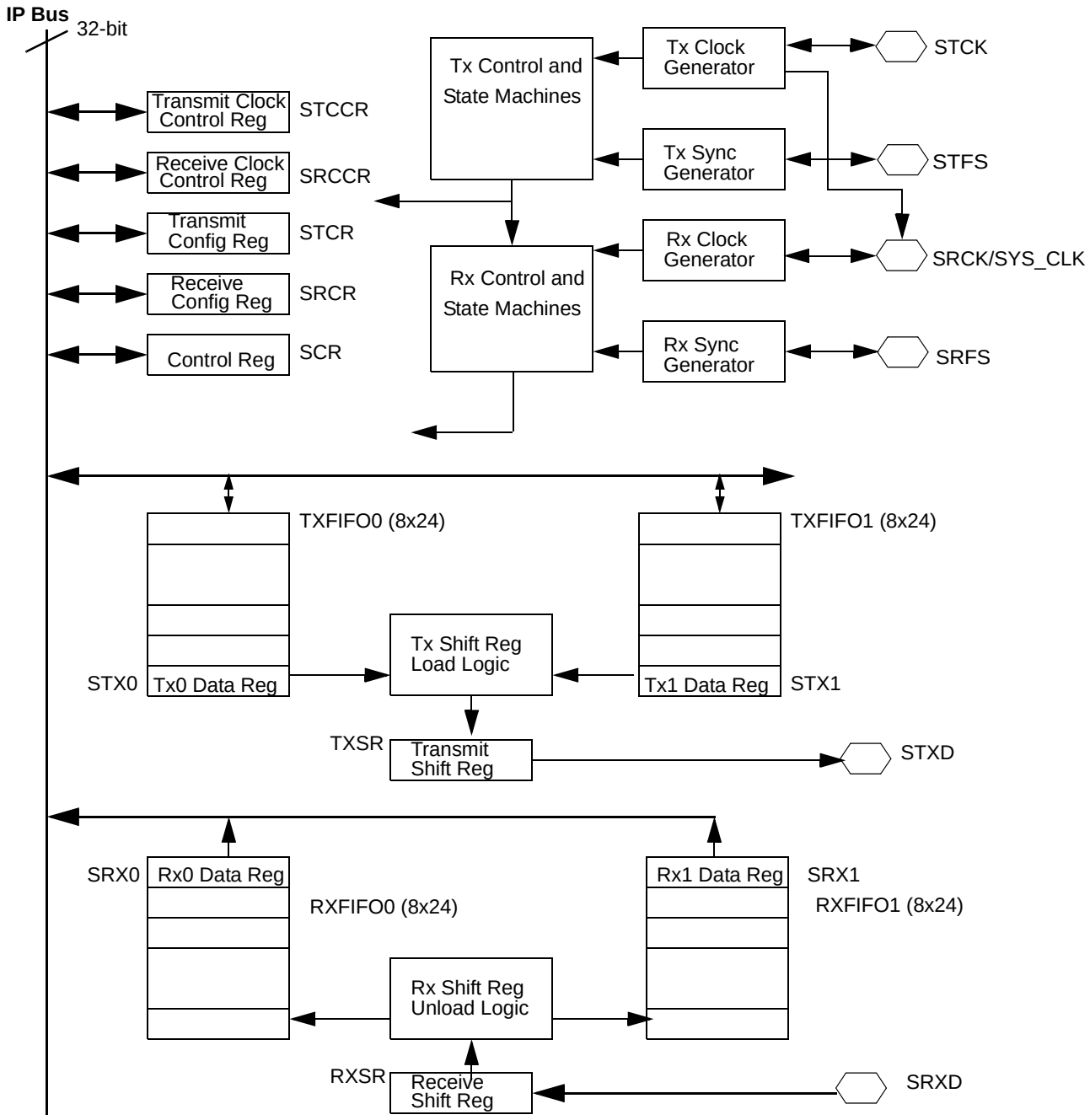


Figure 45-1. SSI Block Diagram

45.1 Overview

The SSI is typically used to transfer samples in a periodic manner. It consists of independent transmitter and receiver sections with independent clock generation and frame synchronization.

This chapter presents the SSI, and discusses its architecture, programming model, operating modes, and initialization.

45.1.1 Features

The SSI includes the following features:

- Independent (asynchronous) or shared (synchronous) transmit and receive sections with separate or shared internal/external clocks and frame syncs, operating in master or slave mode
- Normal mode operation using frame sync
- Network mode operation allowing multiple devices to share the port with as many as thirty-two time slots
- Gated clock mode operation requiring no frame sync
- Two sets of transmit and receive FIFOs. Each of the four FIFOs is 8x24 bits. The two sets of tx/rx FIFOs can be used in network mode to provide 2 independent channels for transmission and reception
- Programmable data interface modes such as I^2S, LSB, MSB aligned
- Programmable word length (8, 10, 12, 16, 18, 20, 22, or 24 bits)
- Programmable options for frame sync and clock generation
- Programmable I^2S modes (master, slave or normal). Oversampling clock, ccm_ssi_clk available as output from SRCK in I^2S master mode
- AC97 support
- Completely separate clock and frame sync selections for the receive and transmit sections. In AC97 standard, the clock is taken from an external source and frame sync is generated internally.
- External ccm_ssi_clk input for use in I^2S master mode. Programmable oversampling clock (SYS_CLK/ccm_ssi_clk) of the sampling frequency available as output in master mode at SRCK, when operated in sync mode.
- Programmable internal clock divider
- Time slot mask registers for reduced CPU overhead (for both transmit and receive)
- SSI power-down feature
- Programmable wait states for CPU accesses
- IP interface for register accesses, compliant to SRS 3.0.2 standard

45.1.2 Modes of Operation

SSI has the following basic operating modes:

- Normal
  - — Asynchronous protocol
  - — Synchronous protocol
- Network
  - — Asynchronous protocol
  - — Synchronous protocol
- Gated clock
  - — Synchronous protocol only

These modes can be programmed by several bits in the SSI control registers. Table 45-1 shows the list of SSI operating modes and some of the typical applications in which they can be used.

Table 45-1. SSI Operating Modes

TX, RX Sections	Serial Clock	Mode	Typical Application
Asynchronous	Continuous	Normal	Multiple synchronous CODECs
Asynchronous	Continuous	Network	TDM CODEC or DSP networks
Synchronous	Continuous	Normal	Multiple synchronous CODECs
Synchronous	Continuous	Network	TDM CODEC or DSP network
Synchronous	Gated	Normal	SPI-type devices; DSP to MCU

The transmit and receive sections of the SSI can be synchronous or asynchronous. In synchronous mode, the transmitter and the receiver use a common clock and frame synchronization signal. The RXBIT0 and RSHFD bits in SRCR still affect shifting-in of received data in synchronous mode. In asynchronous mode, the transmitter and receiver each has its own clock and frame synchronization signals. Continuous or gated clock mode can be selected. In continuous mode, the clock runs continuously. In gated clock mode, the clock is functioning only during transmission.

Normal or network mode can also be selected. In normal mode, the SSI functions with one data word of I/O per frame. In network mode, any number from two to thirty-two data words of I/O per frame can be used. Network mode is typically used in star time division multiplex networks with other processors or CODECs, allowing interface to time division multiplexed networks without additional logic. Use of the gated clock is not allowed in network mode. These distinctions result in the basic operating modes that allow the SSI to communicate with a wide variety of devices.

The SSI supports both normal and network modes, and these can be selected independently of whether the transmitter and receiver are synchronous or asynchronous. Typically, these protocols are used in a periodic manner, where data is transferred at regular intervals, such as at the sampling rate of an external CODEC. Both modes use the concept of a frame. The beginning of the frame is marked with a frame sync when programmed with continuous clock. The frame sync occurs at a periodic interval. The length of the frame is determined by the DC[4:0] bits in either the SRCCR or STCCR register, depending on whether data is being transmitted or received. The number of words transferred per frame depends on the mode of the SSI.

In normal mode, one data word is transferred per frame. In network mode, the frame is divided into anywhere between two and thirty-two time slots, where, in each time slot, one data word can optionally be transferred.

Apart from the above basic modes of operation, SSI supports the following modes that require some specific programming.

- I^2S mode
- AC97 mode, fixed or variable

In (non-I^2S) slave modes (external frame sync), the programmed word length setting of the SSI should be equal to the word length setting of the master. In I^2S slave mode, the programmed word length setting of the SSI can be less than or equal to the word length setting of the I^2S master (external CODEC).

In slave modes, the programmed frame length setting (DC bits) of the SSI can be less than or equal to the frame length setting of the master (external CODEC).

The following sections provide detailed descriptions of the above modes.

45.1.2.1 Normal Mode

Normal mode is the simplest mode of the SSI. It is used to transfer data in one time slot per frame. A time slot is a unit of data and the WL[3:0] bits define the number of bits in a time slot. In continuous clock mode, a frame sync occurs at the beginning of each frame.

The length of the frame is determined by the following factors:

- The period of the serial bit clock (DIV2, PSR, PM[7:0] bits for internal clock or the frequency of the external clock on the STCK port)
- The number of bits per time slot (WL[3:0] bits)
- The number of time slots per frame (DC[4:0] bits)

If normal mode is configured with more than one time slot per frame, data is transferred only in the first time slot. No data is transferred in subsequent time slots. In normal mode, using DC[4:0] values that correspond to more than a single time slot in a frame result only in lengthening the frame. Data transfer takes place only during the first time slot of the frame.

45.1.2.1.1 Normal Mode Transmit

The conditions for data transmission from the SSI in normal mode are the following:

- SSI is enabled (SSIEN = 1).
- Enable FIFO and configure transmit and receive watermark if FIFO is used.
- Write data to transmit data register (STX).
- Transmitter is enabled (TE = 1).
- Frame sync is active (for continuous clock case).
- Bit clock begins (for gated clock case).

When the above conditions occur in normal mode upon frame sync detection or generation, the next data word is transferred into the transmit shift register (TXSR) from the transmit data register 0 (STX0), or from the transmit FIFO 0 register if transmit FIFO 0 is enabled. The new data word is transmitted immediately.

If transmit FIFO 0 is not enabled and the transmit data register empty (TDE0) bit is set, transmit interrupt 0 occurs if the transmit interrupt enable (TIE) and TDE0_EN bits are set.

The transmit FIFO empty 0 (TFE0) bit is set if the transmit FIFO 0 reaches the selected threshold. If transmit FIFO 0 is enabled and the transmit FIFO empty (TFE0) bit is set, transmit interrupt 0 occurs if the transmit interrupt enable (TIE) and TFE0_EN bits are set. If transmit FIFO 0 is enabled and filled with data, 8 data words can be transferred before the core must write new data to the STX0 register.

The STXD port is disabled except during the data transmission period. For a continuous clock, the optional frame sync output and clock outputs are not disabled, even if both receiver and transmitter are disabled.

45.1.2.1.2 Normal Mode Receive

The conditions for data reception from the SSI are the following:

- SSI is enabled (SSIEN = 1).
- Receiver is enabled (RE = 1).
- Frame sync is active (for continuous clock case).
- Bit clock begins (for gated clock case).

With the above conditions in normal mode with a continuous clock, each time the frame sync signal is generated (or detected), a data word is clocked in. With the above conditions and a gated clock, each time the clock begins, a data word is clocked in.

If receive FIFO 0 is not enabled, the received data word is transferred from the receive shift register (RXSR) to the receive data register 0 (SRX0), and the receive data ready 0 (RDR0) flag is set. Receive Interrupt 0 occurs if RIE and RDR0_EN bits are set.

If receive FIFO 0 is enabled, the received data word is transferred to the receive FIFO 0. The receive FIFO full 0 (RFF0) flag is set if the receive data register (SRX0) is full and receive FIFO 0 reaches the selected threshold. Receive interrupt 0 occurs if receive interrupt enable (RIE) and RFF0_EN bits are set.

The core program must read the data from the receive data register 0 (SRX0) before a new data word is transferred from the receive shift register (RXSR); otherwise, the receive overrun error 0 (ROE0) bit is set. If receive FIFO 0 is enabled, the receive overrun error 0 (ROE0) bit is set when the receive FIFO 0 data level reaches the selected threshold and a new data word is ready to be transferred to the receive FIFO 0.

See Figure 45-2 for an illustration of the transmitter and receiver timing for an 8-bit word in the first time slot in normal mode, continuous clock with a late word length frame sync. The transmit data register is loaded with the data to be transmitted. On arrival of the clock, this data is transferred to the transmit shift register, which gets transmitted on arrival of the frame-sync on the STXD output. Simultaneously, the receive shift register shifts in the received data available on the SRXD input. At the end of the time slot, this data is transferred to the receive data register.

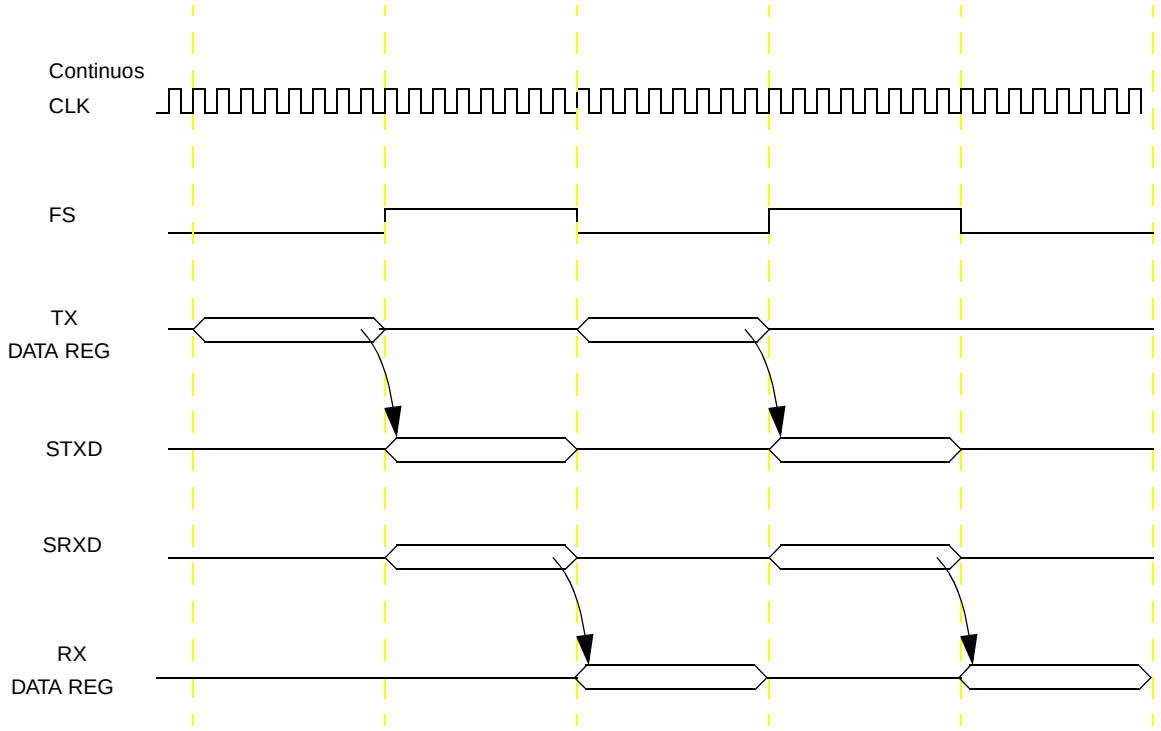


Figure 45-2. Normal Mode Timing—Continuous Clock

Figure 45-3 shows a similar case for internal (SSI generates clock) gated clock mode and Figure 45-4 shows a case for external (SSI receives clock) gated clock mode.

NOTE

A pull-down resistor is required in the gated clock case because the clock port is disabled between transmissions (the clock signal is tri-stated).

The transmit data register is loaded with the data to be transmitted. On arrival of the clock, this data is transferred to the transmit shift register, which gets transmitted on arrival of the frame-sync on the STXD output. Simultaneously, the receive shift register shifts in the received data available on the SRXD input and at the end of the time slot. This data is transferred to the receive data register. In case of internal hated clock mode, the transmit data line and clock output port are put in the high-impedance state at the end of transmission of the last bit (at the completion of the complete clock cycle). In external gated clock mode, the transmit data line is tri-stated at the last inactive edge of the incoming bit clock (during the last bit in a data word).

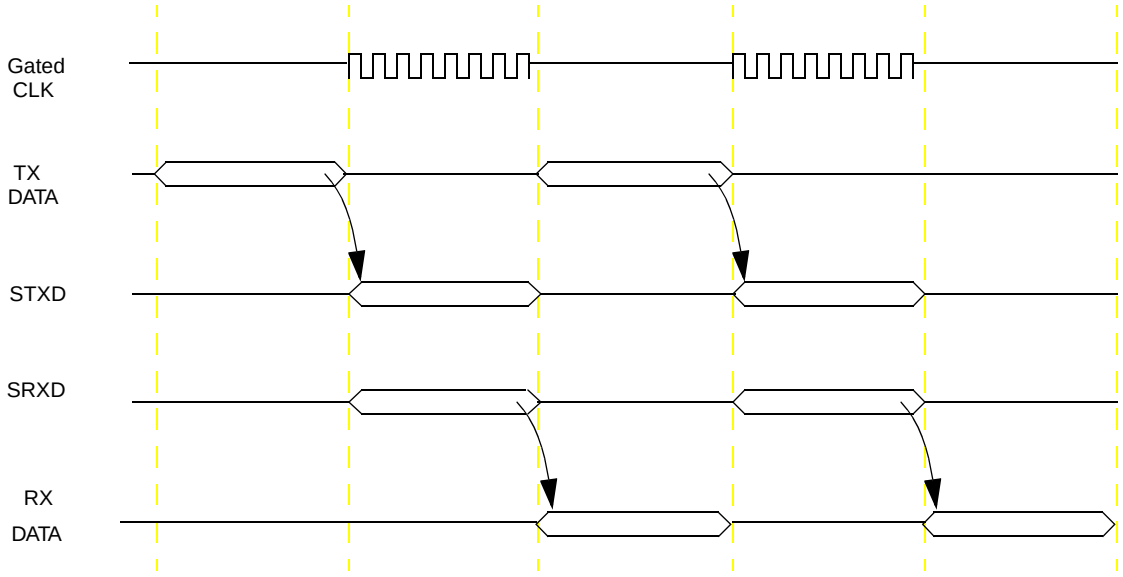


Figure 45-3. Normal Mode Timing—Internal Gated Clock

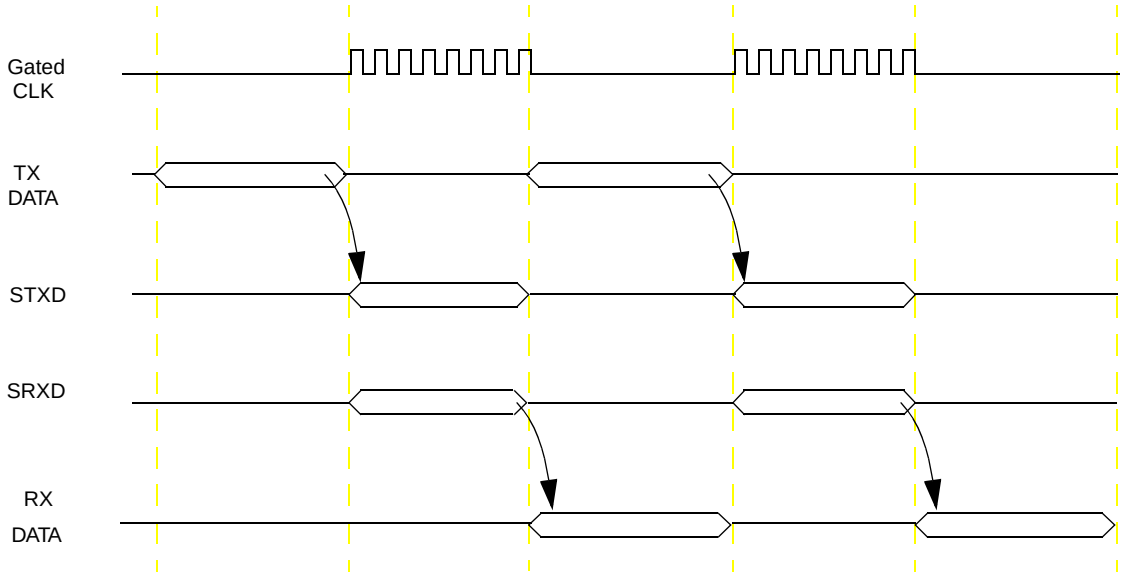


Figure 45-4. Normal Mode Timing—External Gated Clock

45.1.2.2 Network Mode

Network mode is used for creating a time division multiplexed (TDM) network, such as a TDM CODEC network or a network of DSPs. In continuous clock mode, a frame sync occurs at the beginning of each frame. In Network mode, the frame is divided into more than one time slot. During each time slot, one data word can be transferred. Each time slot is then assigned to an appropriate CODEC or DSP on the network.

The DSP can be a master device that controls its own private network, or a slave device that is connected to an existing TDM network and occupies a few time slots.

The frame sync signal indicates the beginning of a new data frame. Each data frame is divided into time slots and transmission and/or reception of one data word can occur in each time slot (rather than in just the frame sync time slot as in normal mode). The frame rate dividers, controlled by the DC[4:0] bits, select two to thirty-two time slots per frame. The length of the frame is determined by the following factors:

- The period of the serial bit clock (PSR, PM[7:0] bits for internal clock, or the frequency of the external clock on the STCK port)
- The number of bits per sample (WL[3:0] bits)
- The number of time slots per frame (DC[4:0] bits)

In network mode, data can be transmitted in any time slot. The distinction of the network mode is that each time slot is identified with respect to the frame sync (data word time). This time slot identification enables the option of transmitting data during the time slot by writing to the STX registers or ignoring the time slot as determined by STMSK register bits. The receiver is treated in the same manner and received data is only transferred to the receive data register/fifo if the corresponding time slot is enabled (through SRMSK).

By utilizing the STMSK and SRMSK registers, software has only to service the SSI during valid time slots. This eliminates any overhead associated with unused time slots. Refer to Section 45.3.3.18, “SSI Transmit Time Slot Mask Register (STMSK)” for more information on STMSK. See Section 45.3.3.19, “SSI Receive Time Slot Mask Register (SRMSK)” for more information on and SRMSK. In network mode, STXD port is in high impedance during inactive time slots, therefore permitting other devices to send data to each other during time slots that are not used by the i.MX31.

NOTE

If internal network mode of AUDMUX module is used, the STXD will be ANDed with STXD from other audio ports, resulting in driving the external port all the time. The pin will not be tri-stated anymore.

In the two-channel mode of operation, the second set of transmit and receive FIFOs and data registers are used to create two separate channels. These channels are completely independent, with a their own set of core interrupts and DMA requests, which are identical to the ones available for the default channel. In this mode, data is transmitted/received in enabled time slots alternately from/to FIFO 0 and FIFO 1, starting from FIFO 0. The first data word is taken from FIFO 0 and transmitted in the first enabled time slot and subsequently, data is loaded from FIFO 1 and FIFO 0 alternately and transmitted. Similarly, the first received data is sent to FIFO 0 and subsequent data is sent to FIFO 1 and FIFO 0 alternately. Time slots can be selected through the transmit and receive time slot mask registers (STMSK and SRMSK). For using this mode of operation, the TCH_EN bit (SCR[8]) needs to be set. Two channel mode is adapted to the processing of network mode with 2 channels, providing 2 separate data streams without the need of software reorganizing the samples (Stereo mode, or 2 different devices connected on the same SSI bus).

45.1.2.2.1 Network Mode Transmit

The transmit portion of SSI is enabled when the SSIEN and the TE bits in the SCR are both set. However, for continuous clock, when the TE bit is set, the transmitter is enabled only after detection of a new frame sync (meaning, transmission starts from the next frame boundary).

Normal start-up sequence for transmission is performing the following steps:

1. Write the data to be transmitted to the STX register. This clears the TDE flag.
2. Set the TE bit to enable the transmitter on the next word boundary (for continuous clock case).
3. Enable transmit interrupts.

Alternatively, the programmer can decide not to transmit in a time slot by configuring the STMSK. The TDE flag is not cleared, but the STXD port remains disabled (tri-state) during the time slot. When the frame sync is detected or generated (continuous clock), the first enabled data word is transferred from the STX register to the TXSR and is shifted out (transmitted). When the STX register is empty, the TDE bit is set, which causes a transmitter interrupt (in case the FIFO is disabled) to be sent if the TIE bit is set. The software can poll the TDE bit or use interrupts to reload the STX register with new data for the next time slot. Failing to reload the STX register before the TXSR is finished shifting (empty) causes a transmitter underrun and the TUE error bit is set. In case the FIFO is enabled, the TFE flag is set in accordance with the watermark setting and this flag causes the transmitter interrupt to occur.

The operation of clearing the TE bit disables the transmitter after completion of transmission of the current frame. Setting the TE bit enables transmission from the next frame. During that time, the STXD port is disabled. The TE bit should be cleared after the TDE bit is set to ensure that all pending data is transmitted.

To summarize, the network mode transmitter generates interrupts every enabled time slot and requires the core program to respond to each enabled time slot. These responses from the core are one of the following:

- Write data in data register to enable transmission in the next time slot.
- Configure the time slot register to disable transmission in the next time slot (unless the time slot is already masked by STMSK register bit).
- Do nothing—transmit underrun occurs at the beginning of the next time slot and the previous data is re-transmitted.

In the two-channel mode of operation, both the channels (data registers, FIFOs, interrupts and DMA requests) operate in the same manner as described above. (The samples go alternately from FIFO 0 and FIFO 1).The only difference in case of the second channel is that the interrupts related to this channel are generated only if this mode of operation is selected (TDE1 is low by default).

45.1.2.2.2 Network Mode Receive

The receiver portion of the SSI is enabled when both the SSIEN and the RE bits in the SCR are set. However, the receive enable takes place only during that time slot if RE is enabled before the second to last bit of the word. If the RE bit is cleared, the receiver is disabled at the end of the current frame. SSI is capable of finding the start of the next frame automatically. When the word is completely received, it is transferred to the SRX register, which sets the RDR bit (receive data ready). Setting the RDR bit causes a receive interrupt to occur if the receiver interrupt is enabled (meaning the RIE bit is set). The second data word (second time slot in the frame), begins shifting in immediately after the transfer of the first data word to the SRX register. The core program must read the data from the receive data register (which clears RDR) before the second data word is completely received (ready to transfer to RX data register) or a receive overrun error occurs (meaning the ROE bit is set).

An interrupt can occur after the reception of each enabled data word or the programmer can poll the RDR flag. The core program response can be one of the following:

- Read RX and use the data.
- Read RX and ignore the data.
- Do nothing—the receiver overrun exception occurs at the end of the current time slot.

NOTE

For a continuous clock, the optional frame sync output and clock output signals are not affected, even if the transmitter or receiver is disabled. TE and RE do not disable the bit clock or the frame sync generation. To disable the bit clock and the frame sync generation, the SSIEN bit in the SCR can be cleared, or the port control logic external to the SSI (for example, in the IOMUX) can be reconfigured.

In the two-channel mode of operation, both the channels (data registers, FIFOs, interrupts and DMA requests) operate in the same manner as described above. The only difference in case of the second channel is that the interrupts related to this channel are generated only in case this mode of operation is selected.

The transmitter and receiver timing for an 8-bit word with continuous clock, FIFO disabled, three words per frame sync in network mode is shown in Figure 45-5.

NOTE

In Table 45-5 the transmitter repeats the value 0x5E because of an underrun condition. For the transmit section, the STMSK value is updated in the last time slot of frame 1, to mask the first two time slots (0x3). This value takes effect from the next time slot and consequently, the next frame transmits data in the third time slot only.

For the receive section, data received on the SRXD pin gets transferred to the receive data register at the end of each time slot. If the FIFO is disabled, the RDR flag gets set and causes a receiver interrupt if RE, RIE, and RDR_EN bits are set. If the FIFO is enabled, then the RFF flag is used for interrupt generation (this flag set in accordance with the watermark settings). Here, all time slots are enabled. The receive data ready flag is set after reception of the first data (0x55). Since the flag is not cleared (receive data register not read by core), the receive overrun error (ROE) flag is set on reception of the next data (0x5E). The ROE flag is cleared on reading the SSI status register followed by reading the receive data register.

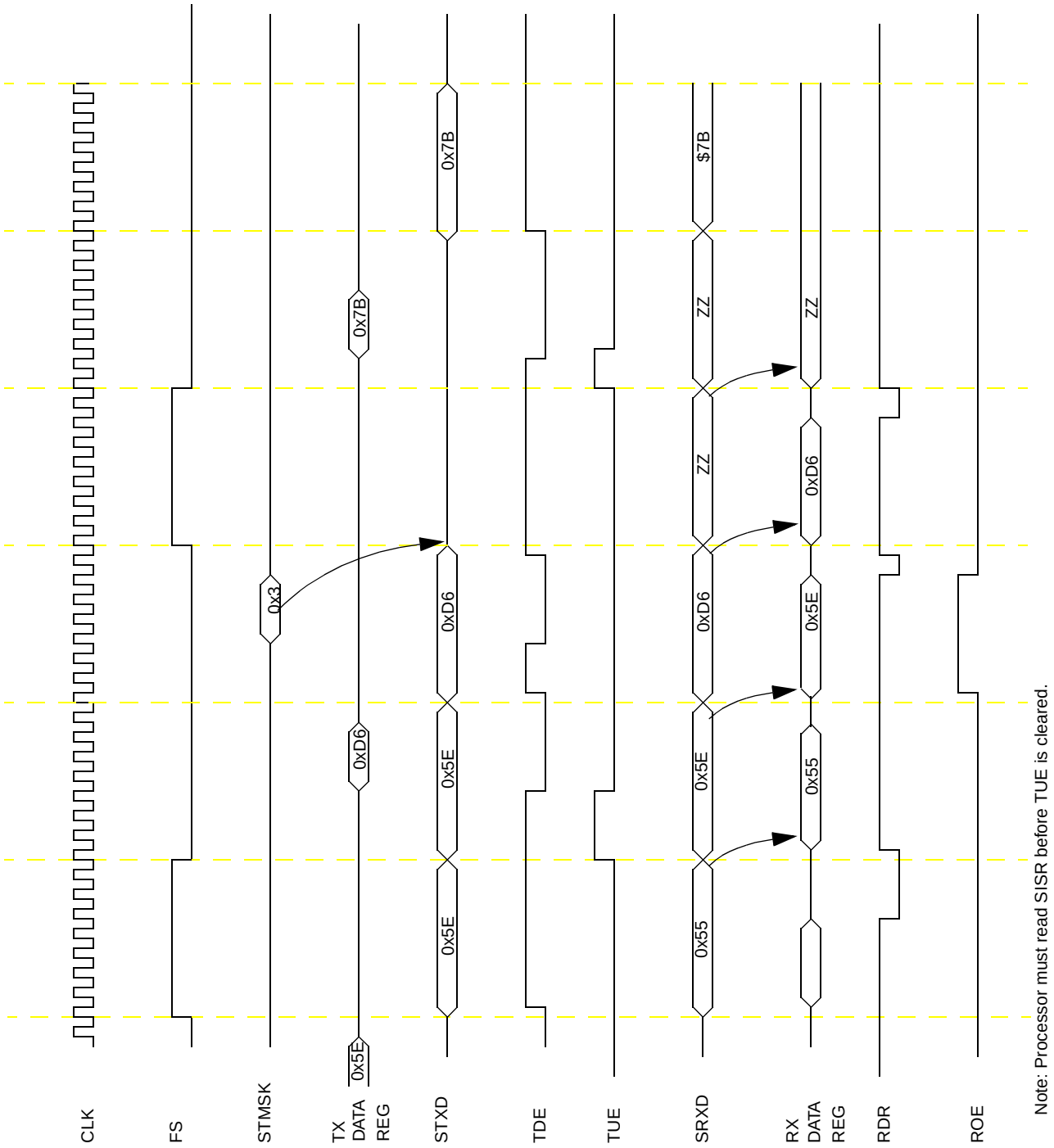


Figure 45-5. Network Mode Timing—Continuous Clock

45.1.2.3 Gated Clock Mode

Gated clock mode is often used to hook up to SPI-type interfaces on Microcontroller Units (MCUs) or external peripheral chips. In gated clock mode, the presence of the clock indicates that valid data is on the STXD or SRXD ports. For this reason, no frame sync is needed in this mode. Once transmission of data has completed, the clock is pulled to the inactive state. Gated clocks are allowed for both the transmit and receive sections with either internal or external clock in normal mode. Gated clocks are not allowed in network mode.

The clock runs when the TE bit and/or the RE bit are appropriately enabled. In the case of an internally generated clock, all internal bit clocks, word clocks, and frame clocks continue to operate. When a valid time slot occurs (such as the first time slot in normal mode), the internal bit clock is enabled onto the appropriate clock port. This allows data to be transferred out in periodic intervals in gated clock mode. With an external clock, the SSI waits for a clock signal to be received. Once the clock begins, valid data is shifted in. Care should be taken to clear all DC bits (0x00000) when SSI is used in gated mode.

For gated clock operated in external clock mode, a proper clock signalling must be applied to the SSI STCK in order for it to function properly. If the SSI uses rising edge transition to clock data (TSCKP=0) and the falling edge transition to latch data (RSCKP=0), the clock must be in an active low state when idle. If the SSI uses falling edge transition to clock data (TSCKP=1) and the rising edge transition to latch data (RSCKP=1), the clock must be in a active high state when idle. Figure 45-6 through Figure 45-9 show the different edge clocking/latching.

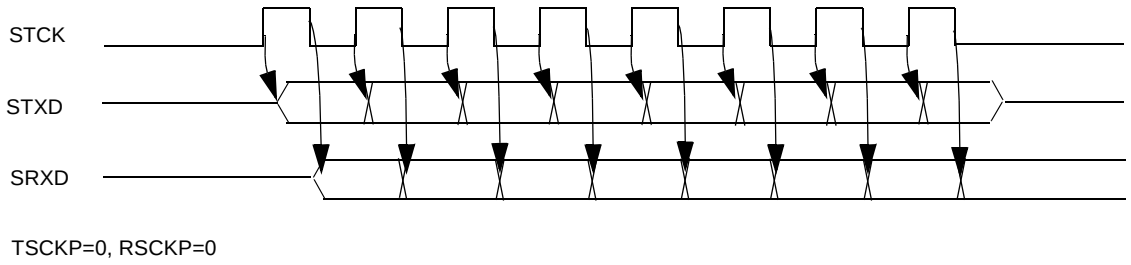


Figure 45-6. Internal Gated Mode Timing—Rising Edge Clocking/Falling Edge Latching

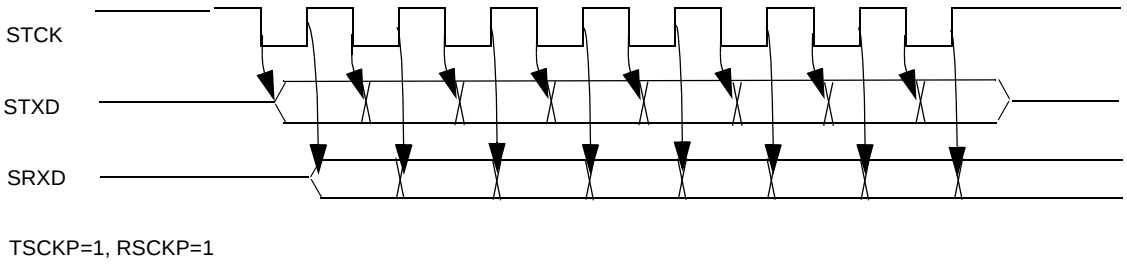


Figure 45-7. Internal Gated Mode Timing—Falling Edge Clocking/Rising Edge Latching

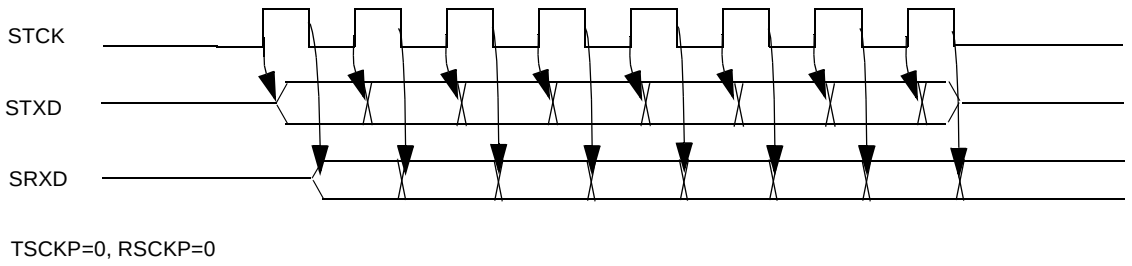


Figure 45-8. External Gated Mode Timing—Rising Edge Clocking/Falling Edge Latching

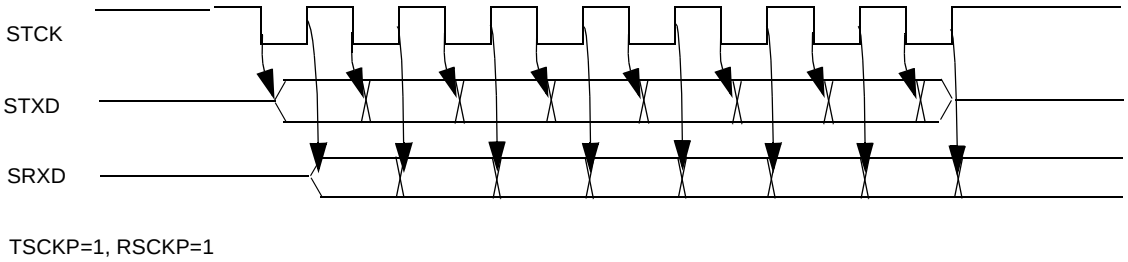


Figure 45-9. External Gated Mode Timing—Falling Edge clocking/Rising Edge Latching

NOTE

- The bit clock ports must be kept free of timing glitches. If a single glitch occurs, all ensuing transfers will be out of synchronization.
- In case of external gated mode, even though the transmit data line is put in the high-impedance state at the last non-active edge of the bit clock, the round trip delay should be sufficient to take care of hold time requirements at the external receiver.

45.1.2.4 I^2S Mode

The SSI is compliant to I^2S bus specification from Philips Semiconductors (February 1986, Revised June 5, 1996). See Figure 45-10 for an illustration of the basic I^2S protocol timing.

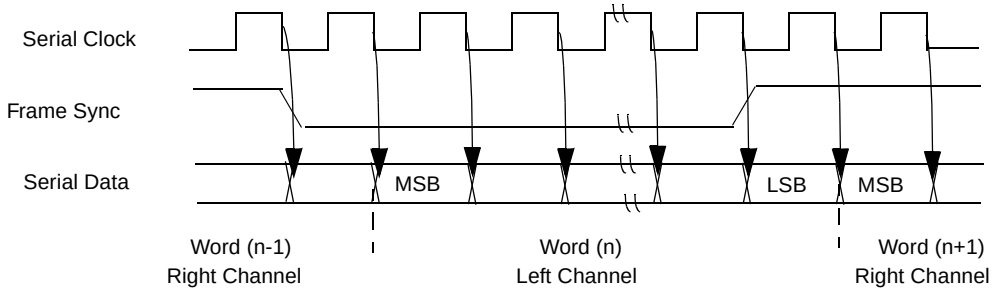


Figure 45-10. I^2S Mode Timing—Serial Clock, Frame Sync, and Serial Data

Select I^2S mode using the options listed in Table 45-2.

Table 45-2. I^2S Mode Selection

I2S_MODE[1]	I2S_MODE[0]	Mode Type
0	0	Normal
0	1	I^2S master
1	0	I^2S slave
1	1	Normal

In normal mode operation, no register bits are forced to any particular state internally, and the user can program the SSI to work in any operating condition.

When I^2S modes are entered—I^2S master (01) or I^2S slave (10)—the following settings are recommended:

- Sync mode (SCR[4] =1)
- Transmit shift direction: MSB transmitted first (STCR[4]=0)
- Receive shift direction: MSB received first (SRCR[4]=0)
- Transmit data clocked at falling edge of the clock (STCR[3]=1)
- Receive data latched at rising edge of the clock (SRCR[3]=1)
- Transmit frame sync active low (STCR[2]=1)
- Receive frame sync active low (SRCR[2]=1)
- Transmit frame sync initiated one bit before data is transmitted (STCR[0]=1)
- Receive frame sync initiated one bit before data is received (SRCR[0]=1)

In I^2S master mode(SCR[6:5]=01), the following additional settings are recommended:

- TXDIR bit (STCR[5]) set to 1 to select internal generated bit clock
- TFDIR bit (STCR[6]) set to 1 to select internal generated frame sync

In I^2S master mode(SCR[6:5]=01), the following settings are internally overridden by the hardware:

- Network mode selected (SCR[3]=1)
- Transmit frame sync length set to one-word-long-frame (STCR[1]=0)
- Receive frame sync length set to one-word-long-frame (SRCR[1]=0)
- Transmit shifting w.r.t. bit 0 of TXSR (STCR[9]=1)
- Receive shifting w.r.t. bit 0 of RXSR (SRCR[9]=1)

The user needs to set the following control bits to configure the bit clock and frame sync:

- PM (STCCR[7:0])
- PSR (STCCR[17])
- DIV2(STCCR[18]
- WL (STCCR[16:13]
- DC (STCCR[12:8])

The word length is fixed to 32 in I^2S Master mode and the WL bits determine the number of bits that will contain valid data (out of the 32 transmitted/received bits in each channel). The fixing of word duration as 32 simplifies the relation between oversampling clock (ccm_ssi_clk) and Frame Sync (ccm_ssi_clk becomes an integer multiple of frame sync).

In I^2S slave mode(SCR[6:5]=10), the following additional settings are recommended:

- TXDIR bit(STCR[5]) set to 0 to select external generated bit clock
- TFDIR bit(STCR[6]) set to 0 to select external generated frame sync

In I^2S slave mode(SCR[6:5]=10), the following settings are done internally overridden by the hardware:

- Normal mode is selected (SCR[3]=0)
- Transmit frame sync length set to one-bit-long-frame (STCR[1]=1)
- Receive frame sync length set to one-bit-long-frame (SRCR[1]=1)
- Transmit shifting w.r.t. bit 0 of TXSR (STCR[9]=1)
- Receive shifting w.r.t. bit 0 of RXSR (SRCR[9]=1)

The user needs to set the following control bits to configure the data transmission:

- WL (STCCR[16:13])
- DC (STCCR[12:8])

The word length is variable in slave mode and the WL bits determine the number of bits that will contain valid data. The actual word length is determined by the external CODEC. The external I^2S master still sends frame sync according to the I^2S protocol (early, word wide, and active low), the SSI internally operates so that each frame sync transition is the start of a new frame (meaning the WL bits determine the number of bits to be transmitted/received). After one data word has been transferred, the SSI waits for the next frame sync transition to start operation in the next time slot. Transmit (STMSK) and receive (SRMSK) mask bits should not be used in I^2S slave mode of operation.

45.1.2.5 AC97 Mode

In AC97 mode of operation, the SSI transmits a 16-bit tag slot at the start of a frame. The rest of the slots (in that frame) are all 20-bits wide. The same sequence is followed while receiving data. Refer to the AC97 specification for details regarding transmit and receive sequences and data formats.

When AC97 mode is enabled, the following settings are internally overridden by the hardware. The programmed register values are not changed by entering AC97 mode, but they no longer apply to the module's operation. Writing to the programmed register fields update their values. These updates can be seen by reading back the register fields; however, these settings do not take effect until the AC97 mode is turned off.

The register bits within the bracket are the equivalent settings:

- Sync mode is entered. (SCR[4] =1)
- Network mode is selected. (SCR[3]=1)
- Transmit shift direction is MSB transmitted first. (STCR[4]=0)
- Receive shift direction is MSB received first. (SRCR[4]=0)

- Transmit data is clocked at rising edge of the clock. (STCR[3]=0)
- Receive data is latched at falling edge of the clock. (SRCR[3]=0)
- Transmit frame sync is active high. (STCR[2]=0)
- Receive frame sync is active high. (SRCR[2]=0)
- Transmit frame sync length is one-word-long-frame. (STCR[1]=0)
- Receive frame sync length is one-word-long-frame. (SRCR[1]=0)
- Transmit frame sync initiated one bit before data is transmitted. (STCR[0]=1)
- Receive frame sync initiated one bit before data is received. (SRCR[0]=1)
- Transmit shifting w.r.t. bit 0 of TXSR. (STCR[9]=1)
- Receive shifting w.r.t. bit 0 of RXSR. (SRCR[9]=1)
- Transmit FIFO is enabled. (STCR[7]=1)
- Receive FIFO is enabled. (SRCR[7]=1)
- TFDIR bit (STCR[6]) is forced to 1 internally to select internal generated frame sync.
- TXDIR bit (STCR[5]) is forced to 0 internally to select external generated bit clock.

Any alteration of these bits individually does not affect the operational conditions of the SSI unless AC97 mode is deselected.

Hence, the only control bits needed to be set by the user to configure the data transmission/reception are the WL (STCCR[16:13]) and DC (STCCR[12:8]) bits. In AC97 mode, the WL bits can only legally take the values corresponding to 16-bit (truncated data) or 20-bit time slots. In case WL bits are set to select 16-bit time slots, the SSI pads the transmit data (four least significant bits) with zeros, and while receiving, stores only the most significant 16 bits in the receive FIFO. If WL bits are set to select 20-bit time slots, the tag slot will still be 16 bits, and the data slots will be configured to 20 bits.

Follow the sequence for programming the SSI to work in AC97 mode:

1. Program the WL bits to a value corresponding to either 16 or 20 bits. The WL bit setting is only for the data portion of the AC97 frame (slots #3 through #12). The tag slot (slot #0) is always 16 bits wide. The command address and command data slots (slots #1 and #2) are always 20 bits wide.
2. Select the number of time slots by programming the DC bits. For AC97 operation, DC bits should be set to a value of ‘0xC’, resulting in 13 time slots per frame.
3. Write data to be transmitted in transmit FIFO 0 (through transmit data register 0).
4. Program the FV, TIF, RD, WR, and FRDIV bits in SACNT register.
5. Update the contents of SACADD, SACDAT, and SATAG (for fixed mode only) registers.
6. Enable the AC97 mode of operation (AC97EN bit in SACNT register).

Once the SSI starts transmitting and receiving data (after being configured in AC97 mode), the programmer needs to service the interrupts as they arise (for examples—updates to command address/data or tag registers, reading of received data and writing more data for transmission). Further details regarding fixed and variable mode implementation follow.

While using AC97 in two-channel mode (TCH_EN=1), it is recommended that the received tag is not stored in the receive FIFO (TIF=0). In case the programmer needs to update the SATAG register and also

issue a RD/WR command (in a single frame), it is recommended that the SATAG register be updated prior to issuing a RD/WR command.

45.1.2.5.1 AC97 Fixed Mode (SACNT[1]=0)

In fixed mode operation, the SSI transmits at the start of a frame in accordance with the frame rate divider bits. For transmission, the programmer selects the frame rate (SACNT[10:5]), and at the start of appropriate frames, tag (slot #0), command address (slot #1), command data (slot #2), and two data samples from the transmit FIFO are transmitted (slots #3, #4).

While receiving, the received tag slot (slot #0) bit 15 is checked to see if the CODEC is ready. If this bit is set, the rest of the frame is received and the relevant registers updated. Otherwise, the entire frame is ignored.

45.1.2.5.2 AC97 Variable Mode (SACNT[1]=1)

In variable mode operation, the SSI transmits at the start of a frame only if there was a request for data in the previous frame. While transmitting, the AC97 slot request bits received during the previous frame are examined and the appropriate data samples are transmitted from the transmit FIFO, the software does not need to write the transmit tag information. Only data to be transmitted needs to be written to the appropriate location.

While receiving, if the CODEC is ready, the frame is received and the slot request bits are stored for scheduling transmission in the next frame.

45.1.2.6 External Frame and Clock Operation

When applying external frame sync and clock signals to SSI, there should be at least 4-bit clock cycles between the enabling of the transmit or receive section and the rising edge of the corresponding frame sync signal. The transition of STFS or SRFS should be synchronized with the rising edge of external clock signal, STCK, or SRCK.

45.1.2.7 Data Alignment Formats Supported

The SSI supports three data formats in order to provide flexibility with handling data. These formats dictate how data is written to (and read from) the data registers. Therefore, data can appear in different places in STX0/1 and SRX0/1 based on the data format and the number of bits per word. Independent data formats are supported for both the transmitter and receiver (that is, the transmitter and receiver can use different data formats).

The supported data formats are the following:

- MSB alignment
- LSB alignment
  - — Zero-extended (receive data only)
  - — Sign-extended (receive data only)

With MSB alignment, the most significant byte of the received or transmitted data is stored in bits 31 through 24 of the data register if the word length is larger than or equal to 16 bits. If the word length is less

than 16 bits and MSB alignment is chosen, the most significant byte of the received or transmitted data is stored in bits 15 through 8 of the data register. With LSB alignment, the least significant byte of the received or transmitted data is stored in bits 7 through 0 of the data register. Data alignment is controlled by the TXBIT0 bit in the STCR and the RXBIT0 bit in the SRCR. See Table 45-3 for the bit assignment for all the data formats supported by the SSI.

Table 45-3. Data Alignment

Format	Bit Number																													
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
8-bit LSB Aligned																									7	6	5	4	3	2	1	0
8-bit MSB Aligned																	7	6	5	4	3	2	1	0								
10-bit LSB Aligned																							9	8	7	6	5	4	3	2	1	0
10-bit MSB Aligned																	9	8	7	6	5	4	3	2	1	0						
12-bit LSB Aligned																					11	10	9	8	7	6	5	4	3	2	1	0
12-bit MSB Aligned																	11	10	9	8	7	6	5	4	3	2	1	0				
16-bit LSB Aligned																	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
16-bit MSB Aligned	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																
18-bit LSB Aligned															17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
18-bit MSB Aligned	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0														
20-bit LSB Aligned													19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
20-bit MSB Aligned	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0												
22-bit LSB Aligned											21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
22-bit MSB Aligned	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0										
24-bit LSB Aligned									23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
24-bit MSB Aligned	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0								

In addition, receive data can either be zero-extended or sign-extended if LSB alignment is selected. With zero-extension, all bits above the most significant bit are 0s. This format is useful when data is stored in a pure integer format. With sign-extension, all bits above the most significant bit are equal to the most significant bit. This format is useful when data is stored in a fixed-point integer format (which implies fractional values). Receive data extension is controlled by the RXEXT bit in the SRCR. Transmit data used with LSB alignment has no concept of sign/zero-extension. Unused bits above the most significant bit are simply ignored.

When configured in I^2S or AC97 mode, the SSI forces the selection of LSB alignment. However, RXEXT still permits a choice between zero-extension and sign-extension.

See Section 45.3.3.10, "SSI Transmit Configuration Register (STCR)" and Section 45.3.3.11, "SSI Receive Configuration Register (SRCR)" for more details.

45.2 External Signal Description

The SSI has no external signals as its serial I/O signals are connected to the AUDMUX. Refer to Chapter 42, "Digital Audio Multiplexer (AUDMUX)" for information on the external audio signals.

SSI1 is connected to Port 1 of the AUDMUX and SSI2 is connected to Port2 of the AUDMUX.

There are 6 signals from each SSI going into the AUDMUX (x being 1 or 2 depending on the SSI that is considered):

- STXDx: Transmit data from the SSI x
- STFSx: Transmit Frame Sync for SSI x
- STCLKx: Transmit Clock for SSI x
- SRXDx: Receive data for SSI x
- SRFSx: Receive Frame Sync for SSIx
- SRCLKx: Receive Clock for SSI x

45.3 Memory Map and Register Definition

Section 45.3.3, "Register Descriptions" provides the detailed descriptions for all of the SSI registers.

45.3.1 SSI Memory Map

Table 45-4 shows the SSI memory map.

Table 45-4. SSI Memory Map

Address	Register	Access	Reset Value	Section/Page
0x43FA_0000 (STX0_1) 0x5001_4000 (STX0_2)	SSI Transmit Data Register	R/W	0x0000_0000	45.3.3.1/45-25
0x43FA_0004 (STX1_1) 0x5001_4004 (STX1_2)	SSI Transmit Data Register 1	R/W	0x0000_0000	
0x43FA_0008 (SRX0_1) 0x5001_4008 (SRX0_2)	SSI Receive Data Register 0	R	0x0000_0000	45.3.3.4/45-29
0x43FA_000C (SRX1_1) 0x5001_400C (SRX1_2)	SSI Receive Data Register 1	R	0x0000_0000	
0x43FA_0010 (SCR1) 0x5001_4010 (SCR2)	SSI Control Register	R/W	0x0000_0000	45.3.3.7/45-32
0x43FA_0014 (SISR1) 0x5001_4014 (SISR2)	SSI Interrupt Status Register	R	0x0000_3003	45.3.3.8/45-34

Table 45-4. SSI Memory Map (continued)

Address	Register	Access	Reset Value	Section/Page
0x43FA_0018 (SIER) 0x5001_4018 (SIER2)	SSI Interrupt Enable Register	R/W	0x0000_3003	45.3.3.9/45-38
0x43FA_001C (STCR1) 0x5001_401C (STCR2)	SSI Transmit Configuration Register	R/W	0x0000_0200	45.3.3.10/45-40
0x43FA_0020 (SRCR1) 0x5001_4020 (SRCR2)	SSI Receive Configuration Register	R/W	0x0000_0200	45.3.3.11/45-42
0x43FA_0024 (STCCR1) 0x5001_4024 (STCCR2)	SSI Transmit Clock Control Register	R/W	0x0004_0000	45.3.3.12/45-44
0x43FA_0028 (SRCCR1) 0x5001_4028 (SRCCR2)	SSI Receive Clock Control Register	R/W	0x0004_0000	
0x43FA_002C (SFCSR1) 0x5001_402C (SFCSR2)	SSI FIFO Control/Status Register	R/W	0x0081_0081	45.3.3.13/45-47
0x43FA_0038 (SACNT1) 0x5001_4038 (SACNT2)	SSI AC97 Control Register	R/W	0x0000_0000	45.3.3.14/45-50
0x43FA_003C (SACADD1) 0x5001_403C (SACADD2)	SSI AC97 Command Address Register	R/W	0x0000_0000	45.3.3.15/45-51
0x43FA_0040 (SACDAT1) 0x5001_4040 (SACDAT2)	SSI AC97 Command Data Register	R/W	0x0000_0000	45.3.3.16/45-52
0x43FA_0044 (SATAG1) 0x5001_4044 (SATAG2)	SSI AC97 Tag Register	R/W	0x0000_0000	45.3.3.17/45-52
0x43FA_0048 (STMSK1) 0x5001_4048 (STMSK2)	SSI Transmit Time Slot Mask Register	R/W	0x0000_0000	45.3.3.18/45-53
0x43FA_004C (SRMSK1) 0x5001_404C (SRMSK2)	SSI Receive Time Slot Mask Register	R/W	0x0000_0000	45.3.3.19/45-54

45.3.2 Register Summary

Figure 45-11 shows the key to the register fields, and Table 45-5 shows the register figure conventions.

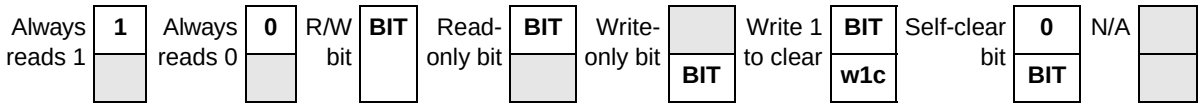


Figure 45-11. Key to Register Fields

Table 45-5. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	

Table 45-5. Register Figure Conventions (continued)

Convention	Description
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 45-6 shows the SSI register summary.

Table 45-6. SSI Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x43FA_0000 (STX0_1) 0x5001_4000 (STX0_2)	R	STX0[31:16]															
	W																
	R	STX0[15:0]															
	W																
0x43FA_0004 (STX1_1) 0x5001_4004 (STX1_2)	R	STX1[31:16]															
	W																
	R	STX1[15:0]															
	W																
0x43FA_0008 (SRX0_1) 0x5001_4008 (SRX0_2)	R	SRX0[31:16]															
	W																
	R	SRX0[15:0]															
	W																
0x43FA_000C (SRX1_1) 0x5001_400C (SRX1_2)	R	SRX1[31:16]															
	W																
	R	SRX1[15:0]															
	W																
0x43FA_0010 (SCR1) 0x5001_4010 (SCR2)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	CLK_IST	TCH_EN	SYS_CLK_EN	I2S MODE[1:0]		SYN	NET	RE	TE	SSIEN
	W																
0x43FA_0014 (SISR1) 0x5001_4014 (SISR2)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	CMDAU	CMDDU	RXT
	W																
	R	RDR1	RDR0	TDE1	TDE0	ROE1	ROE0	TUE1	TUE0	TFS	RFS	TLS	RLS	RFF1	RFF0	TFE1	TFE0
	W																
0x43FA_0018 (SIER) 0x5001_4018 (SIER2)	R	0	0	0	0	0	0	0	0	0	RDMAE	RIE	TDMAE	TIE	CMDAU_EN	CMDDU_EN	RXT_EN
	W																
	R	RDR1_EN	RDR0_EN	TDE1_EN	TDE0_EN	ROE1_EN	ROE0_EN	TUE1_EN	TUE0_EN	TFS_EN	RFS_EN	TLS_EN	RLS_EN	RFF1_EN	RFF0_EN	TFE1_EN	TFE0_EN
	W																

Table 45-6. SSI Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x43FA_001C (STCR1) 0x5001_401C (STCR2)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	TXBIT0	TFEN1	TFEN0	TFDIR	TXDIR	TSHFD	TSCKP	TFSI	TFSL	TEFS
	W																
0x43FA_0020 (SRCR1) 0x5001_4020 (SRCR2)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	RXEXT	RXBIT0	RFEN1	RFEN0	RFDIR	RXDIR	RSHFD	RSCKP	RFSI	RFSL	REFS
	W																
0x43FA_0024 (STCCR1) 0x5001_4024 (STCCR2)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	DIV2	PSR	WL3
	W																
	R	WL2	WL1	WL0	DC4	DC3	DC2	DC1	DC0	PM7	PM6	PM5	PM4	PM3	PM2	PM1	PM0
	W																
0x43FA_0028 (SRCCR1) 0x5001_4028 (SRCCR2)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	DIV2	PSR	WL3
	W																
	R	WL2	WL1	WL0	DC4	DC3	DC2	DC1	DC0	PM7	PM6	PM5	PM4	PM3	PM2	PM1	PM0
	W																
0x43FA_002C (SFCSR1) 0x5001_402C (SFCSR2)	R	RFCNT1[3:0]				TFCNT1[3:0]				RFWM1[3:0]				TFWM1[3:0]			
	W																
	R	RFCNT0[3:0]				TFCNT0[3:0]				RFWM0[3:0]				TFWM0[3:0]			
	W																
0x43FA_0038 (SACNT1) 0x5001_4038 (SACNT2)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	FRDIV[5:0]						WR	RD	TIF	FV	AC97EN
	W																

Table 45-6. SSI Register Summary (continued)

Name		31 15	30 14	29 13	28 12	27 11	26 10	25 9	24 8	23 7	22 6	21 5	20 4	19 3	18 2	17 1	16 0
0x43FA_003C (SACADD1) 0x5001_403C (SACADD2)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	SACADD[18:16]		
	W																
	R	SACADD[15:0]															
	W																
0x43FA_0040 (SACDAT1) 0x5001_4040 (SACDAT2)	R	0	0	0	0	0	0	0	0	0	0	0	0	SACDAT[19:16]			
	W																
	R	SACDAT[15:0]															
	W																
0x43FA_0044 (SATAG1) 0x5001_4044 (SATAG2)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	SATAG[15:0]															
	W																
0x43FA_0048 (STMSK1) 0x5001_4048 (STMSK2)	R	STMSK[31:0]															
	W																
	R																
	W																
0x43FA_004C (SRMSK1) 0x5001_404C (SRMSK2)	R	SRMSK[31:0]															
	W																
	R																
	W																

45.3.3 Register Descriptions

This section consists of register descriptions in address order. Each description includes a standard register diagram with an associated figure number. Details of each register bit and its field function follow the register diagrams in bit order.

45.3.3.1 SSI Transmit Data Registers 0 and 1 (STX0/1)

Figure 45-12 shows the SSI0 STX0 register, Figure 45-13 shows the SSI1 STX1 register, and Table 45-7 shows the registers' field descriptions.

0x43FA_0000 (STX0_1)
0x5001_4000 (STX0_2)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	STX0[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	STX0[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 45-12. SSI0 Transmit Data Register

0x43FA_0004 (STX1_1)
0x5001_4004 (STX1_2)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	STX1[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	STX1[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 45-13. SSI1 Transmit Data Register

Table 45-7. SSI Transmit Data Register Field Descriptions

Field	Description
31–0 STX0 STX1	SSI transmit data. These bits store the data to be transmitted by the SSI. These are implemented as the first word of their respective transmit FIFOs. Data written to these registers is transferred to the transmit shift register (TXSR), when shifting of the previous data is complete. If both FIFOs are in use, data is alternately transferred from STX0 and STX1 to TXSR. Multiple writes to the STX registers do not result in the previous data being over-written by the subsequent data. STX1 can be used only in two-channel mode of operation. Protection from over-writing is present irrespective of whether the transmitter is enabled or not. Example 1: If transmit FIFO 0 is in use and user writes Data1...Data9 to STX0, then Data9 does not over-write Data1. Data1...Data8 are stored in the FIFO while Data9 is discarded. Example 2: If transmit FIFO 0 is not in use and user writes Data1, Data2 to STX0, then Data2 does not over-write Data1 and will be discarded.

NOTE

Enable SSI (SSIEN=1) before writing to SSI transmit data registers.

45.3.3.2 SSI Transmit FIFO 0 and FIF0 1 Registers

The SSI transmit FIFO registers are 8x24-bit registers. These registers are not directly accessible by the end user (except in SSI test mode). The transmit FIFOs are accessed through STX10/1 registers. The transmit shift register (TXSR) receives its values from these FIFO registers. Transmitted data is first-in-first-out. When the transmit interrupt enable (TIE) bit in the SIER and either of the transmit FIFO empty (TFE0 or 1) bits in the SISR are set, the core is interrupted whenever the data level in either of the SSI transmit FIFOs falls below the selected threshold.

45.3.3.3 SSI Transmit Shift Register (TXSR)

The SSI transmit shift register (TXSR) is a 24-bit shift register that contains the data being transmitted. This register is not directly accessible by the end user. When a continuous clock is used, data is shifted out to the serial transmit data (STXD) port by the selected (internal/external) bit clock when the associated (internal/external) frame sync is asserted. When a gated clock is used, data is shifted out to the STXD port by the selected (internal/external) gated clock. The word length control bits (WL[3:0]) in the STCCR (described in SSI transmit and receive clock control registers) determine the number of bits to be shifted out of the TXSR before it is considered empty and can be written to again. This word length can be 8, 10, 12, 16, 18, 20, 22, or 24 bits. The data to be transmitted occupies the most significant portion of the shift register if TXBIT0 is ‘0’; otherwise, it occupies the least significant portion. The unused portion of the register is ignored. Data is always shifted out of this register with the Most Significant Bit (MSB) first when the SHFD bit of the STCR is cleared. If this bit is set, the Least Significant Bit (LSB) is shifted out first. Figure 45-14 through Figure 45-17 show the transmitter loading and shifting operation. The figures show the working for some WL values; the same can be extended for other values.

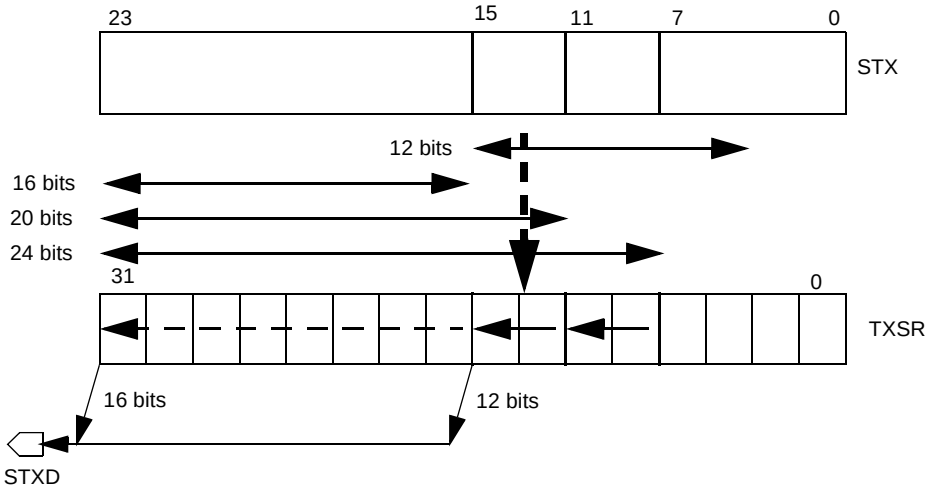


Figure 45-14. Transmit Data Path (TXBIT0=0, TSHFD=0) (MSB Alignment)

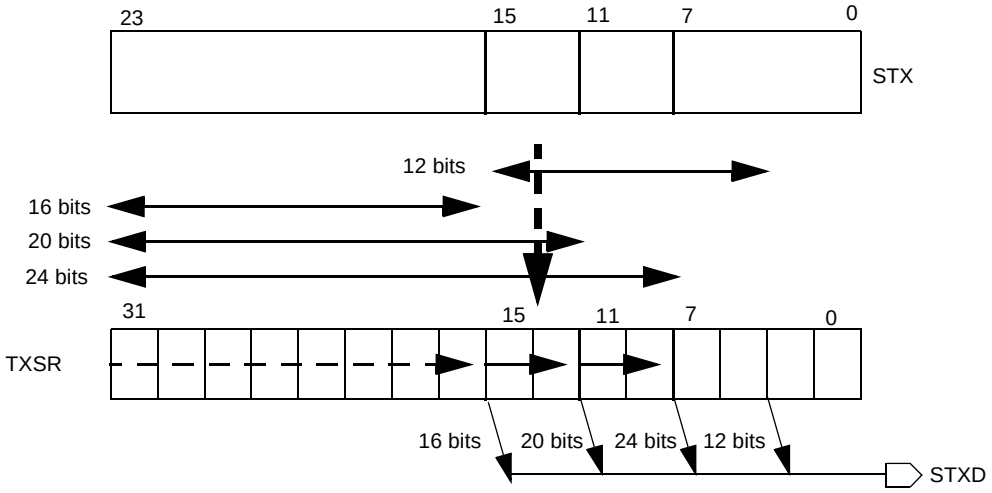


Figure 45-15. Transmit Data Path (TXBIT0=0, TSHFD=1) (MSB Alignment)

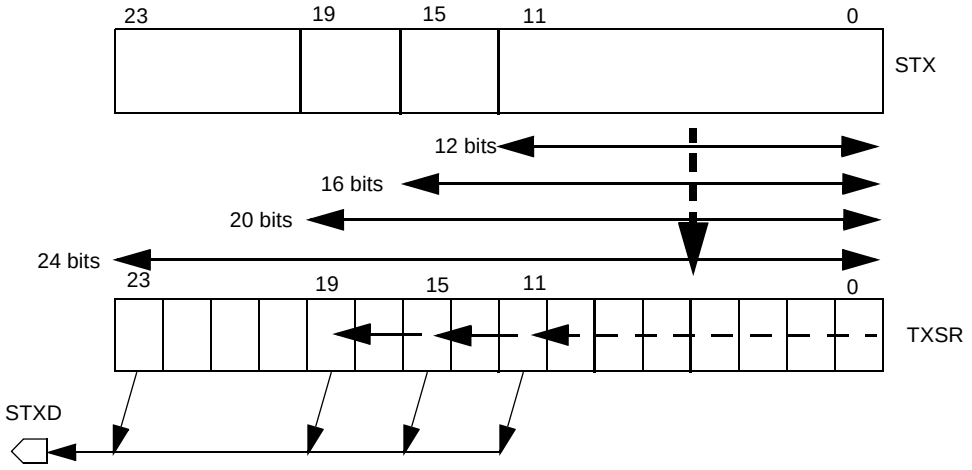


Figure 45-16. Transmit Data Path (TXBIT0=1, TSHFD=0) (LSB Alignment)

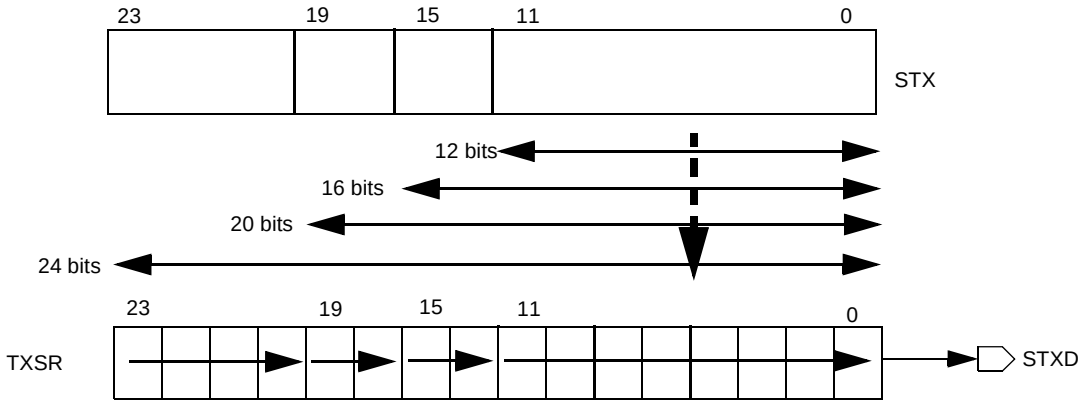


Figure 45-17. Transmit Data Path (TXBIT0=1, TSHFD=1) (LSB Alignment)

45.3.3.4 SSI Receive Data Registers 0 and 1 (SRX0/1)

Figure 45-18 shows the SSI SRX0 register, Figure 45-19 shows the SSI SRX1 register, and Table 45-8 shows the registers' field descriptions.

0x43FA_0008 (SRX0_1)
0x5001_4008 (SRX0_2)

Access: User read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SRX0[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SRX0[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 45-18. SSI0 Receive Data Register

0x43FA_000C (SRX1_1)
0x5001_400C (SRX1_2)

Access: User read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SRX1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SRX1[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 45-19. SSI1 Receive Data Register

Table 45-8. SSI_1 Receive Data Register Field Descriptions

Field	Description
31–0 SRX0 SRX1	SSI receive data. These bits store the data received by the SSI. These are implemented as the first word of their respective receive FIFOs. These bits receive data from the RXSR depending on the mode of operation. In case both FIFOs are in use, data is transferred to each data register alternately. SRX1 can be used only in two-channel mode of operation.

45.3.3.5 SSI Receive FIFO 0 and FIFO 1 Registers

The SSI receive FIFO registers are 8x24-bit registers. These registers are not directly accessible by the end user (except in SSI test mode). They are read from SRX0/1. They always accepts data from the receive shift register (RXSR). The core is interrupted when the data level in either of the SSI receive FIFOs reaches the selected threshold, if the associated interrupt is enabled.

45.3.3.6 SSI Receive Shift Register (RXSR)

The SSI receive shift register (RXSR) is a 24-bit, shift register that receives incoming data from the serial receive data SRXD port. This register is not directly accessible by the end user. When a continuous clock is used, data is shifted in by the selected (internal/external) bit clock when the associated (internal/external) frame sync is asserted. When a gated clock is used, data is shifted in by the selected (internal/external) gated clock. Data is assumed to be received MSB first if the SHFD bit of the SRCR is cleared. If this bit is set, the data is received LSB first. Data is transferred to the appropriate SSI receive data register (SRX0/1) or receive FIFOs (if the receive FIFO is enabled and the corresponding SRX is full) after 8, 10, 12, 16, 18, 20, 22, or 24 bits have been shifted in depending on the WL[3:0] control bits. For receiving less than 24 bits of data, LSB bits are appended with zero. Figure 45-20 through Figure 45-23 show the receiver loading and shifting operation. These figures show the working for some WL values; the same can be extended for other values.

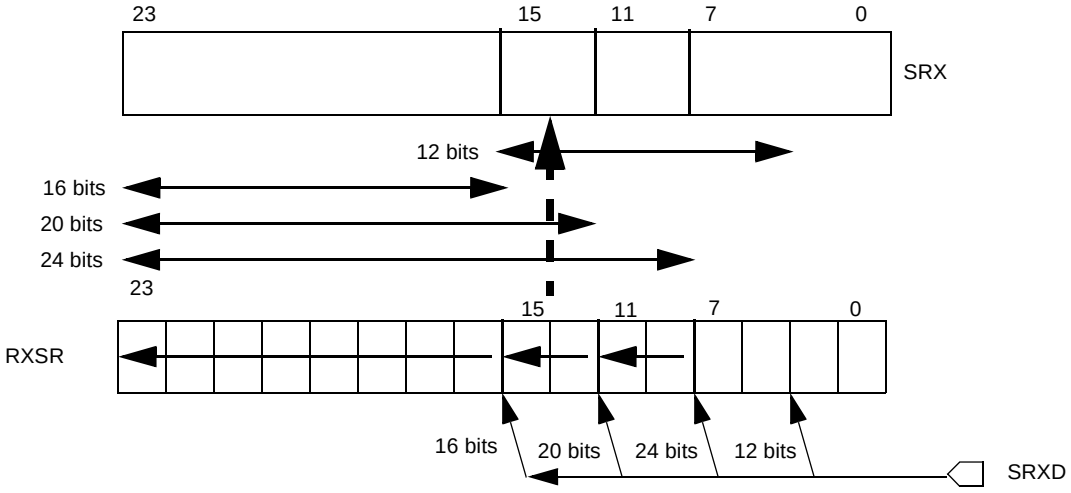


Figure 45-20. Receive Data Path (RXBIT0=0, RSHFD=0) (MSB Alignment)

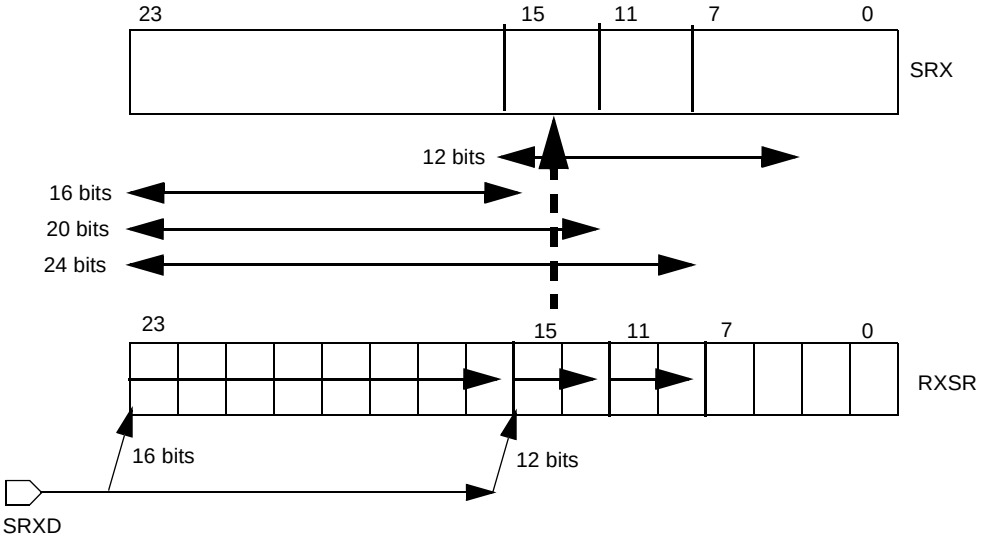


Figure 45-21. Receive Data Path (RXBIT0=0, RSHFD=1) (MSB Alignment)

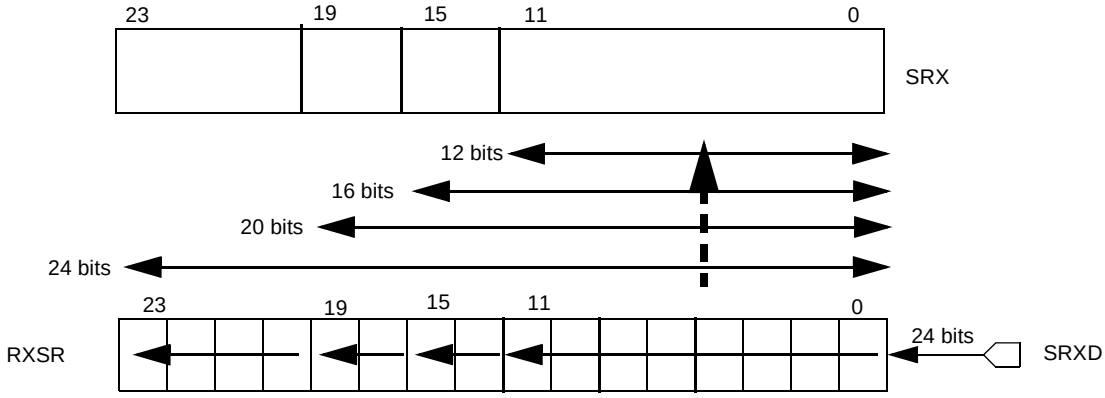


Figure 45-22. Receive Data Path (RXBIT0=1, RSHFD=0) (LSB Alignment)

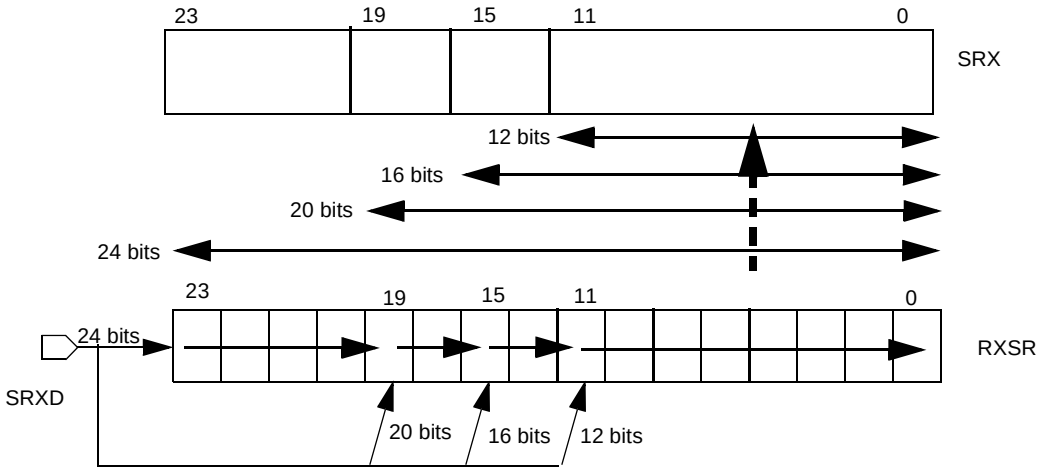


Figure 45-23. Receive Data Path (RXBIT0=1, RSHFD=1) (LSB Alignment)

45.3.3.7 SSI Control Register (SCR)

The SSI control register (SCR) is a 10-bit register used for setting up the SSI. SSI reset is controlled by bit 0 in the SCR. SSI operating modes are also selected in this register, except AC97 mode, which is selected in SACNT register.

Figure 45-24 shows the SCR register, and Table 45-9 shows the register's field descriptions.

0x43FA_0010 (SCR1)
0x5001_4010 (SCR2)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	CLK_IST	TCH_EN	SYS_CLK_EN	I2S MODE[1:0]		SYN	NET	RE	TE	SSEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 45-24. SSI Control Register

Table 45-9. SSI Control Register Field Descriptions

Field	Description
31–10	Reserved
9 CLK_IST	Clock idle state. This bit controls the idle state of the transmit clock port during SSI internal gated mode. 0 Clock idle state is ‘0’. 1 Clock idle state is ‘1’.
8 TCH_EN	Two-channel operation enable. This bit allows SSI to operate in the two-channel mode. In this mode, two time slots should be used out of the possible 32. Any two time slots (from 0 to 31) can be selected. The data in the two time slots is alternately handled by the two data registers (0 and 1). While receiving, the RXSR transfers data to SRX0 and SRX1 alternately and while transmitting, data is alternately transferred from STX0 and STX1 to TXSR. If more than two slots are to be enabled, then for odd number of slots, two-channel operation is deprecated and TCH_EN should be cleared. This will ensure that all data is received and transmitted from one FIFO—FIFO 0. For an even number of slots, two-channel operation can be enabled to optimize usage of both FIFOs or disabled as in the case of odd number of active slots. 0 Two-channel mode disabled. 1 Two-channel mode enabled.
7 SYS_CLK_EN	System clock enable. When set, this bit allows the SSI to output the SYS_CLK (ccm_ssi_clk) at the SRCK port, provided that network mode, synchronous mode, and transmit internal clock mode are set. The relationship between bit clock and SYS_CLK is determined by DIV2, PSR, and PM bits.The ccm_ssi_clk signal is the output of SSIxDIV of the Clock Control Module. This bit can be used to output the oversampling clock on SRCK port in I^2S master mode. 0 SYS_CLK not output on SRCK port. 1 SYS_CLK output on SRCK port.
6–5 I2S MODE[1:0]	I^2S mode select. These bits allow the SSI to operate in normal, I^2S master or I^2S slave mode. Refer to Section 45.1.2.4, “I2S Mode” for a details. Refer to Table 45-2 for details regarding settings.
4 SYN	Synchronous mode. This bit controls whether SSI is in synchronous mode or not. In synchronous mode, the transmit and receive sections of SSI share a common clock port (STCK) and frame sync port (STFS). 0 Asynchronous mode selected. 1 Synchronous mode selected.
3 NET	Network mode. This bit controls whether SSI is in network mode or not. 0 Network mode not selected. 1 Network mode selected.
2 RE	Receive enable. This control bit enables the receive section of the SSI. When this bit is enabled, data reception starts with the arrival of the next frame sync. If data is being received when this bit is cleared, data reception continues until the end of the current frame and then stops. If this bit is set again before the second to last bit of the last time slot in the current frame, then reception continues without interruption. 0 Receive section disabled. 1 Receive section enabled.

Table 45-9. SSI Control Register Field Descriptions (continued)

Field	Description
1 TE	Transmit enable. This control bit enables the transmit section of the SSI. It enables the transfer of the contents of the STX registers to the TXSR and also enables the internal transmit clock. The transmit section is enabled when this bit is set and a frame boundary is detected. When this bit is cleared, the transmitter continues to send data until the end of the current frame and then stops. Data can be written to the STX registers with the TE bit cleared; the corresponding TDE bit will be cleared. If the TE bit is cleared and then set again before the second to last bit of the last time slot in the current frame, data transmission continues without interruption. The normal transmit enable sequence is to write data to the STX register(s) and then set the TE bit. The normal disable sequence is to clear the TE and TIE bits after the TDE bit is set. In gated clock mode, clearing the TE bit results in the clock stopping after the data currently in TXSR has shifted out. When the TE bit is set, the clock starts immediately (for internal gated clock mode). 0 Transmit section disabled. 1 Transmit section enabled.
0 SSIEN	SSI enable. This bit is used to enable/disable the SSI. When disabled, all SSI status bits are preset to the same state produced by the power-on reset, all control bits are unaffected, and the contents of transmit and receive FIFOs are cleared. When SSI is disabled, all internal clocks are disabled (except register access clock). 0 SSI is disabled. 1 SSI is enabled.

45.3.3.8 SSI Interrupt Status Register (SISR)

The SSI interrupt status register (SISR) is a 19-bit register used to monitor the SSI. This register is read-only and is used by the core to interrogate the status of the SSI. All receiver related interrupts are generated only if the receiver is enabled (SCR[2]=1) and all transmitter related interrupts are generated only if the transmitter is enabled (SCR[1]=1). Figure 45-25 shows a list of SSI interrupt sources.

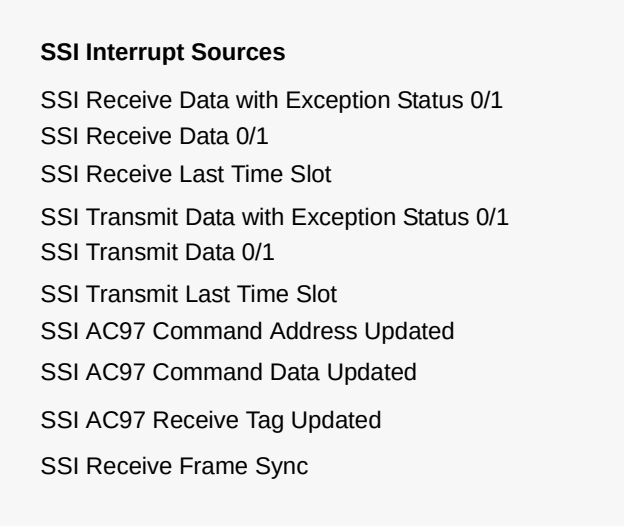


Figure 45-25. SSI Interrupts

NOTE

SSI status flags are updated when SSI is enabled.

- See Section 45.4.3, “Receive Interrupt Enable Bit Description” and Section 45.4.4, “Transmit Interrupt Enable Bit Description” for interrupt source mapping.
- All the flags in the SISR are updated after the first bit of the next SSI word has completed transmission or reception. Some status bits (ROE0/1 and TUE0/1) are cleared by reading the SISR followed by a read or write to either the SRX0/1 or STX0/1 registers.

Figure 45-26 shows the SISR register, and Table 45-10 shows the register’s field descriptions.

0x43FA_0014 (SISR1)
0x5001_4014 (SISR2)

Access: User read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	CMD AU	CMD DU	RXT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RDR1	RDR0	TDE1	TDE0	ROE1	ROE0	TUE1	TUE0	TFS	RFS	TLS	RLS	RFF1	RFF0	TFE1	TFE 0
W																
Reset	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1

Figure 45-26. SSI Interrupt Status Register

Table 45-10. SSI Interrupt Status Register Field Descriptions

Field	Description
31–19	Reserved
18 CMDAU	Command address register updated. This bit causes the command address updated interrupt (when CMDAU_EN bit is set). This status bit is set each time there is a difference in the previous and current value of the received command address. This bit is cleared on reading the SACADD register. 0 No change in SACADD register. 1 SACADD register updated with different value.
17 CMDDU	Command data register updated. This bit causes the command data updated interrupt (when CMDDU_EN bit is set). This status bit is set each time there is a difference in the previous and current value of the received command data. This bit is cleared on reading the SACDAT register. 0 No change in SACDAT register. 1 SACDAT register updated with different value.
16 RXT	Receive tag updated. This status bit is set each time there is a difference in the previous and current value of the received tag. It causes the receive tag interrupt (if RXT_EN bit is set). This bit is cleared on reading the SATAG register. 0 No change in SATAG register. 1 SATAG register updated with different value.

Table 45-10. SSI Interrupt Status Register Field Descriptions (continued)

Field	Description
15 RDR1	Receive data ready 1. This flag bit is set when SRX1 or receive FIFO 1 is loaded with a new value and two-channel mode is selected. RDR1 is cleared when the core reads the SRX1 register. If receive FIFO 1 is enabled, RDR1 is cleared when the FIFO is empty. If RIE and RDR1_EN are set, a receive data 1 interrupt request is issued on setting of RDR1 bit in case receive FIFO 1 is disabled. If the FIFO is enabled, the interrupt is issued on RFF1 assertion. The RDR1 bit is cleared by POR and SSI reset. 0 No new data for core to read. 1 New data for core to read.
14 RDR0	Receive data ready 0. This flag bit is set when SRX0 or receive FIFO 0 is loaded with a new value. RDR0 is cleared when the Core reads the SRX0 register. If receive FIFO 0 is enabled, RDR0 is cleared when the FIFO is empty. If RIE and RDR0_EN are set, a receive Data 0 interrupt request is issued on setting of RDR0 bit in case receive FIFO 0 is disabled. If the FIFO is enabled, the interrupt is issued on RFF0 assertion. The RDR0 bit is cleared by POR and SSI reset. 0 No new data for core to read. 1 New data for core to read.
13 TDE1	Transmit data register empty 1. This flag is set whenever data is transferred to TXSR from STX1 register and two-channel mode is selected. If transmit FIFO 1 is enabled, this occurs when there is at least one empty slot in STX1 or transmit FIFO 1. If transmit FIFO 1 is not enabled, this occurs when the contents of STX1 are transferred to TXSR. The TDE1 bit is cleared when the core writes to STX1. If TIE and TDE1_EN are set, an SSI transmit data 1 interrupt request is issued on setting of TDE1 bit. The TDE1 bit is cleared by POR and SSI reset. 0 Data available for transmission. 1 Data needs to be written by the core for transmission.
12 TDE0	Transmit data register empty 0. This flag is set whenever data is transferred to TXSR from STX0 register. If transmit FIFO 0 is enabled, this occurs when there is at least one empty slot in STX0 or transmit FIFO 0. If transmit FIFO 0 is not enabled, this occurs when the contents of STX0 are transferred to TXSR. The TDE0 bit is cleared when the core writes to STX0. If TIE and TDE0_EN are set, an SSI transmit data 0 interrupt request is issued on setting of TDE0 bit. The TDE0 bit is cleared by POR and SSI reset. 0 Data available for transmission. 1 Data needs to be written by the core for transmission.
11 ROE1	Receiver overrun error 1. This flag is set when the RXSR is filled and ready to transfer to SRX1 register or to receive FIFO 1 (when enabled) and these are already full and two-channel mode is selected. If receive FIFO 1 is enabled, this is indicated by the RFF1 flag. Else, this is indicated by the RDR1 flag. The RXSR is not transferred in this case. The ROE1 flag causes an interrupt if RIE and ROE1_EN are set. The ROE1 bit is cleared by POR and SSI reset. It is also cleared by reading the SISR with the ROE1 bit set, followed by reading the SRX1 register. Clearing the RE bit does not affect the ROE1 bit. 0 Default interrupt is issued to the core. 1 Exception interrupt is issued to the core.
10 ROE0	Receiver overrun error 0. This flag is set when the RXSR is filled and ready to transfer to the SRX0 register or to receive FIFO 0 (when enabled) and these are already full. If receive FIFO 0 is enabled, this is indicated by the RFF0 flag. Else, this is indicated by the RDR0 flag. The RXSR is not transferred in this case. The ROE0 flag causes an interrupt if RIE and ROE0_EN are set. The ROE0 bit is cleared by POR and SSI reset. It is also cleared by reading the SISR with ROE0 bit set, followed by reading the SRX0 register. Clearing the RE bit does not affect the ROE0 bit. 0 Default interrupt is issued to the core. 1 Exception interrupt is issued to the core.

Table 45-10. SSI Interrupt Status Register Field Descriptions (continued)

Field	Description
9 TUE1	Transmitter underrun error 1. This flag is set when the TXSR is empty (no data to be transmitted), the TDE1 flag is set, a transmit time slot occurs, and the SSI is in two-channel mode. When a transmit underrun error occurs, the previous data is retransmitted. In network mode, each time slot requires data transmission (unless masked through STMSK register), when the transmitter is enabled (meaning TE is set). The TUE1 flag causes an interrupt if TIE and TUE1_EN are set. The TUE1 bit is cleared by POR and SSI reset. It is also cleared by reading the SISR with TUE1 bit set, followed by writing to the STX1 register. 0 Default interrupt is issued to the core. 1 Exception interrupt is issued to the core.
8 TUE0	Transmitter underrun error 0. This flag is set when the TXSR is empty (no data to be transmitted), the TDE0 flag is set, and a transmit time slot occurs. When a transmit underrun error occurs, the previous data is retransmitted. In network mode, each time slot requires data transmission (unless masked through STMSK register), when the transmitter is enabled (meaning TE is set). The TUE0 flag causes an interrupt if TIE and TUE0_EN are set. The TUE0 bit is cleared by POR and SSI reset. It is also cleared by reading the SISR with TUE0 bit set, followed by writing to the STX0 register. 0 Default interrupt is issued to the core. 1 Exception interrupt is issued to the core.
7 TFS	Transmit frame sync. This flag indicates the occurrence of transmit frame sync. Data written to the STX registers during the time slot when the TFS flag is set during the second time slot (in network mode) or in the next first time slot (in normal mode). In network mode, the TFS bit is set during transmission of the first time slot of the frame and is then cleared when starting transmission of the next time slot. In normal mode, this bit is always high. This flag causes an interrupt if TIE and TFS_EN are set. The TFS bit is cleared by POR and SSI reset. 0 No occurrence of transmit frame sync. 1 Transmit frame sync occurred during transmission of last word written to STX registers.
6 RFS	Receive frame sync. This flag indicates the occurrence of receive frame sync. In network mode, the RFS bit is set when the first slot of the frame is being received. It is cleared when the next slot begins to be received. In normal mode, this bit is always high. This flag causes an interrupt if RIE and RFS_EN are set. The RFS bit is cleared by POR and SSI reset. 0 No occurrence of receive frame sync. 1 Receive frame sync occurred during reception of next word in SRX registers.
5 TLS	Transmit last time slot. This flag indicates the last time slot in a frame. When set, it indicates that the current time slot is the last time slot of the frame. TLS is set at the start of the last transmit time slot and causes the SSI to issue an interrupt (if TIE and TLS_EN are set). TLS is cleared when the SISR is read with this bit set. The TLS bit is cleared by POR and SSI reset. 0 Current time slot is not last time slot of frame. 1 Current time slot is the last transmit time slot of frame.
4 RLS	Receive last time slot. This flag indicates the last time slot in a frame. When set, it indicates that the current time slot is the last receive time slot of the frame. RLS is set at the end of the last time slot and causes the SSI to issue an interrupt (if RIE and RLS_EN are set). RLS is cleared when the SISR is read with this bit set. The RLS bit is cleared by POR and SSI reset. 0 Current time slot is not last time slot of frame. 1 Current time slot is the last receive time slot of frame.

Table 45-10. SSI Interrupt Status Register Field Descriptions (continued)

Field	Description
3 RFF1	Receive FIFO full 1. This flag is set when receive FIFO 1 is enabled, the data level in receive FIFO 1 reaches the selected receive FIFO watermark 1 (RFWM1) threshold, and the SSI is in two-channel mode. The setting of RFF1 causes an interrupt only when RIE and RFF1_EN are set, receive FIFO 1 is enabled, and the two-channel mode is selected. RFF1 is automatically cleared when the amount of data in receive FIFO 1 falls below the threshold. The RFF1 bit is cleared by POR and SSI reset. When receive FIFO 1 contains 8 words—the maximum it can hold, all further data received (for storage in this FIFO) is ignored until the FIFO contents are read. 0 Space available in receive FIFO 1. 1 Receive FIFO 1 is full.
2 RFF0	Receive FIFO full 0. This flag is set when receive FIFO 0 is enabled and the data level in receive FIFO 0 reaches the selected receive FIFO watermark 0 (RFWM0) threshold. The setting of RFF0 causes an interrupt only when RIE and RFF0_EN are set and receive FIFO 0 is enabled. RFF0 is automatically cleared when the amount of data in receive FIFO 0 falls below the threshold. The RFF0 bit is cleared by POR and SSI reset. When receive FIFO 0 contains 8 words—the maximum it can hold, all further data received (for storage in this FIFO) is ignored until the FIFO contents are read. 0 Space available in receive FIFO 0. 1 Receive FIFO 0 is full.
1 TFE1	Transmit FIFO empty 1. This flag is set when transmit FIFO 1 is enabled, the data level in transmit FIFO 1 falls below the selected transmit FIFO watermark 1 (TFWM1) threshold, and the two-channel mode is selected. The setting of TFE1 causes an interrupt only when TIE and TFE1_EN are set, transmit FIFO 1 is enabled, and two-channel mode is selected. The TFE1 bit is automatically cleared when the data level in transmit FIFO 1 becomes more than the amount specified by the watermark bits. The TFE1 bit is set by POR and SSI reset. 0 Transmit FIFO 1 has data for transmission. 1 Transmit FIFO 1 is empty.
0 TFE0	Transmit FIFO empty 0. This flag is set when transmit FIFO 0 is enabled and the data level in transmit FIFO 0 falls below the selected transmit FIFO watermark 0 (TFWM0) threshold. The setting of TFE0 causes an interrupt only when TIE and TFE0_EN are set and transmit FIFO 0 is enabled. The TFE0 bit is automatically cleared when the data level in transmit FIFO 0 becomes more than the amount specified by the watermark bits. The TFE0 bit is set by POR and SSI reset. 0 Transmit FIFO 0 has data for transmission. 1 Transmit FIFO 0 is empty.

45.3.3.9 SSI Interrupt Enable Register (SIER)

The SSI Interrupt Enable Register (SIER) is a 23-bit register used to set up the SSI interrupts and DMA requests. Figure 45-27 shows the SIER register, and Table 45-11 shows the register's field descriptions.

0x43FA_0018 (SIER)
0x5001_4018 (SIER2)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	RDMAE	RIE	TDMAE	TIE	CMDAU_EN	CMDDU_EN	RXT_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RDR1_EN	RDR0_EN	TDE1_EN	TDE0_EN	ROE1_EN	ROE0_EN	TUE1_EN	TUE0_EN	TFS_EN	RFS_EN	TLS_EN	RLS_EN	RFF1_EN	RFF0_EN	TFE1_EN	TFE0_EN
W																
Reset	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1

Figure 45-27. SSI Interrupt Enable Register

Table 45-11. SSI Interrupt Enable Register Field Descriptions

Field	Description
31–23	Reserved
22 RDMAE	Receive DMA enable. This bit allows the SSI to request for DMA transfers. When enabled, DMA requests are generated when any of the RFF0/1 bits in the SISR are set and if the corresponding RFEN bit is also set. If the corresponding FIFO is disabled, a DMA request is generated when the corresponding RDR bit is set. 0 SSI receiver DMA requests disabled. 1 SSI receiver DMA requests enabled.
21 RIE	Receive interrupt enable. This control bit allows the SSI to issue receiver related interrupts to the Core. Refer to Section 45.4.3, “Receive Interrupt Enable Bit Description” for a detailed description of this bit. 0 SSI receiver interrupt requests disabled. 1 SSI receiver interrupt requests enabled.
20 TDMAE	Transmit DMA enable. This bit allows the SSI to request for DMA transfers. When enabled, DMA requests are generated when any of the TFE0/1 bits in the SISR are set and if the corresponding TFEN bit is also set. If the corresponding FIFO is disabled, a DMA request is generated when the corresponding TDE bit is set. 0 SSI transmitter DMA requests disabled. 1 SSI transmitter DMA requests enabled.
19 TIE	Transmit interrupt enable. This control bit allows the SSI to issue transmitter data related interrupts to the core. Refer to Section 45.4.4, “Transmit Interrupt Enable Bit Description” for a detailed description of this bit. 0 SSI transmitter interrupt requests disabled. 1 SSI transmitter interrupt requests enabled.
18–0 Enable Bits	Enable bit. Each bit controls whether the corresponding status bit in SISR can issue an interrupt to the core or not. 0 Status bit cannot issue interrupt. 1 Status bit can issue interrupt.

45.3.3.10 SSI Transmit Configuration Register (STCR)

The SSI transmit configuration register (STCR) is a read/write control register used to direct the transmit operation of the SSI. STCR controls the direction of the bit clock and frame sync ports, STCK and STFS. Interrupt enable bit for the transmit sections is provided in this control register. The power-on reset clears all STCR bits. However, the SSI reset does not affect the STCR bits. The STCR bits are described in the following paragraphs. See Table 45-4 for the programming model of the SSI. The SSI control register (SCR) must first be set to enable interrupts. Next, the SSI interrupt bit in the interrupt enable register (SIER) must be set to enable the interrupt. Finally, the interrupt can be enabled from within the SSI. In Synchronous mode, STCR bits TFDIR, TXDIR, TSCKP, TFSI, TEFS control the behavior of the frame sync and clocks and the corresponding bits in SRCR (RFDIR, RXDIR, RSCKP, RFSI, REFS) are overridden. However, the behavior of shift registers is still controllable independently on Tx and Rx sections (This is for bits TSHFD and TXBIT0 of STCR and RSHFD and RXBIT0 of SRCR).

Figure 45-28 shows the STCR register, and Table 45-12 shows the register's field descriptions.

0x43FA_001C (STCR1)
0x5001_401C (STCR2)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	TXBIT0	TFEN1	TFEN0	TFDIR	TXDIR	TSHFD	TSCKP	TFSI	TFSL	TEFS
W																
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Figure 45-28. SSI Transmit Configuration Register

Table 45-12. SSI Transmit Configuration Register Field Descriptions

Field	Description
31–10	Reserved
9 TXBIT0	Transmit bit 0. This control bit allows the SSI to transmit the data word from bit position 0 or 15/31 in the transmit shift register. The shifting data direction can be MSB or LSB first, controlled by the TSHFD bit. Refer to Section 45.3.3.3, "SSI Transmit Shift Register (TXSR) description for more details on how to configure TXBIT0 and TSHFD. 0 Shifting with respect to bit 31 (if word length = 16, 18, 20, 22, or 24) or bit 15 (if word length = 8, 10, or 12) of the transmit shift register (MSB aligned). 1 Shifting with respect to bit 0 of transmit shift register (LSB aligned).

Table 45-12. SSI Transmit Configuration Register Field Descriptions (continued)

Field	Description
8 TFEN1	Transmit FIFO enable 1. This bit enables transmit FIFO 1. When enabled, the FIFO allows 8 samples to be transmitted by the SSI per channel (a 9th sample can be shifting out) before TDE1 bit is set. When the FIFO is disabled, an interrupt is generated when a single sample is transferred to the transmit shift register (provided the interrupt is enabled). 0 Transmit FIFO 1 is disabled. 1 Transmit FIFO 1 is enabled.
7 TFEN0	Transmit FIFO enable 0. This bit enables transmit FIFO 0. When enabled, the FIFO allows eight samples to be transmitted by the SSI per channel (a 9th sample can be shifting out) before TDE0 bit is set. When the FIFO is disabled, an interrupt is generated when a single sample is transferred to the transmit shift register (provided the interrupt is enabled). 0 Transmit FIFO 0 is disabled. 1 Transmit FIFO 0 is enabled.
6 TFDIR	Transmit frame direction. This bit controls the direction and source of the transmit frame sync signal. The internally generated frame sync signal is sent out through the STFS port and external frame sync is taken from the same port. 0 Frame sync is external. 1 Frame sync is generated internally.
5 TXDIR	Transmit clock direction. This bit controls the direction and source of the clock signal used to clock the TXSR. The internally generated clock is output through the STCK port. The external clock is taken from this port. 0 Transmit clock is external. 1 Transmit clock is generated internally.
4 TSHFD	Transmit shift direction. This bit controls whether the MSB or LSB will be transmitted first in a sample. Refer to Section 45.3.3.3, “SSI Transmit Shift Register (TXSR) description for more details on how to configure TXBIT0 and TSHFD. **Note:** The CODEC device labels the MSB as bit 0, whereas the core labels the LSB as bit 0. Therefore, when using a standard CODEC, the ore MSB (CODEC LSB) is shifted in first (TSHFD cleared). 0 Data is transmitted MSB first. 1 Data is transmitted LSB first.
3 TSCKP	Transmit clock polarity. This bit controls which bit clock edge is used to clock out data for the transmit section. Refer to Figure 45-6 and Figure 45-7 for examples of configuration of TSCKP and TFSI. 0 Data is clocked out on rising edge of bit clock. 1 Data is clocked out on falling edge of bit clock.
2 TFSI	Transmit frame sync invert. This bit controls the active state of the frame sync I/O signal for the transmit section of SSI. 0 Transmit frame sync is active high. 1 Transmit frame sync is active low.

Table 45-12. SSI Transmit Configuration Register Field Descriptions (continued)

Field	Description
1 TFSL	Transmit frame sync length. This bit controls the length of the frame sync signal to be generated or recognized for the transmit section. The length of a word-long frame sync is same as the length of the data word selected by WL[3:0]. Refer to Figure 45-6 and Figure 45-7 for examples of configuration of TSCKP and TFSI. 0 Transmit frame sync is one-word long. 1 Transmit frame sync is one-clock-bit long.
0 TEFS	Transmit early frame sync. This bit controls when the frame sync is initiated for the transmit section. The frame sync signal is deasserted after one bit-for-bit length frame sync and after one word-for-word length frame sync. In case of synchronous operation, the frame sync can also be initiated on receiving the first bit of data. 0 Transmit frame sync is initiated as the first bit of data is transmitted. 1 Transmit frame sync is initiated one bit before the data is transmitted.

45.3.3.11 SSI Receive Configuration Register (SRCR)

The SSI receive configuration register (SRCR) is a read/write control registers used to direct the receive operation of the SSI. The SRCR controls the direction of the bit clock and frame sync ports, SRCK, and SRFS. The interrupt enable bit for the transmit sections is provided in this control register. The power-on reset clears all SRCR bits; however, SSI reset does not affect the SRCR bits. In Synchronous mode, STCR bits TFDIR, TXDIR, TSCKP, TFSI, TEFS control the behavior of the frame sync and clocks and the corresponding bits in SRCR (RFDIR, RXDIR, RSCKP, RFSI, REFS) are overridden. However, the behavior of shift registers is still controllable independently on Tx and Rx sections (This is for bits TSHFD and TXBIT0 of STCR and RSHFD and RXBIT0 of SRCR).

Figure 45-29 shows the SRCR register, and Table 45-13 shows the register's field descriptions.

0x43FA_0020 (SRCR1)
0x5001_4020 (SRCR2)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	RXEXT	RXBIT0	RFEN1	RFEN0	RFDIR	RXDIR	RSHFD	RSCKP	RFSI	RFSL	REFS
W																
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Figure 45-29. SSI Receive Configuration Register

Table 45-13. SSI Receive Configuration Register Field Descriptions

Field	Description
31–11	Reserved
10 RXEXT	Receive data extension. This control bit allows SSI to store the received data word in sign extended form. This bit affects data storage only in case received data is LSB aligned (SRCR[9]=1). 0 Sign extension is turned off. 1 Sign extension is turned on.
9 RXBIT0	Receive bit 0. This control bit allows SSI to receive the data word at bit position 0 or 15/31 in the receive shift register. The shifting data direction can be MSB or LSB first, controlled by the RSHFD bit. Refer to Section 45.3.3.6, “SSI Receive Shift Register (RXSR)) description for more details on how to configure RXBIT0 and RSHFD 0 Shifting with respect to bit 31 (if word length = 16, 18, 20, 22, or 24) or bit 15 (if word length = 8, 10, or 12) of receive shift register (MSB aligned). 1 Shifting with respect to bit 0 of receive shift register (LSB aligned).
8 RFEN1	Receive FIFO enable 1. This bit enables receive FIFO 1. When enabled, the FIFO allows eight samples to be received by the SSI per channel (a 9th sample can be shifting in) before RDR1 bit is set. When the FIFO is disabled, an interrupt is generated when a single sample is received by the SSI (provided the interrupt is enabled). 0 Receive FIFO 1 is disabled. 1 Receive FIFO 1 is enabled.
7 RFEN0	Receive FIFO enable 0. This bit enables receive FIFO 0. When enabled, the FIFO allows 8 samples to be received by the SSI per channel (a 9th sample can be shifting in) before RDR0 bit is set. When the FIFO is disabled, an interrupt is generated when a single sample is received by the SSI (provided the interrupt is enabled). 0 Receive FIFO 0 is disabled. 1 Receive FIFO 0 is enabled.
6 RFDIR	Receive frame direction. This bit controls the direction and source of the receive frame sync signal. Internally generated frame sync signal is sent out through the SRFS port and the external frame sync is taken from the same port. 0 Frame sync is external. 1 Frame sync is generated internally.
5 RXDIR	Receive clock direction. This bit controls the direction and source of the clock signal used to clock the RXSR. The internally generated clock is output through the SRCK port. The external clock is taken from this port. 0 Receive clock is external. 1 Receive clock is generated internally.
4 RSHFD	Receive shift direction. This bit controls whether the MSB or LSB will be received first in a sample. Refer to Section 45.3.3.6, “SSI Receive Shift Register (RXSR)) description for more details on how to configure RXBIT0 and RSHFD. **Note:** The CODEC device labels the MSB as bit 0, whereas the core labels the LSB as bit 0. Therefore, when using a standard CODEC, core MSB (CODEC LSB) is shifted in first (RSHFD cleared). 0 Data received is MSB first. 1 Data received is LSB first.
3 RSCKP	Receive clock polarity. This bit controls which bit clock edge is used to latch in data for the receive section. Refer to Figure 45-20, Figure 45-21, Figure 45-22, and Figure 45-23 for examples of configuration of RSCKP and RFSI. 0 Data is latched on falling edge of bit clock. 1 Data is latched on rising edge of bit clock.

Table 45-13. SSI Receive Configuration Register Field Descriptions (continued)

Field	Description
2 RFSI	Receive frame sync invert. This bit controls the active state of the frame sync I/O signal for the receive section of SSI. Refer to Figure 45-20, Figure 45-21, Figure 45-22, and Figure 45-23 for examples of configuration of RSCKP and RFSI. 0 Receive frame sync is active high. 1 Receive frame sync is active low.
1 RFSL	Receive frame sync length. This bit controls the length of the frame sync signal to be generated or recognized for the receive section. The length of a word-long frame sync is same as the length of the data word selected by WL[3:0]. 0 Receive frame sync is one-word long. 1 Receive frame sync is one-clock-bit long.
0 REFS	Receive early frame sync. This bit controls when the frame sync is initiated for the receive section. The frame sync is disabled after one bit-for-bit length frame sync and after one word-for-word length frame sync. 0 Receive frame sync is initiated one bit before the data is received. 1 Receive frame sync initiated as the first bit of data is received.

45.3.3.12 SSI Transmit and Receive Clock Control Registers (STCCR and SRCCR)

The SSI transmit and receive control (STCCR and SRCCR) registers are 19-bit, read/write control registers used to direct the operation of the SSI. The Clock and Reset Module (CRM) can source the SSI clock (ccm_ssi_clk) from multiple sources and perform fractional division to support commonly used audio bit rates. The CRM can maintain the ccm_ssi_clk frequency at a constant rate even in cases where the ipg_clk frequency changes. These registers control the SSI clock generator, bit and frame sync rates, word length, and number of words per frame for the serial data.

The STCCR register is dedicated to the transmit section, and the SRCCR register is dedicated to the receive section except in synchronous mode, in which the STCCR register controls both the receive and transmit sections. Power-on reset clears all STCCR and SRCCR bits. SSI reset does not affect the STCCR and SRCCR bits. The control bits are described in the following paragraphs. Although the bit patterns of the STCCR and SRCCR registers are the same, the contents of these two registers can be programmed differently.

Figure 45-30 shows the STCCR register, Figure 45-31 shows the SRCCR register, and Table 45-14 shows the registers' field descriptions.

0x43FA_0024 (STCCR1)
0x5001_4024 (STCCR2)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	DIV2	PSR	WL3
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	WL2	WL1	WL0	DC4	DC3	DC2	DC1	DC0	PM7	PM6	PM5	PM4	PM3	PM2	PM1	PM0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 45-30. SSI Transmit Clock Control Register

0x43FA_0028 (SRCCR1)
0x5001_4028 (SRCCR2)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	DIV2	PSR	WL3
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	WL2	WL1	WL0	DC4	DC3	DC2	DC1	DC0	PM7	PM6	PM5	PM4	PM3	PM2	PM1	PM0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 45-31. SSI Receive Clock Control Register

Table 45-14. SSI Transmit and Receive Clock Control Register Field Descriptions

Field	Description
31–19	Reserved
18 DIV2	Divide by 2. This bit controls a divide-by-two divider in series with the rest of the prescalers. 0 Divider is bypassed. 1 Divider is used to divide clock by 2.
17 PSR	Prescaler range. This bit controls a fixed divide-by-eight prescaler in series with the variable prescaler. It extends the range of the prescaler for those cases where a slower bit clock is required. 0 Prescaler is bypassed. 1 Prescaler is used to divide clock by 8.

Table 45-14. SSI Transmit and Receive Clock Control Register Field Descriptions (continued)

Field	Description
16–13 WL3–WL0	Word length control. These bits are used to control the length of the data words being transferred by the SSI. These bits control the word length divider in the clock generator. They also control the frame sync pulse length when the FSL bit is cleared. In I^2S master mode, the SSI works with a fixed word length of 32, and the WL bits are used to control the amount of valid data in those 32 bits. Refer to Table 45-15 for details of data word lengths supported by SSI.
12–8 DC4–DC0	Frame rate divider control. These bits are used to control the divide ratio for the programmable frame rate dividers. The divide ratio works on the word clock. In normal mode, this ratio determines the word transfer rate. In network mode, this ratio sets the number of words per frame. The divide ratio ranges from 1 to 32 in normal mode and from 2 to 32 in network mode. In normal mode, a divide ratio of 1 (DC=00000) provides continuous periodic data word transfer. A bit-length frame sync must be used in this case. These bits can be programmed with values ranging from “00000” to “11111” to control the number of words in a frame.
7–0 PM7–PM0	Prescaler modulus select. These bits control the prescale divider in the clock generator. This prescaler is used only in internal clock mode to divide the internal clock (ccm_ssi_clk). The bit clock output is available at the clock port. A divide ratio from 1 to 256 (PM[7:0] = 0x00 to 0xFF) can be selected. Refer to Section 45.4.2.2, “DIV2, PSR, and PM Bit Description” for details regarding settings.

Table 45-15. SSI Data Length

WL3	WL2	WL1	WL0	Number of Bits/Word	Supported in Implementation
0	0	0	0	2	No
0	0	0	1	4	No
0	0	1	0	6	No
0	0	1	1	8	Yes
0	1	0	0	10	Yes
0	1	0	1	12	Yes
0	1	1	0	14	No
0	1	1	1	16	Yes
1	0	0	0	18	Yes
1	0	0	1	20	Yes
1	0	1	0	22	Yes
1	0	1	1	24	Yes
1	1	0	0	26	No
1	1	0	1	28	No
1	1	1	0	30	No
1	1	1	1	32	No

45.3.3.13 SSI FIFO Control/Status Register (SFCSR)

Figure 45-32 shows the SFCSR register, and Table 45-16 shows the register's field descriptions.

0x43FA_002C (SFCSR1)
0x5001_402C (SFCSR2)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	RFCNT1[3:0]				TFCNT1[3:0]				RFWM1[3:0]				TFWM1[3:0]			
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RFCNT0[3:0]				TFCNT0[3:0]				RFWM0[3:0]				TFWM0[3:0]			
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1

Figure 45-32. SSI FIFO Control/Status Register

Table 45-16. SSI FIFO Control/Status Register Field Descriptions

Field	Description
31–28 RFCNT1[3:0]	Receive FIFO counter 1. These bits indicate the number of data words in receive FIFO 1. Refer to the following for details regarding settings for receive FIFO counter bits. 0000 0 data word in receive FIFO 0001 1 data word in receive FIFO 0010 2 data word in receive FIFO 0011 3 data word in receive FIFO 0100 4 data word in receive FIFO 0101 5 data word in receive FIFO 0110 6 data word in receive FIFO 0111 7 data word in receive FIFO 1000 8 data word in receive FIFO
27–24 TFCNT1[3:0]	Transmit FIFO counter 1. These bits indicate the number of data words in transmit FIFO. Refer to the following for details regarding settings for transmit FIFO counter bits. 0000 0 data word in transmit FIFO 0001 1 data word in transmit FIFO 0010 2 data word in transmit FIFO 0011 3 data word in transmit FIFO 0100 4 data word in transmit FIFO 0101 5 data word in transmit FIFO 0110 6 data word in transmit FIFO 0111 7 data word in transmit FIFO 1000 8 data word in transmit FIFO

Table 45-16. SSI FIFO Control/Status Register Field Descriptions (continued)

Field	Description
23–20 RFWM1[3:0]	Receive FIFO full watermark 1. These bits control the threshold at which the RFF1 flag will be set. The RFF1 flag is set whenever the data level in receive FIFO 1 reaches the selected threshold. Refer to the following for details regarding settings for receive FIFO watermark bits. 0000 Reserved 0001 RFF set when at least one data word has been written to the receive FIFO. Set when RxFIFO = 1, 2, 3, 4, 5, 6, 7, 8 data words. 0010 RFF set when more than or equal to 2 data words have been written to the receive FIFO. Set when RxFIFO = 2, 3, 4, 5, 6, 7, 8 data words. 0011 RFF set when more than or equal to 3 data words have been written to the receive FIFO. Set when RxFIFO = 3, 4, 5, 6, 7, 8 data words. 0100 RFF set when more than or equal to 4 data words have been written to the receive FIFO. Set when RxFIFO = 4, 5, 6, 7, 8 data words. 0101 RFF set when more than or equal to 5 data words have been written to the receive FIFO. Set when RxFIFO = 5, 6, 7, 8 data words. 0110 RFF set when more than or equal to 6 data words have been written to the receive FIFO. Set when RxFIFO = 6, 7, 8 data words. 0111 RFF set when more than or equal to 7 data words have been written to the receive FIFO. Set when RxFIFO = 7, 8 data words. 1000 RFF set when 8 data words have been written to the receive FIFO (default). Set when RxFIFO = 8 data words
19–16 TFWM1[3:0]	Transmit FIFO empty watermark 1. These bits control the threshold at which the TFE1 flag will be set. The TFE1 flag is set whenever the data level in transmit FIFO 1 falls below the selected threshold. Refer to the following for details regarding settings for transmit FIFO watermark bits. 0000 Reserved 0001 TFE set when there are more than or equal to 1 empty slots in transmit FIFO. (default) Transmit FIFO empty is set when TxFIFO <= 7 data. 0010 TFE set when there are more than or equal to 2 empty slots in transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 6 data. 0011 TFE set when there are more than or equal to 3 empty slots in transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 5 data. 0100 TFE set when there are more than or equal to 4 empty slots in transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 4 data. 0101 TFE set when there are more than or equal to 5 empty slots in transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 3 data. 0110 TFE set when there are more than or equal to 6 empty slots in transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 2 data. 0111 TFE set when there are more than or equal to 7 empty slots in transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 1 data. 1000 TFE set when there are 8 empty slots in transmit FIFO. Transmit FIFO empty is set when TxFIFO=0 data.
15–12 RFCNT0[3:0]	Receive FIFO counter 0. These bits indicate the number of data words in receive FIFO 0. Refer to the following for details regarding settings for receive FIFO counter bits. 0000 0 data word in receive FIFO 0001 1 data word in receive FIFO 0010 2 data word in receive FIFO 0011 3 data word in receive FIFO 0100 4 data word in receive FIFO 0101 5 data word in receive FIFO 0110 6 data word in receive FIFO 0111 7 data word in receive FIFO 1000 8 data word in receive FIFO

Table 45-16. SSI FIFO Control/Status Register Field Descriptions (continued)

Field	Description
11–8 TFCNT0[3:0]	Transmit FIFO counter 0. These bits indicate the number of data words in transmit FIFO 0. Refer to the following for details regarding settings for transmit FIFO counter bits. 0000 0 data word in transmit FIFO 0001 1 data word in transmit FIFO 0010 2 data word in transmit FIFO 0011 3 data word in transmit FIFO 0100 4 data word in transmit FIFO 0101 5 data word in transmit FIFO 0110 6 data word in transmit FIFO 0111 7 data word in transmit FIFO 1000 8 data word in transmit FIFO
7–4 RFWM0[3:0]	Receive FIFO full watermark 0. These bits control the threshold at which the RFF0 flag will be set. The RFF0 flag is set whenever the data level in receive FIFO 0 reaches the selected threshold. Refer to the following for details regarding settings for receive FIFO watermark bits. 0000 Reserved 0001 RFF set when at least one data word has been written to the receive FIFO. Set when RxFIFO = 1, 2, 3, 4, 5, 6, 7, 8 data words. 0010 RFF set when more than or equal to 2 data words have been written to the receive FIFO. Set when RxFIFO = 2, 3, 4, 5, 6, 7, 8 data words. 0011 RFF set when more than or equal to 3 data words have been written to the receive FIFO. Set when RxFIFO = 3, 4, 5, 6, 7, 8 data words. 0100 RFF set when more than or equal to 4 data words have been written to the receive FIFO. Set when RxFIFO = 4, 5, 6, 7, 8 data words. 0101 RFF set when more than or equal to 5 data words have been written to the receive FIFO. Set when RxFIFO = 5, 6, 7, 8 data words. 0110 RFF set when more than or equal to 6 data words have been written to the receive FIFO. Set when RxFIFO = 6, 7, 8 data words. 0111 RFF set when more than or equal to 7 data words have been written to the receive FIFO. Set when RxFIFO = 7, 8 data words. 1000 RFF set when 8 data words have been written to the receive FIFO (default). Set when RxFIFO = 8 data words
3–0 TFWM0[3:0]	Transmit FIFO empty watermark 0. These bits control the threshold at which the TFE0 flag will be set. The TFE0 flag is set whenever the data level in transmit FIFO 0 falls below the selected threshold. Refer to the following for details regarding settings for transmit FIFO watermark bits. 0000 Reserved 0001 TFE set when there are more than or equal to 1 empty slots in transmit FIFO. (default) Transmit FIFO empty is set when TxFIFO <= 7 data. 0010 TFE set when there are more than or equal to 2 empty slots in transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 6 data. 0011 TFE set when there are more than or equal to 3 empty slots in transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 5 data. 0100 TFE set when there are more than or equal to 4 empty slots in transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 4 data. 0101 TFE set when there are more than or equal to 5 empty slots in transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 3 data. 0110 TFE set when there are more than or equal to 6 empty slots in transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 2 data. 0111 TFE set when there are more than or equal to 7 empty slots in transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 1 data. 1000 TFE set when there are 8 empty slots in transmit FIFO. Transmit FIFO empty is set when TxFIFO=0 data.

Table 45-17 indicates the status of the transmit FIFO empty flag, with different settings of the transmit FIFO watermark bits and varying amounts of data in the transmit FIFO.

Table 45-17. Status of Transmit FIFO Empty Flag

Transmit FIFO Watermark (TFWM)	Number of Data in TXFIFO								
	0	1	2	3	4	5	6	7	8
1	1	1	1	1	1	1	1	1	0
2	1	1	1	1	1	1	1	0	0
3	1	1	1	1	1	1	0	0	0
4	1	1	1	1	1	0	0	0	0
5	1	1	1	1	0	0	0	0	0
6	1	1	1	0	0	0	0	0	0
7	1	1	1	0	0	0	0	0	0
8	1	0	0	0	0	0	0	0	0

45.3.3.14 SSI AC97 Control Register (SACNT)

Figure 45-33 shows the SACNT register, and Table 45-18 shows the register's field descriptions.

0x43FA_0038 (SACNT1)
0x5001_4038 (SACNT2)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	FRDIV[5:0]						WR	RD	TIF	FV	AC97EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 45-33. SSI AC97 Control Register

Table 45-18. SSI AC97 Control Register Field Descriptions

Field	Description
31–11	Reserved
10–5 FRDIV[5:0]	Frame rate divider. These bits control the frequency of AC97 data transmission/reception. They are programmed with the number of frames for which the SSI should be idle, after operating in one frame. Through these bits, AC97 frequency of operation, from 48 KHz (000000) to 1 KHz (101111) can be achieved. Sample Value: 001010 (10 Decimal) = SSI operates once every 11 frames.

Table 45-18. SSI AC97 Control Register Field Descriptions (continued)

Field	Description
4 WR	Write command. This bit specifies whether the next frame will carry an AC97 Write Command or not. The programmer should take care that only one of the bits (WR or RD) is set at a time. When this bit is set, the corresponding tag bits (corresponding to command address and command data slots of the next transmit frame) are automatically set. This bit is automatically cleared by the SSI after completing transmission of a frame. 0 Next frame will not have a write command. 1 Next frame will have a write command.
3 RD	Read command. This bit specifies whether the next frame will carry an AC97 read command or not. The programmer should take care that only one of the bits (WR or RD) is set at a time. When this bit is set, the corresponding tag bit (corresponding to command address slot of the next transmit frame) is automatically set. This bit is automatically cleared by the SSI after completing transmission of a frame. 0 Next frame will not have a read command. 1 Next frame will have a read command.
2 TIF	Tag in FIFO. This bit controls the destination of the information received in AC97 tag slot (slot #0). 0 Tag information is stored in the SATAG register. 1 Tag information is stored in receive FIFO 0.
1 FV	Fixed/variable operation. This bit selects whether the SSI is in AC97 fixed mode or AC97 variable mode. 0 AC97 fixed mode. 1 AC97 variable mode.
0 AC97EN	AC97 mode enable. This bit is used to enable SSI AC97 operation. Refer to Section 45.1.2.5, “AC97 Mode” for details of AC97 operation. 0 AC97 mode is disabled. 1 SSI is in AC97 mode.

45.3.3.15 SSI AC97 Command Address Register (SACADD)

Figure 45-34 shows the SACADD register, and Table 45-19 shows the register’s field descriptions.

0x43FA_003C (SACADD1)
0x5001_403C (SACADD2)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	SACADD[18:16]		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SACADD[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 45-34. SSI AC97 Command Address Register

Table 45-19. SSI AC97 Command Address Register Field Descriptions

Field	Description
31–19	Reserved
18–0 SACADD	AC97 command address. These bits store the command address slot information (bit 19 of the slot is sent in accordance with the read and write command bits in SACNT register). These bits can be updated by a direct write from the core. They are also updated with the information received in the incoming command address slot. If the contents of these bits change due to an update, the CMDAU bit in SISR is set.

45.3.3.16 SSI AC97 Command Data Register (SACDAT)

Figure 45-35 shows the SACDAT register, and Table 45-20 shows the register's field descriptions.

0x43FA_0040 (SACDAT1)
0x5001_4040 (SACDAT2)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	SACDAT[19:16]			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SACDAT[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 45-35. SSI AC97 Command Data Register

Table 45-20. SSI AC97 Command Data Register

Field	Description
31–20	Reserved
19–0 SACDAT	AC97 command data. The outgoing command data slot carries the information contained in these bits. These bits can be updated by a direct write from the core. They are also updated with the information received in the incoming command data slot. If the contents of these bits change due to an update, the CMDDU bit in SISR is set. In case of an AC97 read command, these bits are cleared for transmission in time slot #2 of AC97 frame.

45.3.3.17 SSI AC97 Tag Register (SATAG)

Figure 45-36 shows the SATAG register, and Table 45-21 shows the register's field descriptions.

0x43FA_0044 (SATAG1)
0x5001_4044 (SATAG2)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SATAG[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 45-36. SSI AC97 Tag Register

Table 45-21. SSI AC97 Tag Register Field Descriptions

Field	Description
31–16	Reserved
15–0 SATAG	AC97 tag. These bits store the tag information if the tag is not stored in the FIFO 0 (if TIF bit in SACNT register is cleared). Writing to this register (by the core) sets the value of the transmit tag (in AC97 fixed mode). On a read, the core gets the last receive tag value. It is updated at the start of each received frame. The contents of this register are also used to generate the transmit tag in the AC97 variable mode of operation. In case the received tag value changes, the RXT bit in SISR register is set, if enabled.

45.3.3.18 SSI Transmit Time Slot Mask Register (STMSK)

Figure 45-37 shows the STMSK register, and Table 45-22 shows the register's field descriptions.

0x43FA_0048 (STMSK1)
0x5001_4048 (STMSK2)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	STMSK[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	STMSK[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 45-37. SSI Transmit Time Slot Mask Register

Table 45-22. SSI Transmit Time Slot Mask Register Field Descriptions

Field	Description
31–0 STMSK	Transmit mask. These bits indicate which slot has been masked in the current frame. The core can write to this register to control the time slots in which the SSI transmits data. Each bit has information corresponding to the respective time slot in the frame. If a change is made to the register contents, the transmission pattern is updated from the next time slot. Transmit mask bits should not be used in an I^2S slave mode of operation. 0 Valid time slot. 1 Time slot is masked (no data transmitted in this time slot).

45.3.3.19 SSI Receive Time Slot Mask Register (SRMSK)

Figure 45-38 shows the SRMSK register, and Table 45-23 shows the register's field descriptions.

0x43FA_004C (SRMSK1)
0x5001_404C (SRMSK2)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SRMSK[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SRMSK[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 45-38. SSI Receive Time Slot Mask Register

Table 45-23. SSI Receive Time Slot Mask Register Field Descriptions

Field	Description
31–0 SRMSK	Receive mask. These bits indicate which slot has been masked in the current frame. The core can write to this register to control the time slots in which the SSI receives data. Each bit has information corresponding to the respective time slot in the frame. If a change is made to the register contents, the reception pattern is updated from the next time slot. Receive mask bits should not be used in an I^2S slave mode of operation. 0 Valid time slot. 1 Time slot is masked (no data received in this time slot).

45.4 Functional Description

45.4.1 SSI Architecture

The SSI is connected to chip pads through the digital audio MUX (AUDMUX) module. The AUDMUX can be configured to connect the SSI module to the chip pads in various ways. Refer to Figure 45-1 for a block diagram of the SSI.

SSI1 is connected to Port 1 of the AUDMUX and SSI2 is connected to Port2 of the AUDMUX.

There are 6 signals from each SSI going into the AUDMUX (x being 1 or 2 depending on the SSI that is considered):

- STXDx: Transmit data from the SSI x
- STFSx: Transmit Frame Sync for SSI x
- STCLKx: Transmit Clock for SSI x
- SRXDx: Receive data for SSI x
- SRFSx: Receive Frame Sync for SSIx
- SRCLKx: Receive Clock for SSI x

45.4.2 SSI Clocking

The SSI uses the following clocks:

- Bit clock—Used to serially clock the data bits in and out of the SSI port. This clock is either generated internally (from ccm_ssi_clk) or taken from external clock source (through the transmit/receive clock ports).
- Word clock—Used to count the number of data bits per word (8, 10, 12, 16, 18, 20, 22, or 24 bits). This clock is generated internally from the bit clock.
- Frame clock (frame sync)—Used to count the number of words in a frame. This signal can be generated internally from the bit clock, or taken from external source (from the transmit/receive frame sync ports).
- Sys clock—In master mode, this is an integer multiple of frame clock. This is ccm_ssi_clk. It is used in cases when SSI must provide the clock. The ccm_ssi_clk signal is the output of the SSIxDIV dividers in the Clock Control Module.

Care should be taken to ensure that the bit clock frequency (either internally generated by dividing the ccm_ssi_clk or sourced from external device through transmit/receive clock ports) is never greater than 1/4 of the ipg_clk frequency.

In normal mode (SCR[6:5]=00), the bit clock, used to serially clock the data, is visible on the serial transmit clock (STCK) and serial receive clock (SRCK) ports. The word clock is an internal clock used to determine when transmission of an 8, 10, 12, 16, 18, 20, 22, or 24-bit word has completed. The word clock, in turn, then clocks the frame clock, which counts the number of words in the frame. The frame clock can be viewed on the STFS and SRFS frame sync ports because a frame sync is generated after the correct number of words in the frame have passed.

In master and synchronous mode, the unused port SRCK is used as serial system clock (SYS_CLK) enabled by the SCR register bit 15, SYS_CLK_EN. This serial system clock is an oversampling clock of the frame sync clock (STFS). In this mode, the word length (WL), prescaler range (PSR), prescaler modulus (PM), and frame rate (DC) select the ratio of SYS_CLK to sampling clock STFS. In the case of I^2S mode, the oversampling clock ccm_ssi_clk can be made available on this port if the SYS_CLK_EN bit is set. The relationship between the clocks and the dividers is shown in Figure 45-39. The bit clock can be received from an SSI clock port or can be generated from the ccm_ssi_clk through a divider, as shown in Figure 45-40.

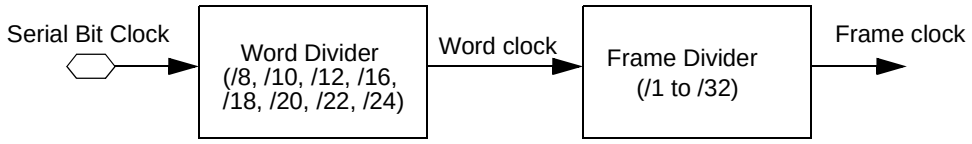


Figure 45-39. SSI Clocking

45.4.2.1 SSI Clock and Frame Sync Generation

Data clock and frame sync signals can be generated internally, or can be obtained from external sources. If internally generated, the SSI clock generator is used to derive bit clock and frame sync signals from the ccm_ssi_clk clock. The SSI clock generator consists of a selectable, fixed prescaler and a programmable prescaler for bit rate clock generation. In gated clock mode, the data clock is valid only when data is being transmitted. Otherwise, the clock port is pulled to the inactive state. A programmable frame rate divider and a word length divider are used for frame rate sync signal generation.

Figure 45-40 shows a block diagram of the clock generator for the transmit section. The serial bit clock can be internal or external, depending on the transmit direction (TXDIR) bit in the SSI transmit configuration register (STCR). The receive section contains an equivalent clock generator circuit.

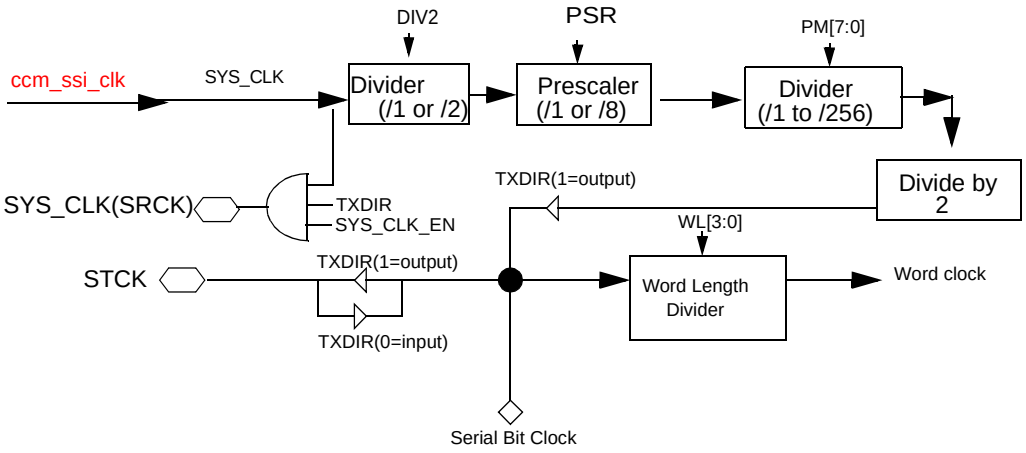


Figure 45-40. SSI Transmit Clock Generator Block Diagram

See Figure 45-41 shows the frame sync generator block for the transmit section. When internally generated, both receive and transmit frame syncs are generated from the word clock and are defined by the

frame rate divider (DC[4:0]) bits and the word length (WL[3:0]) bits of the SSI transmit clock control register (STCCR). The receive section contains an equivalent circuit for the frame sync generator.

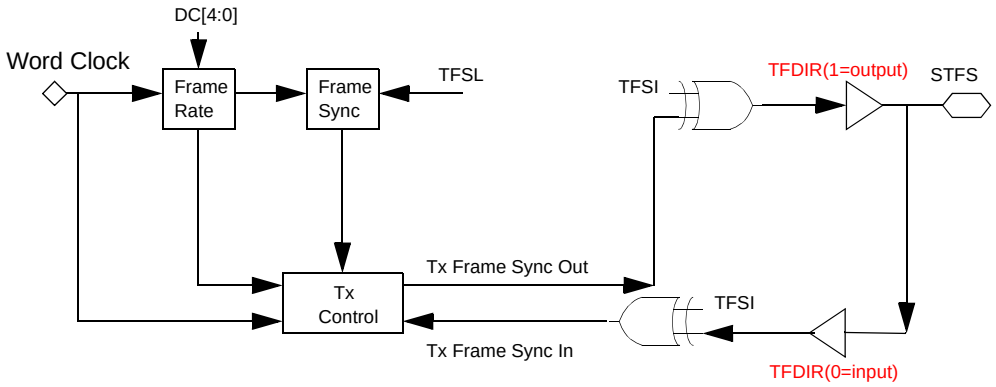


Figure 45-41. SSI Transmit Frame Sync Generator Block Diagram

45.4.2.2 DIV2, PSR, and PM Bit Description

The bit clock frequency can be calculated from the SSI serial system clock (ccm_ssi_clk) using the equation in Figure 45-42.

NOTE

You must ensure that the bit-clock frequency is ¼ times of the ipg_clk frequency. The oversampling clock frequency can go up to jpg_clk frequency. Bits DIV2, PSR, and PM should not be all set to zero at the same time.

$$f_{INT\_BIT\_CLK} = f_{SYS\_CLK} / [(DIV2 + 1) \times (7 \times PSR + 1) \times (PM + 1) \times 2]$$

where PM=PM[7:0]

$$f_{FRAME\_SYN\_CLK} = (f_{INT\_BIT\_CLK}) / [(DC + 1) \times WL]$$

where DC = DC[4:0] and WL = 8, 10, 12, 16, 18, 20, 22, 24

Figure 45-42. SSI Bit Clock Equation

For example, if the SSI working clock SYS_CLK (ccm_ssi_clk) is 12.288, in 8-bit word normal mode with DC[4:0] set to 1 (00001), PM[7:0] set to 47 (0010 1111), the PSR bit cleared, DIV2 bit set to 1, a bit clock rate of 12.288 MHz / [1 x 4 x 48] = 64 kHz is generated. Since the 8-bit word rate is equal to one (normal mode), the sampling rate (FS rate) would then be 64 kHz / [1 * 8] = 8 kHz.

In next example, the SYS_CLK (ccm_ssi_clk) clock is 11.2896 MHz. A 16-bit word network mode with DC[4:0] set to 1 (00001), PM[7:0] set to 3 (0000 0011), the PSR bit is set to 0, DIV2 bit set to 0, and a 11.2896 MHz SYS_CLK clock, a bit clock rate of 11.2896 MHz / [1 x 2 x 4] = 1.4112 MHz is generated. Since the 16-bit word rate is equal to two, the sampling rate (FS rate) would be 1.4112 MHz / [2 * 16] = 44.1 kHz.

Table 45-24 shows programming examples for the clock dividers in the CRM and the SSI to support various bit clock (STCK) frequencies.

Table 45-24. SSI Bit Clock and Frame Rate as a Function of PSR, PM, and DIV2

Bits/ Word	Words/ Frame	Ideal Frame Rate (kHz)	PLL Freq (MHz)	SSIDIV (in CRM)	MCLK/ ccm_ssi_clk Freq (MHz)	DIV2	PSR	PM	WL	DC	Actual Bit_Clk Freq (kHz) STCK	Target Bit_Clk Freq (kHz) STCK	Error (Hz)
16	1	8	294.912	48	12.288	0	0	47	7	0	128	128	0
16	2	8	294.912	48	12.288	0	0	23	7	1	256	256	0
16	4	8	294.912	48	12.288	0	0	11	7	3	512	512	0
16	1	12	294.912	48	12.288	0	0	31	7	0	192	192	0
16	2	12	294.912	48	12.288	0	0	15	7	1	384	384	0
16	4	12	294.912	48	12.288	0	0	7	7	3	768	768	0
16	1	16	294.912	48	12.288	0	0	23	7	0	256	256	0
16	2	16	294.912	48	12.288	0	0	11	7	1	512	512	0
16	4	16	294.912	48	12.288	0	0	5	7	3	1024	1024	0
16	1	24	294.912	48	12.288	0	0	15	7	0	384	384	0
16	2	24	294.912	48	12.288	0	0	7	7	1	768	768	0
16	4	24	294.912	48	12.288	0	0	3	7	3	1536	1536	0
16	1	32	294.912	48	12.288	0	0	11	7	0	512	512	0
16	2	32	294.912	48	12.288	0	0	5	7	1	1024	1024	0
16	4	32	294.912	48	12.288	0	0	2	7	3	2048	2048	0
16	1	48	294.912	48	12.288	0	0	15	7	0	768	768	0
16	2	48	294.912	48	12.288	0	0	3	7	1	1536	1536	0
16	4	48	294.912	48	12.288	0	0	1	7	3	3072	3072	0
16	1	11.025	270.9504	48	11.2896	0	0	31	7	0	176.4	176.4	0
16	2	11.025	270.9504	48	11.2896	0	0	15	7	1	352.8	352.8	0
16	4	11.025	270.9504	48	11.2896	0	0	7	7	3	705.6	705.6	0
16	1	22.05	270.9504	48	11.2896	0	0	15	7	0	352.8	352.8	0
16	2	22.05	270.9504	48	11.2896	0	0	7	7	1	705.6	705.6	0
16	4	22.05	270.9504	48	11.2896	0	0	3	7	3	1411.2	1411.2	0
16	1	44.1	270.9504	48	11.2896	0	0	7	7	0	705.6	705.6	0
16	2	44.1	270.9504	48	11.2896	0	0	3	7	1	1411.2	1411.2	0
16	4	44.1	270.9504	48	11.2896	0	0	1	7	3	2822.4	2822.4	0

NOTE

Table 45-24 shows how various frame rates can be achieved with the SPLL supplying a frequency of 294.912 MHz and 270.9504 MHz (with WL and DC settings as shown).These clocks are recommended as convenient starting points, but the system allows for other input clock frequencies as well.

Table 45-25 shows programming of the CRM and SSI dividers in order to generate the appropriate SYS_CLK and BIT_CLK frequencies for various sampling rates. In these examples, the master mode is selected either by setting I^2S master bit (SCR[6:5]=01) or individually programming the SSI in network, synchronous, transmit internal mode (the table specifically illustrates the I^2S mode frequencies/sample rates). The SYS_CLK clock is ccm_ssi_clk.

NOTE

The I^2S master mode requires that a word length of 32 bits be used (regardless of the actual data type). Consequently, the fixed I^2S frame rate of 64 bits per frame (word length [WL] can be any value) and DC of 1 are assumed.

Table 45-25. SSI System Clock, Bit Clock, Frame Clock in Master Mode

Bits/ Word	Words/ Frame	Ideal Sampling Rate (kHz)	Over Sampling Rate	PLL Freq (MHz)	SSIDIV (in CRM)	Target MCLK Freq (MHz)	Actual MCLK Freq (MHz)	Bits/ Word	Words/ Frame	Actual Sampling Rate (kHz) STFS	Error (Hz)
32	2	22.05	512	270.9504	48	11.2896	11.2896	32	2	44.117	17.65
32	2	24	512	294.912	48	12.288	12.288	32	2	22.058	8.82
32	2	32	384	294.912	48	12.288	12.288	32	2	11.029	4.41
32	2	32	512	294.912	36	16.384	16.384	32	2	48.000	0

NOTE

Table 45-25 shows how various frame rates can be achieved with the SPLL supplying a frequency of 294.912 MHz and 270.9504 MHz. These clocks are recommended as convenient starting points but the system allows for other input clock frequencies as well.

45.4.3 Receive Interrupt Enable Bit Description

When the RIE and RE bits are set, the processor is interrupted when either of the SSI receive FIFO full (RFF0/1) bits in SISR is set (if the corresponding receive FIFO is enabled). If the receive FIFO is not enabled, the interrupt is generated when the corresponding SSI receive data ready (RDR0/1) bit in the SISR is set.

When the receive FIFO is enabled, a maximum of eight values are available to be read (8 values per channel in two-channel mode). If not enabled, then one value can be read from the SRX register (one each

in case of two-channel mode). If the RIE bit is cleared, these interrupts are disabled. However, the RFF0/1 and RDR0/1 bits still indicate the receive data register full condition.

Reading the SRX registers clears the RDR bits, thus clearing the pending interrupt. Two receive data interrupts (two per channel in case of two-channel mode) are available: receive data with exception status and receive data without exception. Table 45-26 and Table 45-27 show the conditions under which these interrupts are generated.

Table 45-26. SSI Receive Data 1 Interrupts

Interrupt	RIE	ROE0	RFF0/RDR0
Receive Data 1(with Exception Status)	1	1	1
Receive Data 1(without exception)	1	0	1

Table 45-27. SSI Receive Data 0 Interrupts

Interrupt	RIE	ROE1	RFF1/RDR1
Receive Data 0 (with Exception Status)	1	1	1
Receive Data 0 (without exception)	1	0	1

45.4.4 Transmit Interrupt Enable Bit Description

The SSI transmit interrupt enable (TIE) control bit determines whether the processor is interrupted when the SSI transmitter needs to be serviced. When the TIE and TE bits are set, the program controller is interrupted when either of the SSI transmit FIFO empty (TFE0/1) flags in SISR are set (if corresponding transmit FIFO is enabled). If the corresponding transmit FIFO is not enabled, an interrupt is generated when the corresponding SSI transmit data register empty (TDE0/1) flag in the SISR is set and transmit enable (TE) bit is set.

When transmit FIFO 0 is enabled, a maximum of eight values can be written to the SSI (8 per channel in case of two-channel mode, using transmit FIFO 1). If not enabled, then one value can be written to the STX0 register (one per channel in case of two-channel mode using STX1). When the TIE bit is cleared, all transmit interrupts are disabled. However, the TDE0/1 bits always indicate the corresponding STX register empty condition, even when the transmitter is disabled by the transmit enable (TE) bit (in the SCR).

Writing data to the STX clears the corresponding TDE bit, thus clearing the interrupt. Two transmit data interrupts are available (four in case of two-channel mode, two per channel): transmit data with exception status and transmit data without exceptions. Table 45-28 and Table 45-29 show the conditions under which these interrupts are generated.

Table 45-28. SSI Transmit Data 1 Interrupts

Interrupt	TIE	TUE1	TFE1/TDE1
Transmit Data 1 (with Exception Status)	1	1	1
Transmit Data 1 (without exception)	1	0	1

Table 45-29. SSI Transmit Data 0 Interrupts

Interrupt	TIE	TUE0	TFE0/TDE0
Transmit Data 0 (with Exception Status)	1	1	1
Transmit Data 0 (without exception)	1	0	1

45.4.5 IP Bus Interface

The SSI has an IP bus interface compliant with SRS 3.0.2 in order to provide a control and data interface. This interface is used by both the processor and DMA controller.

45.4.5.1 Transfer Lengths Supported

The IP bus interface of the SSI supports only 32-bit transfers with all SSI registers other than STX0, STX1, SRX0, and SRX1 (that is, the data registers). With the exception of the data registers, using 8-bit and 16-bit transactions could result in undesired behavior but will not result in a transfer bus error. The data registers (STX0, STX1, SRX0, and SRX1) support 8-bit, 16-bit, and 32-bit transfer lengths without restrictions.

45.4.5.2 Transfer Bus Errors

Transfer bus errors are generated upon response to the following:

- Write transfer to a read-only register.
- Read or write access to a register space beyond the last populated register of the SSI in its memory map (up until the end of the allocated memory address range of the SSI).

45.4.5.3 Clock Rate

The IP bus clock frequency must be at least four times the serial bit clock frequency.

45.5 Initialization/Application Information

The SSI is affected by the following types of reset:

- Power-on reset—The power-on reset is generated by asserting the RESET port. The power-on reset clears the SSIEN bit in SCR, which disables the SSI. All other status and control bits in the SSI are affected as described in SSI programming model in Section 45.3, “Memory Map and Register Definition.”
- SSI reset—The SSI reset is generated when the SSIEN bit in the SCR is cleared. The SSI status bits are preset to the same state produced by the power-on reset. The SSI control bits are unaffected. The control bits in the SCR are also unaffected. The SSI reset is useful for selective reset of the SSI without changing the present SSI control bits and without affecting the other peripherals.

The correct sequence to initialize the SSI is as follows:

1. Issue a power-on or SSI reset (SCR[SSIEN]=0).
2. Disable SSI clocks (ipg_clk, ccm_ssi_clk).
3. Set all control bits for configuring the SSI. (See Table 45-30.)

4. Enable appropriate interrupts/DMA requests through SIER.
5. Set the SCR[SSIEN] bit (=1) to enable the SSI.
6. Enable SSI clocks (ipg_clk, ccm_ssi_clk), as required.
7. In the case of AC97 mode, set the SACNT[AC97EN] bit after programming the SATAG register (if needed for AC97 fixed mode).
8. Set SCR[TE/RE] bits.
9. To ensure proper operation of the SSI, use the “Power-on or SSI reset before changing any of the SSI Control bits listed in Table 45-30.

NOTE

These control bits should not be changed when SSI is enabled.

Table 45-30. SSI Control Bits Requiring SSI To Be Disabled Before Change

Control Register	Bit
SCR	[9]=CLK_IST [8]=TCH_EN [7]=SYS_CLK_EN [6:5]=I2S_MODE [4]=SYN [3]=NET
SIER	[22]=RDMAE [20]=TDMAE
SRCR STCR	[9]=RXBIT0 [9]=TXBIT0 [8]=RFEN1 [8]=TFEN1 [7]=RFEN0 [7]=TFEN0 [6]=RFDIR [6]=TFDIR [5]=RXDIR [5]=TXDIR [4]=RSHFD [4]=TSHFD [3]=RSCKP [3]=TSCKP [2]=RFSI [2]=TFSI [1]=RFSL [1]=TFSL [0]=REFS [0]=TEFS
SRCCR STCCR	[16]=WL3 [15]=WL2 [14]=WL1 [13]=WL0
SACNT	[1]=FV [10:5]=FRDIV

Chapter 46 Graphics Accelerator (MBX R-S)

NOTE

This chapter contains third-party content only, and is provided by—and used with permission from—ARM Ltd. and Imagination Technologies Ltd.

This chapter introduces the MBX R-S 3D Graphics Core (GX020) revision r1p2. It contains the following sections:

- Section 46.1, "About the MBX R-S 3D Graphics Core"
- Section 46.2, "Features of the MBX R-S 3D Graphics Core"
- Section 46.3, "MBX R-S Functional Description"

NOTE

The MBX R-S is not available in the i.MX31L Multimedia Applications Processor.

46.1 About the MBX R-S 3D Graphics Core

The MBX R-S 3D Graphics Core is an *Advanced Microcontroller Bus Architecture* (AMBA) compliant *System-on-a-Chip* (SoC) component. Figure 46-1 shows a top-level block diagram of the MBX R-S 3D Graphics Core.

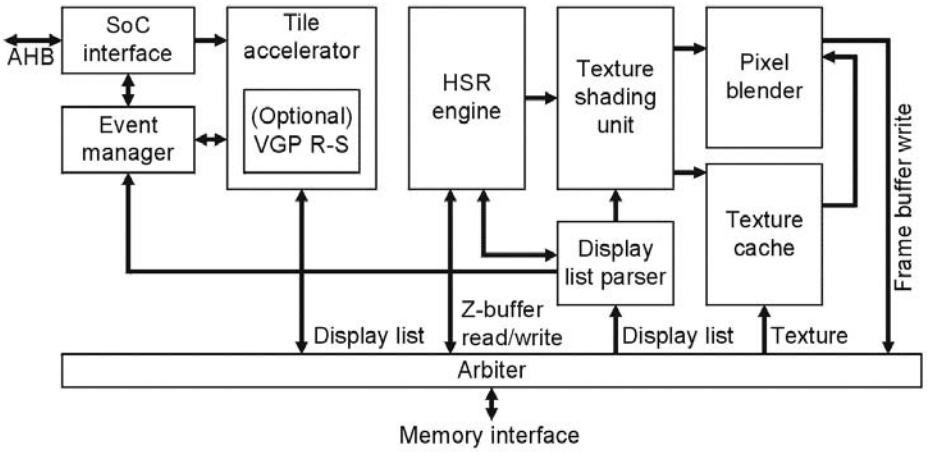


Figure 46-1. MBX R-S 3D Graphics Core Top-level Block Diagram

The MBX R-S 3D Graphics Core consists of the following modules:

- *Tile Accelerator* (TA)
- Event manage

- Display list parser
- *Hidden Surface Removal* (HSR) engine
- Texture shading unit
- Texture cache
- Pixel blender

The MBX R-S 3D Graphics Core operates on 3D scene data (sent as batches of triangles) that are transformed and lit either by the *Central Processing Unit* (CPU) or by the VGP R-S optional. Triangles are written directly to the TA on a *First In First Out* (FIFO) basis so that the CPU is not stalled. The TA performs advanced culling on triangle data by writing the tiled non-culled triangles to the external memory.

The HSR engine reads the tiled data and implements per-pixel HSR with full Z-accuracy. The resulting visible pixels are textured and shaded in *Internal True Color* (ITC) before rendering the final image for display.

46.2 Features of the MBX R-S 3D Graphics Core

The MBX R-S 3D Graphics Core has the following features:

- Deferred texturing
- Screen tiling
- Flat and Gouraud shading
- Perspective correct texturing
- Specular highlights
- Floating-point Z-buffer
- 32-bit ARGB internal rendering and layer buffering
- Full tile blend buffer
- Z-load and store mode
- Per-vertex fog
- 16-bit RGB textures, 1555, 565, 4444, 8332, 88
- 32-bit RGB textures, 8888
- YUV 422 textures
- PVR-TC compressed textures
- One-bit textures for text acceleration
- Point, bilinear, trilinear and anisotropic filtering
- Full range of OpenGL and *Direct3D* (D3D) blend modes
- Dot3 bump mapping
- Alpha test
- Zero-cost full-scene anti-aliasing
- 2Dvia3D

The MBX R-S 3D Graphics Core provides the following benefits:

- Low-memory bandwidth
- High-quality images on low-resolution displays
- Suitable for UMA system
- Easy integration with AMBA multi-layer AHB.

46.3 MBX R-S Functional Description

This section provides a functional description of the MBX R-S 3D Graphics Core. It contains the following sections:

- Section 46.3.1, "Functional Overview"
- Section 46.3.2, "Memory Map"

46.3.1 Functional Overview

Figure 46-2 shows the MBX R-S 3D Graphics Core block diagram. It is a deferred rendering device that uses tile-based display lists. The MBX R-S 3D Graphics Core operates on 3D scene data (sent as batches of triangles) that are transformed and lit either by the CPU or by the optional VGP R-S subcomponent, when included in the MBX R-S 3D Graphics Core.

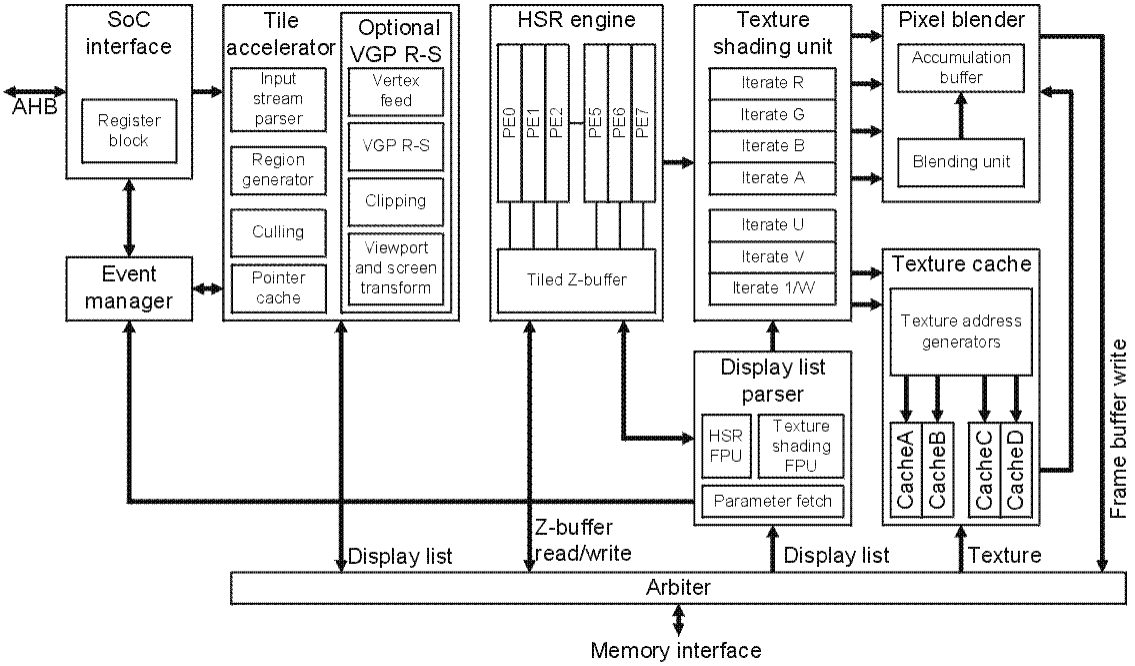


Figure 46-2. MBX R-S Block Diagram

The render surface is broken down into 8x16 tiles (or regions), each one having its own display list containing the data for the objects within that tile. Therefore, before a render can start, the object data for the whole scene must be tiled. Triangles are written directly to the TA through an AHB slave port so that

the CPU is not stalled. The TA performs advanced culling on triangle data by writing the tiled non-culled triangles to the external memory. That is, a bounding box for each object must be calculated and then the object inserted into the display lists for the tiles that overlap the box.

The HSR engine reads the tiled data and implements per-pixel HSR with full Z-accuracy. The resulting visible pixels are textured and shaded in ITC before rendering the final image for display.

The normal steps to rendering 3D scenes are as follows:

1. Host software (application and driver) creates polygon data, creates and loads associated textures, and prepares the scene data for the TA and 3D core to process.
2. The TA reads the 3D parameters and creates a tiled representation of these in frame buffer memory.
3. The *Image Synthesis Processor* (ISP) reads each tile in turn. For each one it reads the 3D object geometry data and performs HSR.
4. The ISP passes tags to the *Texture Shading Processor* (TSP) for all visible parts of polygons. The TSP reads the 3D object data and associated textures, then textures and blends the resulting pixels. Rendered pixels are stored in an accumulation buffer to enable arbitrary numbers of layers to be added to objects before the final pixels are written out to frame buffer.

The MBX R-S 3D Graphics Core includes an advanced event manager that uses SmartBuffer technology so that any level of scene complexity can be handled in a fixed memory buffer size.

46.3.2 Memory Map

The MBX R-S 3D Graphics Core memory map is described in the following sections:

- Section 46.3.2.1, “AHB Slave Interface”
- Section 46.3.2.2, “GX Port Memory Interface”

46.3.2.1 AHB Slave Interface

Its main SoC interface has an input address range of 64 Mbyte which is arranged as shown in Table 46-1.

46.3.2.2 GX Port Memory Interface

The MBX R-S 3D Graphics Core can access memory with or without a *Memory Management Unit* (MMU):

- Native access to 32 Mbyte of external memory with no MMU
- Access up to 4 Gbyte with an MMU. The MMU maps 4 Kbyte pages in a 32 Mbyte linear address space into 4 Kbyte pages in a 4 Gbyte fragmented address space.

Translation is performed using a table with 8192 entries, one for each 4 Kbyte page in a 32 Mbyte linear address space. Each entry is a 32-bit word, so 32 Kbyte must be allocated for the translation table. Table entries are byte-aligned.

The translation table resides in the 4 Gbyte fragmented address space. Eight static registers point to eight 4 Kbyte pages containing the table. A 256-entry, four-way set-associative cache of the translation table is maintained within the MBX R-S 3D Graphics Core core.

The MMU is bypassed on reset. The 32 Mbyte linear address space is mapped directly in to the bottom of the 4 Gbyte address space. When enabled, approximately 256 cycles are required to initialize the cache. A flag, made available in the MMU enable register, is set when the cache is ready. If the cache is subsequently disabled, the MMU returns to the bypass state.

To ensure that updates to the translation table are used immediately, the MMU entry updated must be invalidated using the following registers:

- MMU Index Invalidate
- MMU PHYS Add Invalidate

Figure 46-3 shows the MMU address translation.

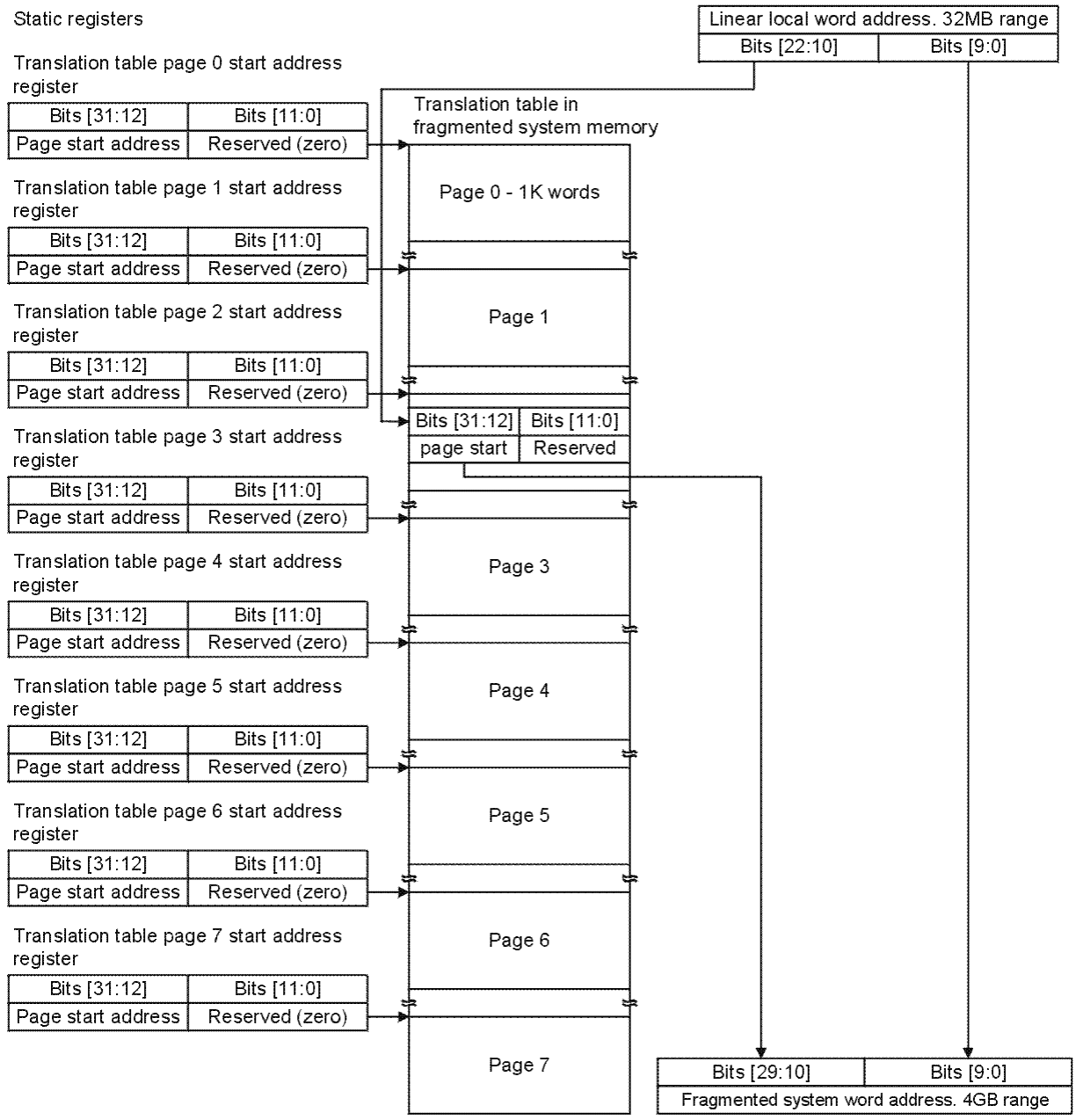


Figure 46-3. MMU Address Translation