

Armadillo-400 Series Software Manual

**Version 1.3.0
2010/11/05**

Atmark Techno, Inc.

Armadillo Developers Site

Armadillo-400 Series Software Manual

Atmark Techno, Inc.

060-0035 AFT Bldg. 6F, N5E2, Chuo-ku, Sapporo
TEL 011-207-6550 FAX 011-207-6570

© 2010 Atmark Techno, Inc.

Version 1.3.0
2010/11/05

1. Preface	9
1.1. Who Should Read This Document	9
1.2. Document Structure	9
1.3. Typographical Conventions	9
1.3.1. Fonts	9
1.3.2. Command Entry Examples	10
1.3.3. Icons	10
1.4. Acknowledgements	10
1.5. Software Usage Precautions	11
1.6. Trademarks	11
2. System Overview	12
2.1. Armadillo-400 Series Basic Specifications	12
2.2. Basic Specifications of Armadillo-420 Basic Model	13
2.3. Basic Specifications of Armadillo-440 LCD Model	15
2.4. Memory Map	18
2.5. Software Make-up	18
2.5.1. Bootloader	19
2.5.2. Kernel	19
2.5.3. Userland	19
2.5.4. Downloader	19
2.6. Boot Modes	19
3. Before Getting Started	21
3.1. Preparation	21
3.2. Connections	21
3.3. Serial Console Software Configuration	23
4. Development Environment Set-up	24
4.1. Installing Cross Development Environment Packages	24
4.2. Installing Packages Required for Atmark-Dist Builds	25
4.3. Installing Cross Development Library Packages	25
5. Rewriting Flash Memory	27
5.1. Flash Memory Writing Regions	27
5.2. Installing Downloader	28
5.2.1. For Linux	28
5.2.2. For Windows	28
5.3. Rewriting Flash Memory With A Downloader	28
5.3.1. Preparation	28
5.3.2. For Linux	28
5.3.3. For Windows	29
5.4. Rewriting Flash Memory With tftpd	30
5.5. Rewriting Flash Memory With netflash	31
5.6. Restoring Bootloader to Factory State	32
5.6.1. Preparation	32
5.6.2. For Linux	32
5.6.3. For Windows	33
6. Building	35
6.1. Building Kernel and Userland Images	35
6.1.1. Preparing Source Code	35
6.1.2. Applying Default Configuration	35
6.1.3. Building	37
6.1.4. Customizing Images	37
6.1.5. Adding An Application To The Userland Image	42
6.2. Building Bootloader Images	43
6.2.1. Preparing Source Code	43

- 6.2.2. Building 43
- 7. Kernel and Userland Placement 44
 - 7.1. Loading From A TFTP Server 44
 - 7.1.1. File Placement 44
 - 7.1.2. Boot Options 44
 - 7.2. Loading From Storage 45
 - 7.2.1. Partitioning 45
 - 7.2.2. Creating Filesystems 46
 - 7.2.3. Kernel Image Placement 47
 - 7.2.4. Creating A Root Filesystem 47
 - 7.2.5. Boot Device and Kernel Parameter Settings 49
 - 7.3. Return Settings To Defaults 49
- 8. Linux Kernel Device Driver Specifications 51
 - 8.1. UART 51
 - 8.2. Ethernet 53
 - 8.3. SD/MMC/SDIO Host 53
 - 8.4. USB 2.0 Host 53
 - 8.5. Frame Buffer 54
 - 8.6. LED Backlight 54
 - 8.7. Touchscreen 54
 - 8.8. Audio 55
 - 8.9. GPIO 56
 - 8.9.1. GPIO sysfs 56
 - 8.9.2. Armadillo-200 Series Compatible GPIO Driver 59
 - 8.10. LED 60
 - 8.10.1. LED Class 60
 - 8.10.2. Armadillo-200 Series Compatible LED Driver 61
 - 8.11. Buttons 61
 - 8.12. Real-time Clock 62
 - 8.13. Watchdog Timer 62
 - 8.14. I2C 62
 - 8.15. SPI 63
 - 8.16. One Wire 64
 - 8.17. PWM 64
 - 8.18. CAN 65
 - 8.19. Keypad 67
 - 8.20. Power Management 68
 - 8.20.1. Treatment of External Devices During Sleep 69
- A. Hermit-At Bootloader 70
 - A.1. version 70
 - A.1.1. version Example 70
 - A.2. info 70
 - A.2.1. info Example 70
 - A.3. memmap 70
 - A.3.1. memmap Example 71
 - A.4. mac 71
 - A.4.1. mac Example 71
 - A.5. md5sum 71
 - A.5.1. md5sum Example 71
 - A.6. erase 72
 - A.6.1. erase Example 72
 - A.7. setenv and clearenv 72
 - A.7.1. setenv/clearenv Example 73
 - A.7.2. Linux Kernel Parameters 73
 - A.8. setbootdevice 73

A.8.1. setbootdevice Example 74

A.9. frob 74

A.10. tftpd 74

 A.10.1. tdtpd Example 75

A.11. tftpboot 75

 A.11.1. tdtboot Example 76

A.12. boot 76

 A.12.1. boot Example 77

A.13. Note On Versions 77

2.1. Armadillo-420/440 Block Diagram	13
2.2. Basic Layout of Armadillo-420 Basic Model	14
2.3. Basic Layout of Armadillo-440 LCD Model	16
3.1. Armadillo-440 LCD Model Connections	22
3.2. Armadillo-420 Basic Model Connections	23
4.1. Install Command	24
4.2. Installed Version Display Command	25
4.3. Cross Development Library Package Creation	26
4.4. Installing Cross Development Library Packages	26
4.5. apt-cross Command	26
5.1. Installing Downloader (Linux)	28
5.2. Download Command	29
5.3. Download Command (With Port Option)	29
5.4. Download Command (Unprotected)	29
5.5. Hermit-At Win32: Download Window	30
5.6. Hermit-At Win32: Download Dialog	30
5.7. tftpd Command Example	31
5.8. netflash Command Example	32
5.9. shoehorn Command Example	32
5.10. shoehorn Command Log	33
5.11. Hermit-At Win32: Shoehorn Window	34
5.12. Hermit-At Win32: shoehorn Dialog	34
6.1. Source Code Preparation	35
6.2. Atmark-Dist Build	37
6.3. Atmark-Dist Configuration	37
6.4. menuconfig: Main Menu	38
6.5. menuconfig: Kernel/Library/Defaults Selection	39
6.6. menuconfig: Do you wish to save your new kernel configuration?	40
6.7. menuconfig: Linux Kernel Configuration	41
6.8. menuconfig: Userland Configuration	42
6.9. Userland Image Customization	43
6.10. Hermit-At Source Archive Extraction	43
6.11. Hermit-At Build Example	43
7.1. tftpboot Command	44
7.2. tftpboot Command Example	45
7.3. Partitioning Procedure	46
7.4. Filesystem Creation Procedure	47
7.5. Kernel Image Placement	47
7.6. Root Filesystem Creation With Debian Archives	48
7.7. Root Filesystem Creation With Atmark-Dist Image	49
7.8. Boot Device Designation	49
7.9. Root Filesystem Designation Example	49
7.10. Assigning Flash Memory As Boot Device	50
7.11. Return Kernel Parameters To Default State With clearenv	50
8.1. GPIO sysfs Interrupt Sample Program	58
8.2. CAN Transmission Speed Calculation	67
A.1. version Syntax	70
A.2. version Example	70
A.3. info Syntax	70
A.4. info Example	70
A.5. memmap Syntax	71
A.6. memmap Example	71
A.7. mac Syntax	71

A.8. mac Example	71
A.9. md5sum Syntax	71
A.10. md5sum Example	72
A.11. erase Syntax	72
A.12. erase Example	72
A.13. setenv/clearenv Syntax	72
A.14. setenv and clearenv Example	73
A.15. setbootdevice Syntax	73
A.16. Assigning Flash Memory As Boot Device	74
A.17. Assigning TFTP Server As Boot Device	74
A.18. Assign SD/MMC Card As Boot Device	74
A.19. tftpd Syntax	74
A.20. tftpd Example	75
A.21. tftpboot Syntax	75
A.22. tftpboot Example	76
A.23. boot Syntax	76
A.24. boot Example	77

1.1. Fonts	10
1.2. Relationship Between Prompt and Execution Environment	10
1.3. Abbreviations Used In Command Entry Examples	10
2.1. Armadillo-400 Series Basic Specifications	12
2.2. Basic Specifications of RTC Option Module	13
2.3. Pin Layout of Armadillo-420 Basic Model Expansion Interfaces	14
2.4. Basic Specifications of Expansion Board	15
2.5. Pin Layout of Armadillo-440 LCD Model Expansion Interfaces	17
2.6. Armadillo-420 Flash Memory Map	18
2.7. Armadillo-440 Flash Memory Map	18
2.8. Jumper Settings	20
3.1. Serial Communication Configuration	23
4.1. List of Packages Required for Atmark-Dist Builds	25
5.1. Region Names and Corresponding Image Files	27
5.2. Downloader List	28
5.3. Region Names and Corresponding Options	31
5.4. Region Names and Corresponding Device Files	32
6.1. Product Name List	36
7.1. Kernel Image Download URLs	47
7.2. Debian Archive Download URLs	48
7.3. Atmark-Dist Image Download URL	49
8.1. Serial Interface Device Files	52
8.2. UART Configuration	52
8.3. SD/MMC/SDIO Host Controller Configuration	53
8.4. Frame Buffer Device Files	54
8.5. Touchscreen Events	55
8.6. Audio Configuration	55
8.7. GPIO_NAME and GPIO Pins	56
8.8. GPIO I/O Direction Configuration	57
8.9. GPIO Interrupt Type Configuration	57
8.10. Armadillo-200 Series Compatible GPIO Driver GPIO List	59
8.11. Armadillo-200 Series Compatible GPIO Driver Device File	60
8.12. Armadillo-200 Series Compatible GPIO Driver ioctl Commands	60
8.13. LED List	61
8.14. LED Node	61
8.15. LED Manipulation Commands	61
8.16. Armadillo-440 LCD Model Button Events	62
8.17. I2C Configuration	63
8.18. SPI Configuration	63
8.19. One Wire Configuration	64
8.20. PWM sysfs	64
8.21. PWM Configuration	65
8.22. CAN sysfs	65
8.23. CAN Configuration	67
8.24. Keypad Configuration	67
8.25. Sleep Modes	68
8.26. Wakeup Basis Designation	68
A.1. Well Used Linux Kernel Parameters	73
A.2. frob Command	74
A.3. tftpd Options	75

1. Preface

The Armadillo Series are small high-performance low-power general purpose boards which incorporate ARM CPU cores. Linux (kernel 2.6) is employed as the standard operating system, providing access to a rich array of software resources and proven stability. All boards include network interfaces as standard which, combined with the Linux network protocol stack, enable simple development of network ready devices.

The Armadillo-400 Series models provide improved performance over existing products of the same class, while at the same time also offering even lower power consumption. The Armadillo-400 Series is comprised of two products, the low cost Armadillo-420 and the Armadillo-440 which can readily support multimedia functionality with the addition of an expansion board.

Armadillo-400 Series boards have interfaces which are often required for embedded devices, such as serial, Ethernet, USB, storage (microSD) and GPIO. In addition to these, multimedia functionality including LCD, touch screen and audio interfaces can be added to Armadillo-440 via an expansion board. Other functionality such as a real-time clock and wireless LAN can also be added with optional modules.

Armadillo-420 is available together with the RTC Option Module as the Armadillo-420 Basic Model. Armadillo-440 together with the LCD Expansion Board is known as the Armadillo-440 LCD Model.

This document provides information required when customizing the software on the Armadillo-400 Series.

For guidance on how to use the default software, please see the "Armadillo-420 Basic Model Development Set Start-up Guide" and the "Armadillo-440 LCD Model Development Set Start-up Guide". For information on hardware specifications, please refer to the "Armadillo-400 Series Hardware Manual".

The product name "Armadillo" will be used in descriptions that apply to the whole Armadillo Series for the remainder of this document.

1.1. Who Should Read This Document

This document is for those planning to customize the software on Armadillo.

1.2. Document Structure

This document comprises of Chapters 1 to 8 and an appendix.

Chapters 1 to 3 deal with the preparation necessary in order to begin development.

Chapters 4 to 6 explain how to set up the development environment, build bootloader, kernel and userland image files from source, and how to write the image files to Armadillo.

Chapter 7 explains how to deploy kernel and userland images to storage devices other than the on-board flash memory.

Chapter 8 describes the specifications of the Linux kernel device drivers unique to Armadillo.

Finally, the Appendix explains the functionality of the bootloader.

1.3. Typographical Conventions

1.3.1. Fonts

Fonts are used in the following ways in this document.

1.1 Fonts

Font Example	Description
Plain text font	Used for standard text
[PC ~]\$ ls	Shell prompt and user input text
text	Text that is either displayed, is to be edited, or is a comment

1.3.2. Command Entry Examples

The command entry examples in this document all have an assumed execution environment which is reflected in the displayed prompt. The directory part “/” will differ depending on the current directory. The home directory of each user is represented by “~”.

1.2 Relationship Between Prompt and Execution Environment

Prompt	Command Execution Environment
[PC /]#	To be executed by a privileged user on the work PC
[PC /]\$	To be executed by a general user on the work PC
[armadillo /]#	To be executed by a privileged user on Armadillo
[armadillo /]\$	To be executed by a general user on Armadillo
hermit>	To be executed on Armadillo in maintenance mode


Commands that may change or vary depending on the relevant environment are written as shown below. Please adjust the commands as necessary.

1.3 Abbreviations Used In Command Entry Examples


Notation	Description
[version]	File version number

1.3.3. Icons

Icons are used in the following way in this document.



This is used for precautions.



This is used for helpful information.

1.4. Acknowledgements

The software used on Armadillo is composed from Free Software / Open Source Software. This Free Software / Open Source Software is the result of efforts from developers from all over the world. We would like to take this opportunity to express our gratitude.

1.5. Software Usage Precautions

About Software Contained In This Product

The software and documentation contained in this product is provided “AS IS” without warranty of any kind including warranty of merchantability or fitness for a particular purpose, reliability, correctness or accuracy. Furthermore, no guarantee is made in regard to any outcomes resulting from the use of this product.

1.6. Trademarks

Armadillo is a registered trademark of Atmark Techno, Inc. All other company names, product names and related trademarks are the property of their respective owners.

2. System Overview

This chapter provides a basic system overview.

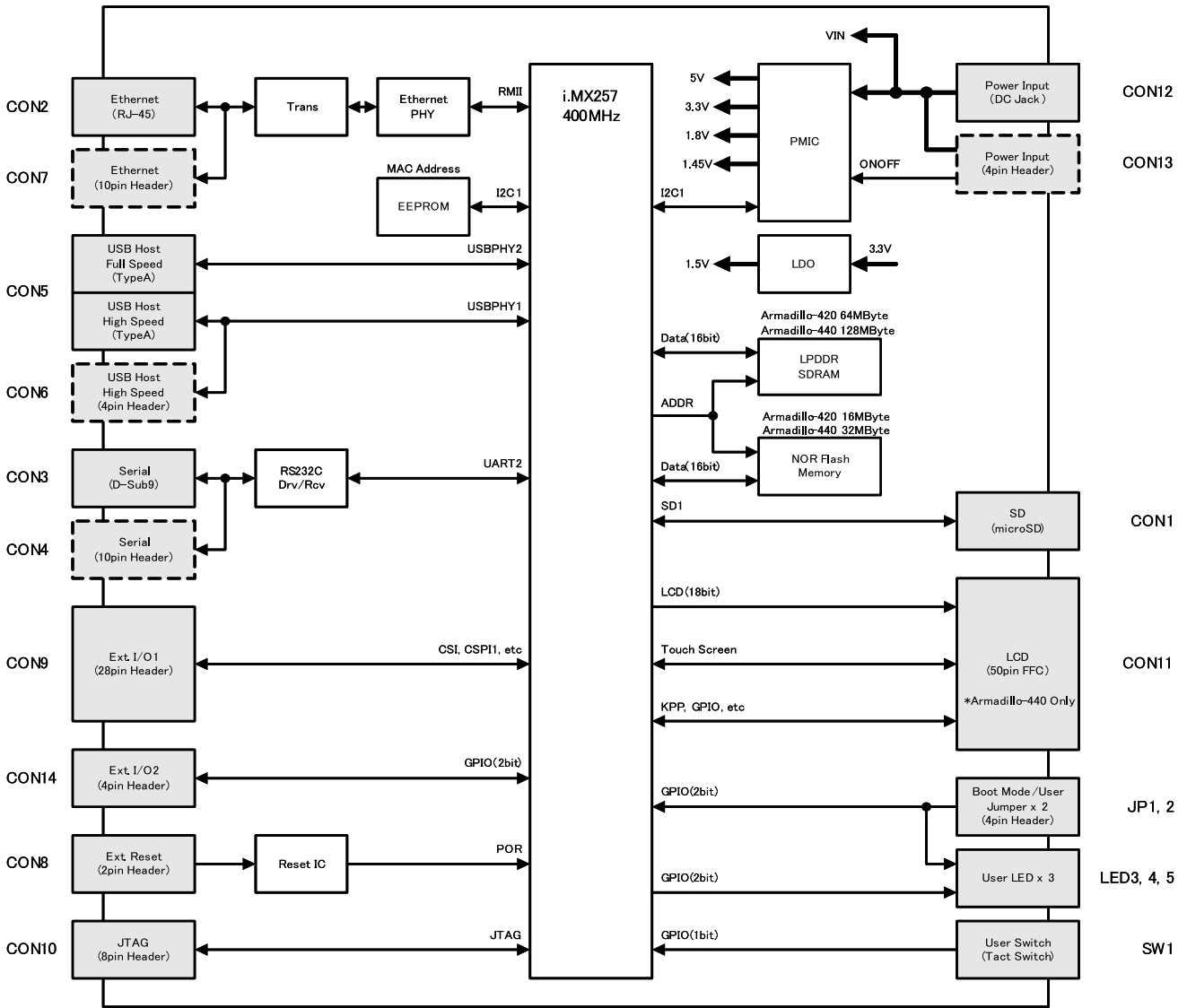
2.1. Armadillo-400 Series Basic Specifications

The basic specifications of the Armadillo-400 Series boards in their default state^[1] are show in 2.1. Armadillo-400 Series Basic Specifications. The block diagram is shown in 2.1. Armadillo-420/440 Block Diagram.

2.1 Armadillo-400 Series Basic Specifications

	Armadillo-420	Armadillo-440
Processor	Freescale i.MX257 (ARM926EJ-S) Instruction / Data Cache: 16KByte / 16KByte Internal RAM: 128KByte	
System Clock	CPU Core Clock: 400MHz BUS Clock: 133MHz	
RAM	LPDDR SDRAM: 64MByte (16bit width)	LPDDR SDRAM: 128MByte (16bit width)
ROM	NOR Flash Memory: 16MByte (16bit width)	NOR Flash Memory: 32MByte (16bit width)
Serial	RS232C Levels x1 port Flow control pins (full modem) 230.4 kbps max	
	3.3V I/O levels x2 ports No flow control pins 4Mbps max	
USB 2.0 Host	High Speed x1 port	
	Full Speed x1 port	
LAN	10BASE-T/100BASE-TX x1 port	
Storage	microSD x1 4bit width, 208Mbps max	
GPIO	3.3V I/O levels x18 pins	
Programmable LEDs	Red x1, Green x1, Yellow x1	
Buttons	Tact switch x1	

^[1]It is possible to change the functions assigned to the I/O pins on the Armadillo-400 Series with multiplexing. For more information please refer to "Armadillo-400 Series Hardware Manual" and 8. Linux Kernel Device Driver Specifications.



2.1 Armadillo-420/440 Block Diagram

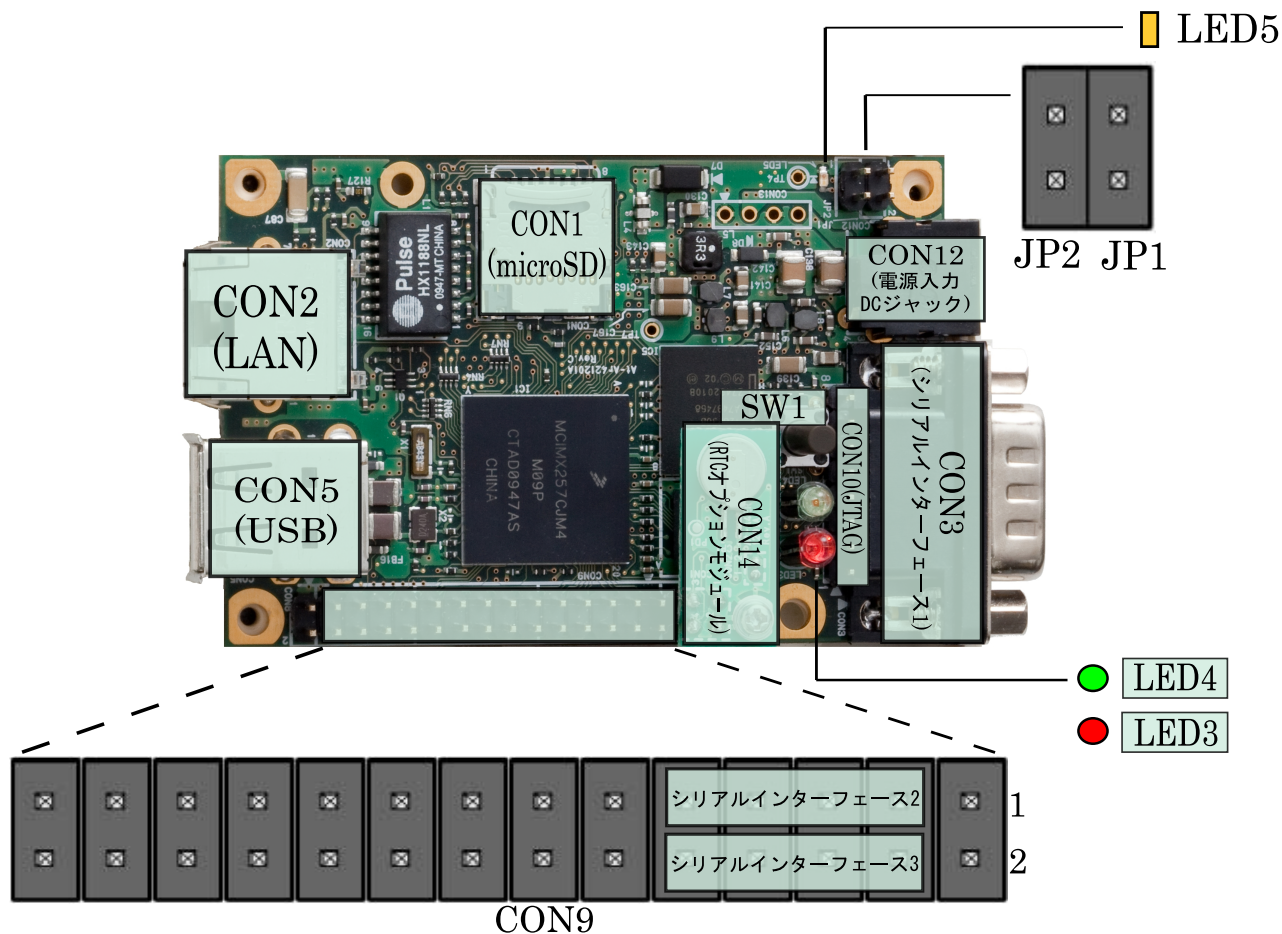
2.2. Basic Specifications of Armadillo-420 Basic Model

The Armadillo-420 Basic Model is comprised of the Armadillo-420 board together with the Armadillo-400 RTC Option Module. The basic specifications of the RTC Option Module are shown in 2.2. Basic Specifications of RTC Option Module.

2.2 Basic Specifications of RTC Option Module

Armadillo-400 RTC Option Module	
Real-time Clock	Backup with supercapacitor

The basic layout of the Armadillo-420 Basic Model is shown in 2.2. Basic Layout of Armadillo-420 Basic Model. The pin layouts of CON9 and CON14 are shown in 2.3. Pin Layout of Armadillo-420 Basic Model Expansion Interfaces. Please make sure to confirm the position of each interface.




2.2 Basic Layout of Armadillo-420 Basic Model

2.3 Pin Layout of Armadillo-420 Basic Model Expansion Interfaces

Pin Number	Function	Notes
CON9 1	GPIO	100 k pull-up
CON9 2	GPIO	100 k pull-up
CON9 3	Serial Interface 2 RXD	
CON9 4	Serial Interface 3 RXD	
CON9 5	Serial Interface 2 TXD	
CON9 6	Serial Interface 3 TXD	
CON9 7	+3.3V	
CON9 8	+3.3V	
CON9 9	GND	
CON9 10	GND	
CON9 11	GPIO	100 k pull-up
CON9 12	GPIO	100 k pull-up
CON9 13	GPIO	100 k pull-up
CON9 14	GPIO	100 k pull-up
CON9 15	GPIO	100 k pull-up
CON9 16	GPIO	100 k pull-up

Pin Number	Function	Notes
CON9 17	GPIO	100 k pull-up
CON9 18	GPIO	100 k pull-up
CON9 19	GND	
CON9 20	+3.3V	
CON9 21	GPIO	100 k pull-up
CON9 22	GPIO	100 k pull-up
CON9 23	GPIO	100 k pull-up
CON9 24	GPIO	100 k pull-up
CON9 25	GPIO	100 k pull-up
CON9 26	GPIO	100 k pull-up
CON9 27	GPIO	
CON9 28	GPIO	
CON14 1	+3.3V	
CON14 2	GND	
CON14 3	I2C2 SCL	22 k pull-up / open-drain
CON14 4	I2C2 SDA	22 k pull-up / open-drain



Serial Interfaces 2 and 3 have +3.3V I/O levels. They can be used at RS232C levels by connecting the optional^[2] RS232C level conversion adapter.

When using the RS232C level conversion adapter, please connect pin 1 (the yellow or green wire) to CON9 1 for Serial Interface 2 and to CON9 2 for Serial Interface 3.

2.3. Basic Specifications of Armadillo-440 LCD Model

The Armadillo-440 LCD Model is comprised of the Armadillo-440 board together with the Armadillo-440 LCD Expansion Board. The basic specifications of the LCD Expansion Board are shown in 2.4. Basic Specifications of Expansion Board.

2.4 Basic Specifications of Expansion Board

	Armadillo-440 LCD Expansion Board
Audio	Playback (stereo) / Capture (mono)
LCD	Resolution: 480 x 272 pixels RGB 565 Color
Touchscreen	4-Wire Resistive
Real-time Clock	Backup with supercapacitor
Buttons	Tact switch x3

The basic layout of the Armadillo-440 LCD Model is shown in 2.3. Basic Layout of Armadillo-440 LCD Model. The pin layouts of CON9 and CON14 are shown in 2.5. Pin Layout of Armadillo-440 LCD Model Expansion Interfaces. Please make sure to confirm the position of each interface.

^[2]A RS232C level conversion adapter is available as an option and is also included in the development set.



2.3 Basic Layout of Armadillo-440 LCD Model

2.5 Pin Layout of Armadillo-440 LCD Model Expansion Interfaces


Pin Number	Function	Notes
CON9 1	GPIO	100 k pull-up
CON9 2	GPIO	100 k pull-up
CON9 3	Serial Interface 2 RXD	
CON9 4	Serial Interface 3 RXD	
CON9 5	Serial Interface 2 TXD	
CON9 6	Serial Interface 3 TXD	
CON9 7	+3.3V	
CON9 8	+3.3V	
CON9 9	GND	
CON9 10	GND	
CON9 11	GPIO	100 k pull-up
CON9 12	GPIO	100 k pull-up
CON9 13	GPIO	100 k pull-up
CON9 14	GPIO	100 k pull-up
CON9 15	GPIO	100 k pull-up
CON9 16	GPIO	100 k pull-up
CON9 17	GPIO	100 k pull-up
CON9 18	GPIO	100 k pull-up
CON9 19	GND	
CON9 20	+3.3V	
CON9 21	GPIO	100 k pull-up
CON9 22	GPIO	100 k pull-up
CON9 23	GPIO	100 k pull-up
CON9 24	GPIO	100 k pull-up
CON9 25	GPIO	100 k pull-up
CON9 26	GPIO	100 k pull-up
CON9 27	GPIO	
CON9 28	GPIO	
CON14 1	+3.3V	
CON14 2	GND	
CON14 3	I2C2 SCL	22 k pull-up / open-drain
CON14 4	I2C2 SDA	22 k pull-up / open-drain



Serial Interfaces 2 and 3 have +3.3V I/O levels. They can be used at RS232C levels by connecting the optional^[3] RS232C level conversion adapter.

When using the RS232C level conversion adapter, please connect pin 1 (the yellow or green wire) to CON9 1 for Serial Interface 2 and to CON9 2 for Serial Interface 3.

^[3]A RS232C level conversion adapter is available as an option and is also included in the development set.



CON14 3 and CON14 4 were configured to be used as GPIO by default in linux-2.6.26-at7 (linux-a400-1.00.bin.gz). In linux-2.6.26-at8 (linux-a400-1.01.bin.gz) and later, this was changed so that they are configured to be used as I2C2 by default. Please be aware of this change when using CON14 3 and CON14 4.

2.4. Memory Map

The default partitioning of flash memory on the Armadillo-400 Series boards is shown in 2.6. Armadillo-420 Flash Memory Map and 2.7. Armadillo-440 Flash Memory Map.

2.6 Armadillo-420 Flash Memory Map

Physical Address	Region Name	Size	Description
0xa0000000 0xa001ffff	bootloader	128KB	Bootloader image is stored here
0xa0020000 0xa021ffff	kernel	2MB	Kernel image is stored here
0xa0220000 0xa0fdffff	userland	13.75MB	Userland image is stored here
0xa0fe0000 0xa0ffffff	config	128KB	Configuration data is stored here

2.7 Armadillo-440 Flash Memory Map

Physical Address	Region Name	Size	Description
0xa0000000 0xa001ffff	bootloader	128KB	Bootloader image is stored here
0xa0020000 0xa021ffff	kernel	2MB	Kernel image is stored here
0xa0220000 0xa1fdffff	userland	29.75MB	Userland image is stored here
0xa1fe0000 0xa1ffffff	config	128KB	Configuration data is stored here

2.5. Software Make-up

The Armadillo-400 Series operates with the software described below.

2.5.1. Bootloader

The bootloader is the first software program to run after the board is turned on. The Hermit-At Bootloader (hereafter referred to as Hermit-At) is used on the Armadillo-400 Series.

Hermit-At has two operating modes: auto-boot mode and maintenance mode. In auto-boot mode, the kernel image is loaded to RAM from a predetermined place and then booted. In maintenance mode it is possible to carry out operations such as updating flash memory and setting boot options. For more information, please refer to [A Hermit-At Bootloader](#).

The bootloader must be stored in the bootloader region of flash memory.

2.5.2. Kernel

Linux 2.6 is used as the default kernel on the Armadillo-400 Series.

The kernel image is stored in the kernel region of flash memory by default. It is also possible to use a kernel image from storage (microSD) or from a TFTP server by altering Hermit-At's boot options.

2.5.3. Userland

The standard userland root filesystem used on the Armadillo-400 Series is an `initrd`^[4] image created from a source code based distribution named Atmark-Dist.

In addition to the standard userland, a Debian GNU/Linux based userland option is also available.

By default, the `initrd` image is stored in the userland region of flash memory and loaded as a RAM disk by Hermit-At. It is possible to use an image from a TFTP server by altering Hermit-At's boot options.

Aside from a RAM disk, it is also possible to use a root filesystem from external storage (microSD / USB) or a NFS server^[5] by setting the appropriate kernel parameters.

The use of a kernel or userland image not stored in flash memory is explained in [7. Kernel and Userland Placement](#).

2.5.4. Downloader

This is an application that runs on the work PC and is used to write to the flash memory on the Armadillo.

For Linux PCs, the Hermit-At downloader and Shoehorn-At are available. The Hermit-At downloader works in cooperation with the target Armadillo to rewrite the on-board flash memory. Shoehorn-At is used to restore the bootloader.

The downloader available for Windows PCs is called Hermit-At Win32. Hermit-At Win32 can be used to both rewrite the on-board flash memory and restore the bootloader on the Armadillo.

2.6. Boot Modes

JP1 is used to select between on-board flash memory boot mode and UART boot mode on the Armadillo-400 Series.

In on-board flash memory boot mode, the bootloader stored in the bootloader region of flash memory is executed at power on.

With the default bootloader (Hermit-At), JP2 is used to select between auto boot mode, where the kernel is automatically booted, and maintenance mode, where it is possible to carry out various configuration.

^[4]Initial RAM disk. On standard Linux systems, an `initrd` is used as a temporary "mini" root filesystem before the root file system stored in external (HDD etc) storage is mounted. On the Armadillo-400 Series, the `initrd` is used as the final root file system.

^[5]When NFS support is enabled in the kernel.

However, even when auto boot mode is selected with JP2, the auto boot cancel function in Hermit-At will cause it to enter maintenance mode if SW1 is depressed at boot time.

The UART boot mode is used for system restore when, for example, the flash memory bootloader has been damaged. For more information, please refer to 5.6. Restoring Bootloader to Factory State.

The jumper settings for each boot mode on the Armadillo-400 Series are shown in 2.8. Jumper Settings.

2.8 Jumper Settings

JP1	JP2	Boot Modes
Open	Open	On-board flash memory boot / auto boot mode
Open	Shorted	On-board flash memory boot / maintenance mode
Shorted	-	UART boot mode

3. Before Getting Started

3.1. Preparation

The following equipment is required in order to begin development of embedded systems based on the Armadillo-400 Series.

Work PC	A PC that runs either Debian GNU/Linux or Windows and has at least one serial port.
Serial Cross Cable	A D-Sub 9 pin (female-to-female) cable to connect the Armadillo and the work PC.
Serial Console Software	Please install a serial console program on the work PC such as minicom on Linux or Tera Term Pro on Windows

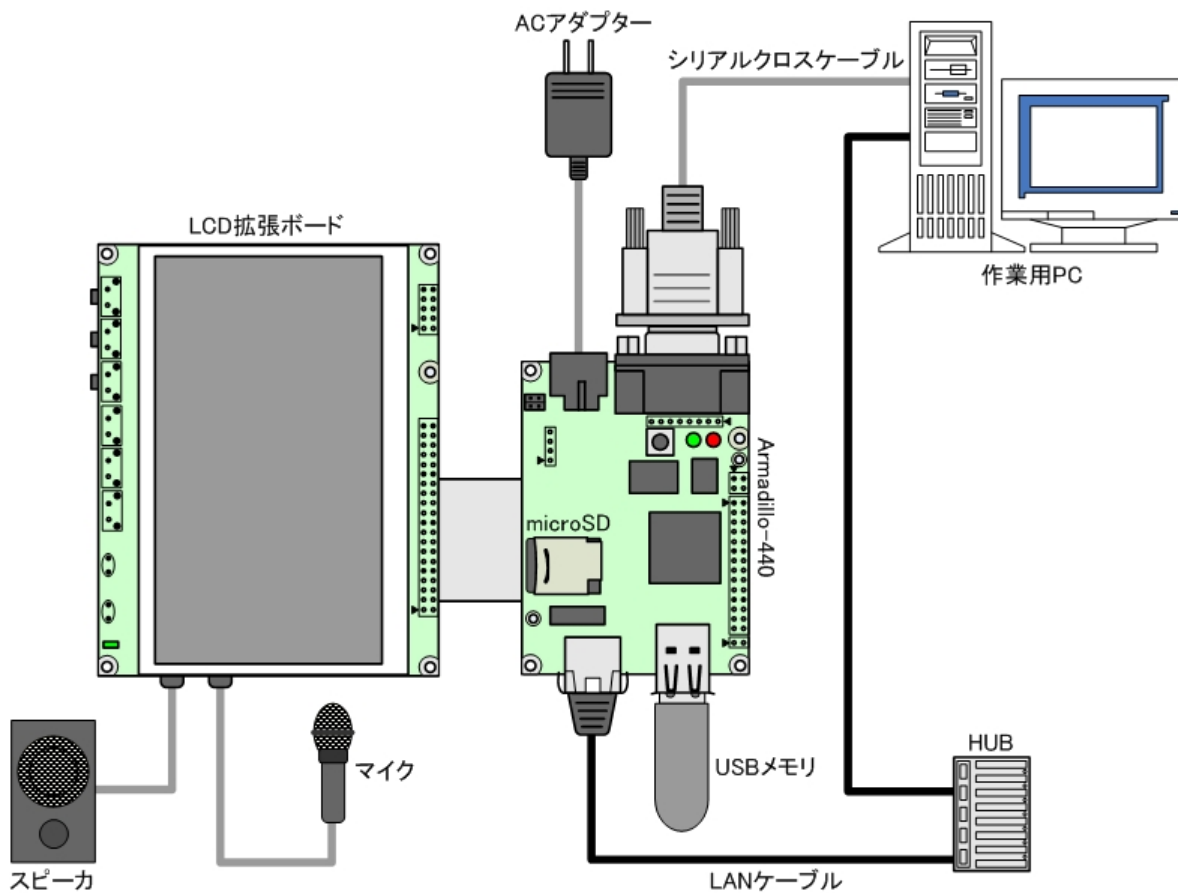
The following equipment is not an absolute requirement, but does allow for improved development efficiency.

LAN Cable	This is required in order to communicate with the Armadillo via LAN. Please connect the Armadillo and work PC via a switching hub ^[1] .
-----------	--

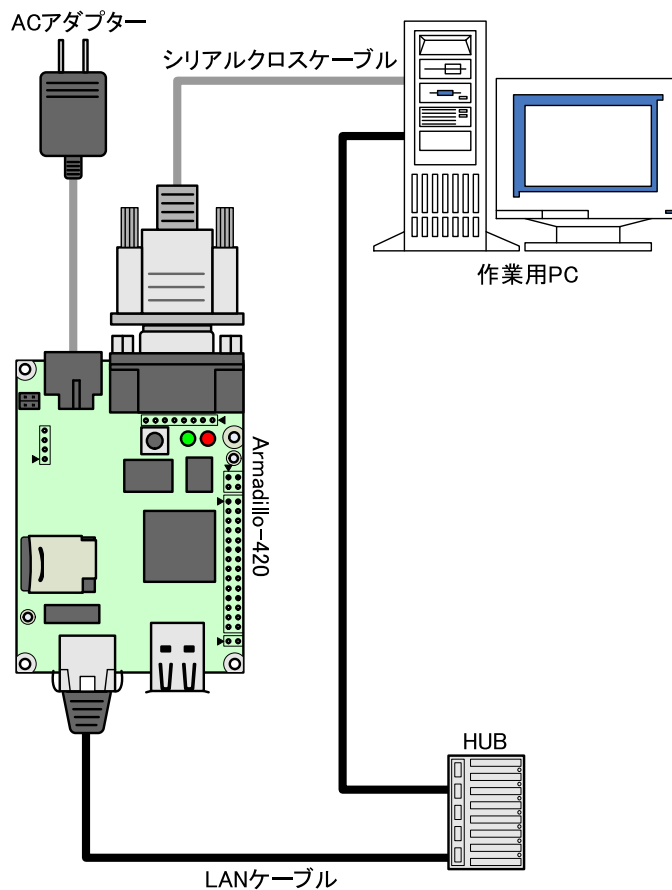
3.2. Connections

Please connect the work PC and peripherals to the Armadillo by referring to the examples in 3.1. Armadillo-440 LCD Model Connections and 3.2. Armadillo-420 Basic Model Connections.

^[1]As Auto MDIX is supported on the Armadillo-400 Series, it is also possible to connect the Armadillo and work PC directly with a LAN cable.



3.1 Armadillo-440 LCD Model Connections



3.2 Armadillo-420 Basic Model Connections

3.3. Serial Console Software Configuration

When connecting the work PC to a serial console on the Armadillo, please configure the serial console software with the settings shown in 3.1. Serial Communication Configuration.

3.1 Serial Communication Configuration

Item	Configuration
Transmission Rate	115,200 bps
Data Length	8 bit
Stop Bit	1 bit
Parity	None
Flow Control	None

4. Development Environment Set-up

This chapter explains how to set up a software development environment for Armadillo on a work PC.

A Debian based Linux environment^[1] (Debian/GNU Linux 5.0 codename Lenny is standard) is required to undertake software development for the Armadillo-400 Series.

If the work PC is Windows, a virtual Linux environment must be set up within Windows.

"VMware" is the recommended way to set up a Linux environment on Windows. An operating system image called ATDE (Atmark Techno Development Environment)^[2] is available with the required development software pre-installed.

Please refer to the "ATDE Install Guide" for information on setting up a Linux environment on Windows.

As the basic development environment is pre-installed, there is no need to carry out the operations listed in 4.1. Installing Cross Development Environment Packages and 4.2. Installing Packages Required for Atmark-Dist Builds when using ATDE.

4.1. Installing Cross Development Environment Packages

Debian (deb) packages are used for application and library management on Debian based Linux systems.

In order to carry out cross development, toolchain packages for cross development and library packages for the target architecture which have been converted for cross development use must be installed on the work PC.

On Debian based Linux systems there are two ARM architectures: arm and armel. The difference between these two lies in the ABI (Application Binary Interface), with the arm architecture using OABI, and the armel architecture using EABI.

As EABI is the standard ABI on the Armadillo-400 Series, Debian armel packages must be installed.

The cross development environment packages are stored on the included DVD under the `cross-dev/deb/` directory. For development with the Armadillo-400 Series, please install the ARM EABI cross development packages in the `armel` directory.

The install must be carried out as a root user. The deb packages are installed by executing the command shown in 4.1. Install Command.

```
[PC ~]$ sudo dpkg --install *.deb
```

4.1 Install Command



sudo is used to execute the specified command as a different user. The above command line executes the **dpkg** command as the root user.

A password is requested after executing the **sudo** command. This is the password of the user executing the command, and not the root password.

^[1]While it is possible to use Linux distributions other than Debian, not all operations listed in this document will work in the same way. The reader will be required to modify the listed operations to suit their Linux environment when using other distributions.

^[2]ATDE v3.0 or later is the recommended development environment for the Armadillo-400 Series.



If a cross development environment for the same target architecture has been previously installed in the Linux environment, please make sure to uninstall this existing environment before installing the new cross development environment.

4.2. Installing Packages Required for Atmark-Dist Builds

Atmark-Dist is the default distribution for Armadillo. The packages listed in 4.1. List of Packages Required for Atmark-Dist Builds must be installed on the work PC in order to build Atmark-Dist.

4.1 List of Packages Required for Atmark-Dist Builds

Package Name	Version
genext2fs	1.4.1-2.1 or later
file	4.26-1 or later
sed	4.1.5-6 or later
perl	5.10.0-19lenny2 or later
bison	1:2.3.dfsg-5 or later
flex	2.5.35-6 or later
libncurses5-dev	5.7+20081213-1 or later

The currently installed version can be displayed by specifying the package name when executing the command shown in 4.2. Installed Version Display Command.

`--list` is an option to `dpkg` used to display package information. The package name pattern for the package to be displayed is specified in place of `package-name-pattern`.

```
[PC ~]$ dpkg --list [package-name-pattern]
```

4.2 Installed Version Display Command

4.3. Installing Cross Development Library Packages

When building applications or libraries not included in Atmark-Dist, library packages that are neither available in the included DVD or from the download site may be required. The following is an introduction on how to create and install the required cross development library packages.

First, the library package on which the cross development package will be based on must be acquired. The package architecture must match the target and the Debian distribution version should match that of the development environment. For the Armadillo-400 Series, the architecture is `armel` and the Debian distribution version is `lenny` (the stable version as at March, 2010).

For example, for the `libjpeg62` library the package name will be `libjpeg62_[version]_armel.deb`.

Debian packages can be searched for and downloaded from the Debian Packages site^[3].

Once the library package has been obtained, it is then converted for cross development use with the `dpkg-cross` command.

^[3]<http://www.debian.org/distrib/packages>

```
[PC ~]$ dpkg-cross --build --arch armel libjpeg62_[version]_armel.deb
[PC ~]$ ls
libjpeg62-armel-cross_[version]_all.deb libjpeg62_[version]_armel.deb
```

4.3 Cross Development Library Package Creation

The created cross development library package is then installed.

```
[PC ~]$ sudo dpkg -i libjpeg62-armel-cross_[version]_all.deb
```

4.4 Installing Cross Development Library Packages



If the **dpkg-cross** command is used in environments other than Debian Lenny, there may be times when an installable package cannot be created.

By using **apt-cross**, all of the above steps can be performed with just one command.

```
[PC ~]$ apt-cross --arch armel --suite lenny --install libjpeg62
```

4.5 apt-cross Command

The target architecture is specified after the **--arch** option, and the Debian distribution version after **--suite**. When **--install** is specified, **apt-cross** automatically installs the converted package. The package name is specified as the last parameter.


5. Rewriting Flash Memory

This chapter explains the steps required to rewrite the on-board flash memory on the Armadillo.

There are two main ways to rewrite the flash memory.

1. Rewriting flash memory by using a downloader program on the work PC to send an image to the target Armadillo.
2. Rewriting flash memory by having the target Armadillo download an image file from a remote server.

The first method is explained in 5.3. Rewriting Flash Memory With A Downloader. The second method is then explained in 5.4. Rewriting Flash Memory With tftpd and 5.5. Rewriting Flash Memory With netflash.



If for some reason flash rewriting fails, the software on the Armadillo may no longer be able to boot properly. Be sure to pay attention to the following points when rewriting flash.

- Do not cut power to the Armadillo while rewriting
- Do not disconnect the serial or LAN cable connecting the Armadillo and work PC while rewriting


If the Armadillo can no longer be booted after a rewrite of the bootloader has failed, please follow the steps listed in 5.6. Restoring Bootloader to Factory State to restore the bootloader.

5.1. Flash Memory Writing Regions

The flash memory address to be written to can be specified by region name. The image files to be specified for each region are shown in 5.1. Region Names and Corresponding Image Files.

5.1 Region Names and Corresponding Image Files

Product	Region Name	File Name
Armadillo-440 LCD Model	bootloader	loader-armadillo4x0-[version].bin
	kernel	linux-a400-[version].bin.gz
	userland	romfs-a440-[version].img.gz
Armadillo-420 Basic Model	bootloader	loader-armadillo4x0-[version].bin
	kernel	linux-a400-[version].bin.gz
	userland	romfs-a420-[version].img.gz



Armadillo-420 and Armadillo-440 both use the same bootloader and kernel image files.

5.2. Installing Downloader

The downloader program is installed on the work PC.

There are multiple types of downloader, as listed in 5.2. Downloader List.

5.2 Downloader List

Downloader	Operating System	Description
Hermit-At Downloader	Linux	This is a CUI application for Linux.
Shoehorn-At	Linux	This is a CUI application for Linux.
Hermit-At Win32	Windows	This is a GUI application for Windows.



The downloader package is pre-installed in ATDE (Atmark Techno Development Environment), so there is no need to install it separately.

5.2.1. For Linux

Install the package files contained in the `downloader/` directory on the included DVD.

```
[PC ~]$ sudo dpkg --install hermit-at_[version]_i386.deb
[PC ~]$ sudo dpkg --install shoehorn-at_[version]_i386.deb
```

5.1 Installing Downloader (Linux)

5.2.2. For Windows

Extract the contents of the `hermit-at-win_[version].zip` archive stored in the `downloader/` directory on the included DVD to an appropriate folder.

5.3. Rewriting Flash Memory With A Downloader

The following explains how to rewrite flash memory with the Hermit-At Downloader and Hermit-At Win32.

The Hermit-At Downloader and Hermit-At Win32 work in cooperation with the bootloader on the Armadillo to rewrite the flash memory from the work PC.

5.3.1. Preparation

Boot the Armadillo in maintenance mode by appropriately setting the jumpers as shown in 2.8. Jumper Settings and then powering the board on.

Ensure that the serial interface on the work PC that is connected to the Armadillo is not being used by another application. If another application is using the serial interface, please either close the connection or close the application in order to free the interface.

5.3.2. For Linux

On Linux, execute the `hermit-at` command as shown in 5.2. Download Command.

The download option is a sub-command of **hermit-at**. The `--input-file` option is used to specify the file to be written to flash. The `--region` option is used to specify the region to be written to. Therefore in the following example `linux.bin.gz` will be written to the kernel region.

```
[PC ~]$ hermit download --input-file linux.bin.gz --region kernel
```

5.2 Download Command

If using a serial interface other than `ttyS0`, the port must be specified with the `--port` option as shown in 5.3. Download Command (With Port Option)^[1].

```
[PC ~]$ hermit download --input-file linux.bin.gz --region kernel --port ttyS1
```

5.3 Download Command (With Port Option)

The bootloader region has a simple protection mechanism to prevent mistaken rewrites. To rewrite the bootloader region, override the protection mechanism by using the `--force-locked` option as shown in 5.4. Download Command (Unprotected)^[1].

```
[PC ~]$ hermit download --input-file loader-armadillo4x0-[version].bin  
--region bootloader --force-locked
```

5.4 Download Command (Unprotected)

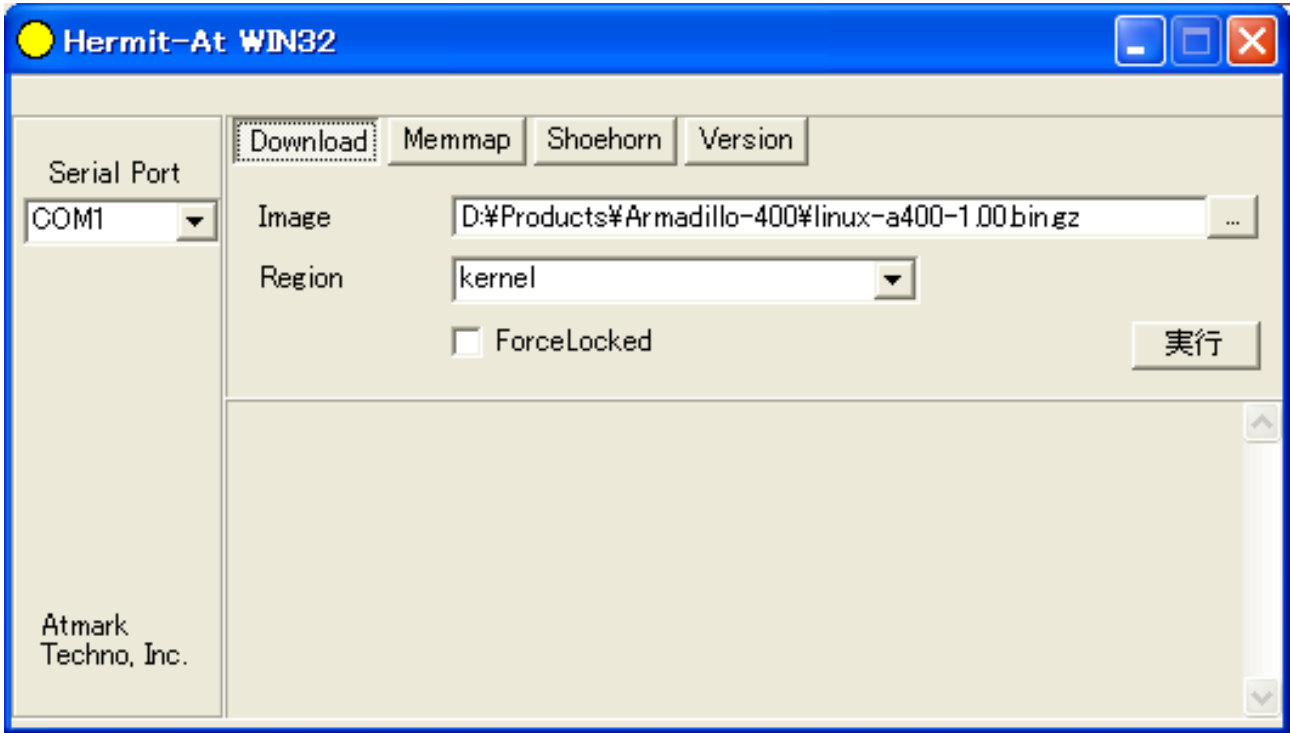


If an incorrect image is written to the bootloader region, it will no longer be possible to boot from on-board flash memory. If this does occur, restore the bootloader by referring to 5.6. Restoring Bootloader to Factory State.

5.3.3. For Windows

For Windows, execute `hermit.exe`. The window shown in 5.5. Hermit-At Win32: Download Window will appear.

^[1]The command is only shown as multiple lines due to formatting constraints and should be entered as one line.

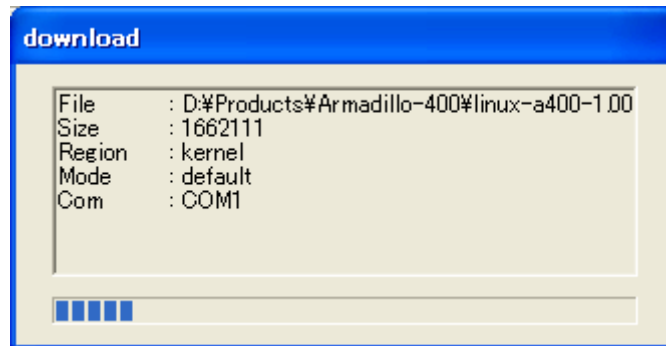


5.5 Hermit-At Win32: Download Window

Under "Serial Port", select the serial interface connected to the Armadillo. If a drop-down list is not displayed, please enter the port name directly.

For "Image", specify the image to be written and for "Region", specify the region to be written to. If specifying the all or bootloader regions, "Force Locked" must be selected.

After all configuration is complete, click the Execute button to start the writing. The download progress is displayed in a dialog box as shown in 5.6. Hermit-At Win32: Download Dialog while the image is being written to flash.



5.6 Hermit-At Win32: Download Dialog


The dialog box will close once the download has finished.

5.4. Rewriting Flash Memory With tftpd

The following explains how to rewrite flash memory by having the target Armadillo download an image from a remote server.

Using the `tftpd` function of the Hermit-At bootloader is a faster way to rewrite flash memory than using a downloader program.

The `tftpd` function gives the Armadillo the ability to download an image made available by a TFTP server on the same network and then write the image to its own flash memory.



A TFTP server (`atftpd`) is started by default on ATDE v3.0 and later. Files placed in the `/var/lib/tftpboot/` directory will be accessible via TFTP.

In order to use the `tftpd` function, first configure the jumpers on the target Armadillo for maintenance mode and boot the board.

Use the serial console program on the work PC to enter the `tftpd` command. In the example shown in 5.7. `tftpd` Command Example, the Armadillo's IP address is set to 192.168.10.10 and `linux.bin.gz` from a TFTP server with the IP address of 192.168.10.1 is written to the kernel region.

```
hermit> tftpd 192.168.10.10 192.168.10.1 --kernel=linux.bin.gz
```

5.7 tftpd Command Example

The bootloader, kernel and userland regions can be specified for rewriting. The region names and their corresponding options are shown in 5.3. Region Names and Corresponding Options.


5.3 Region Names and Corresponding Options

Region	Option
Bootloader	--bootloader
Kernel	--kernel
Userland	--userland

5.5. Rewriting Flash Memory With netflash

It is possible to rewrite flash memory with the **netflash** application once the Armadillo has booted into Linux.

The **netflash** command gives the Armadillo the ability to download an image made available by a HTTP or FTP server on the same network and then write the image to its own flash memory.



A HTTP server (`lighttpd`) is started by default on ATDE v3.0 and later. Files placed in the `/var/www/` directory will be accessible via HTTP.

In order to use the **netflash** command, first login to the Armadillo and then execute the command as shown in 5.8. `netflash` Command Example.

```
[armadillo ~]# netflash -k -n -u -r /dev/flash/kernel [URL]
```

5.8 netflash Command Example

The "-r [device-file-name]" option is used to specify the region to be written to. Please refer to 5.4. Region Names and Corresponding Device Files for the device file names. Details on other options can be found by executing `netflash -h`.

5.4 Region Names and Corresponding Device Files

Region	Device File
Kernel	/dev/flash/kernel
Userland	/dev/flash/userland



As the device file of the bootloader region is read-only by default, it cannot be rewritten with **netflash**.

5.6. Restoring Bootloader to Factory State

If for some reason the contents of the bootloader region have been damaged and the bootloader no longer functions properly, the bootloader can be restored to factory state by using the UART boot mode.

5.6.1. Preparation

Configure the jumpers on the Armadillo for UART boot mode by referring to 2.8. Jumper Settings, but do not boot the Armadillo at this stage.

Ensure that the serial interface on the work PC that is connected to the Armadillo is not being used by another application. If another application is using the serial interface, please either close the connection or close the application in order to free the interface.

5.6.2. For Linux

Execute the command^[2] in 5.9. shoehorn Command Example and then power on the Armadillo.

```
[PC ~]$shoehorn --boot --target armadillo4x0
--initrd /dev/null
--kernel /usr/lib/hermit/loader-armadillo4x0-boot-[version].bin
--loader /usr/lib/shoehorn/shoehorn-armadillo4x0.bin
--initfile /usr/lib/shoehorn/shoehorn-armadillo4x0.init
--postfile /usr/lib/shoehorn/shoehorn-armadillo4x0.post
```

5.9 shoehorn Command Example

After executing the command the log shown in 5.10. shoehorn Command Log will be displayed.

^[2]The command is only shown as multiple lines due to formatting constraints and should be entered as one line.


```

/usr/lib/shoehorn/shoehorn-armadillo4x0.bin: 1272 bytes (2048 bytes buffer)
/usr/lib/hermit/loader-armadillo4x0-boot-v2.0.0.bin: 45896 bytes (45896 bytes
buffer)
/dev/null: 0 bytes (0 bytes buffer)
Waiting for target - press Wakeup now.
Initializing target...
Writing SRAM loader...
Pinging loader
Initialising hardware:
- flushing cache/TLB
- Switching to 115200 baud
- Initializing for Mobile-DDR
Pinging loader
Detecting DRAM
- 32 bits wide
- start: 0x80000000 size: 0x04000000 last: 0x83ffffff
Total DRAM: 65536kB
Loading /usr/lib/hermit/loader-armadillo4x0-boot-v2.0.0.bin:
- start: 0x83000000 size: 0x0000b348 last: 0x8300b347
initrd_start is c0400000
Moving initrd_start to c0400000
Loading /dev/null:
- start: 0xc0400000 size: 0x00000000
Writing parameter area
- nr_pages (all banks): 4096
- rootdev: (RAMDISK_MAJOR, 0)
- pages_in_bank[0]: 2048
- pages_in_bank[1]: 2048
- initrd_start: 0xc0400000
- initrd_size: 0x0
- ramdisk_size: 0x0
- start: 0x80020000 size: 0x00000900 last: 0x800208ff
Pinging loader
Starting kernel at 0x83000000

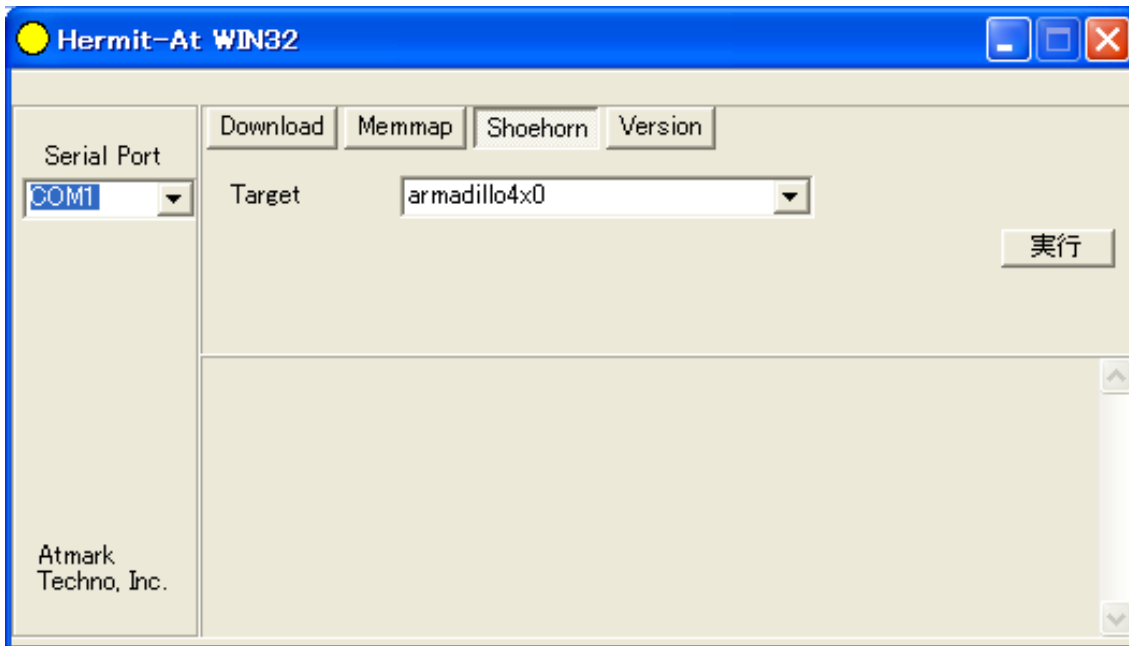
```

5.10 shoehorn Command Log

A successful execution of the **shoehorn** command will result in the UART boot mode version of the Hermit At bootloader (loader-armadillo4x0-boot-*[version]*.bin) being loaded on the Armadillo. With the Armadillo in this state, refer to 5.3. Rewriting Flash Memory With A Downloader to write the bootloader image to flash memory.

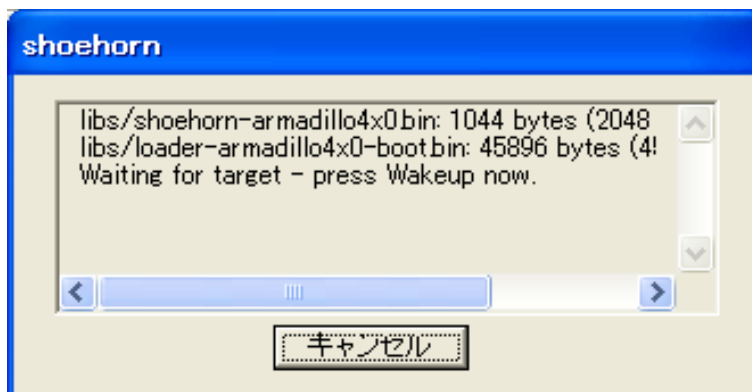
5.6.3. For Windows

Execute hermit.exe and click on the Shoehorn button. The window shown in 5.11. Hermit-At Win32: Shoehorn Window will appear.



5.11 Hermit-At Win32: Shoehorn Window

Select armadillo4x0 for the target and click the execute button.



5.12 Hermit-At Win32: shoehorn Dialog

After the dialog box appears, power on the Armadillo. The dialog box will then close automatically once the Armadillo is ready for downloading.

With the Armadillo in this state, refer to 5.3. Rewriting Flash Memory With A Downloader to write the bootloader image to flash memory.

6. Building

This chapter explains how to build images from source code with the same content as the factory images, and also how to customize these images.

If you have not yet set up a development environment on a work PC, please do so now by referring to 4. Development Environment Set-up.

For more a detailed explanation of the steps involved, please refer to the "Atmark-Dist Developers Guide."

The operations in the following examples should be carried out under the home directory (~ /).



The development process involves working with basic libraries, applications and system configuration files. While all files are altered only under the working directory, in order to ensure the PC operating system is not inadvertently damaged from any mistakes made during development please perform all work as a **general user** and not a root user.

6.1. Building Kernel and Userland Images

The following explains how to make kernel and userland images from the source code based distribution Atmark-Dist and the Linux kernel source code.

6.1.1. Preparing Source Code

First, obtain the source code. Extract `atmark-dist-[version].tar.gz` from the `source/dist` directory and `linux-[version].tar.gz` from the `source/kernel` directory on the included DVD into the working directory.

After extracting the files, register the kernel source in Atmark-Dist. Perform all operations as shown in 6.1. Source Code Preparation.

```
[PC ~]$tar zxvf atmark-dist-[version].tar.gz
[PC ~]$tar zxvf linux-[version].tar.gz
[PC ~]$ ls
atmark-dist-[version].tar.gz  atmark-dist-[version]
linux-[version].tar.gz  linux-[version]
[PC ~]$ ln -s atmark-dist-[version] atmark-dist
[PC ~]$ cd atmark-dist
[PC ~/atmark-dist]$ ln -s ../linux-[version] linux-2.6.x
```

6.1 Source Code Preparation

6.1.2. Applying Default Configuration

Atmark-Dist should then be configured for the target Armadillo. As the Linux kernel is managed by Atmark-Dist, once Atmark-Dist has been configured appropriately the correct kernel configuration will be applied automatically.

Start the configuration by entering the commands shown in the following example^[1].

```
[PC ~/atmark-dist]$ make config
```

First, you will be asked to select a vendor. Please enter "AtmarkTechno".

```
[PC ~/atmark-dist]$ make config
config/mkconfig > config.in
#
# No defaults found
#
*
* Vendor/Product Selection
*
* Select the Vendor you wish to target
*
Vendor (3com, ADI, Akizuki, Apple, Arcturus, Arnewsh, AtmarkTechno, Atmel, Avnet,
Cirrus, Cogent, Conexant, Cwlinux, CyberGuard, Cytek, Exys, Feith, Future, GDB,
Hitachi, Imt, Insight, Intel, KendinMicrel, LEOX, Mecel, Midas, Motorola, NEC,
NetSilicon, Netburner, Nintendo, OPENcores, Promise, SNEHA, SSV, SWARM, Samsung,
SecureEdge, Signal, SnapGear, Soekris, Sony, StrawberryLinux, TI, TeleIP,
Triscend, Via, Weiss, Xilinx, senTec) [SnapGear] (NEW) AtmarkTechno
```

Next, you will be asked to select a product name. Please select the appropriate product name from 6.1. Product Name List.

6.1 Product Name List

Product	Product Name	Notes
Armadillo-440 LCD Model	Armadillo-440	Factory Image
Armadillo-420 Basic Model	Armadillo-420	Factory Image

Armadillo-440 is used in the example below.

```
* * Select the Product you wish to target * AtmarkTechno Products
(Armadillo-210.Base, Armadillo-210.Recover, Armadillo-220.Base,
Armadillo-220.Recover, Armadillo-230.Base, Armadillo-230.Recover,
Armadillo-240.Base, Armadillo-240.Recover, Armadillo-300, Armadillo-440,
Armadillo-500, Armadillo-500-FX.base, Armadillo-500-FX.dev, Armadillo-9,
Armadillo-9.PCMCIA, SUZAKU-V.SZ310, SUZAKU-V.SZ310-SID, SUZAKU-V.SZ310-SIL,
SUZAKU-V.SZ310-SIL-GPIO, SUZAKU-V.SZ410, SUZAKU-V.SZ410-SID, SUZAKU-V.SZ410-SIL,
SUZAKU-V.SZ410-SIL-GPIO, SUZAKU-V.SZ410-SIV) [Armadillo-210.Base]
(NEW) Armadillo-440
```

You will then be asked to select the development environment. Please enter "default". The armel (EABI) development environment will be used for the Armadillo-400 Series when "default" is selected.

```
* * Kernel/Library/Defaults Selection * * * Kernel is linux-2.6.x * Cross-dev
(default, arm-vfp, arm, armel, armmommu, common, h8300, host, i386, i960,
m68knommu, microblaze, mips, powerpc, sh) [default] (NEW) default
```

^[1]While this explains how to configure Atmark-Dist using the command line, it is also possible to use a menu system. For more details on using the menu system, please refer to the "Atmark-Dist Developers Guide."

Next, enter "None" for which libc to build. By selecting "None", the libc already installed in the development environment (glibc) will be used.

```
Libc Version (None, glibc, uC-libc, uClibc) [uClibc] (NEW)None
```

Enter "y" (yes) when asked whether or not to default all settings.

```
Default all settings (lose changes) (CONFIG_DEFAULTS_OVERRIDE) [N/y/?] (NEW)y
```

Enter "n" (no) for the final three questions.

```
Customize Kernel Settings (CONFIG_DEFAULTS_KERNEL) [N/y/?]n
Customize Vendor/User Settings (CONFIG_DEFAULTS_VENDOR) [N/y/?]n
Update Default Vendor Settings (CONFIG_DEFAULTS_VENDOR_UPDATE) [N/y/?]n
```

Once all options have been entered, the configuration will be saved to the build system and you will be returned to the prompt.

6.1.3. Building

To start the build, execute the command shown in 6.2. Atmark-Dist Build in the atmark-dist directory. Once the build completes, image files will have been created under the atmark-dist/images directory. romfs.img is the userland image and linux.bin is the kernel image. romfs.img.gz and linux.bin.gz are compressed versions of these files.

```
[PC ~/atmark-dist]$ make

[PC ~/atmark-dist]$ ls images
linux.bin linux.bin.gz romfs.img romfs.img.gz
```

6.2 Atmark-Dist Build

The created images can be written to the target Armadillo by following the steps in 5. Rewriting Flash Memory. The compressed images are normally used as they use the least amount of flash memory.

6.1.4. Customizing Images

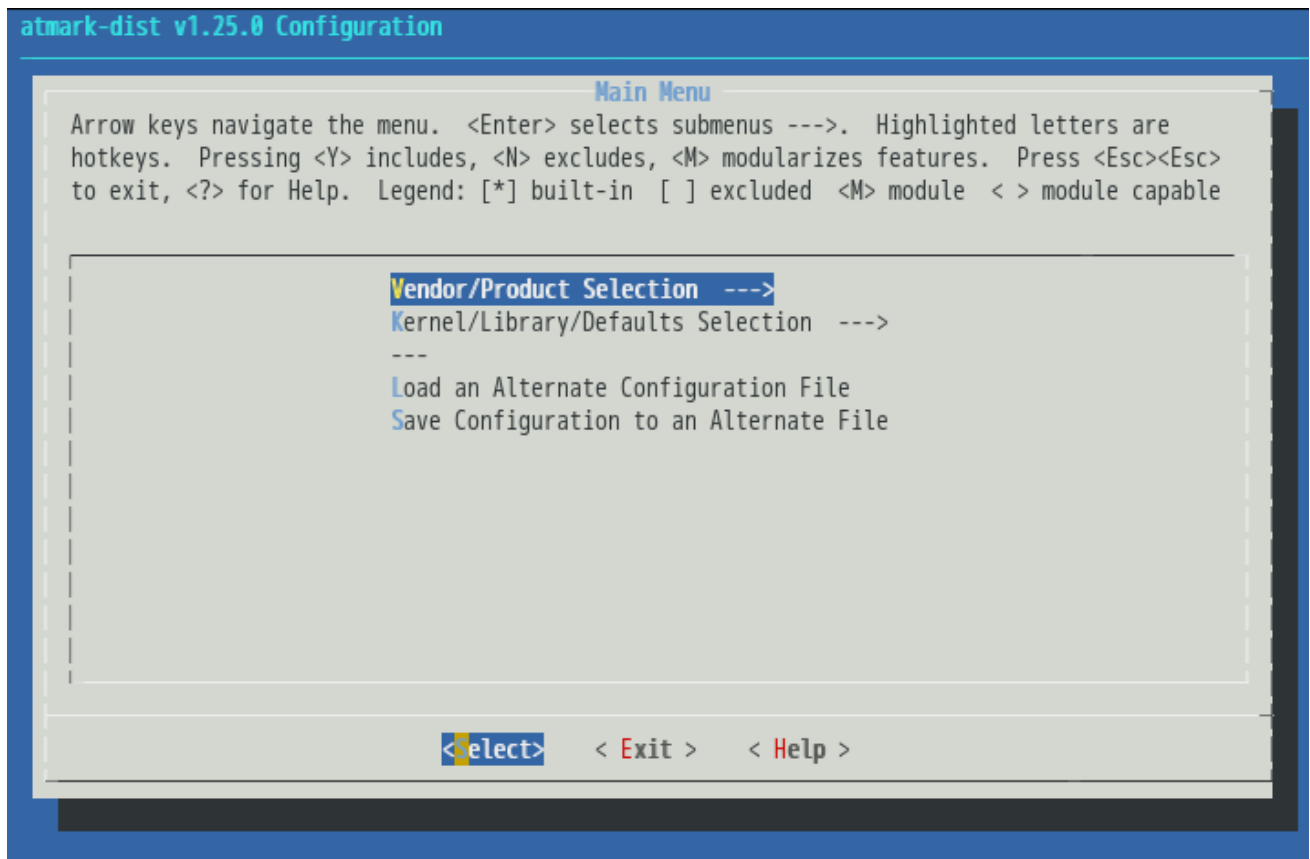
Atmark-Dist includes a variety of applications and libraries which can be included or removed from an image according to the chosen configuration. It is also possible to alter the kernel configuration.

Use the **make menuconfig** command to change the Atmark-Dist configuration.

```
[PC ~/atmark-dist]$ make menuconfig
```

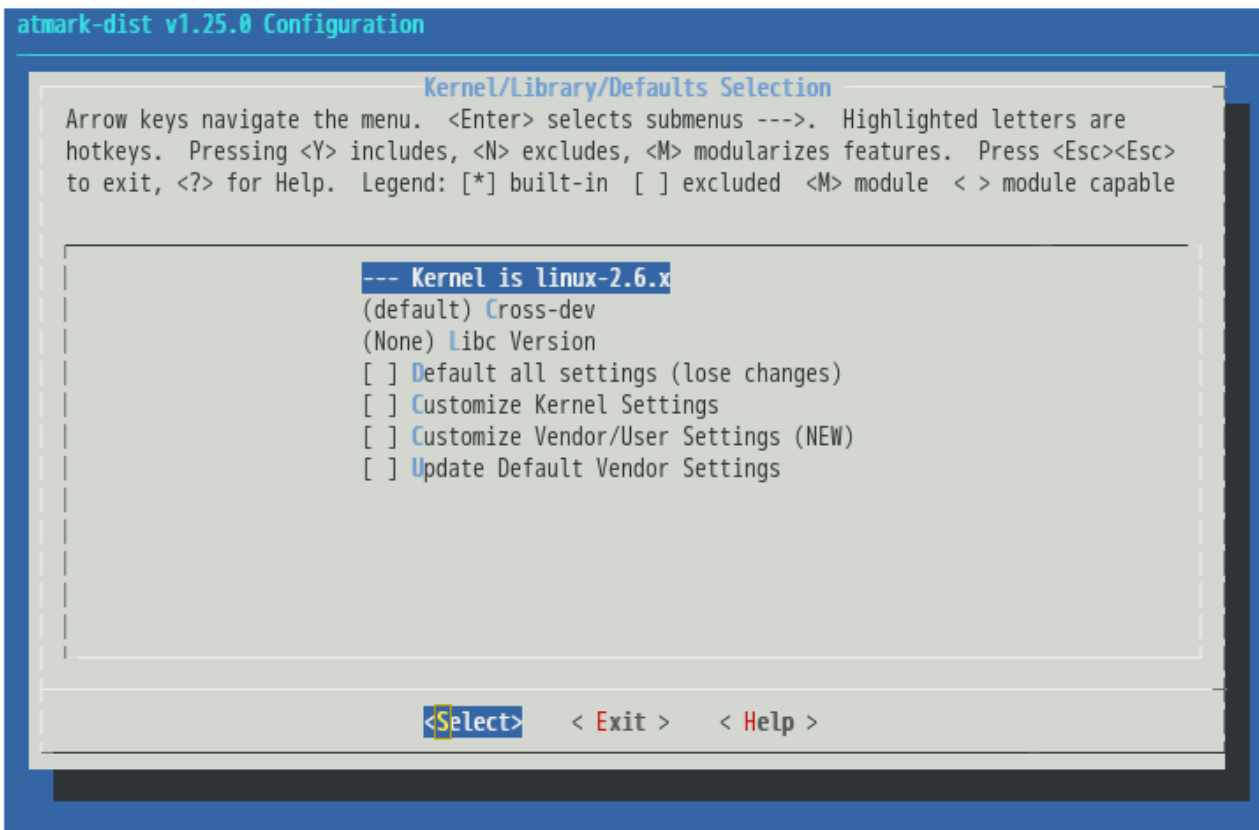
6.3 Atmark-Dist Configuration

After executing the **make menuconfig** command, the Main Menu screen as shown in 6.4. menuconfig: Main Menu is displayed.



6.4 menuconfig: Main Menu

Use the up and down arrow keys to highlight **Kernel/Library/Defaults Selection --->** and press enter. The Kernel/Library/Defaults Selection screen will be displayed.

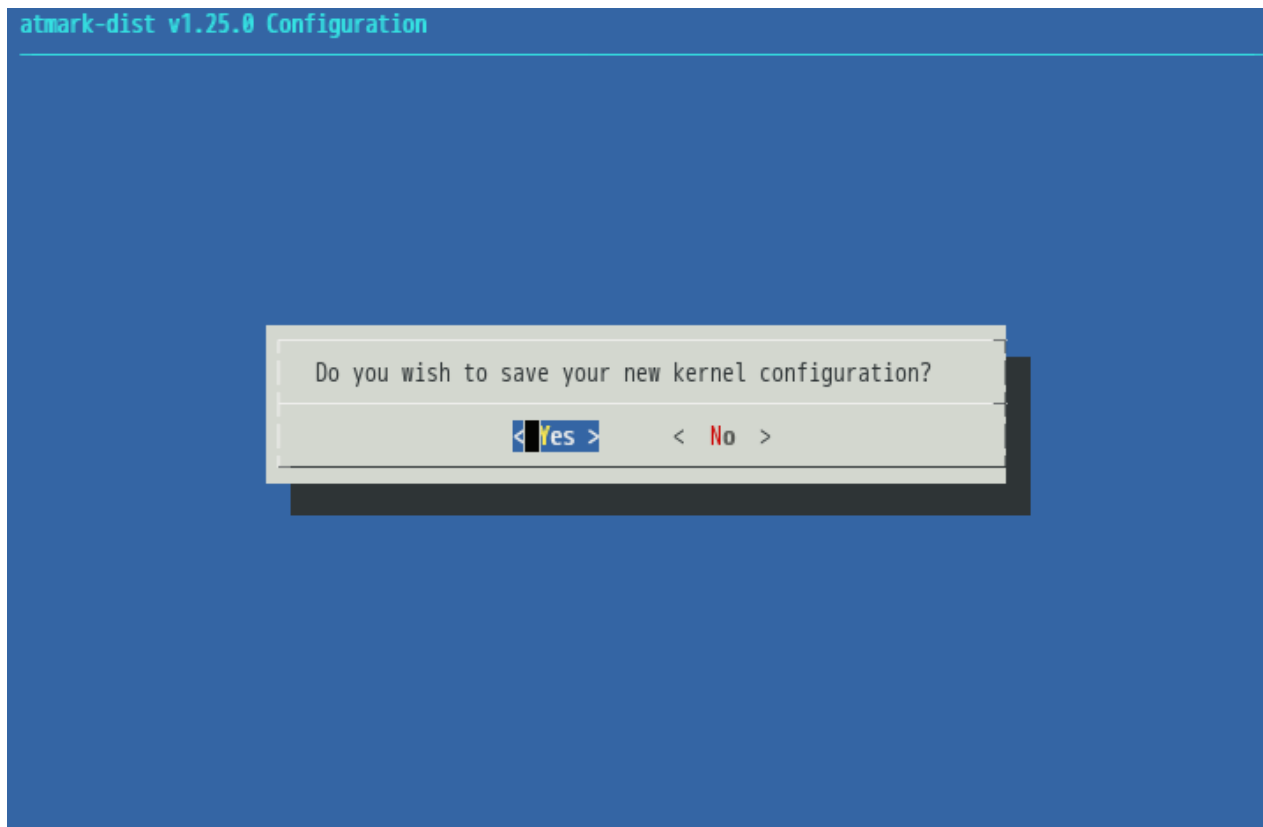


6.5 menuconfig: Kernel/Library/Defaults Selection

To change the kernel configuration, select **Customize Kernel Settings**. To change what applications or libraries are included in the userland, select **Customize Vendor/User Settings**. Each option can be selected by using the up and down arrow keys to highlight the relevant option and then pressing the space bar so that a "*" mark appears before it.

After making the selections, use the left and right arrow keys to highlight **Exit** and press Enter. You will be taken back to the Main Menu screen.

Again in the Main Menu screen highlight **Exit** and press Enter. The **Do you wish to save your new kernel configuration?** confirmation screen will then appear. Highlight **Yes** and press Enter.



6.6 menuconfig: Do you wish to save your new kernel configuration?

If you had selected the **Customize Kernel Settings** option, the Linux Kernel Configuration screen will appear. Changes to the kernel configuration can be made here. After making the required changes, highlight **Exit** on the Linux Kernel Configuration screen and press Enter to exit.


```
.config - Linux Kernel v2.6.26-at7 Configuration

Linux Kernel Configuration

Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are
hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc>
to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < >
module capable

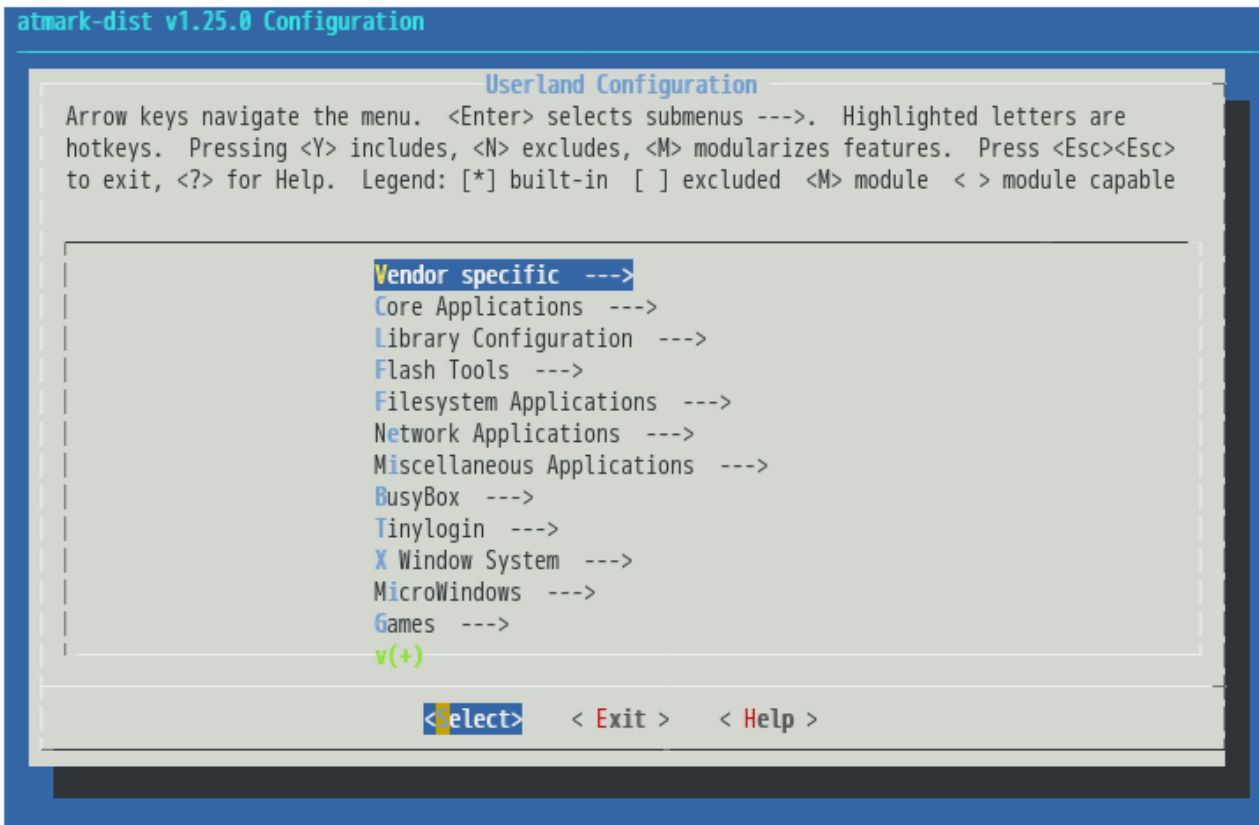
| General setup --->
[*] Enable loadable module support --->
[*] Enable the block layer --->
  System Type --->
  Bus support --->
  Kernel Features --->
  Boot options --->
  Floating point emulation --->
  Userspace binary formats --->
  Power management options --->
  Networking --->
  Device Drivers --->

v(+)
```

<Select> < **Exit** > < **Help** >

6.7 menuconfig: Linux Kernel Configuration

If you had selected the **Customize Vendor/User Settings** option, the Userland Configuration screen will then appear. Changes to what applications are libraries are included in the userland can be made here. After making the required changes, highlight **Exit** on the Userland Configuration screen and press Enter to exit.



6.8 menuconfig: Userland Configuration

The **Do you wish to save your new kernel configuration?** confirmation screen will then appear once again. Highlight Yes and press Enter.

This completes the configuration process.

For more details on the using **make menuconfig** for configuration, please refer to the "Atmark-Dist Developers Guide".

After the configuration process, use the **make** command in the same way described in 6.1.3. Building to create images based off the new configuration.

6.1.5. Adding An Application To The Userland Image

The following explains how to add files such as an original application not included in Atmark-Dist to the userland created in 6.1. Building Kernel and Userland Images.

The following example assumes that an original application has been built with the Out-Of-Tree method^[2] and has the filename `~/sample/hello`.

In Atmark-Dist, the files to be included in the userland image are stored under the `romfs` directory. It is possible to create a userland image which includes the original application by adding the application to this directory and then executing the **make image** command.

^[2]For information on Out-Of-Tree compiles, please refer to "Atmark-Dist Developers Guide".

```
[PC ~/atmark-dist]$ cp ~/sample/hello romfs/bin/
[PC ~/atmark-dist]$ make image
[PC ~/atmark-dist]$ ls images
linux.bin linux.bin.gz romfs.img romfs.img.gz
```

6.9 Userland Image Customization

The hello application is now installed under the /bin directory in the newly created userland image.

6.2. Building Bootloader Images

The following explains how to build a Hermit-At bootloader image from source code. The explanation applies to versions v2.0.0 or later of the source code.

6.2.1. Preparing Source Code

Extract `hermit-at-[version]-source.tar.gz` from the `source/bootloader` directory on the included DVD to the working directory. Complete the operation as shown in 6.10. Hermit-At Source Archive Extraction.

```
[PC ~]$ tar zxvf hermit-at-[version]-source.tar.gz
[PC ~]$ ln -s hermit-at-[version] hermit-at
[PC ~]$ cd hermit-at
[PC ~/hermit-at]$
```

6.10 Hermit-At Source Archive Extraction

6.2.2. Building

In versions v2.0.0 and later of Hermit-At, default configurations for each product have been prepared as defconfig files. To build an image with the same content as the factory image, execute the commands shown in 6.11. Hermit-At Build Example.

```
[PC ~/hermit-at]$ make armadillo4x0_defconfig
[PC ~/hermit-at]$ make
:
:
[PC ~/hermit-at]$ ls src/target/armadillo4x0/*.bin
loader-armadillo4x0-[version].bin
```

6.11 Hermit-At Build Example

The `loader-armadillo4x0-[version].bin` file is the newly created bootloader image.

Versions v2.0.0 and later of Hermit-At also support configuration with the `make menuconfig` command in the same way as Atmark-Dist. Commands can be added or removed and default behavior altered with this configuration method.

7. Kernel and Userland Placement

On the Armadillo-400 Series, by default the kernel and userland images are stored in flash memory and loaded to RAM by the bootloader before the kernel is booted.

It is also possible to have the kernel and userland images loaded from places other than flash memory on the Armadillo-400 Series.

This chapter explains how to load images from other places as well as the boot options required to do so.

7.1. Loading From A TFTP Server

The tftpboot function of the Hermit-At bootloader can download kernel and userland image files from a TFTP server, load the images to RAM and then boot them.

As images are booted without first writing them to flash memory, the tftpboot function can help development efficiency during stages when the images are updated regularly.

7.1.1. File Placement

Place the kernel and userland images in the root directory of the TFTP server.



A TFTP server (atftpd) is started by default on ATDE v3.0 and later. Files placed in the `/var/lib/tftpboot/` directory will be accessible via TFTP.

7.1.2. Boot Options

Set the jumpers on the target Armadillo appropriately for maintenance mode and power on the board.

Using the serial console software on the work PC, enter the following command^[1].

```
hermit> setbootdevice tftp [Armadillo IP address] [tftp server IP address]
        --kernel=kernel_image_file_name --userland=userland_image_file_name
```

7.1 tftpboot Command

Either one or both of the kernel and userland images may be specified.

Where the TFTP server has the IP address 192.168.10.1, the Armadillo has the IP address 192.168.10.10, the kernel image filename is `linux.bin.gz` and the userland image filename is `romfs.img.gz`, the command^[2] will be as shown below.

^[1]The command is only shown as multiple lines due to formatting constraints and should be entered as one line.

^[2]The command is only shown as multiple lines due to formatting constraints and should be entered as one line.

```
hermit> setbootdevice tftp 192.168.10.10 192.168.10.1
        --kernel=linux.bin.gz --userland=romfs.img.gz
```

7.2 tftpboot Command Example

When the TFTP configuration is set with the **setbootdevice** command, the settings are saved and the kernel and userland images will be loaded from the TFTP server on all future boots.

7.2. Loading From Storage

The kernel image can be loaded from a microSD card and the userland root filesystem can be stored on either a microSD card or USB memory on the Armadillo-400 Series.

The following explains how to load a kernel image and root filesystem both stored on a microSD card.

In this example one partition is created on a microSD card and formatted as an EXT3 filesystem. A root filesystem is then created there and the kernel image placed under the `/boot/` directory. The device from which the kernel image will be loaded is specified with Hermit-At's boot options. The placement of the root filesystem is specified with kernel parameters.



The steps described here apply when using versions 2.0.3 and later of the Hermit-At bootloader. As versions 2.0.2 and earlier are not compatible with EXT3 formatted boot partitions, these steps will not be usable.

If version 2.0.2 or earlier of the Hermit-At bootloader must be used for some reason, please refer to the explanation in version 1.2.0 of the Armadillo-400 Series Software Manual.

7.2.1. Partitioning

First, make one primary partition on the microSD card.

Insert the microSD card into the slot^[3] and create the partition as shown in 7.3. Partitioning Procedure.

^[3]The microSD slot on the Armadillo-400 Series is a lockable type. For information on inserting and removing the microSD card, please refer to the "Armadillo-400 Series Hardware Manual".

```

[armadillo ~]# fdisk /dev/mmcblk0
The number of cylinders for this disk is set to 124277.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
   (e.g., DOS FDISK, OS/2 FDISK)

Command (m for help): d ❶
Selected partition 1

Command (m for help): n ❷
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1 ❸
First cylinder (1-124277, default 1): ❹
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-124277, default 124277): ❺
Using default value 124277


Command (m for help): w ❺
The partition table has been altered!

Calling ioctl() to re-read partition table.
mmcblk0: p1
mmcblk0: p1
Syncing disks.

[armadillo ~]#
    
```

7.3 Partitioning Procedure

- ❶ First, delete a pre-existing partition. If there is more than one pre-existing partition, please delete them all.
- ❷ Make the new primary partition on the microSD card.
- ❸ Press enter to use the default value (1) for the first cylinder.
- ❹ Press enter to use the default value (124277) for the last cylinder also.
- ❺ Write the changes to the microSD card.



As the number of cylinders depends on the specifications of the microSD card being used, the number may not be the same as that displayed in the procedure above.

7.2.2. Creating Filesystems

Next, format the partition as an EXT3 filesystem as shown in 7.4. Filesystem Creation Procedure.

```
[armadillo ~]# mke2fs -j /dev/mmcblk0p1
mke2fs 1.25 (20-Sep-2001)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
497984 inodes, 994220 blocks
49711 blocks (5%) reserved for the super user
First data block=0
31 block groups
32768 blocks per group, 32768 fragments per group
16064 inodes per group
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736

Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 35 mounts or
180.00 days, whichever comes first.  Use tune2fs -c or -i to override.
```

7.4 Filesystem Creation Procedure

7.2.3. Kernel Image Placement

When booting from a microSD card, the kernel image must be stored in the /boot directory on the boot partition. Both uncompressed kernel images (Image, linux.bin) and compressed images (Image.gz, linux.bin.gz) are supported.

In the following example the **wget** command is used to obtain the kernel image. The URL that should be specified with the **wget** command differs depending on the product. Please refer to the table below for the correct URL.

7.1 Kernel Image Download URLs

Product	URL
Armadillo-420	http://download.atmark-techno.com/armadillo-420/image/linux-a400-[version].bin.gz
Armadillo-440	http://download.atmark-techno.com/armadillo-440/image/linux-a400-[version].bin.gz

The following example shows the kernel placement for Armadillo-440.

```
[armadillo ~]# mount /dev/mmcblk0p1 /mnt/
[armadillo ~]# mkdir /mnt/boot
[armadillo ~]# cd /mnt/boot
[armadillo /mnt/boot]# wget http://download.atmark-techno.com/armadillo-440/image/
linux-a400-[version].bin.gz
[armadillo /mnt/boot]# mv linux-a440-[version].bin.gz /mnt/boot/linux.bin.gz
[armadillo /mnt/boot]# cd
[armadillo ~]# umount /mnt
```

7.5 Kernel Image Placement


7.2.4. Creating A Root Filesystem

The following explains how to create a root filesystem on a microSD card.

Either Debian/GNU Linux or a filesystem created with Atmark-Dist can be used for the root filesystem.

7.2.4.1. Installing Debian GNU/Linux

To install Debian GNU/Linux, first obtain the archive files from either the debian directory on the included DVD or from the download site. These are files from a standard Debian GNU/Linux install which have been divided up and compressed into a number of archives. The installation can be performed by just extracting the archives to the root filesystem.



A partition with at least 1GB of free space is required when installing Debian GNU/Linux as the root filesystem.

In the following example the wget command is used to obtain the debian archives. The URLs that should be specified with the wget command differ depending on the product. Please refer to the table below for the correct URLs.

7.2 Debian Archive Download URLs

Product	URL
Both Armadillo-420 and Armadillo-440	http://download.atmark-techno.com/armadillo-4x0/debian/debian-lenny-armel-#.tgz ^[1]
	http://download.atmark-techno.com/armadillo-4x0/debian/debian-lenny-armel-a4x0.tgz

^[1]Note: the "#" is replaced with the range 1 to 5.

```
[armadillo ~]# mount /dev/mmcblk0p1 /mnt/
[armadillo ~]# mkdir tmp
[armadillo ~]# mount -t ramfs ramfs tmp
[armadillo ~]# cd tmp
[armadillo ~/tmp]#for N in 1 2 3 4 5 a4x0; do
> wget http://download.atmark-techno.com/armadillo-4x0/debian/debian-lenny-armel-  
{N}.tgz;
> gzip -cd debian-lenny-armel-${N}.tgz | (cd /mnt; tar xf -);
> sync;
> rm -f debian-lenny-armel-${N}.tgz;
> done
[armadillo ~/tmp]# cd
[armadillo ~]# umount tmp
[armadillo ~]# rmdir tmp
[armadillo ~]# umount /mnt
```

7.6 Root Filesystem Creation With Debian Archives

7.2.4.2. Using an Atmark-Dist Image

The following explains how to create a root filesystem on a microSD card by using a root filesystem produced with Atmark-Dist. It is possible to use a microSD with a smaller storage capacity than when installing Debian.

In the following example the wget command is used to obtain the initrd image containing the root filesystem produced with Atmark-Dist. The URL that should be specified with the wget command differs depending on the product. Please refer to the table below for the correct URL.

7.3 Atmark-Dist Image Download URL

Product	URL
Armadillo-420	http://download.atmark-techno.com/armadillo-420/image/romfs-a420-[version].img.gz
Armadillo-440	http://download.atmark-techno.com/armadillo-440/image/romfs-a440-[version].img.gz

```
[armadillo ~]# mount /dev/mmcblk0p1 /mnt/
[armadillo ~]# mkdir tmp
[armadillo ~]# mkdir romfs
[armadillo ~]# mount -t ramfs ramfs tmp
[armadillo ~]# wget http://download.atmark-techno.com/armadillo-440/image/romfs-
a440-[version].img.gz -P tmp
[armadillo ~]# gzip -d tmp/romfs-a440-[version].img.gz
[armadillo ~]# mount -o loop tmp/romfs-a440-[version].img romfs/
[armadillo ~]# (cd romfs/; tar cf - *) | (cd /mnt; tar xf -)
[armadillo ~]# sync
[armadillo ~]# umount romfs
[armadillo ~]# rmdir romfs
[armadillo ~]# umount tmp
[armadillo ~]# rmdir tmp
[armadillo ~]# umount /mnt
```

7.7 Root Filesystem Creation With Atmark-Dist Image

7.2.5. Boot Device and Kernel Parameter Settings

The place from which the kernel image will be loaded is selected with Hermit-At's boot device configuration. The placement of the root filesystem is specified with kernel parameters.

Set the jumpers for maintenance mode and reboot the board.

Execute the command shown in 7.8. Boot Device Designation in order to boot from a kernel image stored in the first partition on a microSD card.

```
hermit> setbootdevice mmcblk0p1
```

7.8 Boot Device Designation

Execute the command shown in 7.9. Root Filesystem Designation Example in order to use a root filesystem stored in the first partition on a microSD card.

```
hermit> setenv console=ttymxc1 root=/dev/mmcblk0p1 noinitrd rootwait
```

7.9 Root Filesystem Designation Example

7.3. Return Settings To Defaults

Enter the command shown in 7.10. Assigning Flash Memory As Boot Device to return to using flash memory as the boot device.

```
hermit> setbootdevice flash
```

7.10 Assigning Flash Memory As Boot Device

To return kernel parameters set with setenv to their default state, execute the clearenv command.

```
hermit> clearenv
```

7.11 Return Kernel Parameters To Default State With clearenv

8. Linux Kernel Device Driver Specifications

This chapter describes the specifications of the Linux kernel device drivers unique to the Armadillo-400 Series.

It is possible to use a wide range of functionality that is not enabled by default on the Armadillo-400 Series by altering the kernel configuration.

In order to use device drivers not enabled by default on the Armadillo-400 Series, the kernel must be configured by following the steps below.

1. Select which pins to assign the function to with board options.

When using `make menuconfig`, board options can be altered under **System Type -> Freescale MXC Implementations -> MX25 Options -> Armadillo-400 Board options** in the Linux Kernel Configuration.

The board options are designed so that multiple functions cannot be assigned to the same pins. Pins that do not have any specific function assigned to them will be configured as GPIO.

2. Enable the host (master) device driver.
3. If required, enable the slave device driver.
4. If required, add device information to `linux-2.6.26-at/arch/arm/mach-mx25/armadillo400.c`.

For information on altering the kernel configuration, please refer to 6.1.4. Customizing Images.

8.1. UART

The i.MX25 processor incorporates five UART modules, UART1 to UART5. On the Armadillo-400 Series, by default UART2 is used for Serial Interface 1 (CON3), UART3 for Serial Interface 2 (CON9), and UART5 for Serial Interface 3 (CON9). Also, UART3 and UART4 can be assigned to CON11 by altering the kernel configuration. When using UART9 or UART11, CTS/RTS hardware flow control can be selectively enabled or disabled.

The UART driver has the following functionality.

- 7/8 bit send and receive
- 1/2 stop bits
- None/Odd/Even parity
- XON/XOFF software flow control
- CTS/RTS hardware flow control
- Modem line control
- Standard Linux serial API
- Max baud rate: 230.4Kbps (Serial Interface 1) / 4Mbps^[1] (Serial Interface 2, 3)

Each serial interface and their corresponding device files are shown in 8.1. Serial Interface Device Files.

^[1]When DMA transfer is enabled. DMA is not enabled by default.

8.1 Serial Interface Device Files

Serial Interface	Device File	Module Used
Serial Interface 1	/dev/ttymxcl	UART2
Serial Interface 2	/dev/ttymxcl2	UART3
Serial Interface 3	/dev/ttymxcl4	UART5
Serial Interface 4	/dev/ttymxcl3	UART4

Kernel configuration options related to the UART functionality are shown in 8.2. UART Configuration.

8.2 UART Configuration

Kernel Configuration Option	Default	Description
SERIAL_MXC	y	Enables the i.MX serial driver
SERIAL_MXC_CONSOLE	y	Enables the system console for the i.MX serial driver
ARMADILLO400_UART3_CON9	y	Enables UART3 on CON9 Uses CON9_3 for UART3_RXD and CON9_5 for UART3_TXD
ARMADILLO400_UART3_HW_FLOW_CON9	n	Enables hardware flow control for UART3 on CON9 Uses CON9_11 for UART3_RTS and CON9_13 for UART3_CTS Depends on ARMADILLO400_UART3_CON9
ARMADILLO400_UART3_CON11	n	Enables UART3 on CON11 Uses CON11_40 for UART3_RXD and CON11_41 for UART3_TXD Cannot be used with ARMADILLO400_UART3_CON9
ARMADILLO400_UART3_HW_FLOW_CON11	n	Enables hardware flow control for UART3 on CON11 Uses CON11_42 for UART3_RTS and CON11_43 for UART3_CTS Depends on ARMADILLO400_UART3_CON11
ARMADILLO400_UART4_CON11	n	Enables UART4 on CON11 Uses CON11_44 for UART4_RXD and CON11_45 for UART4_TXD
ARMADILLO400_UART4_HW_FLOW_CON11	n	Enables hardware flow control for UART4 on CON11 Uses CON11_46 for UART4_RTS and CON11_47 for UART4_CTS Depends on ARMADILLO400_UART4_CON11
ARMADILLO400_UART5_CON9	y	Enables UART5 on CON9 Uses CON9_4 for UART5_RXD and CON9_6 for UART5_TXD

Kernel Configuration Option	Default	Description
ARMADILLO400_UART5_HW_FLOW_CON9	n	Enables hardware flow control for UART5 on CON9 Uses CON9_12 for UART5_RTS and CON9_14 for UART5_CTS Depends on ARMADILLO400_UART5_CON9

8.2. Ethernet

The Armadillo-400 Series Ethernet driver has the following functionality.

- Auto-negotiation support
- Carrier detect support
- Ethtool support
 - Link status
 - 10/100Mbps speed
 - Full/Half Duplex
 - Auto-negotiation enable/disable

8.3. SD/MMC/SDIO Host

The i.MX25 processor incorporates two SD/MMC/SDIO host controllers (eSDHC). On the Armadillo-400 Series, by default eSDHC1 is used for the microSD slot (CON1). Also, eSDHC2 can be assigned to CON9 by altering the kernel configuration.

The Armadillo-400 Series SD/MMC/SDIO host driver has the following functionality.

- 4 bit mode
- Card detect

When a card is inserted in the microSD card slot, it will be registered as `/dev/mmcblkN` (N is 0 or 1).

Kernel configuration options related to the SD/MMC/SDIO host functionality are shown in 8.3. SD/MMC/SDIO Host Controller Configuration.

8.3 SD/MMC/SDIO Host Controller Configuration

Kernel Configuration Option	Default	Description
MMC	y	Enables Linux kernel MMC/SD/SDIO card support
MMC_IMX_ESDHCI	y	Enables the i.MX25 eSDHC driver
ARMADILLO400_SDHC2_CON9	n	Enables SDHC2 on CON9 Uses CON9_15 to CON9_24

8.4. USB 2.0 Host

The Armadillo-400 Series USB 2.0 host driver has the following functionality.

- EHCI compliant
- OTG not supported
- USB High Speed host x1
- USB Full Speed host x1

When a USB device is detected, it will be mapped to `/dev/sd*`.

8.5. Frame Buffer

The video out function on Armadillo-440 is implemented as a frame buffer device.

The frame buffer device driver has the following functionality.

- Double-buffer support
- Max resolution: SVGA

The default settings on the Armadillo-440 LCD Model are shown below.

- Resolution: 480 x 272 pixels
- RGB 565 Color

The frame buffers and their corresponding device files are shown in 8.4. Frame Buffer Device Files.

8.4 Frame Buffer Device Files

Frame Buffer	Device File
Background plane	<code>/dev/fb0</code>
Graphic window	<code>/dev/fb1</code>

8.6. LED Backlight

The LED backlight function on Armadillo-440 is implemented as a backlight class driver.

The backlight can be controlled with the files under the `/sys/class/backlight/pwm-backlight` directory.

The brightness can be adjusted with the `brightness` file. Write a value between 0 (off) and 255 (full brightness) to the `brightness` file to change the brightness.

The current brightness can be obtained by reading the value from the `brightness` file.

The backlight on/off can be controlled with the `bl_power` file. Write 0 to `bl_power` to turn the backlight off and write 1 to turn it back on.

8.7. Touchscreen


The touchscreen function on Armadillo-440 is implemented as an input device and offers an event interface to userland programs.

The events sent by the interface are shown in 8.5. Touchscreen Events.

8.5 Touchscreen Events

Type	Code	Value
EV_KEY(1)	BTN_TOUCH(330)	0 or 1
EV_ABS(3)	ABS_X(0)	100 - 4000
EV_ABS(3)	ABS_Y(1)	100 - 4000
EV_ABS(3)	ABS_PRESSURE(24)	0 or 1

On the Armadillo-440 LCD Model, by default the touchscreen event device is mapped to /dev/input/event1.




The event device file numbers are determined by the order the input devices are detected. Therefore, if another input device such as a USB keyboard is detected at boot time, the event device number of the touchscreen may change.

8.8. Audio

The audio function on the Armadillo-400 Series is implemented as an ALSA device driver. The ALSA device driver has the following functionality.

- Playback (2ch) / Capture (1ch)
- Sampling rates: 48k, 32k, 24k, 16k, 12k, 8k Hz
- Formats: Signed 16/20/24 bit, Little-endian

The audio device can be controlled via the ALSA library (libasound2).



The Armadillo-400 Series audio driver cannot record and play sound at the same time.

On the i.MX25, it is possible to select what pins to assign the audio function to with the audio multiplex function. On the Armadillo-400 Series, the audio multiplex settings can be changed in the kernel configuration. By default, the audio multiplexing is set to use AUD6, with the audio function connected to CON11. By altering the configuration, it is possible to connect the audio function to CON9 by using AUD5.

Kernel configuration options related to the audio functionality are shown in 8.6. Audio Configuration.

8.6 Audio Configuration

Kernel Configuration Option	Default	Description
SOUND	y	Enables Linux kernel sound card support
SND	y	Enables Linux kernel ALSA support
SND_SOC	y	Enables Linux kernel ASoC support
SND_MXC_SOC	y	Enables driver which provides i.MX audio functionality
SND_SOC_ARMADILLO440_WM8978	y	Enables driver which provides audio functionality using the WM8978 codec on the Armadillo-400 Series

Kernel Configuration Option	Default	Description
ARMADILLO400_AUD5_CON11	y	Assign audio function to CON11 Uses CON11_42 to CON11_47
ARMADILLO400_AUD6_CON9	n	Assign audio function to CON9 Uses CON9_15, 17, 21, 22, 23, 24 Cannot be used with ARMADILLO400_AUD5_CON11

8.9. GPIO

The GPIO on the Armadillo-400 Series are implemented as generic GPIO.

There are two interfaces available to manipulate GPIO from userland: GPIO sysfs and an Armadillo-200 Series compatible GPIO driver. The GPIO sysfs driver is enabled by default.

Pins that do not have any other function assigned to them in the kernel configuration are all set to be GPIO.

8.9.1. GPIO sysfs

With GPIO sysfs, I/O direction and output levels can be set and input levels checked with the files under the `/sys/class/gpio/(GPIO_NAME)` directories.

The GPIO_NAME directories and their corresponding GPIO pins are shown in 8.7. GPIO_NAME and GPIO Pins.

8.7 GPIO_NAME and GPIO Pins

GPIO_NAME	GPIO Pin	Initial I/O Direction	Initial Output Level
CON9_1	CON9 pin 1	Input	-
CON9_2	CON9 pin 2	Input	-
CON9_11	CON9 pin 11	Input	-
CON9_12	CON9 pin 12	Input	-
CON9_13	CON9 pin 13	Input	-
CON9_14	CON9 pin 14	Input	-
CON9_15	CON9 pin 15	Input	-
CON9_16	CON9 pin 16	Input	-
CON9_17	CON9 pin 17	Input	-
CON9_18	CON9 pin 18	Input	-
CON9_21	CON9 pin 21	Input	-
CON9_22	CON9 pin 22	Input	-
CON9_23	CON9 pin 23	Input	-
CON9_24	CON9 pin 24	Input	-
CON9_25	CON9 pin 25	Input	-
CON9_26	CON9 pin 26	Input	-
CON9_27	CON9 pin 27	Output	LOW
CON9_28	CON9 pin 28	Output	LOW

I/O direction configuration is done with the `/sys/class/gpio/(GPIO_NAME)/direction` file by writing one of the options shown in 8.8. GPIO I/O Direction Configuration. The current configuration can be obtained by reading the value from the `direction` file.

8.8 GPIO I/O Direction Configuration

Configuration	Description
high	Sets the I/O direction to output and the output level to high. The output level can be obtained and set in this state.
low	Sets the I/O direction to output and the output level to low. The output level can be obtained and set in this state.
out	This is the same as setting "low".
in	Sets the I/O direction to input. The input level can be obtained in this state.

Output levels can be set and input levels checked with the `/sys/class/gpio/(GPIO_NAME)/value` file. A 0 represents a low level and 1 a high level.

Interrupt type configuration is done with the `/sys/class/gpio/(GPIO_NAME)/edge` file by writing one of the options shown in 8.9. GPIO Interrupt Type Configuration. The current configuration can be obtained by reading the value from the `edge` file.

8.9 GPIO Interrupt Type Configuration

Configuration	Description
none	Interrupt detection is not performed.
falling	Falling edge interrupt detection is performed.
rising	Rising edge interrupt detection is performed.
both	Both falling edge and rising edge interrupt detection are performed.

An example of handling GPIO sysfs interrupts in the C language is shown in 8.1. GPIO sysfs Interrupt Sample Program. When executed, the sample program sets the I/O direction of CON9_1 to input, the interrupt type to falling edge, and then waits for an interrupt. When an interrupt is detected the level of the GPIO pin at that time is displayed. The program exits after interrupts have been detected three times.

```

#include <stdio.h>
#include <unistd.h>
#include <poll.h>
#include <fcntl.h>

#define GPIO_DIR "/sys/class/gpio"
#define GPIO_NAME "CON9_1"
#define GPIO_PATH GPIO_DIR "/" GPIO_NAME "/"

int main(void)
{
    int fd;
    int i;

    fd = open(GPIO_PATH "direction", O_RDWR);
    write(fd, "in", 2);
    close(fd);

    fd = open(GPIO_PATH "edge", O_RDWR);
    write(fd, "falling", 7);
    close(fd);

    for (i=0; i > 3; i++) {
        char val;
        struct pollfd pfd;

        fd = open(GPIO_PATH "value", O_RDWR);
        read(fd, &val, 1);

        printf("waiting for interrupt..."); fflush(stdout);

        pfd.fd = fd;
        pfd.events = POLLIN;
        pfd.revents = 0;
        poll(&pfd, 1, -1);

        lseek(fd, 0, SEEK_SET);
        read(fd, &val, 1);
        close(fd);

        printf("OK (%c, %s)\n", val, val == '0' ? "Low" : "High");
        usleep(100000);
    }

    return 0;
}

```


8.1 GPIO sysfs Interrupt Sample Program

- ❶ The interrupt type is set to falling edge by writing "falling" to the edge file.
- ❷ After specifying the interrupt type, the value file is opened.
- ❸ A dummy read is performed once on the value file. Any interrupts occurring after this will be picked up in the polling.

- 4 The poll or select system call is used to wait for an interrupt to occur.
- 5 As a dummy read has been performed, in order to obtain the level of the GPIO pin after an interrupt has occurred the lseek system call must be used to return the offset to the start of the file.

8.9.2. Armadillo-200 Series Compatible GPIO Driver

With the Armadillo-200 Series compatible GPIO driver, GPIO are manipulated by issuing ioctl function calls to the driver specific device file.




The Armadillo-200 Series compatible GPIO driver is not enabled by default. To enable the driver, in the kernel configuration enable the CONFIG_ARMADILLO2X0_GPIO option after disabling the CONFIG_GPIO_SYSFS option and then rebuild the kernel.

The GPIO names and corresponding GPIO pins for the Armadillo-200 Series compatible GPIO driver are shown in 8.10. Armadillo-200 Series Compatible GPIO Driver GPIO List.

8.10 Armadillo-200 Series Compatible GPIO Driver GPIO List

GPIO Name	GPIO Pin	Initial I/O Direction	Initial Output Level
GPIO0	CON9 pin 21	Input	-
GPIO1	CON9 pin 22	Input	-
GPIO2	CON9 pin 23	Input	-
GPIO3	CON9 pin 24	Input	-
GPIO4	CON9 pin 25	Input	-
GPIO5	CON9 pin 26	Input	-
GPIO6	CON9 pin 27	Output	LOW
GPIO7	CON9 pin 28	Output	LOW
GPIO8	CON9 pin 11	Input	-
GPIO9	CON9 pin 12	Input	-
GPIO10	CON9 pin 13	Input	-
GPIO11	CON9 pin 14	Input	-
GPIO12	CON9 pin 15	Input	-
GPIO13	CON9 pin 16	Input	-
GPIO14	CON9 pin 17	Input	-
GPIO15	CON9 pin 18	Input	-



With the Armadillo-200 Series compatible GPIO driver, the GPIO names and the corresponding GPIO pin positions are the same as those on the Armadillo-200 Series. Because of this, only a subsection of the GPIO available on the Armadillo-400 Series can be used with the Armadillo-200 Series compatible GPIO driver.

The details of the device file are shown below.

8.11 Armadillo-200 Series Compatible GPIO Driver Device File

Type	Major Number	Minor Number	Device File
Character device	10	185	/dev/gpio

The file descriptor of the device file is specified as the first argument to `ioctl`, and the command to manipulate the GPIO is specified as the second argument.

8.12 Armadillo-200 Series Compatible GPIO Driver `ioctl` Commands

Command	Description	Third Argument Type
PARAM_SET	Configure GPIO with the state specified in the third argument	struct <code>gpio_param</code>
PARAM_GET	Get GPIO state by following the details specified in the third argument	struct <code>gpio_param</code>
INTERRUPT_WAIT	Wait for GPIO interrupt by following the details specified in the third argument	struct <code>wait_param</code>

The structures "struct `gpio_param`" and "struct `wait_param`" defined in `(kernel source)/include/linux/armadillo2x0_gpio.h` are used for the third argument. "struct `gpio_param`" can be linked with its "next" member to form a singly-linked list in order to control multiple GPIO at once. Be sure to set the "next" member of the last structure of the list to "0 (NULL)". For detailed usage information, please refer to the source code of the GPIO control application (`atmark-dist/vendors/AtmarkTechno/Armadillo-440/gpioctrl`).

8.10. LED

There are two LED drivers for the Armadillo-400 Series: a LED class driver and an Armadillo-200 Series compatible LED driver. The LED class driver is enabled by default.

8.10.1. LED Class

The LEDs can be controlled with the files under the `/sys/class/leds/(LED_NAME)` directory.

The `/sys/class/leds/(LED_NAME)/brightness` file is used to turn the LEDs on and off. Writing 0 to the `brightness` file turns the LED off, and writing any other number turns the LED on.

A trigger mechanism is used to control blinking with the LED class driver. The `mmc0`, `timer`, `heartbeat` and `default-on` triggers can be used on the Armadillo-400 Series. The trigger can be set by writing the respective character string to the `/sys/class/leds/(LED_NAME)/trigger` file.

When the `mmc0` trigger is set, the LED will blink on and off in time with data accesses to the MMC/SD card.

The `timer` trigger can be used to have the LED blink at a certain rate. After the `timer` trigger has been enabled, files `/sys/class/leds/(LED_NAME)/delay_on` and `/sys/class/leds/(LED_NAME)/delay_off` will be created. The length of time (msec) the LED should stay on for each cycle is written to the first file, and the time (msec) it should stay off for to the second file.

When the `heartbeat` trigger is set, the LED will blink regularly while the system is running.

When the `default-on` trigger is set, the LED will remain on continuously while the system is running.


The `LED_NAME` names and their respective LEDs are shown in 8.13. LED List.

8.13 LED List

LED_NAME	Corresponding LED	Default Trigger
red	LED3	None
green	LED4	default-on
yellow	LED5	None

8.10.2. Armadillo-200 Series Compatible LED Driver

With the Armadillo-200 Series compatible LED driver, LEDs are manipulated by issuing ioctl function calls to the driver specific device file.



The Armadillo-200 Series compatible LED driver is not enabled by default. To enable the driver, in the kernel configuration enable the CONFIG_ARMADILLO2X0_LED option after disabling the CONFIG_LEDS_GPIO option and then rebuild the kernel.

The details of the LED device file are shown below.

8.14 LED Node

Type	Major Number	Minor Number	Device File
Character device	10	215	/dev/led

The file descriptor of the device file is specified as the first argument to ioctl, and the command to manipulate the LEDs is specified as the second argument.

8.15 LED Manipulation Commands

Command	Description	Third Argument Type
LED_RED_ON	Turn LED3 (red) on	None
LED_RED_OFF	Turn LED3 (red) off	None
LED_RED_STATUS	Obtain state of LED3 (red)	Buffer to store state (1 byte min)
LED_RED_BLINKON	Start LED3 (red) blinking	None
LED_RED_BLINKOFF	Stop LED3 (red) blinking	None
LED_RED_BLINKSTATUS	Obtain blinking status of LED3 (red)	Buffer to store state (1 byte min)
LED_GREEN_ON	Turn LED4 (green) on	None
LED_GREEN_OFF	Turn LED4 (green) off	None
LED_GREEN_STATUS	Obtain status of LED4 (green)	Buffer to store state (1 byte min)
LED_GREEN_BLINKON	Start LED4 (green) blinking	None
LED_GREEN_BLINKOFF	Stop LED4 (green) blinking	None
LED_GREEN_BLINKSTATUS	Obtain blinking status of LED4 (green)	Buffer to store state (1 byte min)

For detailed usage information on the LED device driver, please refer to the source code of the sample LED control application (atmark-dist/vendors/AtmarkTechno/Armadillo-440/ledctrl).

8.11. Buttons

The button input function on the Armadillo-400 Series is implemented as an input device and offers an event interface to userland programs.


On the Armadillo-440 LCD Model, the Armadillo-440 has one on-board tact switch and the LCD Expansion Board has three buttons mounted.

The buttons and their corresponding events are shown in 8.16. Armadillo-440 LCD Model Button Events.

8.16 Armadillo-440 LCD Model Button Events

Buttons	Type	Code	Value
SW1	EV_KEY(1)	KEY_ENTER(28)	0 or 1
LCD_SW1	EV_KEY(1)	KEY_BACK(158)	0 or 1
LCD_SW2	EV_KEY(1)	KEY_MENU(139)	0 or 1
LCD_SW3	EV_KEY(1)	KEY_HOME(102)	0 or 1

On the Armadillo-440 LCD Model, by default the button event device is mapped to `/dev/input/event0`.



The event device file numbers are determined by the order the input devices are detected. Therefore, if another input device such as a USB keyboard is detected at boot time, the event device number of the button function may change.

8.12. Real-time Clock

A real-time clock is incorporated on the Armadillo-440 LCD Expansion Board and the Armadillo-400 RTC Option Module.

The real-time clock is registered as `/dev/rtc0`. When using the RTC Option Module with the Armadillo-440 LCD Model, the real-time clock on the RTC Option Module is registered as `/dev/rtc0`, and the real-time clock on the LCD Expansion Board becomes `/dev/rtc1`.

8.13. Watchdog Timer

The i.MX25 processor on the Armadillo-400 Series boards includes an internal watchdog timer.

The default bootloader on the Armadillo-400 Series enables the internal watchdog timer straight after the board is powered on. It is set to time out after 10 seconds by default.

The watchdog timer is kicked automatically by the Linux kernel.

If for some reason the Linux kernel freezes and as a result is unable to kick the watchdog timer and a time-out occurs, the system will reset.

8.14. I2C

The i.MX25 processor includes three I2C controllers: I2C1 to I2C3. On the Armadillo-400 Series, by default I2C1 is used as a board internal bus, I2C2 is assigned to CON14 and I2C3 to CON11.

The I2C bus driver has the following functionality.

- I2C master mode
- 400 kbps max

In order to use a slave device connected to an I2C bus, a chip driver for the appropriate device must be enabled. Also, if the chip driver is written in "new style"^[2], struct `i2c_board_info` must be configured appropriately. For the Armadillo-400 Series, please add the details to the `armadillo400_i2cN_board_info` array in `linux-2.6.26-at/arch/arm/maxh-mx25/armadillo400.c` (where N corresponds to the bus number).

Kernel configuration options related to the I2C functionality are shown in 8.17. I2C Configuration.

8.17 I2C Configuration

Kernel Configuration Option	Default	Description
ARMADILLO400_I2C2_CON14	y	Enables I2C2 on CON14 Uses CON14_3 for I2C2_SCL and CON14_4 for I2C2_SDA
ARMADILLO400_I2C3_CON11	y	Enables I2C3 on CON11 Uses CON11_48 for I2C3_SCL and CON11_49 for I2C3_SDA

8.15. SPI

The i.MX25 processor includes three SPI controllers: CSPI1 to CSPI3. On the Armadillo-400 Series, CSPI1 and CSPI3 can be assigned to CON9 in the kernel configuration.

The SPI master driver has the following functionality.

- SPI master mode
- Multiple slave selects

The SPI master driver is not enabled by default. In order to use a slave device connected to a SPI bus, both the SPI master driver and a driver for the slave device must be enabled. Also, a struct `spi_board_info` must be configured appropriately. On the Armadillo-400 Series, please add the details to the `armadillo400_spiN_board_info` array in `linux-2.6.26-at/arch/arm/maxh-mx25/armadillo400.c` (where N corresponds to the bus number).

Kernel configuration options related to the SPI functionality are shown in 8.18. SPI Configuration.

8.18 SPI Configuration

Kernel Configuration Option	Default	Description
SPI	n	Enables Linux kernel SPI support
SPI_MXC	n	Enables the i.MX SPI master driver
ARMADILLO400_SPI1_CON9	n	Enables SPI1 on CON9 Uses CON9_3 for CSPI1_MOSI, CON9_5 for CSPI1_MISO, CON9_13 for CSPI1_SCLK and CON9_26 for CSPI1_RDY
ARMADILLO400_SPI1_SS0_CON9_25	n	Uses CON9_25 for SPI1_SS0 Depends on ARMADILLO400_SPI1_CON9
ARMADILLO400_SPI1_SS1_CON9_11	n	Uses CON9_11 for SPI1_SS1 Depends on ARMADILLO400_SPI1_CON9
ARMADILLO400_SPI3_CON9	n	Enables SPI3 on CON9 Uses CON9_4 for CSPI3_MOSI, CON9_6 for CSPI3_MISO, CON9_12 for CSPI3_SCLK and CON9_14 for CSPI3_RDY

^[2]Refer to `linux-2.6.26-at/Documentation/i2c/writing-clients`.

Kernel Configuration Option	Default	Description
ARMADILLO400_SPI3_SS0_CON9_16	n	Uses CON9_16 for SPI3_SS0 Depends on ARMADILLO400_SPI3_CON9
ARMADILLO400_SPI3_SS1_CON9_18	n	Uses CON9_18 for SPI3_SS1 Depends on ARMADILLO400_SPI3_CON9
ARMADILLO400_SPI3_SS2_CON9_15	n	Uses CON9_15 for SPI3_SS2 Depends on ARMADILLO400_SPI3_CON9
ARMADILLO400_SPI3_SS2_CON9_17	n	Uses CON9_17 for SPI3_SS3 Depends on ARMADILLO400_SPI3_CON9

8.16. One Wire

CON9_2 and CON9_26 can be used as one wire masters on the Armadillo-400 Series. The one wire controller included in the i.MX25 processor provides the master functionality for CON9_2, and the GPIO one wire driver provides the same for CON9_26.

The one wire master drivers are not enabled by default. In order to use a slave device connected to a one wire bus, both a one wire master driver and a driver for the slave device must be enabled.

Kernel configuration options related to the one wire functionality are shown in 8.19. One Wire Configuration.

8.19 One Wire Configuration

Kernel Configuration Option	Default	Description
W1	n	Enables Linux kernel one wire support
W1_MASTER_MXC	n	Enables the i.MX25 internal one wire master controller driver Please enable when using CON9_2
W1_MASTER_GPIO	n	Enables GPIO one wire master driver Please enable when using CON9_26
ARMADILLO400_W1_CON9_2	n	Use CON9_2 as one wire
ARMADILLO400_W1_CON9_26	n	Use CON9_26 as one wire

8.17. PWM

The i.MX25 processor includes four PWM modules: PWM1 to PWM4. On the Armadillo-400 Series, by default PWM1 is used for the LED backlight (CON11_12). PWN2 can be assigned to CON9_25 and PWM4 to CON14_3 by altering the kernel configuration.

With the i.MX25 PWM driver, configuration can be changed by writing an appropriate value to the files under /sys/class/mxc_pwm/(PWM_NAME). Files used for configuration are shown in 8.20. PWM sysfs.

8.20 PWM sysfs

File Name	Description
period_ns	PWN period set in nsec Acceptable range is from 17 to 2,147,483,647 (from 20usec to 2sec approx.)
duty_ns	PWM on time (off time when invert = 1) set in nsec Acceptable range is 0 < duty_ns < period_ns
invert	PWM output reversed when set to 1

File Name	Description
enable	PWM output enabled when set to 1 Output stopped when set to 0 period_ns, duty_ns and invert can be changed even while PWM is enabled

Kernel configuration options related to the PWM functionality are shown in 8.21. PWM Configuration.

8.21 PWM Configuration

Kernel Configuration Option	Default	Description
MXC_PWM	y	Enable the i.MX25 PWM driver
MXC_PWM_CLASS	y	Enable PWM configuration via sysfs
ARMADILLO400_PWM2_CON9_25	n	Uses CON9_25 as PWM2 The PWM_NAME is CON9_25
ARMADILLO400_PWM4_CON14_3	n	Uses CON14_3 as PWM4 The PWM_NAME is CON14_3

8.18. CAN

The i.MX25 processor includes two CAN controllers (FlexCAN): CAN1 and CAN2. On the Armadillo-400 Series, CAN2 can be assigned to CON14 by altering the kernel configuration. The CAN driver is not enabled by default.

The CAN functionality is provided by the SocketCAN framework. For information on SocketCAN, please refer to linux-2.6.26-at/Documentation/networking/can.txt.

The CAN driver has the following functionality.

- Base and Extended frame format support
- 1Mbps max

Configuration can be changed by writing an appropriate value to the files under /sys/devices/platform/FlexCAN.1/. Files used for configuration are shown in 8.22. CAN sysfs^[3].

8.22 CAN sysfs

File Name	Description	Default Value	Use Conditions
br_clksrc	Specify clock source When bus is specified, 66.5MHz is used for clock source When osc is specified, 24MHz is used for clock source	bus	A
br_presdiv	Specify clock source prescaler divider 1 to 8 can be specified	7	A
br_propseg	Set propagation segment value 1 to 8 can be specified	5	A
br_pseg1	Set phase buffer segment 1 value 1 to 8 can be specified	5	A
br_pseg2	Set phase buffer segment 2 value 1 to 8 can be specified	8	A

^[3]Please do not use any file not described here as they are only provided for backwards compatibility.

File Name	Description	Default Value	Use Conditions
br_rjw	Set resynchronization jump width 1 to 4 can be specified	3	A
bitrate	Displays transmission speed (bps) Cannot be written to	500000	None
std_msg	Set whether or not to support standard format 1 for support, 0 for no support	1	A
ext_msg	Set whether or not to support extended format 1 for support, 0 for no support	1	A
maxmb	Set max message buffer number 2 to 64 can be specified	64	A
rx_maxmb	Set receive message buffer size Send message buffer size is: maxmb - rx_maxmb 1 to maxmb - 1 can be specified	32	A
state	Display current status Displayed with format: "interface status::error status" Interface status is either "Start" (up) or "Stop" (down) Error status is one of following: "normal", "error passive", "bus off"	Stop::normal	None
boff_rec	Set whether or not to recover from bus off automatically 0 to recover automatically, 1 to not recover	1	A
listen	Set whether or not to enable listen mode (receive only) 1 to enable listen mode, 0 to disable	0	A
loopback	Set whether or not to enable loopback mode 1 to enable loopback mode, 0 to disable	0	A
smp	Set sampling behavior 0 for determining received bit value from 1 sample 1 for determining received bit value by majority count from 3 samples	1	A
srx_dis	Set whether or not to receive sent frames 0 to receive sent frames, 1 to not receive	1	A
set_resframe	Set data frame to reply with after receiving remote frame Set with format: "ID#DATA" ID specified with 3 digit hex (standard format) or 8 digit hex value Data specified as 0 to 8 groups of 2 digit hex values Data can be separated by "."	None	B
del_resframe	Delete data frame set with set_resframe Specify ID of data frame to delete with 3 digit hex (standard format) or 8 digit hex value	None	B
show_resframe	Display data frame set with set_resframe Cannot be written to	None	C
wakeup	Set whether or not to enable CAN receive wakeup while suspended 1 to enable wakeup, 0 to disable	0	A

File Name	Description	Default Value	Use Conditions
wak_src	Set whether or not to use low-pass filter while suspended 1 to enable filter, 0 to disable	0	A

- Condition A: can be set when the network interface is in the off state (ifconfig canX off).
- Condition B: can be set when the network interface is in the on state (ifconfig canX on).
- Condition C: can be referenced when the network interface is in the on state (ifconfig canX on).

Transmission speed is determined by the following formula.

```
src_clk = 66,500,000 (br_clksrc = bus condition)
src_clk = 24,000,000 (br_clksrc = osc condition)
Transmission Speed[bps] = src_clk / br_presdiv / (1 + br_propseg + br_pseg1 + br_pseg2)
```

8.2 CAN Transmission Speed Calculation


Kernel configuration options related to the CAN functionality are shown in 8.23. CAN Configuration.

8.23 CAN Configuration

Kernel Configuration Option	Default	Description
CAN	n	Enable Linux kernel CAN support
CAN_RAW	n	Enable RAW_CAN protocol support
CAN_BCM	n	Enable CAN_BCM protocol support
CAN_FLEXCAN	n	Enable the i.MX25 FlexCAN driver
ARMADILLO400_CAN2_CON14	n	Uses CON14 for CAN2 Uses CON14_3 for CAN2_TXCAN and CON14_4 for CAN2_RXCAN

8.19. Keypad

The i.MX25 processor includes a keypad controller. On the Armadillo-400 Series, the keypad can be assigned to CON11 by altering the kernel configuration. The keypad driver is not enabled by default.



When the keypad driver is enabled, the event device will be mapped to /dev/input/event0. This will cause the default button and touchscreen event device numbers to change.

Kernel configuration options related to the keypad functionality are shown in 8.24. Keypad Configuration.

8.24 Keypad Configuration

Kernel Configuration Option	Default	Description
INPUT	y	Enable Linux kernel input layer support
INPUT_EVDEV	y	Enable input layer event device support
KEYBOARD_MXC	n	Enable the i.MX25 keypad driver

Kernel Configuration Option	Default	Description
ARMADILLO400_KEYPAD_CON11	n	Enable keypad on CON11 Please enable at least one COL and ROW each
ARMADILLO400_KEYPAD_ROW0_CON11_40	n	Uses CON11_40 as ROW0
ARMADILLO400_KEYPAD_ROW1_CON11_41	n	Uses CON11_41 as ROW1
ARMADILLO400_KEYPAD_ROW2_CON11_42	n	Uses CON11_42 as ROW2
ARMADILLO400_KEYPAD_ROW3_CON11_43	n	Uses CON11_43 as ROW3
ARMADILLO400_KEYPAD_COL0_CON11_44	n	Uses CON11_44 as COL0
ARMADILLO400_KEYPAD_COL1_CON11_45	n	Uses CON11_45 as COL1
ARMADILLO400_KEYPAD_COL2_CON11_46	n	Uses CON11_46 as COL2
ARMADILLO400_KEYPAD_COL3_CON11_47	n	Uses CON11_47 as COL3
ARMADILLO400_KEYPAD_ROW4_CON11_48	n	Uses CON11_48 as ROW4
ARMADILLO400_KEYPAD_ROW5_CON11_49	n	Uses CON11_49 as ROW5

To set what ROW and COL range is used for the keypad, specify the appropriate values in the `armadillo440_keypad_data` variable in `linux-2.6.26-at/arch/arm/maxh-mx25/armadillo400.c`. Specify the button to event key mappings in the `armadillo440_keymapping` variable.

8.20. Power Management

The Armadillo-400 Series support the Linux power management sleep functionality. In the sleep state, the execution of applications is paused and the kernel enters the suspend state. Power consumption is kept at a minimum during the sleep state as the operation of external devices is halted. When returning to the normal execution state from the sleep state, the kernel's resume logic is run and applications are returned to a running state.

Sleep mode can be entered by writing either "standby" or "mem" to the `/sys/power/state` file. The system will return to the normal execution state from sleep mode when a wakeup interrupt occurs.

The differences in state of each mode are shown in 8.25. Sleep Modes. The suspend-to-RAM sleep mode provides for a greater reduction in power consumption compared to the power-on suspend mode.

8.25 Sleep Modes

Sleep Modes	Character String Written to state File	i.MX25 Power Mode	Wakeup Basis
power-on suspend	standby	Doze Mode	Serial input, touchscreen input and button input
suspend-to-RAM	mem	Stop Mode	Button input

For devices that can be used as a basis for wakeups, whether or not they do trigger a wakeup can be specified with their `power/wakeup` sysfs entry. Write "enabled" to the `power/wakeup` file and the device will function as a basis for wakeups, and write "disabled" and they will not.

Please refer to 8.26. Wakeup Basis Designation for details.

8.26 Wakeup Basis Designation

Device	sysfs File	Initial State
Serial Interface 1	<code>/sys/devices/platform/mxcintuart.1/tty/ttymxc1/power/wakeup</code>	enabled
Serial Interface 2	<code>/sys/devices/platform/mxcintuart.2/tty/ttymxc2/power/wakeup</code>	disabled
Serial Interface 3	<code>/sys/devices/platform/mxcintuart.4/tty/ttymxc4/power/wakeup</code>	disabled
Touchscreen	<code>/sys/devices/platform/imx_adc.0/power/wakeup</code>	enabled

Device	sysfs File	Initial State
Buttons	/sys/devices/platform/gpio-keys.0/power/wakeup	enabled
Keypad	/sys/devices/platform/mxc_keypad.0/power/wakeup	enabled
FlexCAN	/sys/devices/platform/FlexCAN.1/wakeup	disabled

8.20.1. Treatment of External Devices During Sleep

On the Armadillo-400 Series, the power supply to all external devices is halted as part of the suspend logic.

Because of this, USB devices must be put into a state where they can be safely disconnected before the system is put into the sleep state. That is, USB memory must be unmounted first. As the devices will be detected again when the system resumes, USB devices can be disconnected and reconnected while the system is in the sleep state.

As opposed to this, microSD cards can be left mounted when the system is put into the sleep state. In order for this to be possible, the SD host driver does not probe for cards at resume time, and instead assumes the same card is still inserted. Because of this, microSD cards cannot be removed or inserted when the system is in the sleep state.

For Ethernet devices, at resume time the same process is followed as when a cable is reconnected. Therefore, Auto-negotiation will take place at resume time if it is enabled.

By default, +3.3V_IO output is stopped when the system is put into the sleep state. However, if Serial Port 2 or Serial Port 4 are selected as a wakeup basis, the +3.3V_IO output will continue even during the sleep state.

A Hermit-At Bootloader

Hermit-At is a functional downloader and bootloader used on Atmark Techno products. The Hermit-At bootloader prompt is displayed when an Armadillo board is booted in maintenance mode. From the prompt it is possible to enter commands for a variety of operations such as updating flash memory and setting Linux kernel parameters. The following gives details on the most well used functions.

A.1. version

This command displays version information.

```
Syntax: version
```

A.1 version Syntax

A.1.1. version Example

```
hermit> version  
Hermit-At v2.0.0 (armadillo4x0) compiled at 23:03:08, Mar 08 2010
```

A.2 version Example

A.2. info

This command displays board information.

```
Syntax: info
```

A.3 info Syntax

A.2.1. info Example

```
hermit> info  
Board Type: 0x00000440  
Hardware ID: 0x00000300  
  DRAM ID: 0x00000002  
    Jumper: 0x00000001  
  Tact-SW: 0x00000000
```

A.4 info Example

A.3. memmap

This command displays the flash memory and DRAM memory map.

```
Syntax: memmap
```

A.5 memmap Syntax

A.3.1. memmap Example

```
hermit> memmap
0xa0000000:0xa1ffffff FLA all bf:8K bl:4x32K/1,255x128K/1
0xa0000000:0xa001ffff FLA bootloader bf:8K bl:4x32K/1
0xa0020000:0xa021ffff FLA kernel bf:8K bl:16x128K
0xa0220000:0xa1fdffff FLA userland bf:8K bl:238x128K
0xa1fe0000:0xa1ffffff FLA config bf:8K bl:1x128K
0x80000000:0x87ffffff RAM dram-1
```

A.6 memmap Example

A.4. mac

This command displays the MAC address.

```
Syntax: mac
```

A.7 mac Syntax

A.4.1. mac Example

```
hermit> mac
00:11:0c:00:00:00
```

A.8 mac Example

A.5. md5sum

This command calculates and displays the md5sum value of the specified memory region.

```
Syntax: md5sum <start address> <size>
```

A.9 md5sum Syntax

A.5.1. md5sum Example

To calculate and display the md5sum value of the first 1024 bytes of the bootloader region, execute the command shown in A.10. md5sum Example.

```
hermit> memmap
0xa0000000:0xa1ffffff FLA all bf:8K bl:4x32K/1,255x128K/1
0xa0000000:0xa001ffff FLA bootloader bf:8K bl:4x32K/1
0xa0020000:0xa021ffff FLA kernel bf:8K bl:16x128K
0xa0220000:0xa1fdffff FLA userland bf:8K bl:238x128K
0xa1fe0000:0xa1ffffff FLA config bf:8K bl:1x128K
0x80000000:0x87ffffff RAM dram-1
hermit> md5sum 0xa0000000 1024
fd44ce938f65726dc59669f537154429
```

A.10 md5sum Example

A.6. erase

This command erases flash memory.

```
Syntax: erase [address]
```

A.11 erase Syntax

A.6.1. erase Example

```
hermit> erase 0xa0fe0000
```

A.12 erase Example

A.7. setenv and clearenv

These commands are used to set Linux kernel parameters. The parameters set with setenv are passed to the kernel at boot time. Executing clearenv clears any configuration. The parameters are saved in flash memory and are therefore maintained even after rebooting.

```
Syntax: setenv [kernel parameter]...
```

Explanation: Sets kernel parameters. Shows current configuration if executed without any options specified.

```
Syntax: clearenv
```

Explanation: Clears all set options.

A.13 setenv/clearenv Syntax

A.7.1. setenv/clearenv Example

```
hermit> setenv console=ttymxc1
hermit> setenv
1: console=ttymxc1
hermit> clearenv
hermit> setenv
hermit>
```

A.14 setenv and clearenv Example

A.7.2. Linux Kernel Parameters

Example Linux kernel parameters are shown in A.1. Well Used Linux Kernel Parameters. For information on other options, please refer to `linux-2.6/Documentation/kernel-parameters.txt`.

A.1 Well Used Linux Kernel Parameters

Option	Description
console	Specify device to be used for kernel console.
root	Specify root filesystem related settings.
rootdelay	Seconds to wait before attempting to mount root filesystem.
rootwait	Wait until the root filesystem is accessible before attempting to mount it.
noinitrd	Specify what should happen with the initrd data after the kernel has booted.
nfsroot	Specify root filesystem place and NFS options when using NFS.



When `ttymxc1,2,4` is specified for the console option, the serial interface used by Hermit-At will also change to that interface after the next boot.

A.8. setbootdevice

This command is used to specify the boot device holding the Linux kernel. This setting is saved in flash memory and is therefore maintained even after rebooting.

Syntax: `setbootdevice flash`

Explanation: Extract the kernel image stored in the kernel flash memory region to RAM and boot it

Syntax: `setbootdevice tftp <client IP address> <server IP address> [--kernel=<path>] [--userland=<path>]`

Explanation: Download kernel and/or userland image from TFTP server, extract to RAM and boot

Syntax: `setbootdevice mmcblkOpN`

Explanation: Extract kernel image from `/boot/` directory in partition N of MMC/SD card to RAM and boot

A.15 setbootdevice Syntax

A.8.1. setbootdevice Example

To boot the kernel image stored in flash memory, execute the command shown in A.16. Assigning Flash Memory As Boot Device.

```
hermit> setbootdevice flash
```

A.16 Assigning Flash Memory As Boot Device

To download and boot a kernel image named linux.bin.gz from a TFTP server (IP address: 192.168.10.10), execute the command shown in A.17. Assigning TFTP Server As Boot Device.

```
hermit> setbootdevice 192.168.10.10 192.168.10.1 --kernel=linux.bin.gz
```

A.17 Assigning TFTP Server As Boot Device

To boot a kernel image stored in the first partition of a SD/MMC card, execute the command shown in A.18. Assign SD/MMC Card As Boot Device.

```
hermit> setbootdevice mmcblk0p1
```

A.18 Assign SD/MMC Card As Boot Device

A.9. frob

This command is used to enter a mode for reading or altering data at a specified address.

A.2 frob Command

frob Command	Description
peek [addr]	Read 32bit data from specified address
peek16 [addr]	Read 16bit data from specified address
peek8 [addr]	Read 8bit data from specified address
poke [addr] [value]	Write 32bit data to specified address
poke16 [addr] [value]	Write 16bit data to specified address
poke8 [addr] [value]	Write 8bit data to specified address

A.10. tftpd

This command is used to download an image file from a TFTP server and write it to flash memory.

```
Syntax: tftpd <client IP address> <server IP address> <option> [option]...
Explanation: Sets the local IP address to the specified client address, then downloads an image file from the TFTP server at the specified server IP address and writes it to flash memory.
```

A.19 tftpd Syntax

A.3 tftpd Options

Option	Description
--bootloader=filepath	Specify file to be written to the bootloader region in place of filepath.
--kernel=filepath	Specify file to be written to the kernel region in place of filepath.
--userland=filepath	Specify file to be written to the userland region in place of filepath.
--fake	Download the files, but do not actually write the files to flash memory.

A.10.1. tftpd Example

```

hermit> tftpd 192.168.10.10 192.168.10.1 --kernel=linux.bin.gz

Client: 192.168.10.10
Server: 192.168.10.1
Region(kernel): linux.bin.gz

initializing net-device...OK
Filename : linux.bin.gz
.....
.....
.....
Filesize : 1841551

programing: kernel
#####

completed!!
    
```

A.20 tftpd Example

A.11. tftpboot

This command is used to download an image file from a TFTP server, extract it to RAM and then boot it. As opposed to the tftpd command, the file is not written to flash memory. Also, the settings are not saved like they are when using the setbootdevice command.

```

Syntax: tftpboot <client IP address> <server IP address> <option> [option]...
Explanation: Sets the local IP address to the specified client address, then downloads the specified image file from the TFTP server at the specified server IP address, extracts it to RAM and boots it.
    
```

A.21 tftpboot Syntax

The same options as shown in A.3. tftpd Options can be used with this command. When --fake is specified, the file is downloaded but not booted.

A.11.1. tftpboot Example

```

hermit> tftpboot 192.168.10.10 192.168.10.1 --kernel=linux.bin.gz

Client: 192.168.10.10
Server: 192.168.10.1
Region(kernel): linux.bin.gz

initializing net-device...OK
Filename : linux.bin.gz
.....
.....
.....
Filesize : 1841551

Uncompressing kernel:net.....
.....done.
Uncompressing ramdisk.....
.....
.....
.....
.....
.....
.....
.....
.....done.
Linux version 2.6.26-at6 (2.6.26) (atmark@sv-build) (gcc version 4.3.2 (Debian
4.3.2-1.1) ) #6 PREEMPT Wed Mar 10 19:19:13 JST 2010
:
:

```

A.22 tftpboot Example

- ❶ The kernel and userland images are being extracted to RAM.
- ❷ The kernel is booted and kernel boot log displayed.

A.12. boot

This command boots the Linux kernel image from the boot device specified with the setbootdevice command.

```
Syntax: boot
```

A.23 boot Syntax

A.12.1. boot Example

```

hermit> boot
Uncompressing kernel.....done.
Uncompressing ramdisk.....
.....
.....done.
Doing console=ttymxc1
Linux version 2.6.26-at6 (2.6.26) (atmark@sv-build) (gcc version 4.3.2 (Debian
4.3.2-1.1) ) #6 PREEMPT Wed Mar 10 19:19:13 JST 2010
CPU: ARM926EJ-S [41069264] revision 4 (ARMv5TEJ), cr=00053177
Machine: Armadillo-440
Memory policy: ECC disabled, Data cache writeback
CPU0: D VIVT write-back cache
CPU0: I cache: 16384 bytes, associativity 4, 32 byte lines, 128 sets
CPU0: D cache: 16384 bytes, associativity 4, 32 byte lines, 128 sets
Built 1 zonelists in Zone order, mobility grouping on. Total pages: 32512
Kernel command line: console=ttymxc1
MXC IRQ initialized
:
:

```

- ①
- ②
- ③

A.24 boot Example

- ① The kernel and userland images are being extracted to RAM.
- ② Kernel parameters set with the setenv command are displayed. All messages up to this point are from Hermit-At.
- ③ The kernel is booted and kernel boot log displayed.

A.13. Note On Versions

When a bootloader image based on hermit-at v2.0.0 source (loader-armadillo4x0-v2.0.0.bin etc) and a kernel image based on linux-2.6.26-at7 source (linux-a400-1.00.bin.gz etc) are used on Armadillo-440 Rev.C1 or later boards (S/N 100201-2195 or greater), a problem exists where the kernel may not boot. This problem only occurs with the above combination.

On Armadillo-440 Rev.C1 or later boards (S/N 100201-2195 and greater), please use a bootloader image based on hermit-at v2.0.1 or later source (loader-armadillo4x0-v2.0.1.bin or later).^[1]

^[1]On Armadillo-440 Rev.C1 and later boards, loader-armadillo4x0-v2.0.1.bin or a later version is written to flash memory in the default factory state.

1.0.0	03/12/2010	<ul style="list-style-type: none"> • Initial Release
1.1.0	04/28/2010	<ul style="list-style-type: none"> • Added information on Armadillo-420 to 1. Preface, 2. System Overview, 3. Before Getting Started, 5. Rewriting Flash Memory, 6. Building and 7. Kernel and Userland Placement. • Added notes on function change of pins 3,4 on CON14 to 2.3. Basic Specifications of Armadillo-440 LCD Model. • Corrected directory name in 6.10. Hermit-At Source Archive Extraction. • Fixed URLs in 7.1. Kernel Image Download URLs, 7.5. Kernel Image Placement, 7.2. Debian Archive Download URLs, 7.6. Root Filesystem Creation With Debian Archives, 7.3. Atmark-Dist Image Download URL and 7.6. Root Filesystem Creation With Debian Archives. • Added note about using board revision Rev.C1 to A.13. Note On Versions.
1.2.0	06/08/2010	<ul style="list-style-type: none"> • Made correction to description of CON9 5 functionality in 2.3. Pin Layout of Armadillo-420 Basic Model Expansion Interfaces and 2.5. Pin Layout of Armadillo-440 LCD Model Expansion Interfaces. • Added notes on changing configuration to 6.1.4. Customizing Images. • Added notes on drivers that can be configured in the kernel configuration to 8. Linux Kernel Device Driver Specifications.
1.3.0	08/20/2010	<ul style="list-style-type: none"> • Made corrections to explanation of RS232 connections in 2.2. Basic Specifications of Armadillo-420 Basic Model • Made corrections to 8.2. CAN Transmission Speed Calculation • Added use conditions to 8.2.2. CAN sysfs • Added UART4 to 8.1. Serial Interface Device Files • Updated explanation in 7.2. Loading From Storage • Corrected inconsistent use of various terms • Added explanation on how to use drivers not enabled by default to 8. Linux Kernel Device Driver Specifications • Add notes on how to handle interrupts with GPIO sysfs to 8.9.1. GPIO sysfs

Armadillo-400 Series Software Manual
Version 1.3.0
2010/11/05

Atmark Techno, Inc.

060-0035 AFT Bldg. 6F, N5E2, Chuo-ku, Sapporo TEL 011-207-6550 FAX 011-207-6570
