



OPB SIL00 (v1.02a)

Introduction

SUZAKU でプログラミングを行った際、動作確認に苦労したことはないでしょうか？ SUZAKU I/O シリーズの LED/SW ボードは単色 LED、押しボタンスイッチ、ロータリコードスイッチ、7 セグメント LED、シリアルポートが搭載されています。SUZAKU を使った開発において簡単な動作確認、FPGA や Linux の学習用としてご利用いただけるように設計されております。

OPB-SIL00 を使うことにより LED/SW ボードにスロット機能を簡単に実現できます。

Features

OPB-SIL00 は以下のような特徴を持っています。

- SUZAKU I/O シリーズ LED/SW ボード専用
- 7 セグメント LED の数字をデコードして出力
- 7 セグメント LED をダイナミック点灯
- 押しボタンスイッチ、ロータリコードスイッチの入力を受け取る
- トリガー信号により、単色 LED を順次点灯
- 割り込み

Block Diagram

OPB-SIL00 のブロック図を示します。

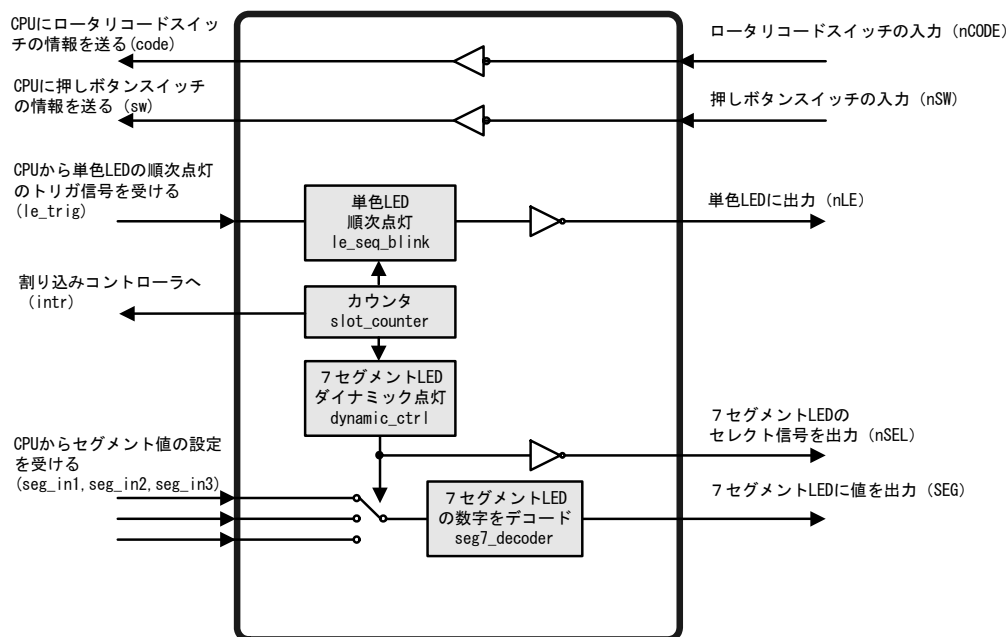


Figure 1 : OPB-SIL00 Block Diagram

Parameters

OPB-SIL00 のパラメータを示します。

Table 1 : Design Parameters

Feature/Description	Parameter Name	Allowable Values	Default Values	VHDL Type
OPB SIL00 Base Address	C_BASEADDR	Valid OPB Address	None	std_logic_vector
OPB SIL00 High Address	C_HIGHADDR	Valid OPB Address	None	std_logic_vector
OPB Address Width	C_OPB_AWIDTH	32	32	integer
OPB Data Width	C_OPB_DWIDTH	32	32	integer

I/O Signals

OPB-SIL00 の I/O Signals を示します。

Table 2 : I/O Signals

Signal Name	I/O	Initial State	Description
OPB_Clk	I		OPB main bus clock
OPB_Rst	I		OPB main bus reset
SI_DBus(0:C_OPB_DWIDTH-1)	I	0	OPB SIL00 Data Bus
SI_errAck	O	0	OPB SIL00 Error Acknowledge
SI_retry	O	0	OPB SIL00 Retry
SI_toutSup	O	0	OPB SIL00 Timeout Suppress
SI_xferAck	O	0	OPB SIL00 Transfer Acknowledge
OPB_ABus(0:C_OPB_AWIDTH-1)	I		OPB Address Bus
OPB_BE(0:[C_OPB_DWIDTH/8]-1)	I		OPB Byte Enable
OPB_DBus(0:OPB_DWIDTH-1)	I		OPB Data Bus
OPB_RNW	I		OPB Read/Not Write
OPB_select	I		OPB Select
OPB_seqAddr	I		OPB Sequential Address
IP2INTC	O	0	OPB SIL00 Interrupt Active high signal
SEG(0:7)	O		7Seg LED Signal Output
nSEL(0:2)	O		7Seg LED Select Signal Output
nLE(0:3)	O		LED Signal Output
nSW(0:2)	I		Push SW Signal Input
nCODE(0:3)	I		Rotary SW Signal Input

Register

OPB-SIL00 の LED、スイッチ制御レジスタを示します。

Table 3 : Register Overview

Register Name	Description	OPB Address	Access
seg_in1	7Seg LED1 Data Register	C_BASEADDR+0x00	Read/Write
seg_in2	7Seg LED2 Data Register	C_BASEADDR+0x01	Read/Write
seg_in3	7Seg LED3 Data Register	C_BASEADDR+0x02	Read/Write
le_trig	LED Trigger Signal Register	C_BASEADDR+0x03	Read/Write
sw	Push SW Data Register	C_BASEADDR+0x04	Read
code	Rotary SW Data Register	C_BASEADDR+0x05	Read

※8bit アクセスのみ対応、16bit 及び 32bit アクセスはできません。

• seg_in1~3(7Seg LED1~3 Data Register)

0~F を 7 セグメント LED に読み書きします。



Figure 2 : 7SegLED Data Register

Table 4 : 7SegLed Data Register Description

Bits	Name	Core Access	Description	Reset Value
0 : 3	Unused	Read/Write	Unused	0x0
4	seg_in1~3(0)	Read/Write	7 セグメント LED のデータを読み書きします。 0~F の 16 進値を設定すると、セグメント LED に 0~F の数字を表示します。	0
5	seg_in1~3(1)	Read/Write		0
6	seg_in1~3(2)	Read/Write		0
7	seg_in1~3(3)	Read/Write		0

• le_trig(LED Trigger Register)

‘0’で消灯、‘1’で単色 LED1~4 が順次点灯します。

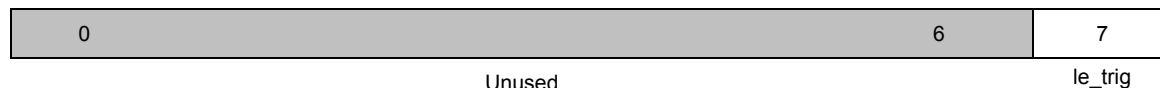


Figure 3 : LED Trigger Register

Table 5 : LED Trigger Register Description

Bits	Name	Core Access	Description	Reset Value
0 : 6	Unused	Read/Write	Unused	000000
7	le_trig	Read/Write	0 : 単色 LED 消灯 1 : 単色 LED 順次点灯	0

• sw(Push SW Data Register)

押しボタンスイッチ(SW1～SW3)の値を受け取ります。

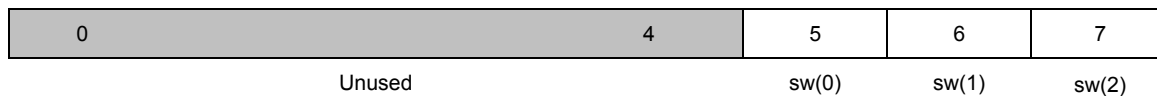


Figure 4 : Push SW Data Register

Table 6 : Push SW Data Register Description

Bits	Name	Core Access	Description	Reset Value
0 : 4	Unused	Read/Write	Unused	00000
5	sw(0)	Read	SW3 の ON/OFF を読み込む 0 : OFF 1 : ON	0
6	sw(1)	Read	SW2 の ON/OFF を読み込む 0 : OFF 1 : ON	0
7	sw(2)	Read	SW1 の ON/OFF を読み込む 0 : OFF 1 : ON	0

• code(Rotary SW Data Register)

ロータリコードスイッチの値を受け取ります。

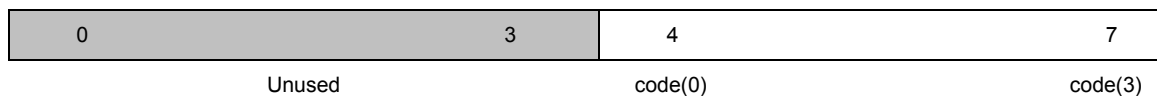


Figure 5 : Rotary SW Data Register

Table 7 : Rotary SW Data Register Description

Bits	Name	Core Access	Description	Reset Value
0 : 3	Unused	Read/Write	Unused	00000
4	code(0)	Read	ロータリコードスイッチの値を 0～F の 16 進値で読み込みます。	0
5	code(1)	Read		0
6	code(2)	Read		0
7	code(3)	Read		0

Interrupt

OPB-SIL00 の割り込みレジスタを示します。

Table 8 : Interrupt Parameters

Register Name	Description	PLB Address	Access
DIER	Device Interrupt Enable Register	C_BASEADDR+0x08	Read/Write
DGIER	Global Interrupt Enable	C_BASEADDR+0x1C	Read/Write
IPISR	IP Interrupt Status Register	C_BASEADDR+0x20	Read/Tpggle on Write
IPIER	IP Interrupt Enable Register	C_BASEADDR+0x28	Read/Write

• DIER(Device Interrupt Enable Register) Offset 0x08

0	27	28	29	30	31
Unused		LVIP	IPIRE	IE(1)	IE(0)

Figure 6 : Device Interrupt Enable Register

Table 9 : Device Interrupt Enable Register Description

Bits	Name	Core Access	Description	Reset Value
0 : 27	Unused	Read	Unused	zeroes
28	LVIP	Read/Write	Level Interrupt Enable 0 : disabled 1 : enabled	0
29	IPIRE	Read/Write	Interrupt Request Enable 0 : disabled 1 : enabled	0
30	IE(1)	Read/Write	Interrupt Enable 1 0 : disabled 1 : enabled	0
31	IE(0)	Read/Write	Interrupt Enable 0 0 : disabled 1 : enabled	0

• DGIER(Device Global Interrupt Enable Register) Offset 0x1C

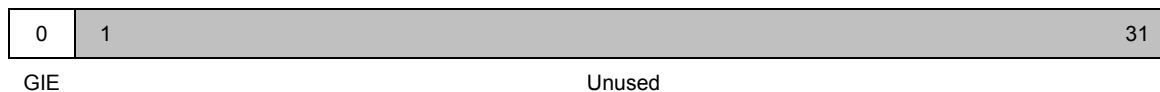


Figure 7 : Device Global Interrupt Enable Register

Table 10 : Device Global Interrupt Enable Register Description

Bits	Name	Core Access	Description	Reset Value
0	GIE	Read/Write	Global Interrupt Enable. 0 : Interrupt disabled; no interrupt possible from this device. 1 : Device Interrupt enabled.	0
1 : 31	Unused	Read	Unused	zeroes

• IPISR(IP Interrupt Status Register) Offset 0x20

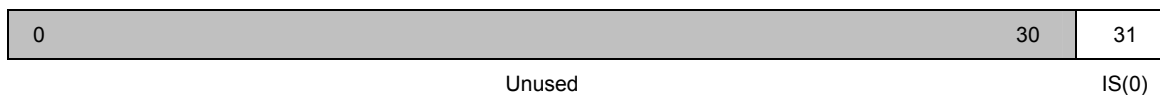


Figure 8 : IP Interrupt Status Register

Table 11 : IP Interrupt Status Register Description

Bits	Name	Core Access	Description	Reset Value
0 : 30	Unused	Read	Unused	zeroes
31	IS(0)	Read/TOW (Toggle on Write) 1 : Write	Interrupt Status 0 : not active. 1 : active	0

• IPIER(IP Interrupt Enable Register) Offset 0x28

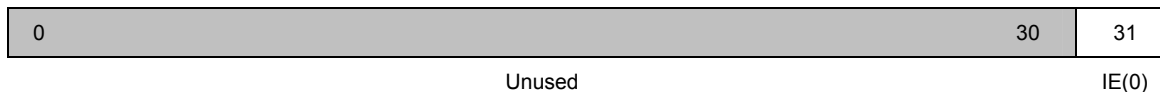


Figure 9 : IP Interrupt Enable Register

Table 12 : IP Interrupt Enable Register Description

Bits	Name	Core Access	Description	Reset Value
0 : 30	Unused	Read	Unused	zeroes
31	IE(0)	Read/Write	Interrupt Enable 0 : disabled 1 : enabled	0

Sample Program

• Slot Machine

OPB-SIL00 によるスロット機能実現のためのサンプルプログラム（C 言語）を示します。

Example 1 : Slot Machine Sample C Source

```
#include <ctype.h>
typedef unsigned char u8;
#define SLOT_BASE_ADDR 0xffffd000

/* slot status */
#define SLOT_READY (0)
#define SLOT_RUN (1)
#define SLOT_STOP (2)

/* switch */
#define CHATTERING_CLEAR_CNT (2)
#define SWITCH_NUM (3)
#define SWITCH(no) (1 << (no))
#define SWITCH_ALL (SWITCH(0) | SWITCH(1) | SWITCH(2))
#define READ_SWITCH (*(volatile u8 *) (SLOT_BASE_ADDR + 0x4))

/* 7seg-led */
#define SEG7_NUM (3)
#define SEG7(no) (1 << (no))
// #define SEG7(no) (1 << (SEG7_NUM - 1 - (no)))
#define SEG7_ALL (SEG7(0) | SEG7(1) | SEG7(2))
#define WRITE_SEG7(no,v) (*(volatile u8 *) (SLOT_BASE_ADDR + (no)) = (v))

/* led */
#define WRITE_LEDS(v) (*(volatile u8 *) (SLOT_BASE_ADDR + 0x3) = (v))

/* rotary switch */
#define READ_RSWITCH (*(volatile u8 *) (SLOT_BASE_ADDR + 0x5))

static u8 clear_switch_chattering(void)
{
    static u8 count[SWITCH_NUM];
    static u8 status = 0;
    u8 reg_sw = READ_SWITCH;
    int i;

    for (i = 0; i < SWITCH_NUM; i++) {
        if (reg_sw & SWITCH(i)) {
            if (count[i] < CHATTERING_CLEAR_CNT)
                count[i]++;
        } else {
            if (count[i] > 0)
                count[i]--;
        }
    }

    if (count[i] == CHATTERING_CLEAR_CNT)
```

```
        status |= SWITCH(i);
    else if (count[i] == 0)
        status &= ~SWITCH(i);
    }

    return status;
}

static void blink_leds(u8 st, u8 value[])
{
    u8 v = 0;
    if (st == SLOT_READY) {
        if (((value[0] & 0xf) == (value[1] & 0xf)) &&
            ((value[0] & 0xf) == (value[2] & 0xf)))
            v = 1;
    }
    WRITE_LEDS(v);
}

static u8 change_rotation_speed(void)
{
    u8 speed;

    switch (READ_RSWITCH) {
    case 0:
        speed = 0x3f;
        break;
    case 1:
        speed = 0x1f;
        break;
    default:
        speed = 0x0f;
        break;
    }

    return speed;
}

static int get_number_pushed_switches(u8 st)
{
    int num = 0;
    int i;

    for (i = 0; i < SWITCH_NUM; i++) {
        if ((st >> i) & 0x1)
            num++;
    }
    return num;
}

int slot(void)
{
    static u8 schedule = 0;
    static u8 sw_status = 0;
    static u8 slot_status = SLOT_READY;
```



```
static u8 seg7_status = 7;    /* 0: unpushed, 1: pushed */
static u8 seg7_val[SEG7_NUM] = {3,2,1};
int i;

schedule++;

sw_status = clear_switch_chattering();

switch (slot_status) {
case SLOT_READY:
    if (get_number_pushed_switches(sw_status) > 1)
        slot_status = SLOT_RUN;
    break;
case SLOT_RUN:
    if (sw_status == 0)
        slot_status = SLOT_STOP;
    break;
case SLOT_STOP:
    if (seg7_status == SEG7_ALL)
        slot_status = SLOT_READY;
    break;
}

if (slot_status == SLOT_STOP) {
    for (i = 0; i < SWITCH_NUM; i++) {
        if (sw_status & SWITCH(i))
            seg7_status |= SEG7(i);
    }
} else if (slot_status == SLOT_RUN) {
    seg7_status = 0;
}

if ((schedule & change_rotation_speed()) == change_rotation_speed()) {
    for (i = 0; i < SEG7_NUM; i++) {
        if ((seg7_status & SEG7(i)) == 0)
            seg7_val[i]++;
    }
}

for (i = 0; i < SEG7_NUM; i++)
    WRITE_SEG7(i, seg7_val[i]);

blink_leds(slot_status, seg7_val);

return 0;
}
```

• Interrupt

OPB-SIL00 で割り込みを利用する際の MHS およびサンプルプログラム (C 言語) を示します。

PowerPC

Example 2 : PowerPC Interrupt MHS File

```
BEGIN ppc405
# 中略
PORT EICC405EXTINPUTIRQ = eicc405extinputirq
END

BEGIN opb_intc
# 中略
PORT Irq = eicc405extinputirq
PORT Intr = sil_intr
END

BEGIN opb_sil00
# 中略
PORT IP2INTC_Irpt = sil_intr
END
```

MicroBlaze

BRAM だけで行う場合は、全セクション用に 64kbyte (0xFFFF0000—0xFFFFFFFF) を用意、または、vectors セクションに 16kbyte (0xFFFF0000—0xFFFF3FFF) + その他セクションに 16kbyte (0xFFFFC000—0xFFFFFFFF) を用意してください。

Example 3 : MicroBlaze Interrupt MHS File

```
BEGIN microblaze
PARAMETER INSTANCE = microblaze_i
# 中略
PORT INTERRUPT = interrupt

BEGIN opb_intc
PARAMETER INSTANCE = system_intc
# 中略
PORT Irq = interrupt
PORT Intr = sil_intr
END

BEGIN opb_sil00
PARAMETER INSTANCE = opb_sil00_0
# 中略
PORT IP2INTC_Irpt = sil_intr
END
```

Example 4 : Interrupt Sample C Source

```

#include <ctype.h>
#include "medium.h"
#include "slot.h"
#include "xparameters.h"
#include "medium.h"
#include "xintc_l.h"
#ifdef __PPC__
#include "xpseudo_asm_gcc.h"
#include "xexception_l.h"
#endif

int sil_interrupt_handler(void * baseaddr_p)
{
    slot();
}

int interrupt_init(void)
{
    /*enable interrupt */
#ifdef __PPC__
    mtevp(0xFFFF0000);
    for(index = XEXC_ID_FIRST; index < XEXC_ID_LAST + 1; index++){
        XExc_RegisterHandler(index, (XExceptionHandler)NullHandler, XNULL);
    }

    XExc_RegisterHandler(XEXC_ID_NON_CRITICAL_INT, (XExceptionHandler)XIntc_DeviceInterruptHandler,
        (void*)XPAR_SYSTEM_INTC_DEVICE_ID);
    XExc_mEnableExceptions(XEXC_NON_CRITICAL);
#else
    microblaze_enable_interrupts();
#endif

    /* Start the interrupt controller */
    XIntc_mMasterEnable(XPAR_SYSTEM_INTC_BASEADDR);
    /* Enable timer interrupt in the interrupt controller */
    XIntc_mEnableIntr(XPAR_SYSTEM_INTC_BASEADDR, XPAR_OPB_SIL00_0_IP2INTC_IRPT_MASK);
    /* start the intr sil */
    /* Enable all interrupt source from user logic.*/
    OPB_SIL00_mWriteReg(XPAR_OPB_SIL00_0_BASEADDR, OPB_SIL00_INTR_IER_OFFSET, 0x00000001);
    /* Enable all possible interrupt sources from device.*/
    OPB_SIL00_mWriteReg(XPAR_OPB_SIL00_0_BASEADDR, OPB_SIL00_INTR_DIER_OFFSET, INTR_TERR_MASK |
INTR_DPTO_MASK | INTR_IPIR_MASK);
    /* Set global interrupt enable.*/
    OPB_SIL00_mWriteReg(XPAR_OPB_SIL00_0_BASEADDR, OPB_SIL00_INTR_DGIER_OFFSET, INTR_GIE_MASK);
}

int interrupt_clean(void)
{
    /* disable interrupt */
#ifdef __PPC__
    XExc_mDisableExceptions(XEXC_NON_CRITICAL);
#else
    microblaze_disable_interrupts();
#endif
}

```

```
/* Disable timer interrupt in the interrupt controller */
XIntc_mDisableIntr(XPAR_SYSTEM_INTC_BASEADDR, XPAR_OPB_SIL00_0_IP2INTC_IRPT_MASK);
/* stop the timer */
OPB_SIL00_mWriteReg(XPAR_OPB_SIL00_0_BASEADDR, OPB_SIL00_INTR_IER_OFFSET, 0);
OPB_SIL00_mWriteReg(XPAR_OPB_SIL00_0_BASEADDR, OPB_SIL00_INTR_DIER_OFFSET, 0);
OPB_SIL00_mWriteReg(XPAR_OPB_SIL00_0_BASEADDR, OPB_SIL00_INTR_DGIER_OFFSET, 0);
}

int main(void)
{
    interrupt_init();
    while(1);
    //interrupt_clean();
    return 0;
}
```

Revision History

Date	Version	Revision
2006/ 8/11	1.0	初版作成
2007/ 3/27	1.1	誤記訂正
2007/10/10	1.2	Interrupt のアドレス修正 + 0x100 PPC の Interrupt Sample C Source の修正 (割り込み Disable 追加)
2008/02/15	1.3	Interrupt のアドレス修正 コアのバージョン変更 v1.01d --> v1.02a