
EAGLE 100 Single Board Computer

CPLD Application Note

April, 2009

Table of Contents

1.0	Introduction	1
1.1	Expansion via Programmable Logic	1
1.2	Processor Interface.....	1
1.3	Electrical Interface.....	1
2.0	Reference Implementations	1
2.1	Monitor	1
2.2	SPI Port Extender	2
2.3	ISA Bus Master	3
3.0	Firmware Updates	3
3.1	Modifying and Rebuilding	3
3.2	Xilinx JTAG	3
3.3	microSD Card	4
4.0	Extended I/O API	5
4.1	XioInit	5
4.2	XioWrite	5
4.3	XioWriteBlock.....	5
4.4	XioRead.....	5
4.5	XioReadBlock.....	6
4.6	IsaInit	6
4.7	IsaWrite	6
4.8	IsaWriteBlock.....	6
4.9	IsaRead	7
4.10	IsaReadBlock.....	7
4.11	LcdInit	7
4.12	LcdCommand	7
4.13	LcdWrite.....	8

Table of Contents

4.14	LcdWriteByte	8
4.15	LcdWriteStr	8
4.16	LcdGoTo	8
4.17	LcdWriteAt	9
4.18	LcdClear	9
4.19	LcdDisplay	9
4.20	LcdCursor	9
4.21	LcdBacklight	10

1.0 Introduction

1.1 Expansion via Programmable Logic

The Eagle 100 includes a Xilinx XC9572XL Complex Programmable Logic Device (CPLD) that can augment the microcontroller capabilities using high speed logic. In this scenario, the CPLD can perform repetitive dedicated tasks and allow the processor to concentrate in more supervisory functions. Examples include event monitors, pwm generators, motor controllers and expansion bus interfaces. Micromint provides several reference implementations with source code that can be adapted to specific application requirements.

1.2 Processor Interface

The ARM microcontroller uses 5 pins for communication with the CPLD as shown in Table 1-1.

LM3S6918		XC9572XL		Direction (CPLD)	Description
Pin	Function	Pin	Function		
28	PA2/SSI0CLK		SPI0CLK	In	SSI Clock
29	PA3/SSI0FSS		IO_CS	In	I/O Chip Select
30	PA4/SSI0RX		SPI0RX	Out	SSI Master Receive (miso)
31	PA5/SSI0TX		SPI0TX	In	SSI Master Transmit (mosi)
35	PA7/I2C1SDA		IO_INTR	Out	I/O Interrupt

Table 1-1: Processor to CPLD interface

The reference implementation uses an SSI (Synchronous Serial Interface) port at 4 MHz. The serial interface uses 9-bit frames including a 1-bit command flag and an 8-bit data word. (SSI). Up to 63 CPLD I/O pins in connectors J12 and J13 are available for applications.

1.3 Electrical Interface

The XC9572XL is a 3.3V device with 5V tolerant I/O. Please refer to the following Xilinx documents for guidelines on thresholds, termination and impedance.

XC9572XL 3.3V High-Performance CPLD Data Sheet

http://www.xilinx.com/support/documentation/data_sheets/ds057.pdf

Xilinx CPLD I/O User Guide

http://www.xilinx.com/support/documentation/user_guides/ug445.pdf

XC9500XL CPLD Documentation

<http://www.xilinx.com/support/documentation/xc9500xl.htm>

2.0 Reference Implementations

2.1 Monitor

This monitor application uses the CPLD to watch events and trigger an alarm when a condition is reached. By delegating the monitoring task to the CPLD, the processor can concentrate on other tasks. If properly implemented, this distributed processing can lead to very efficient applications.

Introduction

The event being monitored is the number of pulses on pin 30 (PXC7) of the extended I/O connector J12. It uses two CPLD registers: the current count (RXA) and the count limit (RXB). Once the count limit is reached, the interrupt line (IO_INTR) is asserted. A status of the count changes is printed on serial port COM1. The count changes are monitored every 100 ms. The alarm event is handled asynchronously using an interrupt service routine (ISR).

To run this application, download the *monitor.svf* firmware to the CPLD using the procedure in section 3 of the manual and the *cpld_monitor.bin* firmware to the microcontroller using a JTAG or the Luminary Flash Programmer.

2.2 SPI Port Extender

The SPI port extender application expands the available I/Os to the processor using both the extended I/O connector J12 and the PC/104 connector J13. The PC/104 signals are reclassified according to table 2-1.

J13 Pin#	PC/104 Firmware	Description	XIO Firmware	Description
A2	SD7	ISA Data Port	XF7	Extended I/O Port XF
A3	SD6	ISA Data Port	XF6	Extended I/O Port XF
A4	SD5	ISA Data Port	XF5	Extended I/O Port XF
A5	SD4	ISA Data Port	XF4	Extended I/O Port XF
A6	SD3	ISA Data Port	XF3	Extended I/O Port XF
A7	SD2	ISA Data Port	XF2	Extended I/O Port XF
A8	SD1	ISA Data Port	XF1	Extended I/O Port XF
A9	SD0	ISA Data Port	XF0	Extended I/O Port XF
A11	AEN	ISA Address Enable	XH4	Extended I/O Port XH
A22	SA9	ISA Address Port	XH1	Extended I/O Port XH
A23	SA8	ISA Address Port	XH0	Extended I/O Port XH
A24	SA7	ISA Address Port	XG7	Extended I/O Port XG
A25	SA6	ISA Address Port	XG6	Extended I/O Port XG
A26	SA5	ISA Address Port	XG5	Extended I/O Port XG
A27	SA4	ISA Address Port	XG4	Extended I/O Port XG
A28	SA3	ISA Address Port	XG3	Extended I/O Port XG
A29	SA2	ISA Address Port	XG2	Extended I/O Port XG
A30	SA1	ISA Address Port	XG1	Extended I/O Port XG
A31	SA0	ISA Address Port	XG0	Extended I/O Port XG
B2	PC104_RESET	ISA Reset	XH7	Extended I/O Port XH
B13	IOW	ISA I/O Write	XH6	Extended I/O Port XH
B14	IOR	ISA I/O Read	XH5	Extended I/O Port XH
B23	IRQ5*	Port F bit 0	IRQ5*	Port F bit 0
B24	IRQ4*	Port D bit 1	IRQ4*	Port D bit 1
B25	IRQ3*	Port D bit 0	IRQ3*	Port D bit 0
B28	ISA_BALE	ISA Buffered Address Latch Enable	XH3	Extended I/O Port XH
B30	ISA_OSC	ISA Oscillator	XH2	Extended I/O Port XH

(*) Controlled by MCU, not CPLD firmware.

Table 2-1: PC/104 connector pin out for Extended I/O

For applications that do not require PC/104 expansion, this maximizes the number of available I/O on the board. It allows up to 63 I/Os as listed in Table 2-2. A

Port	Connector	Width	Direction	Notes
XA	J12	8	Input / Output	
XB	J12	8	Input / Output	
XC	J12	8	Input	
XD	J10	8	Input / Output	GPIO or LCD data
XE	J10	4	Input / Output	GPIO or LCD control
XF	J13	8	Input	
XG	J13	8	Input	
XH	J13	8	Input	
XI	J13	3	Input	

Table 2-2: Extended I/Os available (xtender.svf)

The application writes a four line message to the LCD twice a second, scrolling the text on every instance. It also displays a marching pattern in port XA with its reverse in port XB. The state of port XC is read and printed on serial port COM1.

To run this application, download the *xtender.svf* firmware to the CPLD using the procedure in section 3 of the manual and the *cpld_xio.bin* firmware to the microcontroller using a JTAG or the Luminary Flash Programmer.

2.3 ISA Bus Master

The ISA bus master application allows expansion using PC/104 standard I/O cards on connector J13. The extended I/O connector is still available with the restrictions on table 2-3.

Port	Connector	Width	Direction
XA	J12	8	Input
XB	J12	8	Input
XC	J12	8	Input
XD	J10	8	Input
XE	J10	4	Input

Table 2-3: Extended I/Os available (pc104.svf)

The application writes to an SSD display at address 0x0106 on a PC/104 I/O card. It also copies the state of a DIP switch at address 0x0107 on the card to port XA on the extended I/O connector and an SSD display on port 0x0106. A status is printed twice per second on serial port COM1

To run this application, copy the *pc104.svf* firmware to the CPLD using the procedure in section 3 of the manual and the *cpld_isa.bin* firmware to the microcontroller using a JTAG or the Luminary Flash Programmer.

3.0 Firmware Updates

3.1 Modifying and Rebuilding

To modify and rebuild the reference applications, the Xilinx ISE 10.1 tools should be used. Three projects are included with the source code for each of the reference implementations: monitor.ise, xtender.ise and pc104.ise. The project allows you to build the module after your changes to create a programming file in JED format.

3.2 Xilinx JTAG

The Eagle 100 includes a 6x1 JTAG header for CPLD programming combining JP1 and JP2 that is compatible with Xilinx tools. The proper connector orientation will have VDD and ground on JP2 pins 2 and 1 respectively. Other

compatible JTAGs can be used, such as the Digilent JTAG-USB Cable with its ExPort tool. Note that some CPLD programming tools requires a file in the SVF format. You can use the Xilinx iMPACT utility to convert the JED programming file to SVF.

3.3 microSD Card

To allow customers to update their CPLD firmware without a Xilinx JTAG, Micromint includes a *cpld_update* application that allows updates to the programmable logic using the microSD card. Follow the procedure listed below to use this application.

1. Copy the updated firmware in XSVF format to the microSD card and insert it on the Eagle 100 SBC.
2. Insert four jumpers on JP1 to connect the ARM SSI port to the CPLD JTAG port.
3. Connect a serial terminal or terminal emulator to port COM1 of the Eagle 100.
4. Load the *cpld_update.bin* firmware on the board using a JTAG or the Luminary Flash Programmer. Make sure you have a file with the current firmware in use to restore it afterwards.
5. Reset the board and use the “updatepl” command to update the CPLD as per the example in Figure 3-1.
6. Remove the jumpers on JP1, restore the previous firmware in use and reset the board.

The source code of the *cld_update* utility is included in the DriverLib directory to allow customization for special requirements.

```
COM8 - PuTTY
Welcome to the Eagle SBC
Type 'help' for available commands.

/> help

Available commands for the Eagle SBC
-----
help      : Display list of commands
h         : alias for help
?         : alias for help
ls        : Display list of files
cat       : Show contents of a text file
updatepl  : Updates the CPLD programmable logic

/> updatepl test02~1.xsv
SUCCESS - Completed XSVF execution.
```

Figure 3-1: Command line for *cpld_update*

4.0 Extended I/O API

4.1 XioInit

Initializes SSI port.

Prototype:

```
void XioInit(void);
```

Description:

Setups SSI0 port for transmission to CPLD using 9-bit frames. Must be executed before any extended I/O functions.

4.2 XioWrite

Write one byte to an extended I/O port

Prototype:

```
unsigned long XioWrite(unsigned long port, unsigned long data);
```

Description:

The default CPLD firmware implements ten 8-bit I/O ports XIO_PXA to XIO_PXJ and four internal 8-bit registers XIO_RXA to XIO_RXD. This function allows you to write data to any of those 14 locations.

4.3 XioWriteBlock

Write a block of data to an extended I/O port

Prototype:

```
unsigned long XioWriteBlock(unsigned long port, unsigned long* ptr, unsigned long count);
```

Description:

This function allows you to write extended I/O data in blocks with better performance than byte per byte.

4.4 XioRead

Read a byte from an extended I/O port

Prototype:

```
unsigned long XioRead(unsigned long port);
```

Description:

Allows you to read any of the extended I/O ports (XIO_PXA to XIO_PXJ) or internal registers (XIO_RXA to XIO_RXD).

4.5 XioReadBlock

Read a block of data into a buffer

Prototype:

unsigned long XioReadBlock(unsigned long port, unsigned long* ptr, unsigned long count);

Description:

This function allows you to read extended I/O data in blocks with better performance than byte per byte.

4.6 IsaInit

Initialize PC/104 controller

Prototype:

void IsaInit(void);

Description:

Initializes registers to allow the board to be a bus master over the PC/104 bus. Must be executed before other PC/104 functions.

4.7 IsaWrite

Write data to card at a specified I/O address

Prototype:

unsigned long IsaWrite(unsigned long addr, unsigned long data);

Description:

Writes a byte of data to the expansion card at the specified address.

4.8 IsaWriteBlock

Write block of data to card at a specified I/O address

Prototype:

unsigned long IsaWriteBlock(unsigned long addr, unsigned long* ptr, unsigned long count);

Description:

This function allows you to write data to PC/104 cards in blocks with better performance than byte per byte.

4.9 IsaRead

Read data from card at a specified I/O address

Prototype:

```
unsigned long IsaRead(unsigned long addr);
```

Description:

Reads a byte of data from the expansion card at the specified address.

4.10 IsaReadBlock

Read block of data from card at a specified I/O address

Prototype:

```
unsigned long IsaReadBlock(unsigned long addr, unsigned long* ptr, unsigned long count);
```

Description:

This function allows you to read from PC/104 cards in blocks with better performance than byte per byte.

4.11 LcdInit

Initialize HD44780 LCD Controller

Prototype:

```
void LcdInit(void);
```

Description:

Initializes the LCD registers. Must be called before any other LCD function.

4.12 LcdCommand

Write command byte to LCD

Prototype:

```
unsigned long LcdCommand(unsigned long data);
```

Description:

Writes a command to the LCD. Can be used to send commands that are not supported by the current API.

4.13 LcdWrite

Write data byte to LCD

Prototype:

unsigned long LcdWrite(unsigned long data);

Description:

Writes a data byte to the current position in the LCD display.

4.14 LcdWriteByte

Write command byte to LCD

Prototype:

unsigned long LcdWriteByte(unsigned long data, unsigned long ctl);

Description:

Write data to the current position in the LCD display using specific control flags. This is a low level function that is normally not used from applications.

4.15 LcdWriteStr

Write string to LCD

Prototype:

unsigned long LcdWriteStr(char* str);

Description:

Writes a null terminated string to the current position in the LCD display.

4.16 LcdGoTo

Position LCD cursor assuming 20x4 display

Prototype:

unsigned long LcdGoTo(unsigned long col, unsigned long row);

Description:

Moves the LCD cursor to the desired column and row.

4.17 LcdWriteAt

Write string to LCD at a specified coordinate

Prototype:

unsigned long LcdWriteAt(unsigned long col, unsigned long row, char* str)

Description:

Moves the LCD cursor to the desired column and row to write a null terminated string.

4.18 LcdClear

Clear LCD display

Prototype:

unsigned long LcdClear(void);

Description:

Clears the contents of the LCD display.

4.19 LcdDisplay

Turn display on or off

Prototype:

unsigned long LcdDisplay(unsigned long val);

Description:

Makes the display data not visible. Data can still be written to the LCD display and it will be visible when the display is turned on.

4.20 LcdCursor

Turn cursor on or off

Prototype:

unsigned long LcdCursor(unsigned long val);

Description:

Makes a blinking cursor visible at the current display position.

4.21 LcdBacklight

Turn backlight on or off

Prototype:

```
unsigned long LcdBacklight(unsigned long val);
```

Description:

Turning backlight to improve visibility in dim environments. This will increase power requirements.