EUS100LX Industrial Control Unit User Guide

Rev. 1.01 – June 2005

DSP&FPGA
Custom Design

1.1 Key features

The EUS100LX is a low-cost standalone platform that enables users to evaluate and develop applications for the ETRAX100LX processor and Spartan3 FPGA. Schematics, logic equations and application notes are available to ease firmware development and reduce time to market.

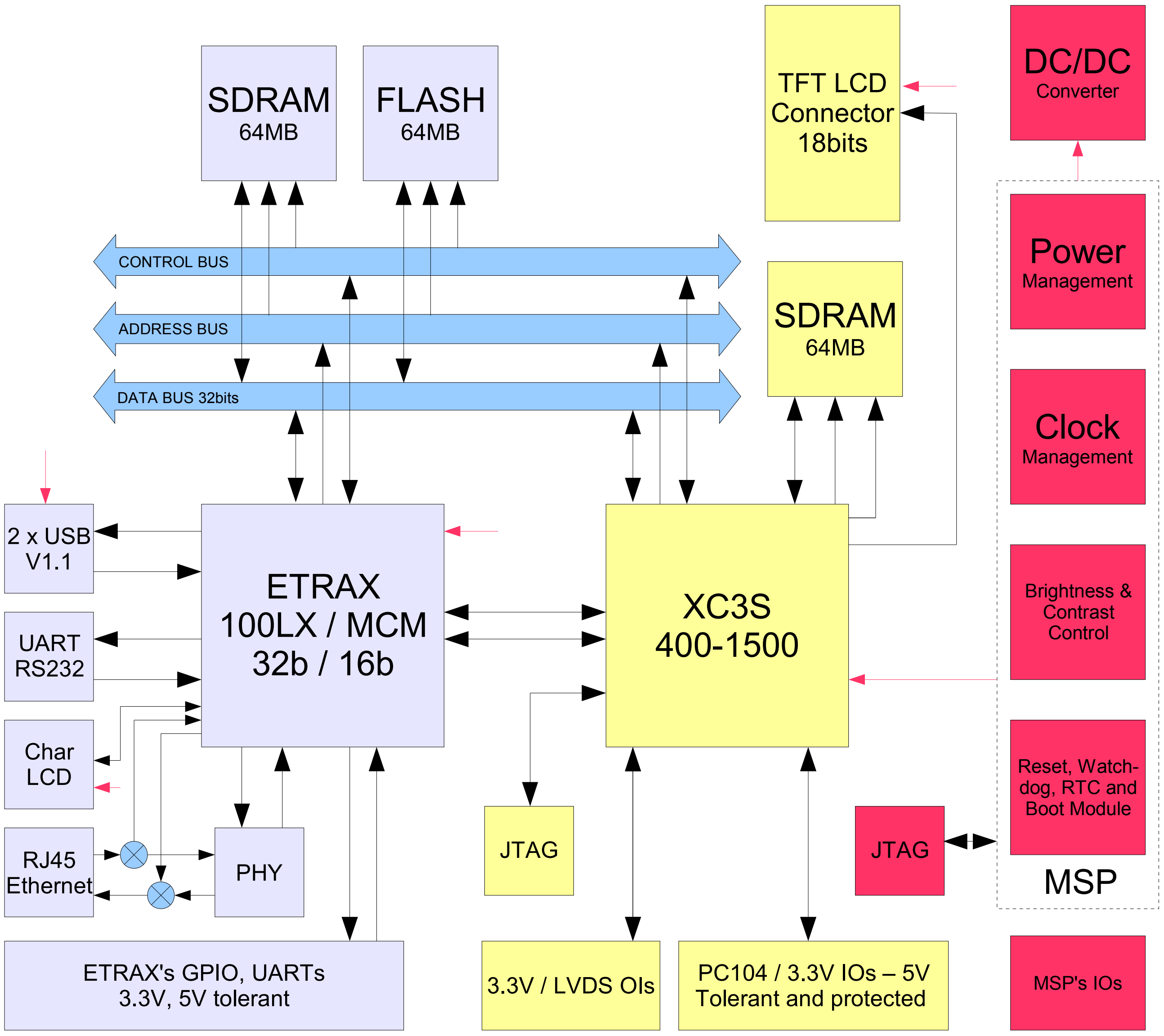


Figure 1: Block Diagram EUS100LX

The EUS100LX comes with a full complement of on board devices that suit a wide variety of application environments. Key features include:

- ETRAX 100LX or ETRAX 100LX MCM4+16 processor
- 32MB SDRAM
- 8MB Flash memory
- FPGA Spartan 3 – XC3S400/1500, connected to the system bus and IRQ
- 16MB independent SDRAM connected to the FPGA
- MSP430 microcontroller for power management & watchdog

- PC104 format compliance
- 10/100Mb Ethernet port
- 2 x USB 1.1 port
- Up to four RS232 from ETRAX & MSP
- Four user LEDs
- Ethernet cable provides direct connection to the host system for initial configuration
- On-board JTAG connector – Xilinx parallel cable IV
- Character LCD support with digital contrast control
- 14 LVDS pairs or 28 LVTTL IOs
- 90 isolated FPGA's IOs, 5V tolerant. Used for PC104 or user configurable
- Shared IOs from ETRAX, MSP and FPGA (USB, Ethernet, Serial ports)
- Expansion connectors for daughter card use
- Single 5V power supply @ 500mA
- Low power modes are supported – less then 3µA
- Utility for FPGA boot & communication
- System provides update through FTP
- Board schematics and Linux source available from DSP&FPGA website
- All source code available under GPL or OHGPL

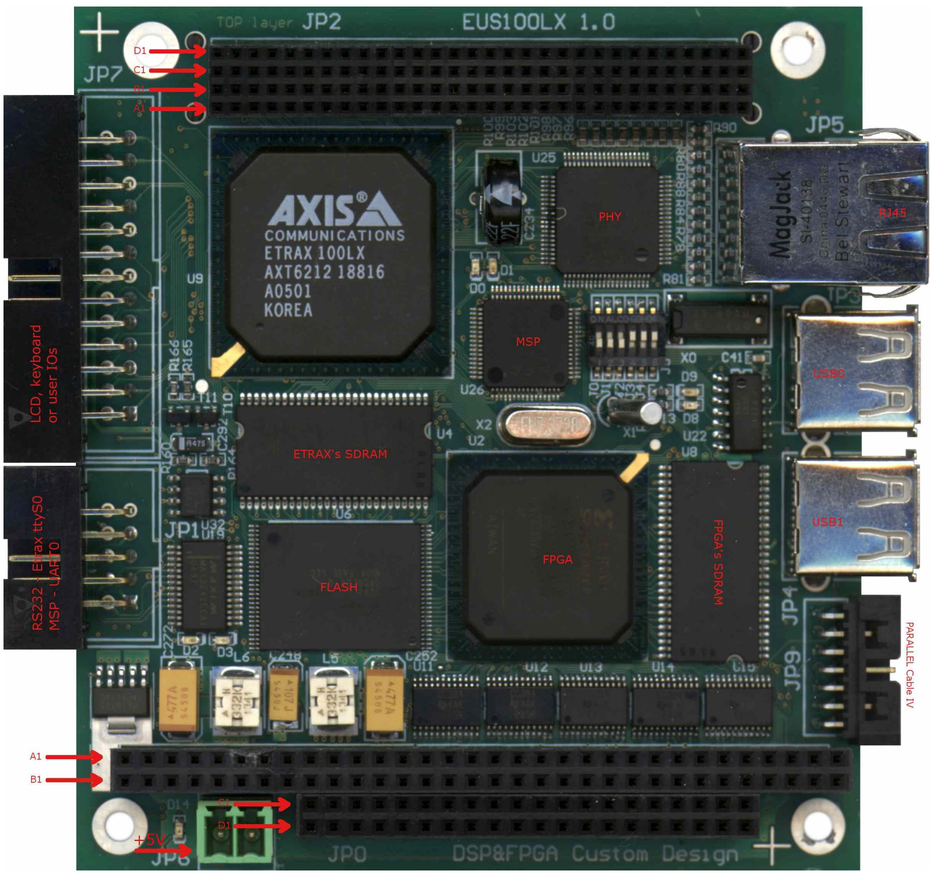


Figure 2: EUS100LX Board - Top Side

1.2 Functional Overview

The processor ETRAX100LX on the EUS100LX board interfaces to on-board peripherals through the 32-bit wide bus. The SDRAM, Flash and FPGA are each connected to one of the buses.
Input and output functions are implemented in the FPGA which resides between the processor and the I/O connector.
A microcontroller MSP430 is used to configure the board by reading and writing to its internal registers. Furthermore, the MSP controls resets, power management, power-up sequencing and watchdog capability.
An 5V external power supply is used to power the board. On-board switching voltage regulators provide the 1.2V FPGA core voltage, 3.3V FPGA I/O supplies and ETRAX100LX processor voltage. The board also has an LDO regulator which provides +2.5V VCCaux and LVDS FPGA voltage. The board is held in reset until these supplies are within operating specifications.
The EUS100LX includes 4 LEDs which can be used to provide the user with interactive feedback. Two LEDs are accessing from the FPGA and two ones from the MSP.

1.3 Memory Map

The ETRAX 100LX processor has a large byte addressable address space. Program code and data can be placed anywhere in the unified address space. The memory map shows the address space of the EUS board with specific details of how each region is used.

Table 1: EUS100LX Memory Map

Name & Address Range	**Size**	**Description**
CSE0 – 0x00000000 – 0x03FFFFFF	64MB	Flash memory 0
CSE1 – 0x04000000 – 0x07FFFFFF	64MB	Flash memory 1 (not used)
CSR0 – 0x08000000 – 0x0BFFFFFF	64MB	FPGA boot space
CSR1 – 0x0C000000 – 0x0FFFFFFF	64MB	LCD Controller / ExtDMA0 (FPGA)
CSP0 – 0x10000000 – 0x13FFFFFF	64MB	FPGA Registers and Statuses (FPGA)
CSP4 – 0x20000000 – 0x23FFFFFF	64MB	Reserved
CSDx – 0x40000000 – 0x7FFFFFFF	1G	SDRAM

Note: Base address + 0x80000000 to bypass the cache

1.4 Configuration Switch Settings

The EUS100LX has six 2 position configuration switches that allow users to control the operational state of the board. The configuration switches are labeled J0 to J5. Switches J0-1 enable MSP430 programming. Switch J2 enables supply for back-up capacitor. J3 and J4 configure the boot mode that will be used when the ETRAX starts executing. Switch J5 enables watchdog capability.

Mode	**J0**	**J1**	**J2**	**J3**	**J4**	**J5**
MSP430 programming	On	On	Off	X	X	Off
Normal operation (default)	**Off**	**Off**	**On/Off**	**On**	**On**	**Off**
External watchdog timer	Off	Off	On/Off	X	X	On

Boot method	J3	J4	Description
Parallel	Off	Off	Isn't supported
Network	Off	On	Receive an Ethernet packets 100/10Mb
Serial	On	Off	Serial port 0 - 9600bps, 8 bits, no parity
Normal	On	On	Execution starts at address 0x80000002

1.5 Power Supply

The EUS100LX operates from a single +5V ±5% external power supply connected to the main power input JP6. A power consumption is about 500mA.

1.6 Programming the FPGA

The FPGA on the EUS100LX is programmed via the ETRAX processor. The Xilinx XC3S400-1500 FPGA supports an 8-bit wide SelectMAP programming interface and the processor uses this parallel interface to program the on board FPGA. In standard configuration the FPGA contents are stored in the Flash -> /mnt/flash/*.mcs.

1.7 FPGA Examples and templates

The board is provided with source codes. The source codes are placed in a FPGA_tmps directory and divided into two sections – the common files (src) and the individual examples. Source directory contains constrain file which defines a pin placement and IO standard definitions and some vhdl files.

Table 2: VHDL Examples

Example dir	Name	Description
Register	register.vhd	Generate three 64kB address windows in CSP0. The first space contains three 32 bit wide registers which are accessible from processor by reading and writing. Second address space returns 32 bit wide system timer value which is incremented every 20ns. The last space returns constant value (0x01234567).
Distram	distram.vhd	Generate three 64kB address windows in CSP0. The first space contains three 32 bit wide registers which are accessible from processor by reading and writing. Second address space returns 32 bit wide system timer value which is incremented every 20ns. The last address space is used for 32bits wide distributed RAM (32 x RAM64X1S). The RAM is accessed by reading and writing to the CSP0 + 0x20000 address space. Provided example supports only 32 bit wide accesses. A LEDx is accessed by reading and writing to the FPGA register (reg_0).
Distram_be	distram_be.vhd	Generate three 64kB address windows in CSP0. The first space contains three 32 bit wide registers which are accessible from processor by reading and writing. Second address space returns 32 bit wide system timer value which is incremented every 20ns. The last address space is used for 32bits wide distributed RAM (32 x RAM64X1S). The RAM is accessed by reading and writing to the CSP0 + 0x20000 address space. Provided example supports 8/16/32 bit wide accesses (aligned/unaligned). A LEDx is accessed by reading and writing to the FPGA register (reg_0).
Bram	Bram.vhd	Generate three 64kB address windows in CSP0. The first space contains three 32 bit wide registers which are accessible from processor by reading and writing. Second address

Example dir	Name	Description
		space returns 32 bit wide system timer value which is incremented every 20ns. The last address space is used for block SelectRAM (RAMB16_S18_S18). The RAM is dual port 18 bit wide and accessed by reading and writing to the CSP0 + 0x20000 address space. Provided example supports only 32 bit wide accesses. A LEDx is accessed by reading and writing to the FPGA register (reg_0).
Ios	Ios.vhd	Generate three 64kB address windows in CSP0. The first space contains three 32 bit wide registers which are accessible from processor by reading and writing. Second address space returns 32 bit wide system timer value which is incremented every 20ns. The last address space is used for reading 8 bit wide data form the input ports X(15:8). An output port X(7:0) is accessed by writing to the FPGA register (reg_0(7:0)). A LEDx shows negated value of input port X(8).
picoblaze	Picoblaze.vhd	Generate three 64kB address windows in CSP0. The first space contains three 32 bit wide registers which are accessible from processor by reading and writing. Reg_0(0) is used as a picoblaze reset (active high). Reg_1(7:0) is uses as an input port for piciblaze processor. Reg_1(8) is used as an interrupt signal to picoblaze engine. The interrupt is edge sensitive and reacts to a rising edge. Second address space returns 32 bit wide system timer value which is incremented every 20ns. The last address space is used for writing 18 bit wide data to picoblaze program memory. A LEDx shows picoblaze output port_4_0 status.

Generating FPGA bitstream

File which describes the project/s is file with an extension *ise (register.ise).*This file can be opened by ISE software provided by Xilinx – www.xilinx.com. In the project window are placed project files which contain source files with an extension *vhd* or *v* and user constrain file with extension *ucf.* User can add or modify these files.

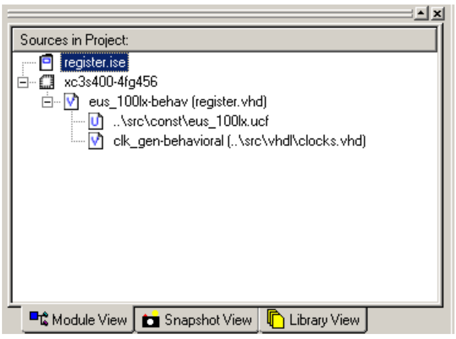

Sources in Project:
register.ise
xc3s400-4fg456
eus_100lx-behav (register.vhd)
..\src\const\eus_100lx.ucf
clk_gen-behavioral (..\src\vhdl\clocks.vhd)

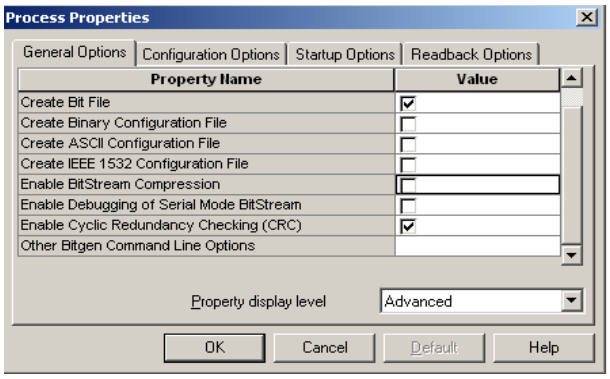
Figure 3: Project windows

For proper operation a user must have generate an appropriate configuration bitstream. The bitstream is generated by promgen tool or simply by clicking to *Generate PROM, ACE, or JTAG file.* The process generates files with extensions *bit* and *mcs*. The *bit* file can be downloaded to the FPGA by JTAG parallel cable IV through a connector JP9. The *mcs (intel hex)* can be downloaded by DSP&FPGA tools from an on-board flash or FTP client.

Recommended settings for bitgen/promgen are:

- *unused iob pins:* ***Float (other setting can cause a crash of the system)***
- *drive DONE pin: High (recommended not mandatory)*
- *fileFormat -value "mcs"*
- *swapBit -value "FALSE"*
- *fillValue -value "FF" or others*
- *attr dir -value "UP"*

These settings are held in files with extension *ipf.*



Property Name	Value
Create Bit File	✓
Create Binary Configuration File	☐
Create ASCII Configuration File	☐
Create IEEE 1532 Configuration File	☐
Enable BitStream Compression	☐
Enable Debugging of Serial Mode BitStream	☐
Enable Cyclic Redundancy Checking (CRC)	✓
Other Bitgen Command Line Options	

Property display level: Advanced

Figure 4: Bitgen/promgen settings

Tools for FPGA

There are two basic tools handling FPGA. One for downloading the bitstream into a fabric and one for basic communication with structures created inside the FPGA.

Downloading tool is called *boot*

boot [options] [file.mcs]

options:

-d dev	device (default is /dev/virtex0)
-h	print help
-p b	don't boot, set only prog value to 'b' (1 or 0)
-v	verbose mode (informs user about programming status, errors, etc)

Communication tool is called *bus*

bus [opts]

-a x	set bus address
-b n	set bus access width, one of 1, 2, 4 (default is 4)
-c n	how many values to process
-f s	copy data FROM hexa text file to memory
-h	print help
-v	verbose operation
-w x	write value

The functions of the tools mentioned above will be shown at the examples. First of all we have to boot the FPGA. This can be done by a boot tool or directly through FTP.

```
boot /mnt/flash/picoblaze.mcs -v
```

after execution: you see something like this:

```
boot: open design file /mnt/flash/picoblaze.mcs
boot: decoded 212392 bytes
boot: open device
boot: write configuration stream
boot: acknowledge startup
boot: success
```

or through FPT server

```
ftp <ip.address>
login
put picoblaze.mcs fpga
```

BTW: This process can be done by Total Commander or other FTP client.

By the *bus* tool you can read or write to the various address spaces. The FPGA examples are mapped to the CSP0. The CSP0 space starts at address 0x90000000 (if the cache is bypassed).

bus -a 0x90000000 -c 3 (this command reads data from address 0x90000000, 0x90000004 and 0x90000008) after execution might be observed:

```
00000001
22222222
33333333
```

Now you can write new value to the register 0:
bus -a 0x90000000 -w 0x12345678
then you can read it back
bus -a 0x90000000 and see
12345678

At address 0x90010000 you can read system timer ticks by:
bus -a 0x90010000 alternatively bus -a 0x90010000 -c 10

The last address space hides picoblaze program memory. One port (A) is used by the picoblaze instruction side and the second port (B) is used by Etrax. This configuration enables a user change content of the instruction memory and downloading new programs. For instance user can write new program by typing *bus -a 0x90020000 -f <program name.hex>*. Before writing you have to reset the processor.

The program sources are available at files:
basicpb.fmt – simple example with blinking LED (fixed delay)
basicpbi.fmt – simple example with blinking LED (delay is read from an input port)
int.fmt – example with blinking LED (delay is setting in the interrupt routine)

This programs must be compiled with Kcpsm3.exe compiler. The most important is a hex output which is writable to the instruction BRAM by the *bus* utility. If you would build your own programs and vhdl codes – don't forget add modified forms <ROM_form.vhd, ROM_form.v> to your compiler directory.

Table 3: PicoBlaze program address space

Address	Subentry	Description
0x90000000	reg_0(0)	PicoBlaze reset (active high)
0x90000004	reg_1(7:0)	PicoBlaze input port 0
0x90000004	reg_1(8)	PicoBlaze interrupt
0x90000008	reg_2(31 downto 0)	Register
0x90010000	sys_cnt	System timer
0x90020000	BRAM	PicoBlaze instruction memory