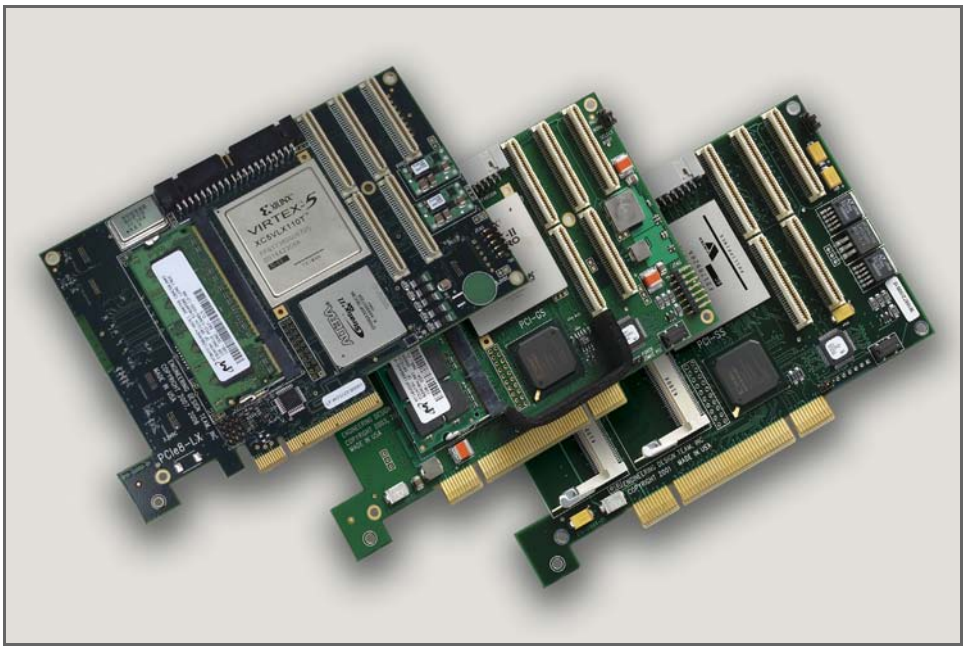
edt
a HEICO company

User's Guide

PCI / PCIe Main Board

PCIe8 LX, PCI GS, or PCI SS



For PCI or PCI Express systems

Doc. 008-02785-02b
Rev. 2011 March 14

Engineering Design Team (EDT), Inc.
3423 NE John Olsen Ave
Hillsboro, OR 97124
p 503-690-1234 / 800-435-4320
f 503-690-1243
www.edt.com

Terms of Use Agreement

Definitions. This agreement, between Engineering Design Team, Inc. (“Seller”) and the user or distributor (“Buyer”), covers the use and distribution of the following items provided by Seller: a) the binary and all provided source code for any and all device drivers, software libraries, utilities, and example applications (collectively, “Software”); b) the binary and all provided source code for any and all configurable or programmable devices (collectively, “Firmware”); and c) the computer boards and all other physical components (collectively, “Hardware”). Software, Firmware, and Hardware are collectively referred to as “Products.” This agreement also covers Seller’s published Limited Warranty (“Warranty”) and all other published manuals and product information in physical, electronic, or any other form (“Documentation”).

License. Seller grants Buyer the right to use or distribute Seller’s Software and Firmware Products solely to enable Seller’s Hardware Products. Seller’s Software and Firmware must be used on the same computer as Seller’s Hardware. Seller’s Products and Documentation are furnished under, and may be used only in accordance with, the terms of this agreement. By using or distributing Seller’s Products and Documentation, Buyer agrees to the terms of this agreement, as well as any additional agreements (such as a nondisclosure agreement) between Buyer and Seller.

Export Restrictions. Buyer will not permit Seller’s Software, Firmware, or Hardware to be sent to, or used in, any other country except in compliance with applicable U.S. laws and regulations. For clarification or advice on such laws and regulations, Buyer should contact: U.S. Department of Commerce, Export Division, Washington, D.C., 20230, U.S.A.

Limitation of Rights. Seller grants Buyer a royalty-free right to modify, reproduce, and distribute executable files using the Seller’s Software and Firmware, provided that: a) the source code and executable files will be used only with Seller’s Hardware; b) Buyer agrees to indemnify, hold harmless, and defend Seller from and against any claims or lawsuits, including attorneys’ fees, that arise or result from the use or distribution of Buyer’s products containing Seller’s Products. Seller’s Hardware may not be copied or recreated in any form or by any means without Seller’s express written consent.

No Liability for Consequential Damages. In no event will Seller, its directors, officers, employees, or agents be liable to Buyer for any consequential, incidental, or indirect damages (including damages for business interruptions, loss of business profits or information, and the like) arising out of the use or inability to use the Products, even if Seller has been advised of the possibility of such damages. Because some jurisdictions do not allow the exclusion or limitation of liability for consequential or incidental damages, the above limitations may not apply to Buyer. Seller’s liability to Buyer for actual damages for any cause whatsoever, and regardless of the form of the action (whether in contract, product liability, tort including negligence, or otherwise) will be limited to fifty U.S. dollars ($50.00).

Limited Hardware Warranty. Seller warrants that the Hardware it manufactures and sells shall be free of defects in materials and workmanship for a period of 12 months from date of shipment to initial Buyer. This warranty does not apply to any product that is misused, abused, repaired, or otherwise modified by Buyer or others. Seller’s sole obligation for breach of this warranty shall be to repair or replace (F.O.B. Seller’s plant, Beaverton, Oregon, USA) any goods that are found to be non-conforming or defective as specified by Buyer within 30 days of discovery of any defect. Buyer shall bear all installation and transportation expenses, and all other incidental expenses and damages.

Limitation of Liability. *In no event shall Seller be liable for any type of special consequential, incidental, or penal damages, whether such damages arise from, or are a result of, breach of contract, warranty, tort (including negligence), strict liability, or otherwise.* All references to damages herein shall include, but not be limited to: loss of profit or revenue; loss of use of the goods or associated equipment; costs of substitute goods, equipment, or facilities; downtime costs; or claims for damages. Seller shall not be liable for any loss, claim, expense, or damage caused by, contributed to, or arising out of the acts or omissions of Buyer, whether negligent or otherwise.

No Other Warranties. Seller makes no other warranties, express or implied, including without limitation the implied warranties of merchantability and fitness for a particular purpose, regarding Seller’s Products or Documentation. Seller does not warrant, guarantee, or make any representations regarding the use or the results of the use of the Products or Documentation or their correctness, accuracy, reliability, currentness, or otherwise. All risk related to the results and performance of the Products and Documentation is assumed by Buyer. The exclusion of implied warranties is not permitted by some jurisdictions. The above exclusion may not apply to Buyer.

Disclaimer. Seller’s Products and Documentation, including this document, are subject to change without notice. Documentation does not represent a commitment from Seller.

Contents

PCI / PCIe Main Board

Overview

The EDT PCI / PCIe Main Board (PCIe8 LX, PCI GS, or PCI SS) is an interface that connects a mezzanine board to a PCI or PCIe host computer. It supports the mezzanine board with additional memory, field-programmable gate arrays (FPGAs), and high-speed DMA with up to 16 user-configurable channels.

NOTE In this guide, “LX” is the PCIe8 LX (with any FPGA); “GS” is the PCI GS; and “SS” is the PCI SS.

Each EDT board pair (main board + mezzanine board) provides the FPGAs described below. Alternatively, an EDT main board can be used without a mezzanine board, as a standalone hardware accelerator for additional FPGA and memory resources.

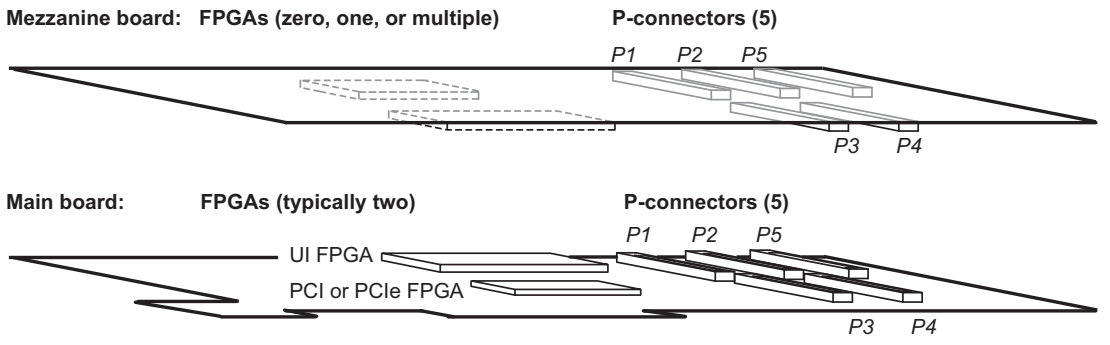
FPGAs: Main Board + Mezzanine Board

As shown in Figure 1, your EDT board pair has the following FPGAs:

- The mezzanine board may have zero, one, or multiple user-programmable FPGAs. For loading instructions, consult the user’s guide for your particular mezzanine board (see Related Resources on page 3).
- The main board has two FPGAs:
  - — The *user interface (UI) FPGA* links the mezzanine board’s FPGA to the main board’s PCI / PCIe FPGA. For loading instructions, see UI FPGA Configuration (.bit) Files on page 4.
  - — The *PCI / PCIe FPGA* communicates with the host computer over the PCI or PCIe bus and implements the DMA engine, which transfers data between the board and the host. This FPGA loads automatically, at powerup, the correct firmware from the main board’s FPGA configuration flash memory (“flash memory”).

NOTE Each FPGA must be loaded with firmware that works with your interface and with the firmware in the other FPGAs. Details are covered in this user’s guide and in the user’s guide for your mezzanine board.

Figure 1. Generic EDT Board Pair



Not to scale. Generic representations only; actual boards may vary.

DMA Interface and Requirements

The PCI / PCIe Main Board is a high-performance DMA device that requires adequate bandwidth for reliable operation.

Starting with PCIe 2.0 and depending on your BIOS, a PCIe bus “maximum payload size” setting may be available in your CMOS setup pages (consult your computer manufacturer’s documentation for availability and usage). If so, EDT recommends a setting of 256 bytes to increase maximum bandwidth up to 2-3% over the bandwidth associated with a typical default setting of 128. However, if your PCIe bus is running another device with a smaller maximum payload size, this setting may adversely affect that device’s performance.

For general specifications related to bus type, data rates, and I/O, see Table 1.

Table 1. PCI / PCIe Main Board Specifications and Features

	Bus type	DMA channels	Typical data rates	SRAM	DRAM	FPGAs	I / O
PCIe8 LX	PCIe (8 or 16 lanes)	16	Up to 1.2 GB/s	8 MB	0, 1, or 2 GB	• 1 PCI or PCIe • 1 user inter-face (UI)	Dependent on mezzanine board
PCI GS	PCI / PCI-X (66 MHz)	1	210 MB/s	Up to 8 MB	Up to 1 GB		
		4	210 MB/s				
		16	75 MB/s				
PCI SS	PCI / PCI-X (66 MHz)	1	210 MB/s	Up to 8 MB	None		
		4	210 MB/s				
		16	75 MB/s				
NOTE	If you use a bus slower than that specified, your board may work, but at reduced bandwidth.						

The main board implements the DMA interface by using the UI FPGA and the PCI / PCIe FPGA, shown in Figure 1 and explained further in Firmware on page 3.

When data comes in from the user device, the UI FPGA sends it to the first in / first out (FIFO) buffers. These buffers smooth the data transfer between the user device and the PCI / PCIe bus, and they accommodate data during the transition from one DMA transfer to the next.

Host DMA transfers are queued in hardware, minimizing the amount of FIFO required. To ensure maximum throughput, the EDT API library, PCD device driver, and FPGA configuration files all support pipelining.

- The library routines and the driver preallocate kernel resources for DMA (for example, memory), rather than waiting for an application to request a DMA transfer (usually with an EDT library routine call such as `edt_read, edt_write, or edt_start_buffers`). When one DMA transfer ends, the resources remain allocated and available for use by the next DMA transfer.
- A portion of host memory can be configured as ring buffers. A ring buffer is a set of buffers preallocated for DMA and reused in round-robin fashion.
- The FPGA fabric provides two sets of DMA registers so that when one DMA transfer starts, the registers required for the next are already prepared, enabling zero-latency transitions between DMA transfers.

To set the number of ring buffers and their size, use the EDT library call `edt_configure_ring_buffers`. Configure the ring buffers according to your application’s DMA requirements — a useful configuration is often four one-megabyte ring buffers. With four ring buffers, you can use one for the current DMA transfer, one for the pending DMA, and one for the application, with one extra to ensure zero-latency transitions.

You can fine-tune your application to the latency requirements of a particular system by increasing or decreasing the size of the ring buffers; slow systems may need larger ring buffers, while fast systems may achieve better performance with a greater number of smaller ones.

Some host systems may restrict your ability to allocate large ring buffers, or large numbers of them. For example, some Windows systems will limit DMA resources to a maximum of 64 megabytes in all. If you suspect this might be a problem in your system, be sure that your code will check for error returns after calling `edt_configure_ring_buffers` and before calling `edt_start_buffers`.

Related Resources

The resources below may be helpful or necessary for your applications.

EDT Resources

Description	*Detail*	*Web link*
• Mezzanine board documentation	Datasheets & user's guides	www.edt.com (find specific board)
• Time Distribution board documentation	Datasheet & user's guide	www.edt.com/timedist.html
• Application Programming Interface	HTML & PDF	www.edt.com/manuals.html
• Installation packages: Windows, Linux, Solaris, Mac	Software / firmware	www.edt.com/software.html

Standards / Specifications

Description	*Pertains to*	*Documentation*	*Web link*
• PCI Special Interest Group (PCI SIG)	PCI / PCIe bus	See weblink at right	www.pcisig.com
• I^2C (v. 2.1)	CPLD on mezzanine board	See weblink at right	www.nxp.com/acrobat_download/literature/9398/39340011.pdf

Parts

Description	*Part number*	*Manufacturer*	*Web link*
• Clock generator (PLL)	ICS307	Integrated Circuit Systems / Scantec	www.scantec.de/Hi-Q-News/2004/ICS-Versaclock/ics307.pdf
• Clock generator (PLL)	Si570	Silicon Labs	www.silabs.com
• CMC connector	71439-1164	Molex	www.molex.com
• Hex inverter	74ACT04	Texas Instruments	www.fairchildsemi.com/pf/74/74ACT04.html

Firmware

The PCI / PCIe Main Board comes with firmware, in the form of FPGA configuration (`.bit`) files, for both of its FPGAs (UI and PCI / PCIe). Each FPGA must be loaded with the correct firmware for its interface, and the firmware in each FPGA must be compatible with the firmware in the other.

NOTE At powerup, if your main board is a PCI GS or SS, it automatically detects the voltage of your PCI bus slot and loads the correct file (`_3v.bit` or `_5v.bit`) for you. Thus, you need not be concerned with these files; the file names below are the correct arguments to supply to the firmware-loading utilities.

PCI / PCIe FPGA Configuration (.bit) Files

For your main board's PCI / PCIe FPGA: The correct default PCI / PCIe FPGA configuration file is loaded automatically at powerup from flash memory.

Table 2 shows the automatically-loading default file for each type of main board.

Table 2. Automatically-Loading PCI / PCIe FPGA Configuration File for Each Main Board

	1-channel DMA	4-channel DMA	16-channel DMA
PCIe8 LX	–	–	`pe8lx16.bit` (lowercase LX + 16)
PCI GS	`pcigs1.bit`	`pcigs4.bit`	`pcigs16_classic.bit`
PCI SS	`pciss1.bit`	`pciss4.bit`	`pciss16_classic.bit`

More PCI / PCIe FPGA configuration files are in the `flash` subdirectory of your EDT installation package.

UI FPGA Configuration (.bit) Files

For your main board's UI FPGA: The correct firmware depends on which mezzanine board you use. To find and load the correct file (from the `bitfiles` subdirectory in your EDT installation package), see Configuring the PCI / PCIe Main Board on page 6.

Software

PCD Device Driver

The PCD device driver is the software running on the host that allows the host operating system to communicate with the main board. The driver is loaded into the kernel upon installation, and thereafter runs as a kernel module. The driver name and subdirectory is specific to each supported operating system, and the correct driver is loaded automatically in the correct manner for your operating system.

Software Initialization (.cfg) Files

Software initialization (`.cfg`) files are editable text files that run like scripts to configure EDT boards and prepare for DMA transfer. The commands in a software initialization file are defined in a C application named `initpcd`. When you invoke `initpcd`, you use the flag `-f` to specify which file to use.

A typical software initialization file loads an FPGA configuration file into the UI FPGA and sets up various registers to prepare the board for DMA transfers. In some cases, a software initialization file may load an FPGA configuration file into an FPGA on the mezzanine board.

Your EDT installation package contains multiple software initialization files, including at least one that is customized to work with the FPGA configuration file for the particular EDT main board you are using. In other words, each FPGA configuration file has a matching software initialization file.

Software initialization files are located in the `pcd_config` subdirectory in your EDT installation package. The software initialization files specific to your PCI / PCIe Main Board are listed in the user's guide for its companion EDT mezzanine board (see Related Resources on page 3). Instructions for using the files are provided in Configuring the PCI / PCIe Main Board on page 6.

You can use commands defined in `initpcd` (and typically found in the software initialization files) to:

- load specific FPGA configuration files (as with the command `bitfile:`);
- write specified hexadecimal values to specified registers (as with the command `command_reg:`);
- enable or disable byte-swapping or short-swapping to accommodate the requirements of different operating systems for bit ordering (as with the command `byteswap:`); or
- invoke arbitrary commands (as with the command `run_command:`).

If your mezzanine board has an FPGA, you can load the appropriate firmware on it by running the command `mezzload`, which determines the appropriate firmware for that mezzanine board in your system.

For example:

```
bitfile: ssd16io.bit
command_reg: 0x08
byteswap: 1
run_command: set_ss_vco -F 1000000 2
run_command: mezzload
```

For complete usage details, enter `initpcd --help`.

C source for `initpcd` is included so that you can add your own commands, if you wish. You can then edit your own software initialization file to use your new commands and specify that `initpcd` use your new file when configuring your board. If you would like us to include your new software initialization commands in subsequent releases of `initpcd`, email us at tech@edt.com.

Sample Applications

Along with the PCD device driver, the FPGA configuration files, and the software initialization files, the EDT installation package includes applications and utilities that you can use to initialize and configure the board, access the registers, and verify that the board is properly installed and functioning. For many of these applications and utiilities, C source is also provided, so that you can use them as starting points to write your own applications. The most commonly useful are listed below; for the complete list, see the README file.

For new installations, we recommend using the latest software download (see Related Resources on page 3) . For existing applications, upgrade only if you have a specific reason to do so.

DMA Applications

`rd16`	Performs simple multichannel ring buffer input.
`wr16`	Performs simple multichannel ring buffer output.
`simple_read`	Performs DMA input without using ring buffers. Data is therefore subject to interruptions, depending on system performance.
`simple_write`	Performs DMA output without using ring buffers. Data is therefore subject to interruptions, depending on system performance.
`simple_getdata`	Serves as an example of various DMA-related operations, including reading data from the connector interface and writing it to a file, as well as measuring input rate.
`simple_putdata`	Serves as an example of various DMA-related operations, including reading data from a file and writing it out to the connector interface.

`test_timeout`	Under normal operation, timeouts cancel DMA transfers. This application exemplifies giving notification when a timeout occurs, without canceling DMA

Utilities

`initpcd`	A utility for initializing and configuring the PCI / PCIe Main Board.
`pdb`	A utility that enables interactive reading and writing of the UI FPGA registers for a PCI / PCIe Main Board.
`set_ss_vco`	A utility for programming the output clock or clocks on the PCI / PCIe Main Board to specific frequencies used by the UI FPGA for input and output (see Generating an Output Clock on page 9).

Basic Testing

To verify that your EDT board pair (main board + mezzanine board) is properly installed, EDT provides C source, executable, and FPGA configuration files (see Verifying Installation on page 8), including at least:

`sslooptest`	Verifies installation for 4- or 16-channel PCI / PCIe Main Boards. Finds and runs the appropriate loopback test for the board model in use.

Building or Rebuilding an Application

In your EDT installation package, the executable and PCD source files are located in the top-level PCD directory. Thus, if you need to build or rebuild an application, run `make` in that directory.

- Linux users: You can use the `gcc` compiler typically included with the Linux installation.
- Windows or Solaris users: You must install a C compiler (we recommend, respectively, the Microsoft Visual C compiler or the Sun WorkShop C compiler; if you wish to use `gcc`, contact tech@edt.com).

After you've built (or rebuilt) an application, use the `--help` command line option to see a list of usage options and descriptions.

Configuring the PCI / PCIe Main Board

To operate properly, the PCI / PCIe Main Board must be loaded with the appropriate FPGA configuration files for both FPGAs, as explained in Firmware on page 3. Before loading the UI FPGA, however, you may wish to check the firmware in the PCI / PCIe FPGA to verify that it is correct and up to date.

Checking or Updating the PCI / PCIe FPGA Firmware

If you upgrade to a new driver, or switch to an FPGA configuration file with special functionality, you may need to reprogram the flash memory. To do so, use the application `pciload` as below.

NOTE The presence of a newer version of the firmware with a new driver does not always mean that the firmware must be updated; any mandatory updates will be indicated clearly in the README file.

NOTE The following procedure applies to standard firmware only. If you are running a custom firmware file and need to update it, you must first get a custom firmware configuration file from EDT.

- For Unix, `pciload` is an application in the installation directory `/opt/EDTpcd`.

- For Windows, double-click the Pcd Utilities icon to bring up a command shell in the installation directory `\EDT\Pcd`.
- For Mac, `pciload` is an application in the installation directory `/Applications/EDT/pcd`.

To see currently installed and recognized EDT boards and drivers, run:

```
pciload
```

The program outputs the date and revision number of the firmware in the flash memory. To compare the PCI / PCIe FPGA firmware in the package with the firmware already loaded on the board, enter:

```
pciload verify
```

The program compares the firmware in the flash memory against the firmware in the EDT installation directory. If they match, there's no need to upgrade the firmware. If they differ, you'll see error messages. Such messages do not necessarily indicate a problem; if your application is operating correctly, you may not need to upgrade the firmware. If you do wish to update the standard firmware, follow the steps below.

1. Enter:

   ```
   pciload update
   ```

2. To upgrade or switch to a custom firmware file, enter:

   `pciload` *`firmware_filename`*

   ...replacing *`firmware_filename`* with the name of the PCI / PCIe FPGA configuration file, either with or without the `.bit` file extension — for example:

   `pciload` *`pcigs1.bit`* (or, without the extension, `pciload` *`pciss1`*)

   If the host computer holds more than one board, you can specify the correct board to load with the optional *`unit_number`* argument (by default, 0 for the first or only board in a host):

   `pciload -u` *`unit_number filename`*

   Be sure to select the `.bit` file that matches the number of DMA channels you are running:

	1-channel DMA	**4-channel DMA**	**16-channel DMA**
PCIe8 LX	–	–	`pce8lx16.bit` (lowercase LX + 16)
PCI GS	`pcigs1.bit`	`pcigs4.bit`	`pcigs16_classic.bit`
PCI SS	`pciss1.bit`	`pciss4.bit`	`pciss16_classic.bit`

3. At the prompt, press Enter to confirm the loading operation. (If the file date is older than the ID date of the flash memory, you may need to press Enter twice.)

   The board reloads the firmware from the flash memory only during power-up, so after running `pciload`, the old firmware remains in the PCI / PCIe FPGA until the system has power-cycled.

NOTE You must cycle the power – not simply reboot – to complete your update of the firmware.

4. If you wish to see a list of all `pciload` options, enter:

   ```
   pciload --help
   ```

Loading the UI FPGA Firmware and Configuring the Board

The utility `initpcd` loads the UI FPGA configuration files, programs the registers, sets the clocks if needed, and prepares the PCI / PCIe Main Board to perform DMA. This utility takes, as an argument, a software initialization file, and then automatically runs the relevant commands.

If you use `initpcd` to configure the PCI / PCIe Main Board, your application can concern itself solely with performing DMA and other application-specific operations; therefore it will omit board-specific operations and be portable to other EDT boards that peform DMA.

To configure the PCI / PCIe Main Board, enter:

```
initpcd -u unit_number -f pcd_config/filename.cfg
```

...replacing *`unit_number`* with the number of the board (by default, 0), and replacing *`filename`* with one of the initialization files listed in Firmware on page 3 – for example:

```
initpcd -f ocm48.cfg
```

NOTE Software initialization files are editable text files. If the files provided do not meet your needs, copy and modify the one that is most similar to your required configuration; then run `initpcd` with your new file.

Using Custom FPGA Configuration Files

You can substitute your own FPGA configuration file, if necessary. If you wish to develop your own VHDL design, contact EDT. When you're done, be sure to create a new software initialization file for your new firmware file and update the `pcd_config` directory to include it.

Verifying Installation

To verify proper installation and functionality of your PCI / PCIe Main Board, follow the procedures below.

The application `sslooptest` is used to run a loopback test on any PCI / PCIe Main Board loaded with a 4- or 16-channel FPGA configuration file (`pciss4.bit`, `pciss16.bit`, `pcigs4.bit`, `pcigs16.bit`, or `pe8lx16.bit`) – along with the accompanying EDT mezzanine board, if any. The application and its C source are included in the EDT API library (see Related Resources on page 3).

This application determines board configuration, loads the appropriate FPGA configuration file, generates test data, and verifies installation of the board and its components with no external device connected.

NOTE The application `sslooptest` overwrites the FPGA configuration file in the main board's UI FPGA. Therefore, after verifying installation, you'll need to reconfigure the board before you can use it again.

To verify installation:

1. Leave the main board (with its attached mezzanine board, if any) in the host computer, but disconnect any external device and its cable.
2. At the command prompt, enter:

   ```
   sslooptest -u unit_number
   ```

   The outcome will vary, depending on which main board and mezzanine board you are using. Errors are directed to `stdout` and indicated in the test status output, as described below.

   For a functional board, the output contains such lines as:

   ```
   Total errs=0 bufs=4000; Channel errs(NNNx) bufs(YYYY)
   ```

   `Total errs` shows the error count so far; `bufs` shows the number of buffers in use.

   The three characters after `Channel errs` show the absence (`N`) or presence (`Y`) of a data error in a specific channel (0–3); an `x` indicates the corresponding channel is not in use.

Similarly, a `Y` after `Channel...bufs` shows a buffer in use; an `x` indicates the corresponding channel is not in use. An `N` indicates that DMA is not occurring in a specific channel.

NOTE The number of `x`, `Y`, or `N` characters in the parentheses after `errs` and `bufs` will vary, according to the number of channels in use.

3. After the process has completed, reconfigure the board using `initpcd` (or your own application) to disable loopback.
4. Reconnect the board to the external device, if any.

Generating an Output Clock

The PCI / PCIe Main Board has four programmable phase-locked loop (PLL) ICS307 clock generators, followed by a 14-bit programmable divider, which allow you to specify a precise frequency for data transfer. Each PLL can generate clock signals of 168 Hz to 100 MHz (or up to 200 MHz, with some FPGA configuration files); most of those frequencies can be achieved with little error. The reference clock to the ICS307 is 10.3861 MHz; for details, see the link under Related Resources on page 3.

NOTE The program `set_ss_vco.c` includes an example of setting the output clock frequency.

These library routines help compute the output clock frequency (see API under Related Resources):

`edt_find_vco_frequency_ics307`
: Computes the PLL parameter for the ICS307, based on an input clock frequency and a target frequency.

`edt_set_out_clk_ics307`
: Sets the frequency output using parameters computed by `edt_find_vco_frequency_ics307`.

`edt_set_frequency_ics307`
: A convenience function; sets the frequency on the desired channel by first calling `edt_find_vco_frequency_ics307` or `edt_find_vco_frequency_ics307_nodivide`, then `edt_set_out_clk_ics307`.

In addition, the fixed 40 MHz crystal oscillator is available to the UI FPGA and can be used by your custom FPGA configuration file. For details, contact EDT.

NOTE The PCIe8 LX main board has an additional programmable clock: a Silicon Labs Si570 PLL that can generate clock frequencies in the range of 10 – 280 MHz. Contact EDT for instructions on how to incorporate this clock into your custom UI design flow.

Configuration Space and PCI Bus Addresses

The configuration space is a 64-byte portion of memory that is required for configuring the bus and handling errors. Table 3 shows the structure as implemented for the PCI / PCIe Main Board.

NOTE The information in this section is governed by the PCI bus specification. For a link to that information, see Related Resources on page 3.

Table 3. Configuration Space – Addresses

Bit:	31 16		15 0	
0x00	*Device ID* SS: 1- or 4-channel = 40; 16-channel = 41 GS: 1- or 4-channel = 50; 16-channel = 51 LX: 16-channel = 97		*Vendor ID* 0x123D	
0x04	Status (see Table 4)		Command (see Table 5)	
0x08	Class Code = 0x088000			Revision ID = 0 (will be updated)
0x0C	BIST = 0x00	Header Type= 0x00	Latency Timer (set by OS)	Cache Line Size (set by OS)
0x10	DMA Base Address Register (set by OS)			
0x14	UI FPGA Memory-mapped I/O Base Address Register (set by OS)			
	[not implemented]			
0x3C	Max_Lat = 0x04	Min_Gnt = 0x04	Interrupt Pin = 0x01	Interrupt Line (set by OS)

Tables 4 and 5 show the values for the status and command fields.

Table 4. Configuration Space – Status Fields

Bit	Name	Value	Bit	Name	Value
15	Detected Parity Error	Implemented	9	DEVSEL Timing	1
14	Signaled System Error	Implemented	8	Data Parity Error Detected	Implemented
13	Received Master Abort	Implemented	7	Fast Back-to-back Capable	0
12	Received Target Abort	Implemented	6	UDF Supported	0
11	Signaled Target Abort	Implemented	5	66 MHz Capable	1
10	DEVSEL Timing	0	4–0	[reserved]	0

Table 5. Configuration Space – Command Fields

Bit	Name	Value	Bit	Name	Value
15–10	[reserved]	0	4	Memory Write and Invalidate Enable	0
9	Fast Back-to-back Enable	Implemented	3	Special Cycles	0
8	SERR# Enable	Implemented	2	Bus Master	Implemented
7	Wait Cycle Control	0	1	Memory Space	Implemented
6	Parity Error Response	Implemented	0	IO Space	0
5	VGA Palette Snoop	0			

Table 6 shows the address assignments of the interface registers with a 1- or 4-channel FPGA configuration file. Table 7 shows the address assignments of the interface registers with a 16-channel FPGA configuration file. The addresses in these tables are offsets from the PCI base address. This base address is initialized, at boot time, by the PCI / PCIe host operating system.

NOTE Your applications must not modify registers 0x80 and 0x84, which are used by the EDT utility `pciload` to update the flash memory. The results of running `pciload` do not take effect until after the board has been powered off and then on again.

Table 6. PCI Addresses for 1- or 4-channel Operation

Bit::	31 … 16	15 … 0
0xCC	UI FPGA data	
0xC8	PCI interrupt status	
0xC4	PCI interrupt and UI FPGA configuration	
0x84	[not used]	flash memory data
0x80	flash memory address	
0x60–7C	channel 3 DMA registers *(set up as channel 0 is)*	
0x40–5C	channel 2 DMA registers *(set up as channel 0 is)*	
0x20–3C	channel 1 DMA registers *(set up as channel 0 is)*	
0x1C	channel 0 scatter-gather DMA next count and control	
0x18	channel 0 scatter-gather DMA current count and control	
0x14	channel 0 scatter-gather DMA next address	
0x10	channel 0 scatter-gather DMA current address	
0x0C	channel 0 main DMA next count and control	
0x08	channel 0 main DMA current count and control	
0x04	channel 0 main DMA next address	
0x00	channel 0 main DMA current address	

Byte:	3	2	1	0
Word:	1		0	

Table 7. PCI / PCIe Addresses for 16-channel Operation

Bit:	31 … 16	15 … 0
0x280–3FC	channels 4–15 DMA registers*(set up as channels 0–3 are)*	
0x260–27C	channel 3 DMA registers *(set up as channel 0 is)*	
0x240–25C	channel 2 DMA registers *(set up as channel 0 is)*	
0x220–23C	channel 1 DMA registers *(set up as channel 0 is)*	
0x21C	channel 0 scatter-gather DMA next count and control	
0x218	channel 0 scatter-gather DMA current count and control	
0x214	channel 0 scatter-gather DMA next address	
0x210	channel 0 scatter-gather DMA current address	
0x20C	channel 0 main DMA next count and control	
0x208	channel 0 main DMA current count and control	
0x204	channel 0 main DMA next address	
0x200	channel 0 main DMA current address	
0xD0–1FC	[not used]	
0xCC	UI FPGA data	
0xC8	PCI interrupt status	
0xC4	PCI interrupt and UI FPGA configuration	
0x84	[not used]	flash memory data
0x80	flash memory address	
0x00–7C	[not used]	

Byte:	3	2	1	0
Word:	1		0	

Scatter-gather DMA

For PCI / PCIe DMA devices in Intel-based computers, memory is accessed through physical addresses. Because the operating system uses a memory manager to connect the user program to memory, memory pages that appear contiguous to the user program are actually scattered throughout physical memory. Because DMA accesses physical addresses, a DMA read operation must gather data from noncontiguous pages, and a write must scatter the data back to the appropriate pages.

The PCD device driver uses information from the operating system to accomplish this. The operating system passes the driver a list of the physical addresses for the user program memory pages. With this information, the driver builds a scatter-gather table, which the DMA device uses sequentially.

Most other PCI / PCIe computers offer memory management for the PCI / PCIe bus also, so the operating system needs to pass only the address and count for DMA. The addresses appear contiguous to the bus.

The scatter-gather DMA list is stored in memory. The scatter-gather DMA channel copies the list as required into the main DMA registers. The format of the scatter-gather DMA list in memory (as shown in Table 8) is:

- Each page entry takes eight bytes, so the scatter-gather DMA count is always evenly divisible by eight.
- The first word consists of the 32-bit start address of a memory page.
- The most significant 16 bits of the second word contain control data.
- The least significant 16 bits of the second word contain the count.

Only bit 16 contains control information. When set to 1, with bit 28 set in 0x1C Scatter-gather DMA Next Count and Control, bit 16 causes the main DMA interrupt to be set when the marked page is complete.

Table 8. Scatter-gather DMA List Format

Bit:	63	32	31	16	15 0
Content:	address		control (unused)	DMA inter-rupt	count

All of the main DMA registers are read-only; only the corresponding scatter-gather DMA registers must write to them. To initiate a DMA transfer, the driver performs the following general operations:

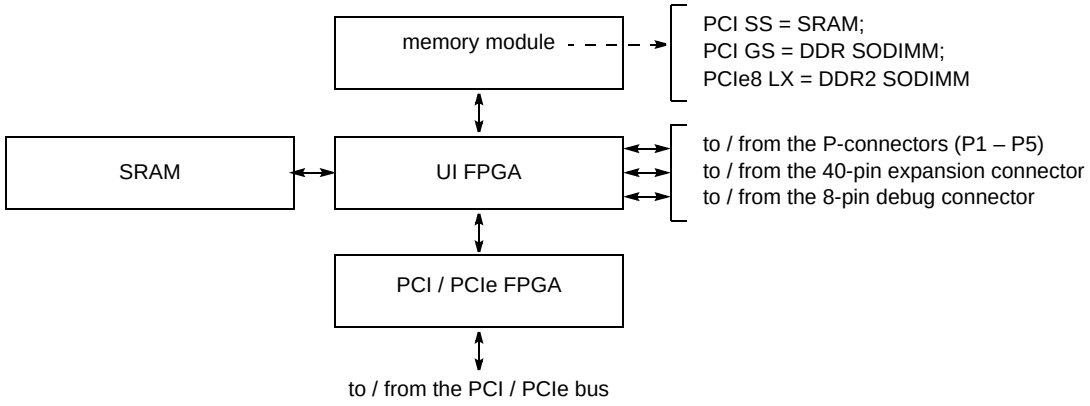
1. It sets up one or more scatter-gather DMA lists in host memory, as explained above and in Table 8.
2. It writes the address of the first entry in the list to 0x14 Scatter-gather DMA Next Address.
3. It writes the length of the scatter-gather DMA list to 0x1C Scatter-gather DMA Next Count and Control, setting the interrupts as you require. Setting bit 29 of this register to 1 starts the DMA.
4. If the DMA list is greater than one page, it loads the address of the first entry of the next page and its length, as described in steps 2 and 3 above, when bit 29 of 0x1C Scatter-gather DMA Next Count and Control is asserted.

Hardware Interfaces

This section discusses the hardware interfaces for an EDT board pair (mezzanine board + PCI / PCIe Main Board) and can also be used to help you design your own mezzanine board for a PCI / PCIe Main Board.

For details on how the two boards work together, see FPGAs: Main Board + Mezzanine Board on page 1. For details on how the major components connect to the UI FPGA, see Figure 2 below.

Figure 2. Diagram – PCI / PCIe Main Board + Mezzanine Board



Mezzanine Board Interface

Your PCI / PCIe Main Board links to its mezzanine board via five 64-pin, CMC-style *P-connectors* as below.

Mechanical Interface

Figure 3 shows the relationship between the mezzanine and main board.

Figure 3. Mezzanine Connectors and Board Dimensions – In Inches Unless Otherwise Noted

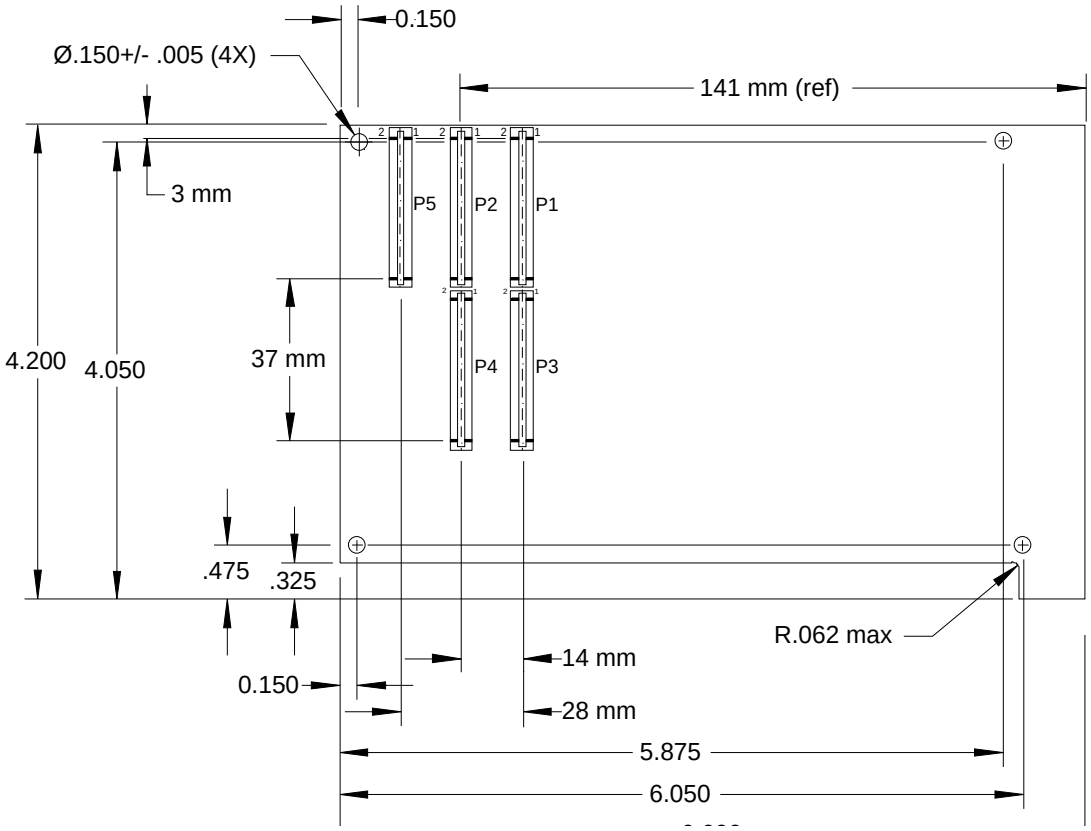


Table 9 shows the connectors and hardware for linking a PCI / PCIe Main Board to a mezzanine board.

Table 9. Connectors and Hardware for Linking PCI / PCIe Main Board to Mezzanine

EDT part #	Part Type	Manufacturer part #	Qty. per board	Description
012-11661-00	CMC receptacle (connector)	Molex 71439-1164	ECB, 5 ea. (at least 2 required)	Electrical connection to mezzanine board
020-01686-00	Round spacer; 12 mm x 4.5 mm al. thread	–	4	Mechanical connection to mezzanine board, included with EDT main board.
020-01700-00	Screw, M2.5 mm x 6 mm Phillips flathead m/s	–	4	Screws for the above spacer, included with EDT main board.
020-01694-00	Flat nylon washer #4 0.120 ID x 0.250 OD x 0.015 thick	–	1	Shim for Purcell P122 back panel; may not be required for a different back panel.
020-01714-00	Screw, 4-40 x 3/16", slot hex washer head m/s, zinc	–	1	Screw to attach Purcell back panel.
020-01073-00	Standoff, 2-56 to 4-40 14000-H-SS-.140-12 Mod B+4-40x.188 Must be passivated.	–	2	SCSI connector attachment, mates with SCSI cable jackscrew.
020-01522-00	External star washer #4, aluminum	–	4	Lock washer for above and below parts.
020-00405-00	Hex bushing, threaded, 4-40 x 3/16" m/f standoff, 3/16" af, 5/16" thread length, zinc yellow	–	2	DE15 connector attachment
027-01674-00	Ultra-thin fan	SunonKDE0502 PEB1-8 or Digikey 259-1001-ND	1	Small mezzanine fan
012-01032-00	Connector, right-angle, 68-pin SCSI-type	Tyco 787169-7	1	68-pin SCSI connector
012-01488-00	Connector, DE15	Tyco 748390-5	1	15-pin D-type connector
017-01688-00	Back panel (board-specific)	Purcell brackets P122-EDT-001	1	For Combo mezzanine board; built from Purcell P122 blank stock
017-01712-00	Back panel (board-specific)	Purcell brackets P122-EDT-002	1	For LVDS/RS422 mezzanine board; built from Purcell P122 blank stock
020-01698-00	Screw, #4 x 5/16", Phillips flathead, self-tapping, type B	–	3	Attachment for small fan. Must also be secured with Locktite 262 or similar adhesive.

Board Identification

The CMC BUSMODE connections act as a four-bit board identifier by which the PCI / PCIe Main Board can see which mezzanine board is connected to it. For details, see the register .

To implement your own board ID, select a value that is not used and decide how you wish to implement the ID in hardware. If you follow the schema described in this section, you can use the software interface implemented in `extbdid.c`.

CPLD Hardware Interface

The interface between the UI FPGA on the PCI / PCIe Main Board and the CPLD on the mezzanine board is a two-wire interface consisting of clock and data, set up as an I^2C bus. (For the I^2C bus specification, see Related Resources on page 3.) The main board uses pins 40 and 42 on the P5 connector for this purpose.

You can access this bus using the upper four bits of the board ID register at address 0x7F, as below.

Bit	Description
7	Tristate for data I/O Set to read from CPLD; clear to write to CPLD.
6	Clock signal to CPLD. The main board UI FPGA is always the bus master.
5	Bidirectional data signal for the I^2C bus to or from the CPLD.
4	Set in main board UI FPGA to indicate that the firmware currently loaded supports the two-wire I^2C interface to the CPLD.

If the software reads a value of two in the lower four bits of the board ID register, as shown in Table 3 - Configuration Space – Addresses, then the software uses the upper four bits described in Table 7 - PCI / PCIe Addresses for 16-channel Operation, to send an I^2C start condition to the CPLD, read the extended board ID information (highest bit first), and send an I^2C stop condition.

CPLD Data Format

Information is stored in the CPLD as eight 32-bit blocks of data. The first of these blocks is required and always readable; others are optional.

The first block is formatted as shown in Figure 4.

Figure 4. First Data Block in CPLD

Bit:	31 20	19 8	7 0
Data:	number of blocks to read — always includes the first block (itself)	board ID	board revision ID — a binary number in the range 0–255

You can use the other seven 32-bit data blocks to store any arbitrary information about the mezzanine board (for example, the options that are installed with it). Each block contains a unique identifier which software can use to index a lookup table containing the information.

Power

Power to the mezzanine board is supplied through the five P-connectors.

For PCI GS and PCI SS: P1, P2, and P3 use the power pin assignments that are specified in the CMC standard, while P4 and P5 use 12 V and ground assignments that are unique to the main board.

For PCIe8 LX: P1, P2, and P3 use the power pin assignments that are specified in the CMC standard, with one exception for P1, pin 2 (as shown in Table 12), while P4 and P5 use 12 V and ground assignments that are unique to the main board.

The power available to the mezzanine board depends on the PCI / PCIe slot, and on the power consumption of the main board FPGA design.

For a bus using PCI revision 2.2 or later, the limits for PCI SS revision 20 or later and PCI GS are as shown in Table 10. The limits for the PCIe8 LX are shown in Table 11.

Table 10. Power Limits for PCI SS or PCI GS

Voltage	Current	Limit
+3.3	System-dependent	Varies with the FPGA's design and clock frequency.
+5	System-dependent	Requires none.
+12	500 mA	See PCI specification.
–12	50 mA	See PCI specification.

For PCI SS revisions earlier than 20, or rev. 20 wired for compliance with PCI 2.1, a 3.3 V power supply is generated from the 5 V power supply. On these boards, the power available to the mezzanine board is limited to what remains after the main board consumes its power, typically 1 A or less.

Table 11. Power Limits for PCIe8 LX

Voltage	Current	Limit
+3.3	Not used	–
+3.3 V AUX	Not used	–
+12	System-dependent	Varies with the FPGA's design and clock frequency.

Signals

External signals are connected to the UI FPGA on the PCI / PCIe Main Board. The signal function is determined by which mezzanine board you use, and which FPGA configuration file you load. To see how certain components connect to the UI FPGA, see Figure 2 under Hardware Interfaces on page 12.

The FPGA I/O signals (shown in the P1–P5 connector pinouts, Tables 12–16) can be any standard supported by the FPGA with 3.3 V VCCO. The FPGA GCLKIN signal is typically connected to a reference clock signal, but need not be; on the main board, it is connected to the FPGA global clock input pins, but a mezzanine board need not use it.

Also, as shown in Table 17, connectors P3–P5 are wired in pairs so you can use LVDS pairs for FPGA-supported differential signaling (PCI GS and PCI SS only – not supported by the PCIe8 LX board).

Table 12. Pinout – P1 Connector to PCI / PCIe FPGA on LX, GS, SS

Pin	Function	LX	GS	SS	Pin	Function	LX	GS	SS
01	FPGA I/O	F34	E30	N36	02	power	+12 V	–12 V	–12 V
03	ground	–	–	–	04	FPGA I/O	E34	F30	N37
05	FPGA I/O	G34	C29	P36	06	FPGA I/O	H34	F29	P38
07	Board ID 0	–	–	–	08	+5 V	–	–	–
09	FPGA I/O	G36	C33	P39	10	FPGA I/O	E35	D31	R36
11	ground	–	–	–	12	FPGA I/O	F35	E31	R38
13	FPGA I/O	D37	C35	R39	14	ground	–	–	–
15	ground	–	–	–	16	FPGA I/O	E38	D35	T36
17	FPGA I/O	E37	D34	T37	18	+5 V	–	–	–
19	+3.3 V	–	–	–	20	FPGA I/O	F37	E34	T38
21	FPGA I/O	F36	D33	T39	22	FPGA I/O	H36	F33	U35
23	FPGA I/O	J36	G31	U36	24	ground	–	–	–
25	ground	–	–	–	26	FPGA I/O	H35	H31	U37
27	FPGA I/O	G42	H38	U38	28	FPGA I/O	F42	H37	AW36
29	FPGA I/O	G41	J39	AW35	30	+5 V	–	–	–
31	+3.3 V	–	–	–	32	FPGA I/O	F41	J38	AW34
33	FPGA I/O	J41	K39	AW33	34	ground	–	–	–
35	ground	–	–	–	36	FPGA I/O	H41	K38	AW32
37	FPGA I/O	K42	L39	AW31	38	+5 V	–	–	–
39	ground	–	–	–	40	FPGA I/O	J42	L38	AW30
41	FPGA I/O	M41	N39	AW29	42	FPGA I/O	L42	N38	AW28
43	FPGA I/O	N41	P39	AW27	44	ground	–	–	–
45	+3.3 V	–	–	–	46	FPGA I/O	M42	P38	AW26
47	FPGA I/O	L41	M38	AW25	48	FPGA I/O	P41	H38	AW24
49	FPGA I/O	R40	R39	AW23	50	+5 V	–	–	–
51	ground	–	–	–	52	FPGA I/O	T42	T37	AW22
53	FPGA I/O	U41	T38	AW21	54	FPGA I/O	U42	U38	AU28
55	FPGA I/O	V41	U39	AU27	56	ground	–	–	–
57	+3.3 V	–	–	–	58	FPGA I/O	W42	V38	AU26
59	FPGA I/O	Y42	V39	AU25	60	FPGA I/O	AA42	Y36	AU24
61	FPGA I/O	AA41	Y37	AU23	62	+5 V	–	–	–
63	ground	–	–	–	64	FPGA GCLKIN	K29	D20	AU22

Table 13. Pinout – P2 Connector to PCI / PCIe FPGA on LX, GS, SS

P2	Function	LX	GS	SS	P2	Function	LX	GS	SS
01	+12 V	–	–	–	02	FPGA I/O	E40	D28	U39
03	FPGA I/O	E39	E28	V35	04	FPGA I/O	F40	G29	V36
05	FPGA I/O	F39	H29	V37	06	ground	–	–	–
07	ground	–	–	–	08	FPGA I/O	G38	H33	V38
09	FPGA I/O	G39	H34	V39	10	FPGA I/O	H38	J33	W36
11	Board ID 1	–	–	–	12	+3.3 V	–	–	–
13	FPGA I/O	H39	J34	W37	14	Board ID 2	–	–	–
15	+3.3 V	–	–	–	16	Board ID 3	–	–	–
17	FPGA I/O	J40	H36	W38	18	ground	–	–	–
19	FPGA I/O	J38	K36	W39	20	FPGA I/O	K38	K35	Y38
21	ground	–	–	–	22	FPGA I/O	H40	J35	AA36
23	FPGA I/O	N39	L34	AA37	24	+3.3 V			
25	FPGA I/O	M39	L35	AA38	26	FPGA I/O	K40	J37	AA39
27	+3.3 V	–	–	–	28	FPGA I/O	K39	K37	AB37
29	FPGA I/O	L39	L37	AV35	30	ground	–	–	–
31	FPGA I/O	M38	L36	AU34	32	FPGA I/O	P40	M34	AV34
33	ground	–	–	–	34	FPGA I/O	N40	M33	AV33
35	FPGA I/O	R38	P36	AV32	36	+3.3 V	–	–	–
37	ground	–	–	–	38	FPGA I/O	R39	P35	AV31
39	FPGA I/O	T41	P37	AV30	40	ground	–	–	–
41	+3.3 V	–	–	–	42	FPGA I/O	T40	N37	AV29
43	FPGA I/O	U39	R37	AV28	44	ground	–	–	–
45	FPGA I/O	W41	U36	AV27	46	FPGA I/O	T39	R36	AV26
47	ground	–	–	–	48	FPGA I/O	V40	T36	AV25
49	FPGA I/O	AA39	Y33	AV24	50	+3.3 V	–	–	–
51	FPGA I/O	W38	W35	AV23	52	FPGA I/O	W40	V36	AV22
53	+3.3 V	–	–	–	54	FPGA I/O	V39	W34	AV21
55	FPGA I/O	Y38	W37	AT27	56	ground	–	–	–
57	FPGA I/O	AA37	W31	AT26	58	FPGA I/O	Y39	W36	AT25
59	ground	–	–	–	60	FPGA I/O	Y37	Y31	AT24
61	FPGA I/O	Y40	V37	AT23	62	+3.3 V	–	–	–
63	ground	–	–	–	64	FPGA I/O	AA40	Y32	AT22

Table 14. Pinout – P3 Connector to PCI / PCIe FPGA on LX, GS, SS

P3	Function	LX	GS	SS	P3	Function	LX	GS	SS
01	FPGA I/O	AB42	AA37	AW18	02	ground	–	–	–
03	ground	–	–	–	04	FPGA I/O	AB41	AA36	AV19
05	FPGA I/O	AC38	AA35	AW17	06	FPGA I/O	AB39	AA34	AV18
07	FPGA I/O	AD42	AB39	AW16	08	ground	–	–	–
9	+3.3 V	–	–	–	10	FPGA I/O	AC41	AB38	AV17
11	FPGA I/O	AC39	AB37	AW15	12	FPGA I/O	AC40	AB36	AV16
13	FPGA I/O	AD41	AC39	AW14	14	ground	–	–	–
15	ground	–	–	–	16	FPGA I/O	AE42	AC38	AV15
17	FPGA I/O	AG41	AC36	AW13	18	FPGA I/O	AF40	AD36	AV14
19	FPGA I/O	AD40	AD38	AW12	20	ground	–	–	–
21	+3.3 V	–	–	–	22	FPGA I/O	AE40	AD37	AV13
23	FPGA I/O	AF42	AE39	AW11	24	FPGA I/O	AF41	AE38	AV12
25	FPGA I/O	AK42	AH38	AW10	26	ground	–	–	–
27	ground	–	–	–	28	FPGA I/O	AG42	AF38	AV11
29	FPGA I/O	AJ40	AF37	AW9	30	FPGA I/O	AH40	AG37	AV10
31	FPGA I/O	AJ41	AG39	AW8	32	ground	–	–	–
33	ground	–	–	–	34	FPGA I/O	AJ42	AG38	AV9
35	FPGA I/O	AM39	AG35	AW7	36	FPGA I/O	AL39	AH36	AV8
37	FPGA I/O	AH41	AF39	AW6	38	ground	–	–	–
39	+3.3 V	–	–	–	40	FPGA I/O	AP42	AK38	AV7
41	FPGA I/O	AM42	AJ39	AW5	42	FPGA I/O	AL42	AJ38	AV6
43	FPGA I/O	AN41	AJ37	AW4	44	ground	–	–	–
45	ground	–	–	–	46	FPGA I/O	AM41	AJ36	AV5
47	FPGA I/O	AP40	AJ35	AV4	48	FPGA I/O	AN40	AJ34	AV3
49	FPGA I/O	AP41	AK39	AU4	50	ground	–	–	–
51	ground	–	–	–	52	FPGA I/O	AL41	AH37	AR4
53	FPGA I/O	AT42	AK36	AT3	54	FPGA I/O	AR42	AK35	AR3
55	FPGA I/O	AT40	AK37	AT2	56	ground	–	–	–
57	+3.3 V	–	–	–	58	FPGA I/O	AR40	AL37	AR2
59	FPGA I/O	AU41	AL39	AT1	60	FPGA I/O	AT41	AL38	AK1
61	FPGA I/O	AV41	AM38	AR1	62	ground	–	–	–
63	ground	–	–	–	64	FPGA I/O	AU42	AM37	AJ4

Table 15. Pinout – P4 Connector to PCI / PCIe FPGA on LX, GS, SS

P4	Function	LX	GS	SS	P4	Function	LX	GS	SS
01	+12 V	–	–	–	02	FPGA I/O	AB36	Y29	AW20
03	FPGA I/O	AD35	AA31	AU21	04	FPGA I/O	AC36	AA30	AT21
05	FPGA I/O	AB38	AA33	AU19	06	ground	–	–	–
07	FPGA I/O	AD37	AB33	AT19	08	FPGA I/O	AB37	AA32	AU18
09	FPGA I/O	AK25	AC31	AT18	10	FPGA I/O	AD36	AB32	AR18
11	ground	–	–	–	12	FPGA I/O	AK24	AC30	AU17
13	FPGA I/O	AD38	AB35	AT17	14	FPGA I/O	AE37	AB34	AR17
15	FPGA I/O	AE38	AC35	AU16	16	ground	–	–	–
17	FPGA I/O	AG38	AD34	AT16	18	FPGA I/O	AE39	AC34	AU15
19	FPGA I/O	AF37	AE35	AT15	20	FPGA I/O	AF39	AD33	AU14
21	ground	–	–	–	22	FPGA I/O	AG37	AE34	AT14
23	FPGA I/O	AH38	AF36	AU13	24	FPGA I/O	AJ37	AF35	AT13
25	FPGA I/O	AL24	AG34	AU12	26	ground			
27	FPGA I/O	AT30	AG31	AU11	28	FPGA I/O	AL25	AG33	AT11
29	FPGA I/O	AK39	AH34	AU10	30	FPGA I/O	AR30	AH32	AT10
31	ground	–	–	–	32	FPGA I/O	AJ38	AH33	AU9
33	FPGA I/O	AK37	AH30	AT9	34	FPGA I/O	AK38	AJ31	AU8
35	FPGA I/O	AL37	AJ33	AT8	36	ground	–	–	–
37	FPGA I/O	AM38	AK34	AU7	38	FPGA I/O	AM37	AJ32	AT7
39	FPGA I/O	AP26	AK32	AU6	40	FPGA I/O	AN38	AK33	AT6
41	ground	–	–	–	42	FPGA I/O	AN26	AK31	AP4
43	FPGA I/O	AP25	AL34	AP3	44	FPGA I/O	AN25	AL33	AP2
45	FPGA I/O	AP38	AL35	AP1	46	ground	–	–	–
47	FPGA I/O	AT29	AM33	AN4	48	FPGA I/O	AN39	AM36	AN3
49	FPGA I/O	AR39	AN34	AN2	50	FPGA I/O	AU29	AM34	AN1
51	ground	–	–	–	52	FPGA I/O	AT39	AN33	AM4
53	FPGA I/O	AR28	AL31	AM3	54	FPGA I/O	AR29	AM32	AN2
55	FPGA I/O	AN24	AH29	AM1	56	ground	–	–	–
57	FPGA I/O	AR27	AP33	AL4	58	FPGA I/O	AM24	AN31	AL3
59	FPGA I/O	AU39	AR34	AL2	60	FPGA I/O	AT26	AR33	AL1
61	ground	–	–	–	62	FPGA I/O	AV40	AT34	AK4
63	FPGA I/O	AT27	AR32	AK3	64	FPGA I/O	AU28	AT32	AK2

Table 16. Pinout – P5 Connector to PCI / PCIe FPGA on LX, GS, SS

P5	Function	LX	GS	SS	P5	Function	LX	GS	SS
01	+12 V	–	–	–	02	FPGA I/O	E32	C21	AB38
03	FPGA I/O	E33	C22	AC36	04	FPGA I/O	F31	E21	AC37
05	FPGA I/O	F32	D21	AC38	06	ground	–	–	–
07	FPGA I/O	H33	F21	AC39	08	FPGA I/O	G33	F20	AD36
09	FPGA I/O	G31	G22	AD37	10	FPGA I/O	G32	G21	AD38
11	ground	–	–	–	12	FPGA I/O	K30	E22	AD39
13	FPGA I/O	L30	D22	AE37	14	FPGA I/O	K37	F22	AE38
15	FPGA I/O	J37	E23	AF36	16	ground	–	–	–
17	FPGA I/O	J35	C23	AF37	18	FPGA I/O	K35	D23	AF38
19	FPGA I/O	K28	F24	AF39	20	FPGA I/O	L29	G23	AG36
21	ground	–	–	–	22	FPGA I/O	L37	E24	AG37
23	FPGA I/O	M37	E25	AG39	24	FPGA I/O	M27	J24	AH37
25	FPGA I/O	M28	H24	AH38	26	ground	–	–	–
27	FPGA I/O	L15	C25	AJ36	28	FPGA I/O	L16	D25	AJ37
29	FPGA I/O	M17	E26	AJ38	30	FPGA I/O	L17	F26	AJ39
31	ground	–	–	–	32	FPGA I/O	P38	G25	AK36
33	FPGA I/O	N38	F25	AK38	34	FPGA I/O	N36	D26	AK39
35	FPGA I/O	P36	C26	AL36	36	ground	–	–	–
37	FPGA I/O	P37	K33	AL38	38	FPGA I/O	P35	H25	AM36
39	FPGA I/O	T36	L33	AM37	40	FPGA I/O	R37	K34	AM38
41	ground	–	–	–	42	FPGA I/O	R35	L32	AN36
43	FPGA I/O	T35	M32	AN37	44	FPGA I/O	U34	N31	AN38
45	FPGA I/O	U33	N34	AN39	46	ground	–	–	–
47	FPGA I/O	V34	P34	AP36	48	FPGA I/O	T34	N33	AP37
49	FPGA I/O	U38	N20	AP38	50	FPGA I/O	T37	L21	AP39
51	ground	–	–	–	52	FPGA I/O	V35	P33	AR36
53	FPGA I/O	W33	R35	AR37	54	FPGA I/O	V33	R34	AR38
55	FPGA I/O	W37	T34	AR39	56	ground	–	–	–
57	FPGA I/O	W35	U35	AT38	58	FPGA I/O	W36	T33	AT39
59	FPGA I/O	AA36	V35	AU36	60	FPGA I/O	Y35	U34	AV36
61	ground	–	–	–	62	FPGA I/O	AA35	V34	AT29
63	FPGA I/O	AA32	W33	AR23	64	FPGA I/O	Y32	W32	AR22

Table 17. Pinout - LVDS Pairs on CMC (PCI GS and PCI SS Only)

P3 Pairs			
Pin	Signal	Signal	Pin
1	1–	1+	4
5	2–	2+	6
7	3–	3+	10
11	4–	4+	12
13	5–	5+	16
17	6–	6+	18
19	7–	7+	22
23	8–	8+	24
25	9–	9+	52
29	10–	10+	30
31	11–	11+	34
35	12–	12+	36
37	14–	14+	28
41	15–	15+	42
43	16–	16+	46
47	17–	17+	48
49	18–	18+	40
53	19–	19+	54
55	20–	20+	58
59	21–	21+	60
61	22–	22+	64

P4 Pairs			
Pin	Signal	Signal	Pin
3	1–	1+	4
5	2–	2+	8
7	3–	3+	10
9	4–	4+	12
13	5–	5+	14
15	6–	6+	18
17	7–	7+	20
19	8–	8+	22
23	9–	9+	24
25	10–	10+	28
27	11–	11+	30
29	12–	12+	32
33	14–	14+	34
35	15–	15+	38
37	16–	16+	40
39	17–	17+	42
43	18–	18+	44
45	19–	19+	48
47	20–	20+	50
49	21–	21+	52
53	22–	22+	54
57	23–	23+	60
59	24–	24+	62
63	25–	25+	64

P5 Pairs			
Pin	Signal	Signal	Pin
3	1–	1+	2
5	2–	2+	4
7	3–	3+	8
9	4–	4+	10
13	5–	5+	12
15	6–	6+	14
17	7–	7+	18
19	8–	8+	20
23	9–	9+	22
25	10–	10+	24
27	11–	11+	28
29	12–	12+	30
33	14–	14+	32
35	15–	15+	34
37	16–	16+	40
39	17–	17+	42
43	18–	18+	44
45	19–	19+	48
47	20–	20+	52
49	21–	21+	50
53	22–	22+	54
55	23–	23+	58
57	24–	24+	60
59	25–	25+	62
63	26–	26+	64

External I/O Interface

For external I/O or debugging, the PCI / PCIe Main Board has a 40-pin QSH or ATA connector (see Tables 18 and 19 below) which mates with common .05-inch insulation displacement connectors. For ATA, the connector and ground assignments allow you to use high-quality but inexpensive PC-standard ATA cables.

- PCIe8 LX and PCI GS – these are 3.3 V by default, but you can cut a wire and add a jumper to change them to 2.5 V I/O standards; for details, contact EDT.
- PCI SS – you can use the I/O pins on the UI FPGA for any 3.3 V I/O standard.

Table 18. Pinout – QSH Connector to PCI / PCIe FPGA on PCIe8 LX

QSH	LX	Function	QSH	LX	Function
01	F4	RX_REFCLK +	02	M14	TX_REFCLK +
03	F3	RX_REFCLK –	04	M13	TX_REFCLK –
05	A5	RX_RIO [0]+	06	B6	TX_RIO [0]+
07	A4	RX_RIO [0]–	08	B5	TX_RIO [0]–
09	A2	RX_RIO [1]+	10	B1	TX_RIO [1]+
11	A3	RX_RIO [1]–	12	B2	TX_RIO [1]–
13	E1	RX_RIO [2]+	14	D2	TX_RIO [2]+
15	F1	RX_RIO [2]–	16	E2	TX_RIO [2]–
17	H1	RX_RIO [3]+	18	J2	TX_RIO [3]+
19	G1	RX_RIO [3]–	20	H2	TX_RIO [3]–
21	L14	RX_CLK0 +	22	N13	TX_CLK0 +
23	K15	RX_CLK0 –	24	P13	TX_CLK0 –
25	N15	RX_DATA [0]+	26	M16	TX_DATA [0]+
27	N14	RX_DATA [0]–	28	N16	TX_DATA [0]–
29	P18	RX_DATA [1]+	30	L24	TX_DATA [1]+
31	P17	RX_DATA [1]–	32	M24	TX_DATA [1]–
33	G16	RX_DATA [2]+	34	F16	TX_DATA [2]+
35	H16	RX_DATA [2]–	36	F17	TX_DATA [2]–
37	G18	RX_DATA [3]+	38	E18	TX_DATA [3]+
39	G17	RX_DATA [3]–	40	E17	TX_DATA [3]–
41	M18	RX_DATA [4]+	42	M19	TX_DATA [4]+
43	N18	RX_DATA [4]–	44	N19	TX_DATA [4]–
45	K18	RX_DATA [5]+	46	J18	TX_DATA [5]+
47	K19	RX_DATA [5]–	48	H18	TX_DATA [5]–
49	L20	RX_DATA [6]+	50	N20	TX_DATA [6]+
51	L19	RX_DATA [6]–	52	P20	TX_DATA [6]–
53	M26	RX_CLK [1]+	54	N30	TX_CLK [1]+
55	L27	RX_CLK [1]–	56	M29	TX_CLK [1]–
57	N25	RX_DATA [7]+	58	P26	TX_DATA [7]+
59	P25	RX_DATA [7]–	60	N26	TX_DATA [7]–
61	P27	RX_DATA [8]+	62	N28	TX_DATA [8]+
63	P28	RX_DATA [8]–	64	N29	TX_DATA [8]–
65	K24	RX_DATA [9]+	66	K25	TX_DATA [9]+
67	L25	RX_DATA [9]–	68	J25	TX_DATA [9]–
69	H26	RX_DATA [10]+	70	K27	TX_DATA [10]+
71	J26	RX_DATA [10]–	72	L26	TX_DATA [10]–
73	G27	RX_DATA [11]+	74	J28	TX_DATA [11]+
75	F27	RX_DATA [11]–	76	J27	TX_DATA [11]–
77	G28	RX_DATA [12]+	78	G29	TX_DATA [12]+
79	H28	RX_DATA [12]–	80	F29	TX_DATA [12]–

•

Table 19. Pinout – ATA Connector to PCI / PCIe FPGA on LX, GS, SS

ATA	LX	GS	SS	Function / Notes	ATA	LX	GS	SS	Function / Notes
01	M19	C10	D39	0+(other half is on pin 25)	02	–	–	–	ground
03	F16	E10	E39	1+	04	F17	F10	H36	1–
05	G16	C11	F39	2+	06	H16	D11	J36	2–
07	E17	F11	G39	3–	08	E18	E11	G36	3+
09	G17	H11	H39	4–	10	G18	G11	K36	4+
11	H18	J12	J39	5–	12	J18	H12	L36	5+
13	K19	D14	K39	6–	14	K18	C14	M37	6+
15	L20	E14	L39	7+	16	L19	F14	P37	7–
17	M18	C15	M39	8+	18	N18	D15	R37	8–
19	–	–	–	ground	20	–	–	–	key pin (missing, to allow keying of mating connector)
21	M24	F15	N39	9–	22	–	–	–	ground
23	L24	E15	N38	9+	24	–	–	–	ground
25	N19	D10	G36	0– (other half is on pin 1)	26	–	–	–	ground
27	K24	E16	Y39	10+	28	L25	E17	AB36	10–
29	H26	D19	AB39	11+ (other half is on pin 39)	30	–	–	–	ground
31	G27	F16	AE39	12+	32	F27	G17	AE36	12–
33	H28	D17	AH39	13–	34	G28	C17	AG38	13+ (if ATA cable has no wire for this pin, cable's host connector end is grounded)
35	F29	C19	AL39	14–	36	G29	C18	AK37	14+
37	H30	E18	AM39	15–	38	H29	D18	AL37	15+
39	J26	E19	AC35	11– (other half is on pin 29)	40	–	–	–	ground

Debug Interface

The PCI / PCIe Main Board has an 8-pin debug connector designed to drive LEDs, as shown in Table 20.

- For PCI SS, debug connector outputs are driven from a 74ACT04 hex inverter using series 200-Ω resistors (for part information, see Related Resources on page 3).
- For PCI GS and PCIe8 LX, the LED signals are driven directly to the 8-pin debug connector.

Table 20. Pinout – Debug Connector to PCI / PCIe FPGA on PCIe8 LX, PCI GS, or PCI SS

Debug		Function	LX	GS	SS
1	(current limited to 800 mA by a polysilicon fuse)	+5 V			
2		LED0	J25	G14	F37
3		LED1	K25	H14	G37
4		LED2	L26	G15	H37
5		LED3	K27	H15	J37
6		LED4	J27	F18	K37
7		LED5	J28	G18	L37
8		ground			

Time Code Interface

When used with the firmware to which the time code functionality has been added, as implemented in `msp430_serial.vhd`, the debug pins become the time code interface pins. The pinouts are shown in the user's guide for the Time Distribution board (see Related Resources on page 3).

NOTE The time code feature is not supported by the PCI SS board.

Registers

The PCI / PCIe Main Board has two memory spaces: the memory-mapped registers, and the configuration space. Expansion ROM and I/O space are not implemented.

Applications can access the board registers through the EDT API library routines, especially `edt_reg_read()` and `edt_reg_write(),` using the symbolic name listed under “Access” for each register.

NOTE This section details the configuration space and registers implemented in the EDT FPGA configuration files listed under Firmware on page 3. If your organization has developed its own FPGA configuration files, consult the appropriate department in your organization and use that information instead.

0x00 Main DMA Current Address

Access / Notes: 32-bit, read-only / EDT_DMA_CUR_ADDR

This register is copied automatically from 0x04 Main DMA Next Address after main DMA completes.

Bit	Name	Description
31–0	[no name]	The address of the current DMA, or the last used address if no DMA is currently active.

0x04 Main DMA Next Address

Access / Notes: 32-bit, read-only / EDT_DMA_NXT_ADDR

The scatter-gather DMA fills this register when required from the scatter-gather DMA list.

Bit	Name	Description
31–0	[no name]	Read the starting address of the next DMA.

0x08 Main DMA Current Count and Control

Access / Notes: 32-bit, read-only / EDT_DMA_CUR_CNT

This register is automatically copied from 0x0C Main DMA Next Count and Control after main DMA completes.

Bit	Name	Description
31–16	[no name]	Read-only versions of bits 31–16 of 0x18 Scatter-gather DMA Current Count and Control.
15–0	[no name]	The number of words still to be transferred in the current DMA

0x0C Main DMA Next Count and Control

Access / Notes: 32-bit, read-only / EDT_DMA_NXT_CNT

The scatter-gather DMA fills this register when required from the scatter-gather DMA list.

Bit	Name	Description
31–16	[no name]	Read-only versions of bits 31–16 of 0x1C Scatter-gather DMA Next Count and Control.
15–0	[no name]	The number of words still to be transferred in the current DMA.

0x10 Scatter-gather DMA Current Address

Access / Notes: 32-bit, read-only / EDT_SG_CUR_ADDR

This register is automatically copied from 0x14 Scatter-gather DMA Next Address when that register is valid and the current scatter-gather DMA completes.

Bit	Name	Description
31–0	[no name]	The address of the current DMA, or the last used address if no DMA is currently active.

0x14 Scatter-gather DMA Next Address

Access / Notes: 32-bit, read-write / EDT_SG_NXT_ADDR

The driver software writes this register (see step 2 of Scatter-gather DMA on page 12).

Bit	Name	Description
31–0	[no name]	The starting address of the next DMA.

0x18 Scatter-gather DMA Current Count and Control

Access / Notes: 32-bit, read-only / EDT_SG_CUR_CNT

The driver software can read this register for debugging or to monitor DMA progress.

Bit	Name	Description
31–16	[no name]	Read-only versions of bits 31–16 of 0x1C Scatter-gather DMA Next Count and Control.
15–0	[no name]	The number of words still to be transferred in the current DMA.

0x1C Scatter-gather DMA Next Count and Control

Access / Notes: 32-bit, read-write / EDT_SG_NXT_CNT

The driver software writes this register (see step 3 under Scatter-gather DMA on page 12).

Bit	Name	Description
31	EDT_EN_RDY	Enable scatter-gather next empty interrupt. A value of 1 enables bit 29 of this register to set bit 12 in 0xC8 PCI Interrupt Status, thus causing an interrupt if bit 15 is set in 0xC4 PCI Interrupt and UI FPGA Configuration. A value of 0 disables bit 29 from causing an interrupt.
30	EDT_DMA_DONE	Read-only. A value of 0 indicates that a scatter-gather DMA transfer is currently in progress. A value of 1 indicates that the current scatter-gather DMA is complete.

29	EDT_DMA_START	Write a 1 to this bit to indicate that the values of this register and 0x14 Scatter-gather DMA Next Address are valid; doing so sets this bit to 0, indicating either that the copy is in progress, or that the device is waiting for the current DMA transfer to complete. In either case, this register and 0x14 Scatter-gather DMA Next Address are not available for writing. Reading a value of 1 indicates that 0x1C Scatter-gather DMA Next Count and Control and 0x14 Scatter-gather DMA Next Address have been copied into 0x18 Scatter-gather DMA Current Count and Control and 0x10 Scatter-gather DMA Current Address, and that the former registers (0x1C and 0x14) are once more available for writing.
28	EDT_EN_MN_DONE	A value of 1 enables the main DMA page done interrupt (bit 18).
27	EDT_EN_SG_DONE	Enable scatter-gather DMA done interrupt. A value of 1 enables bit 30 (EDT_DMA_DONE) of this register to set bit 12 (PCD_DMA_INTR) of 0xC8 PCI Interrupt Status), thus causing an interrupt if bit 15 (PCI_EN_INTR) of 0xC4 PCI Interrupt and UI FPGA Configuration is set. A value of 0 disables bit 30 from causing an interrupt.
26	EDT_DMA_ABORT	A value of 1 stops the DMA transfer in progress and cancels the next one, clearing bits 29 and 30. Always 0 when read.
25	EDT_DMA_MEM_RD	A value of 1 specifies a read operation; 0 specifies write.
24	EDT_BURST_EN	A value of 0 means bytes are written to memory as soon as they are received. A value of 1 means bytes are saved to write the most efficient number at once.
23	EDT_MN_DMA_ DONE	Read-only. A value of 1 indicates that the main DMA is not active.
22	EDT_MN_NXT_EMP	Read-only. A value of 1 indicates that 0x04 Main DMA Next Address and 0x0C Main DMA Next Count and Control are empty.
21–19	[no name]	Reserved for EDT internal use.
18	EDT_PG_INT	Read-only. A value of 1 indicates that the page interrupt is set, enabled by bit 28 (EDT_EN_MN_DONE) of this register, and that the main DMA has completed transferring a page for which bit 16 (EDT_NXTPG_INT) was set in the scatter-gather DMA list (see Table 8). If bit 15 (EDT_PCI_EN_INTR) is enabled in 0xC4 PCI Interrupt and UI FPGA Configuration, this bit causes a PCI interrupt. Clear this bit by disabling bit 28 (page done interrupt) in this register.
17	EDT_CURPG_INT	Read-only. A value of 1 indicates that bit 16 was set in the scatter-gather DMA list entry for the current main DMA page (see Table 8).
16	EDT_NXTPG_INT	Read-only. A value of 1 indicates that bit 16 (see Table 8) was set in the scatter-gather DMA list entry for the next main DMA page.
15–0	[no name]	The number of bytes in the next scatter-gather DMA list.

0x20 PLL Programming

Access / Notes: 8-bit, read-write / EDT_SS_PLL_CTL

Used by the program `set_ss_vco` to program the serial interface of the four PLLs.

Bit	Name	Description
7	PLL_SCLK	Connected to all four PLL serial clock inputs.
6	PLL_DATA	Connected to all four PLL serial data inputs.
5–4	[no name]	Not used.
3–0	PLL_STROBE	Connected to the strobe inputs of PLL 3-0, respectively.

0x7F Board ID

Access / Notes: 8-bit read-write / EDT_BOARDID

Used to identify EDT mezzanine boards. A value of 0x2 in the lowest four bits indicates an extended board ID, hard-wired into a nonvolatile complex programmable logic device (CPLD). The `extbdid` application seeks the identifier in the board ID register; if it finds a value of 0x2, then it seeks the extended board ID from the CPLD instead.

Bit	Access	Name	Description
7–4	RW	[no name]	Used by `extbdid.exe`.
3–0	R only	BOARD_ID	See the table below for details on EDT board ID and extended board ID (CPLD).

Table 21. EDT Board ID and Extended Board ID (CPLD)

Bd ID Register, Bits 3–0	Ext. Bd ID	Mezzanine Board	* Main Boards It Works With	Detail
0 0 0 0 0x0	–	RS422	LX, GS, SS	–
0 0 0 1 0x1	–	LVDS	LX, GS, SS	–
0 0 1 0 0x2	–	Reserved	–	For extended board IDs (below).
– – – – –	0x0A	SRXL	LX, GS, SS	–
– – – – –	0x10	16TE3	LX, GS, SS	–
– – – – –	0x11	OC192	LX, GS	–
– – – – –	0x12	3x3G	LX, GS, SS	–
– – – – –	0x13	MSDV	LX, GS, SS	–
– – – – –	0x14	SRXL2 (rev01 & 02)	LX, GS, SS	Contact EDT to exchange for later revision.
– – – – –	0x15	Net10G	LX, GS	–
– – – – –	0x16	DRX	AMC	–
– – – – –	0x17	DDSP	LX, GS, SS	–
– – – – –	0x18	SRXL2 (rev03+)	LX, GS	For the IDM + LBM option.
– – – – –	0x19	SRXL2 (rev03+)	LX, GS	For the IDM + IDM option.
– – – – –	0x1A	SRXL2 (rev03+)	LX, GS	For the IMM + IMM option.
– – – – –	0x1B	SRXL2 (rev03+)	LX, GS	For the IMM + LBM option.
– – – – –	0x1C	SRXL2 (rev03+)	LX, GS	For the IDM + IMM option.
– – – – –	0x1D	DRX16	LX, GS	For the IDX + IDX option.
– – – – –	0x1E	OCM2.7G	AMC	–
– – – – –	0x1F	3P	LX	–
0 0 1 1 0x3	–	Reserved	–	–
0 1 0 0 0x4	–	SSE	LX, GS, SS	–
0 1 0 1 0x5	–	HRC	LX, GS, SS	For E4, STS3, STM1 / OC3 I/O.
0 1 1 0 0x6	–	OCM	LX, GS, SS	–
0 1 1 1 0x7	–	Combo 2	LX, GS, SS	For LVDS I/O.
1 0 0 0 0x8	–	ECL/LVDS-E/RS422-E	LX, GS, SS	For ECL, LVDS, RS422, E1/T1 I/O.
1 0 0 1 0x9	–	TLK1501	[Legacy]	–
1 0 1 0 0xA	–	Reserved	–	–
1 0 1 1 0xB	–	Combo 3	LX, GS, SS	For RS422 I/O.
1 1 0 0 0xC	–	Combo 3	LX, GS, SS	For LVDS I/O.
1 1 0 1 0xD	–	Combo 3	LX, GS, SS	For ECL I/O.
1 1 1 0 0xE	–	Combo 2	LX, GS, SS	For RS422 I/O.
1 1 1 1 0xF	–	Combo	LX, GS, SS	For ECL I/O.

* LX = PCIe8 LX / FX; GS = PCI GS; SS = PCI SS; AMC = AMC FX5

0x80 Flash Memory Address

Access / Notes: 32-bit, read-write / EDT_FLASHROM_ADDR

Use with 0x84 Flash Memory Data to update the flash memory.

Bit	Name	Description
31–25	[no name]	Reserved for EDT internal use.
24	[no name]	A value of 1 causes the data in the flash memory data register to be written to the address specified by bits 0 through 23. A value of 0 reads the data.
23–0	[no name]	Address of location in flash memory that the next read or write will access.

0x84 Flash Memory Data

Access / Notes: 32-bit, read-write / EDT_FLASHROM_DATA

Use with 0x80 Flash Memory Address to update the flash memory.

Bit	Name	Description
31–9	[no name]	Not used.
8	[no name]	A read-only bit indicating the position of the jumper that enables access to the protected area of the flash memory that contains the executable program. A value of 1 indicates that the main board can load a new program.
7–0	[no name]	The new program to load into flash memory with a write operation, or the data that was read (specified, respectively, by setting or clearing bit A24 in 0x80 Flash Memory Address).

0xC4 PCI Interrupt and UI FPGA Configuration

Access / Notes: 32-bit, read-write / EDT_REMOTE_OFFSET

To program the UI FPGA:

1. Clear bits 19 and 18 (EDT_RMT_PROG and EDT_RMT_INIT).
2. Wait for bit 20 (EDT_RMT_DONE) to be clear.
3. Set bits 19 and 18.
4. Loop until bit 21 (EDT_RMT_INIT state) is set.
5. Wait four microseconds.
6. Set bit 17 (EDT_EN_CCLK) and write program data, one bit at a time, to bit 16 (EDT_RMT_DATA).
7. After all data is written, keep writing 1's to bit 16 until bit 20 (EDT_RMT_DONE) is set.

The programming has failed if it has not completed after 32 clock cycles.

Bit	Name	Description
31–22	[no name]	Not used.
21	EDT_RMT_STATE	UI FPGA INIT pin state. This bit is read-only.
20	EDT_RMT_DONE	UI FPGA DONE pin.
19	EDT_RMT_PROG	UI FPGA PROG pin.
18	EDT_RMT_INIT	UI FPGA INIT pin.
17	EDT_EN_CCLK	Enable one configuration clock cycle to UI FPGA.
16	EDT_RMT_DATA	UI FPGA program data.
15	EDT_PCI_EN_INTR	Enable PCI interrupt.

14	EDT_RMT_EN_INTR	Enable UI FPGA interrupt.
13–9	[no name]	Not used.
8	EDT_RFIFO_ENB	After the UI FPGA has been programmed to your satisfaction: Set and then clear bit 3 (CMD_EN) of the 0x00 Command register in the UI FPGA; then set this bit (bit 8) to enable the burst data FIFO.
7	[no name]	Not used.
6–0	EDT_RMT_ADDR	128-byte address of the appropriate UI FPGA register.

0xC8 PCI Interrupt Status

Access / Notes: 32-bit, read-only / EDT_DMA_STATUS

The driver uses this register initially to determine the source of a PCI interrupt.

Bit	Name	Description
31–16	[no name]	Not used.
15	PCD_PCI_INTR	PCI interrupt. When asserted, the main board is asserting an interrupt on the PCI bus.
14	[no name]	Not used.
13	PCD_RMT_INTR	UI FPGA interrupt. When asserted, the UI FPGA interrupt is set. If bits 14 and 15 of 0xC4 PCI Interrupt and UI FPGA Configuration are asserted, the UI FPGA causes a PCI interrupt.
12	PCD_DMA_INTR	End of DMA interrupt. Asserted when at least one of the DMA interrupts is asserted in 0x1C Scatter-gather DMA Next Count and Control. Causes a PCI interrupt if bit 15 of 0xC4 PCI Interrupt and UI FPGA Configuration is enabled.
11–0	[no name]	Not used.

0xCC UI FPGA Data

Access / Notes: 32-bit, read-write / EDT_REMOTE_DATA

The driver uses this register initially to determine the source of a PCI interrupt.

Bit	Name	Description
31–8	[no name]	Not used.
7–0	[no name]	Read or write data for UI FPGA, using the address specified in bits 0–6 (EDT_RMT_ADDR) of 0xC4 PCI Interrupt and UI FPGA Configuration.

Revision Log

Below is a history of modifications to this guide.

Date	Rev	By	Pp	Detail
20110314	02b	PH,TL	15,17	Updated Power section and P1 pinout table to show that P1, pin 2 is +12 volts for the PCIe8 LX main board.
20100902	02a	PH,TL, RH	2	Under “DMA Interface and Requirements,” added this paragraph: “Starting with PCIe 2.0 and depending on your BIOS, a PCIe bus ‘maximum payload size’ setting may be available in your CMOS setup pages (consult your computer manufacturer’s documentation for availability and usage). If so, EDT recommends a setting of 256 bytes to increase maximum bandwidth up to 2-3% over the bandwidth associated with a typical default setting of 128. However, if your PCIe bus is running another device with a smaller maximum payload size, this setting may adversely affect that device’s performance.”
20100902	02a	PH	29-31	Reordered certain registers correctly, according to their hexadecimal addresses.
20100902	02a	PH,TL	All	Corrected product name to PCIe8 LX (without FX / SX appellations).
20100600	02	PH,TL	All	Incorporated changes from major product update, including new SX/FX FPGAs.
20070000	01	LW	All	Created new guide.