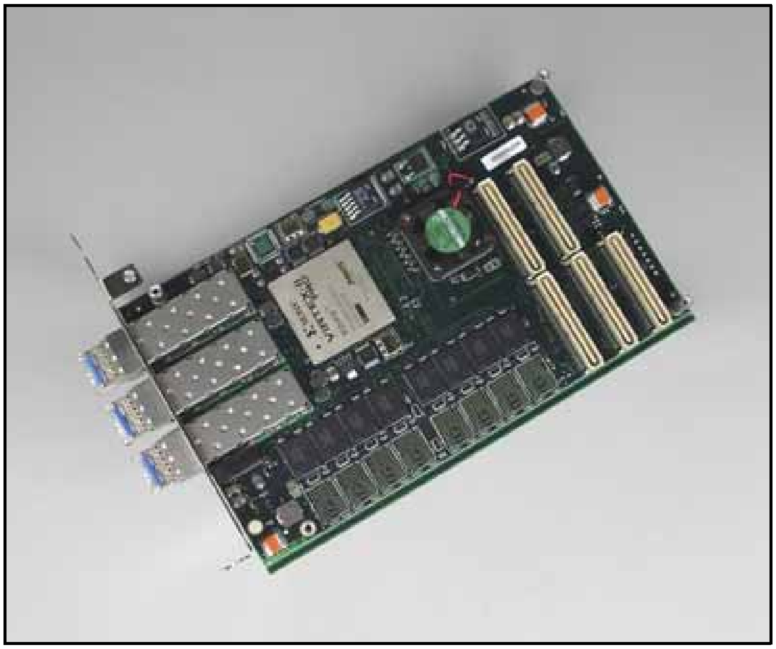
User's Guide

3x3G

Mezzanine Board



Three channels @ 1 GbE (or 3 Gb/s serial data) for use with a PCI / PCIe Main Board

Doc. 008-02748-04
Rev. 2011 Aug 17

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

3x3G Mezzanine Board

Overview

The 3x3G Mezzanine Board is an interface that pairs with an EDT PCI / PCIe main board to provide three bidirectional channels, each supporting either serial data (3.125 Gb/s) or 1GbE.

Transceivers and Supported Data Formats

EDT's factory-loaded default firmware allows either channel 1 or channel 0 to be configured to receive and transmit full-bandwidth 3.125 Gbit signals using the optional 2 GB memory buffer. Channel 2 can receive at 2.5 Gb/s, or 3.125 Gb/s with different firmware (for details, contact EDT). Only one channel at a time can use the memory buffer; to change the default from channel 0, use register 0x90 DDR Memory Control.

The 3x3G has three channels (numbered as shown in Connector Pinout on page 8). Each channel has one connector with its own small form pluggable (SFP) transceiver.

NOTE Although the manufacturer claims that transceivers are hot-swappable, we recommend powering down the board, if at all possible, before changing any transceiver.

Each SFP socket can be plugged with an SFP supporting any one of these options:

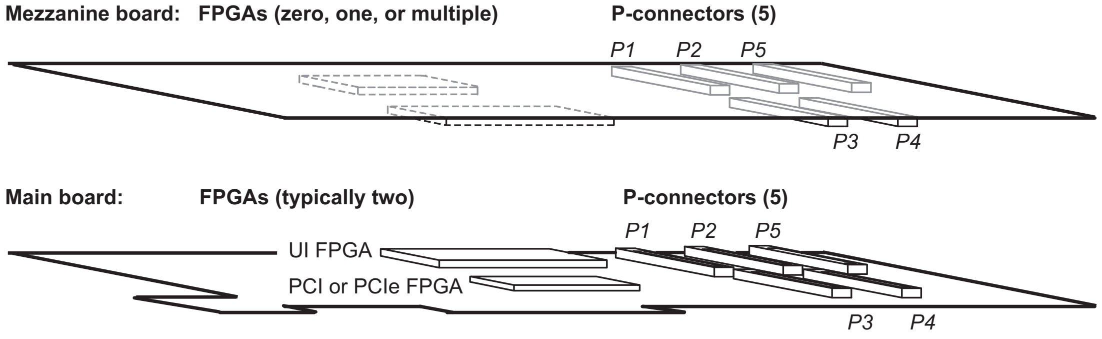
- 3.125 Gb/s serial data (via LC connector) – for normal or extended range (850 nm or 1310 nm);
- 1GbE fiber-optic (via LC connector) – for normal or extended range (850 nm or 1310 nm);
- 1GbE electrical (via RJ45 connector). Before installing the board in the host computer, remove these SFPs (taking note of their positions); then, after installation, plug each SFP back into its original socket.

For part information, see Related Resources on page 2.

FPGAs: Mezzanine Board + Main Board

In general, an EDT board pair has various FPGAs, as shown in Figure 1.

Figure 1. Generic EDT Board Pair



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

In specific, your 3x3G board pair has the following FPGAs.

- The 3x3G mezzanine board has one user-programmable FPGA – the *3x3G FPGA.* For loading instructions, see Installation on page 3.
- The PCI / PCIe Main Board has two FPGAs:
  - — The *user interface (UI) FPGA* links the 3x3G FPGA to the main board's PCI or PCIe FPGA. For loading instructions, consult the PCI / PCIe User's Guide (see Related Resources on page 2).
  - — The *PCI or 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, with the correct firmware from the main board's FPGA configuration flash memory ("flash memory").

The FPGAs will be discussed in more detail throughout the rest of this guide.

Related Products

The 3x3G is designed to work with a variety of other EDT products, including (but not limited to):

- A PCI / PCIe Main Board (PCI SS, PCI GS, or PCIe8 LX / FX), for DMA and other resources
- A Time Distribution Board, for precise timestamping of the data.

For details on these and other products of interest, see Related Resources on page 2.

Related Resources

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

EDT Resources

Description	*Detail*	*Web link*
• 3x3G specifications	Datasheet	www.edt.com (search for product)
• PCI / PCIe Main Board information	Datasheet and user's guide	www.edt.com/main_boards.html
• Time Distribution Board information	Datasheet and user's guide	www.edt.com/timedist.html
• Application Programming Interface	HTML and PDF versions	www.edt.com/manuals.html
• Installation packages: Windows, Linux, Solaris, Mac	Software / firmware downloads	www.edt.com/software.html

Parts

Description	*Part number*	*Manufacturer*	*Web link*
• 3x3G FPGA	XC2VP7 or XC2VP30	Xilinx	www.xilinx.com
• SFP (fiber, 850 nm)	FTLF8524P2xNV	Finisar	www.finisar.com
• SFP (fiber, 1310 nm)	FTLF1321P1BTL	"	"
• SFPs - electrical	FCMJ-8521-3	"	"

Installation

To install the 3x3G, fit the fiber-optic connectors through the host back panel and then plug into the PCI / PCIe connector. The fiber-optic connector furthest from the PCI / PCIe connector is channel 0, and the closest is channel 2; for details, see Figure 2 under Connector Pinout on page 8.

About the Software and Firmware

The 3x3G comes with firmware files to configure the PCI FPGA (having the extension `.bit`) for the 1-, 4-, or 16-channel DMA interfaces, as well as a variety of example applications and utilities, discussed below.

The following firmware files are specific to the 3x3G:

`3x3g.bit`	Configures the UI FPGA on the main board to communicate with the 3x3G mezzanine board.

For all board configurations, the `pcd_config` subdirectory contains sample configuration files, including:

`3x3g.cfg`	Configuration file for `initpcd` to use to configure the 3x3G.

The firmware file names you see in the top-level PCD directory do not match the file names given above because PCI bus slots come in two varieties: those supplying 3 V power, and those supplying 5 V power. Different firmware is required for the two kinds of slots, but the correct firmware file is chosen automatically when you run `pciload` or any other EDT-supplied firmware loading utility.

For example, you may see files named `cda16_3v.bit` and `cda16_5v.bit`, but the correct argument to supply to load the firmware is `cda16.bit`.

In some cases, you may also see additional firmware files incorporating changes required for various board revisions, or files with the same name in different subdirectories. You need not be concerned with any of these variations of name or path, however. In all cases, the names given above are the correct arguments to supply to the firmware-loading utilities.

The PCD Device Driver

The PCD device driver is the software running on the host that allows the host operating system to communicate with the 3x3G. 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; the installation script handles those details for you, automatically installing the correct device driver in the correct operating system-specific manner.

FPGA Configuration Files

FPGA configuration files define the firmware required for the PCI FPGA and the UI FPGA. The PCI FPGA firmware files are in the `flash` subdirectory of the top-level PCD directory. UI FPGA firmware files are in the `bitfiles` subdirectory of the top-level PCD directory.

Each FPGA must be loaded with the firmware specific to the chosen interface, and the firmware in one FPGA must be compatible with the firmware in the other. By default, the correct FPGA configuration file for the PCI FPGA is loaded atthe factory. However, you'll need to load the required FPGA configuration file for the UI FPGA yourself.

The firmware files specific to your 3x3G are listed at the beginning of this section. Instructions for loading them are provided in Configuring the 3x3G.

Software Initialization Files

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

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. Some software initialization files may also load an FPGA configuration file into an FPGA residing on the mezzanine board.

A variety of software initialization files are included with the EDT software, at least one of which is customized for each main board or main board / mezzanine board combination — that is, each FPGA configuration file has a matching software initialization file. Software initialization files are located in the pcd_config subdirectory of the top-level PCD directory. The software initialization files specific to your 3x3G are listed at the beginning of this section. Instructions for their use are provided in Configuring the 3x3G.

Commands defined in initpcd and typically found in software initialization files allow for specific FPGA configuration files to be loaded (for example, bitfile:), write specified hexadecimal values to specified registers (for example, command_reg:), enable or disable byte-swapping or short-swapping to accommodate different operating systems' requirements for bit ordering (for example, byteswap:), or invoke arbitrary commands (for example, run_command:). For example:

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

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 and Utilities

Along with the driver, the FPGA configuration files, and the software initialization files, the software CD includes a number of applications and utilities that you can use to initialize and configure the board, access registers, or test the board. 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 described below; see the README file for the complete list.

NOTE To build a new application, we recommend downloading the latest EDT installation package (see Related Resources). To rebuild an existing application, avoid version issues by using the same package used to build it, or relink / recompile the application using our latest download package.

Sample 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` An example of a variety of DMA-related operations, including reading the data from the connector interface and writing it to a file, as well as measuring input rate.

`simple_putdata` An example of a variety of 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

`set_ss_vco` A utility for programming the output clock or clocks on the 3x3G to specific frequencies used by the UI FPGA for input and output.

Utility Files

`initpcd` A utility for initializing and configuring the 3x3G.

`pdb` A utility that enables interactive reading and writing of the UI FPGA registers.

Basic Testing Files

EDT provides files to perform basic tests, such as verifying proper installation, on your EDT board (for details, see Basic Testing). These files include at least:

`sslooptest` Tests most EDT boards. Determines the board model and runs the correct loopback test.

Building or Rebuilding an Application

Executable and PCD source files are in the top-level EDT PCD directory. To build or rebuild an application, therefore, run `make` in this directory.

Windows and Solaris users must install a C compiler. For Windows, we recommend the Microsoft Visual C compiler; for Solaris, the Sun WorkShop C compiler. Linux users can use the `gcc` compiler typically included with your Linux installation. If Solaris or Windows users wish to use `gcc`, contact `tech@edt.com`.

After you've built an application, use the `--help` command line option for a list of usage options and descriptions.

Configuring the 3x3G

To ensure proper functioning, all EDT boards must be loaded with the correct FPGA configuration files for their FPGAs. At powerup, the PCI / PCIe FPGA is loaded automatically with the correct file from flash memory; however, you must load the UI FPGA yourself. Before doing so, you may wish to check the firmware in the PCI / PCIe FPGA to ensure that it is correct and up to date.

Checking or Updating the PCI / PCIe FPGA Firmware

When upgrading to a new device driver, or switching to a FPGA configuration file with special functionality, you may need to reprogram the PCI / PCIe interface flash memory using `pciload`.

NOTE The presence of a newer version of the firmware with a new driver does not necessarily mean that the firmware must be updated; the README file will tell you if a package contains a mandatory upgrade.

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

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

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

```
pciload
```

The program will output the date and revision number of the firmware in the flash memory.

To compare the PCI / PCIe FPGA firmware in the package with what is already loaded on the board, enter:

```
pciload verify
```

If they match, there's no need to upgrade the firmware. If they differ, you'll see error messages. This does not necessarily indicate a problem; if your application is operating correctly, you may not need to upgrade the firmware. However, if you do wish to update the standard firmware, enter:

```
pciload update
```

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

   ```
   pciload firmware_filename
   ```

   ...replacing *`firmware_filename`* with the name of the PCI / PCIe Main Board FPGA configuration file, with or without the `.bit` file extension.

NOTE 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), as shown:

```
pciload -u unit_number filename
```

2. At the prompt, press Enter to confirm the loading operation. (If the file date is older than the date of the file already in 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 Updating the firmware requires cycling power, not simply rebooting.

For a list of all `pciload` options, enter:

```
pciload --help
```

Loading the UI FPGA Firmware and Configuring the 3x3G

The utility `initpcd` loads the UI FPGA configuration files, programs the registers, sets the clocks (if necessary), and gets the 3x3G ready to perform DMA. This utility takes, as an argument, a software initialization file, and then automatically runs the pertinent commands.

If you use `initpcd` to configure the 3x3G, your application can concern itself solely with performing DMA and other application-specific operations; it will therefore omit 3x3G-specific operations and be portable to other EDT boards that peform DMA.

To configure the 3x3G, 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 About the Software and Firmware; for example:

```
initpcd -f 3x3g.cfg
```

NOTE Software initialization files are editable text files. If the files provided do not meet your needs, copy and modify the one that's closest 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.

Loading the UI FPGA via the Custom Utility

In the instructions below, replace the italicized placeholders with the values required in your own case.

To load the UI FPGA in the 3x3G and configure the channels, enter:

```
3x3gload -u unit_number
```

To configure the board by unit number and channel specifically, and specifying any firmware file you wish:

```
x3gload -u unit_number -c channel_number filename
```

The `3x3gload` program detects and loads the main board UI FPGA with `x3g.bit` if it is not already loaded. If it is already loaded and you want to reload it, use the `bitload` utility:

```
bitload -u unit_number x3g
```

To help you get started writing your application, the EDT installation package contains source files and executables for three example applications:

`rd16` Creates a ring buffer, then captures 100 buffers of 128 KB each from the specified EDT board and channel and writes it to the specified file. Invoke this application thus:

```
rd16 -u unit_number -c channel_number -o output_file
```

`wr16` Reads data from the buffers and writes it to the the specified EDT board and channel. Invoke this application thus:

```
wr16 -u unit_number -c channel_number
```

`wrfile` Reads data from the specified input file and writes it to the specified EDT board and channel. Invoke this application thus:

```
wrfile -u unit_number -c channel_number -i input_file
```

NOTE Configure the 3x3G as described in Configuring the 3x3G on page 5 before running these applications.

Basic Testing

The loopback test determines the board configuration, loads the appropriate FPGA configuration file, generates test data, and tests the board and its components with no external device connected. Test files are included; see About the Software and Firmware on page 3.

NOTE The loopback test overwrites the FPGA configuration file in the UI FPGA. Before you can use the board again, you'll need to reconfigure it after the test has completed.

To perform the loopback test:

1. Leave the board pair (main + mezzanine board) in the host computer, but disconnect any external device and its cable.
2. In a command window, enter:

   ```
   sslooptest -u unit number
   ```

   The outcome will vary for each type of board. Errors are redirected to the file `sslooptest.err` in the current directory; if no such file exists, the test completed without errors.

   Loopback test output for a functional board contains lines such as:

   ```
   Total errs=0 bufs=4000; Channel errs(NNx) bufs(YYx)
   ```

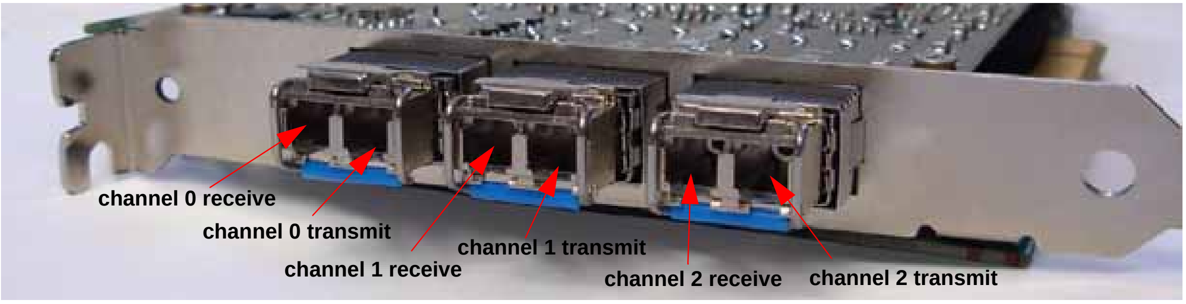
   `Total errs` shows the error count so far; `bufs` shows the number of buffers in use. The three characters after `Channel errs` show the absence (`N`) or presence (`Y`) of a data error in a specific channel (0–3); an `x` indicates a channel is not in use.

   Similarly, a `Y` after `Channel... bufs` shows a buffer in use; an `x`, that the corresponding channel is not in use. An `N` indicates that DMA is not occurring in a specific channel.
3. After the test, reconfigure the board using `initpcd` (or your own application) to disable loopback.
4. Reconnect the board to the external device.

Connector Pinout

The fiber-optic transceivers connect to the channels as shown in Figure 2.

Figure 2. Connector Pinout



Registers

This section describes the registers for the 3x3G.

0x00 Command

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

Bit	Access	Name	Description
7–4	–	[no name]	Not used.
3	RW	CMD_EN	Set to enable the FPGA configuration file. Clear to reset all channels, flush the FIFOs, and clear all under- and overflow bits.
2–0	–	[no name]	Not used.

00x0F Configuration

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

Use this register and 0x16 Least Significant Bit First to control how the data is ordered.

This register implements swapping. The structure of a 32-bit data word without swapping is:

short 1																short 0															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
byte 3								byte 2								byte 1								byte 0							

If bit 3 (SSWAP) is set, short 0 precedes short 1. If bit 0 (BSWAP) is set, byte 2 precedes byte 3, and byte 0 precedes byte 1. If both are set, then the bytes are ordered as: 0, 1, 2, 3.

Bit	Access	Name	Description
7–4	–	[no name]	Not used.
3	RW	SSWAP	Short swap. Swaps the two 16-bit short words in a 32-bit data word – so that short 1 precedes short 0. Does not change the order of the bits within each short.
2–1	–	[no name]	Not used.
0	RW	BSWAP	Byte swap. Swaps bytes 0 and 1, and also bytes 3 and 4, in a 32-bit data word – so that the bytes are ordered as: 1, 0, 3, 2. Does not change the position of the bits within each byte.

0x10 Channel Enable

Access / Notes: 8-bit read-write / SSD16_CHEN

Bit	Access	Name	Description
7–3	–	[no name]	Not used.
2–0	RW	CH_ENABLE[2–0]	Set a bit to enable the corresponding I/O channel. Clear a bit to reset the respective channel.

0x16 Least Significant Bit First

Access / Notes: 8-bit read-write / SSD16_LSB
Use this register and 00x0F Configuration to control how the data is ordered.

Bit	Access	Name	Description
7–3	–	[no name]	Not used.
2–0	RW	LSB_FIRST[2–0]	When set, the least significant bit of the 32-bit data word is first, and the most significant bit is last. When clear for a channel, the most significant bit of a 32-bit word is first.

0x1A Overflow

Access / Notes: 8-bit read-write / SSD16_OVER

Bit	Access	Name	Description
7–3	–	[no name]	Not used.
2–0	RW	OVERFLOW[2–0]	A value of 1 in a bit indicates that the corresponding channel's internal FIFO has probably overflowed since the previous CMD_EN or CHANNEL_ENABLE. (Because reading the FIFO occurs asynchronousy with respect to writing, it is not possible to determine with complete certainty that an overflow has occurred.) Data received while the FIFO is in overflow is discarded. Reset by first disabling, then re-enabling, the channel (see 0x10 Channel Enable).

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	Access	Name	Description
7	RW	PLL_SCLK	Connected to all four PLL serial clock inputs.
6	RW	PLL_DATA	Connected to all four PLL serial data inputs.
5–4	–	[no name]	Not used.
3–0	RW	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 1. EDT Board ID and Extended Board ID (CPLD)

Bd ID Register, Bits 3–0	Ext. BdID	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 SX; GS = PCI GS; SS = PCI SS; AMC = AMC FX5

0x80 Channel 0 SFP Status and Control

Access / Notes: 8-bit read-write / [no access name; identical to 0x83 and 0x86, except channel number]

Bit	Access	Name	Description
7	–	[no name]	Not used.
6	R only	[no name]	When set, the SFP detects transmission faults.
5	R only	[no name]	When set, the SFP detects a loss of signal (LOS).
4	R only	[no name]	When set, the SFP is present.
3–1	–	[no name]	Not used.
0	RW	[no name]	Set to disable transmit at the SFP (i.e., the laser).

0x81 Channel 0 RocketIO Control

Access / Notes: 8-bit read-write / X3G_RIOCTL_CH0 [identical to 0x84 and 0x87, except channel number]

Bit	Access	Name	Description
7	RW	RX_POLARITY	Set to swap the receiver’s differential polarity, if necessary for compatibility with transmitter.
6	–	[no name]	Not used.
5–4	RW	LOOPBACK1[1–0]	Set bit 5 to enable parallel loopback. Set bit 4 to enable serial loopback. Clear both bits for normal operation. If both bits are set, parallel loopback has priority.
3–2	–	[no name]	Not used.
1–0	RW	SET_RATE	Set the maximum output rate, as follows: 00 = oscillator rate / 4 (default) 01 = oscillator rate / 2 10 = oscillator rate 11 = oscillator rate x 2 A slower data rate may be needed to stay within the PCI / PCIe bus maximum bandwidth. In the default firmware, channels 0 and 1 use oscillator 1, and channel 2 uses oscillator 0.

0x82 Channel 0 RocketIO Status

Access / Notes: 8-bit read-only / X3G_RIOSTAT_CH0 [identical to 0x85 and 0x88, except channel number]

Bit	Access	Name	Description
7–4	–	[no name]	Not used.
3	R only	DCM_STAT	When set, indicates the digital clock manager is locked.
2	–	[no name]	Not used.
1–0	R only	CH_LOS[1–0]	Indicates receiver is not synchronized with input signal: When set, bit 0 indicates receiver is trying to lock to input signal. When set, bit 1 indicates loss of synchronization.

0x83 Channel 1 SFP Status and Control

Access / Notes: 8-bit read-write / [no access name – identical to 0x80 and 0x86, except channel number; see 0x80 Channel 0 SFP Status and Control

Bit	Access	Name	Description
7	–	[no name]	Not used.
6	R only	[no name]	When set, the SFP detects transmission faults.
5	R only	[no name]	When set, the SFP detects a loss of signal (LOS).
4	R only	[no name]	When set, the SFP is present.
3–1	–	[no name]	Not used.
0	RW	[no name]	Set to disable transmit at the SFP (i.e., the laser).

0x84 Channel 1 RocketIO Control

Access / Notes: 8-bit read-write / X3G_RIOCTL_CH1 [identical to 0x81 and 0x87, except channel number; see 0x81 Channel 0 RocketIO Control]

0x85 Channel 1 RocketIO Status

Access / Notes: 8-bit read-only / X3G_RIOSTAT_CH1 [identical to 0x82 and 0x88, except channel number; see 0x82 Channel 0 RocketIO Status]

0x86 Channel 2 SFP Status and Control

Access / Notes: 8-bit read-write / [no access name – identical to 0x80 and 0x83, except channel number; see 0x80 Channel 0 SFP Status and Control]

Bit	Access	Name	Description
7	–	[no name]	Not used.
6	R only	[no name]	When set, the SFP detects transmission faults.
5	R only	[no name]	When set, the SFP detects a loss of signal (LOS).
4	R only	[no name]	When set, the SFP is present.
3–1	–	[no name]	Not used.
0	RW	[no name]	Set to disable transmit at the SFP (i.e., the laser).

0x87 Channel 2 RocketIO Control

Access / Notes: 8-bit read-write / X3G_RIOCTL_CH2 [identical to 0x81 and 0x84, except channel number; see 0x81 Channel 0 RocketIO Control]

0x88 Channel 2 RocketIO Status

Access / Notes: 8-bit read-only / X3G_RIOSTAT_CH2 [identical to 0x82 and 0x85, except channel number; see 0x82 Channel 0 RocketIO Status]

Bit	Access	Name	Description

7–4	–	[no name]	Not used.
3	R only	DCM_STAT	When set, indicates the digital clock manager is locked.
2	–	[no name]	Not used.
1–0	R only	CH_LOS[1–0]	Indicates receiver is not synchronized with input signal: When set, bit 0 indicates receiver is trying to lock to input signal. When set, bit 1 indicates loss of synchronization.

0x8F Frequency Selection

Access / Notes: 8-bit read-write / X3G_FREQ_SEL

The default firmware assigns channels 0 and 1 to use oscillator 1, and channel 2 to use oscillator 0.

Bit	Access	Name	Description
7–6	–	[no name]	Not used.
5–4	RW	FREQ_OSC1[1–0]	Set the frequency of oscillator 1 (by default, used by channels 0 and 1), as follows: 00 = 106.25 MHz (default) 01 = 125 MHz 10 = 124.2 MHz 11 = 156.25 MHz
3–2	–	[no name]	Not used.
1–0	RW	FREQ_OSC0[1–0]	Set the frequency of oscillator 0 (by default, used by channel 2), as follows: 00 = 106.25 MHz (default) 01 = 125 MHz 10 = 124.2 MHz 11 = 156.25 MHz

0x90 DDR Memory Control

Access / Notes: 8-bit read-write / X3G_DDR_CTL

Allows the DDR memory to be used as a GB FIFO for one of the channels, permitting data snapshots at the maximum data rate. The proper initialization sequence for the memory is:

1. Clear bits 0 and 1.
2. Set bit 0.
3. Set bit 1.

Bit	Access	Name	Description
7–6	–	[no name]	Not used.
5–4	RW	CH_SELECT	Select the channel for which the 2 GB DDR memory serves as a FIFO, as follows: 00 = channel 0 (default) 01 = channel 1 10 = channel 2 11 = none
3–2	–	[no name]	Not used.
1	RW	MEMORY_INIT	Set to initialize the DDR memory; clear to reset the memory.
0	RW	MEMORY_ENABLE	Set to enable the DDR memory; clear to reset the memory.

0x91 PRBS15 Generator

Access / Notes: 8-bit read-write / X3G_PRBS15

Bit	Access	Name	Description
7–3	–	[no name]	Not used.
2–0	RW	PRBS15_EN[2–0]	Enables PRBS15 test code generation for the corresponding channel.

Revision Log

Below is a history of modifications to this guide.

Date	Rev	By	Pp	Detail
20110817	04	PH	12-14	• Added registers 0x80, 83, 86 (channels 0, 1, 2 – SFP status and control). • Split single register 0x81, 84, 87 (channels 0, 1, 2 –Rocket IO status) into three • Split single register 0x82, 85, 88 (channels 0, 1, 2 – Rocket IO control) into three.
		PH	10	• In register 0x16, bits 2–0, corrected access name to “LSB_FIRST[2–0]”.
20100817	03.10	PH	i	• Added border to product photo.
	03.10	PH	2	• Changed “Companion Products” to “Related Products” and revised text.
	03.10	PH,SB	11	• 0x7F Board ID: Added 0x1F 3P; moved it to top of page (affecting pagination).
	03.10	PH,SB	12	• 0x8F Frequency Selection: - Added “=” signs to separate the first two digits from the MHz. - Incorporated the default assignment details into bit descriptions (5–4 and 3–2).
20100600	02	PH	1	• Moved “Transceivers” section to page 1, with signal TX / RX information.
20100600	02	PH	i–2	• Updated title pages (new format). • Updated “Related Resources” (new format). • Added “FPGAs: Mezzanine Board + Main Board” section and figure.
20100610	02	PH	8	• Changed head “Testing Procedures” to “Basic Testing.”
20100610	02	PH,MM, TL,BO	3-6	Text inset “apps & utilities” – reinserted, with these changes: • Deleted “xtest” per MM. • Changed subhead “Testing Files” to “Basic Testing Files” per TL. Text inset “config” – reinserted, with these changes: • Changed “PROM” to “flash memory” & rewrote Step 2 to say “If the file date is older than the date of the file already in flash memory...” per TL / BO.
20100600	02	PH	All	• Text: Changed “Xilinx” to “FPGA” and "PCI" to “PCI / PCIe” where applicable.
20100610	02	PH	End	• Registers: - All – updated to new format - 0x7F Board ID – updated to new content - 0x0F Configuration & 0x16 LSB – updated notes & data word structure table
20100610	02	PH	End	• Added Revision Log.
20060000	00-01	LW	All	• Created new guide.