User's Guide

16TE3
Mezzanine Board

Sixteen-channel interface for E3 / T3 for use with a PCI / PCIe Main Board

Doc. 008-02007-03
Rev. 2010 June 15

EDT^{TM} and Engineering Design Team^{TM} are trademarks of Engineering Design Team, Inc. All other trademarks, service marks, and copyrights are the property of their respective owners†.

International Distributors

Sky Blue Microsystems GmbH
Geisenhausenerstr. 18
81379 Munich, Germany
+49 89 780 2970, info@skyblue.de
www.skyblue.de

In Great Britain:
Zerif Technologies Ltd.
Winnington House, 2 Woodberry Grove
Finchley, London N12 0DR
+44 115 855 7883, info@zerif.co.uk
www.zerif.co.uk

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

16TE3 Mezzanine Board

Overview

The 16TE3 Mezzanine Board has sixteen line interface units to support sixteen E3 or T3 (also called DS3) signals. The line interface units (LIUs) receive coded signals, extract the clock, and decode the original data.

- For E3 signals, the bit rate is 34.368 Mb and the error correction code is HDB3.

- For T3 signals, the bit rate is 44.736 Mb and the error correction code is B3ZS.

For details on the E3 / T3 standards and the LIUs, see Related Resources on page 2.

Each interface can be either input or output; if duplex capability is required, only eight inputs and eight outputs are available due to connector limitations. Each interface corresponds to a single, transformer-isolated signal that uses two wires – either a twisted pair or coaxial cable.

The registers map the sixteen serial interfaces to the sixteen DMA channels as either input or output; unformatted data is received into, or transmitted from, memory buffers maintained in system memory. Signal direction is switched through the PCI / PCIe FPGA on the PCI / PCIe Main Board. For details about the register controls, see Registers on page 11.

NOTE The 16TE3 does not protect against lightning or power line shorts; the signals provide only the typical industrial electrostatic discharge protection. Therefore, when cabled to external building wiring or in situations where shorts to power wires are possible, you should provide external protection.

NOTE To avoid unwanted current on the ground conductor, the coaxial E3 and T3 provide an AC coupling from the coaxial shield to the logic ground. Although some standards suggest a DC connection to logic ground, we do not recommend this, as ground levels between equipment can vary by several volts.

FPGAs on the PCI / PCIe Main Board

The 16TE3 is paired with an EDT PCI / PCIe Main Board, which has the following field-programmable gate arrays (FPGAs):

- The *user interface (UI) FPGA* links the 16TE3 to the main board's PCI or PCIe FPGA.

- 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").

For details, consult the PCI / PCIe Main Board User's Guide (see Related Resources on page 2).

Companion Products

For additional resources, the 16TE3 is designed to work with these EDT products:

- Required – a PCI / PCIe Main Board (PCI SS, PCI GS, or PCIe8 LX / FX), for DMA and other resources

- Optional – a Time Distribution Board, for precise timestamping of the data.

For details on these products, see Related Resources.

Related Resources

The resources, documents, and part information below may be helpful or necessary for your applications.

EDT Resources

Description	Detail	Web link
• 16TE3 specifications	Datasheet (on product page)	www.edt.com/16te3.html
• 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
• Line interface units (LIUs)	TDK 78P2344AJ	Teridian Semiconductor Corp. (formerly TDK)	www.tdksemiconductor.com/push_file_for_download.cfm/FDS2344JATA07v2-2.pdf?file_directory=54%5DZCGIE%29H%28V%26K%3A%3A%2DJE7%21%405VC%40%40BR%0A&file_name=54%3A%22LQ%3C08X%2B%3ERF%5BO%2A%5E%24%26LR%40%3BAEQ%5E%28%0A

Standards / Specifications

Description	Pertains to	Documentation	Web link
• International Telecommunication Union (ITU)	Characteristics & wave shape	ITU-T G.703	www.itu.int/
	Signal loss	ITU-T G.775	"
	Jitter & wander control (in digital networks based on 2048 Kb/s hierarchy)	ITU-T G.823	"
	Jitter tolerance	ITU-T G.824	"
• Alliance for Telecommunications Industry Solutions (ATIS) & American National Standards Institute (ANSI)	E3 / T3 interfaces	ANSI T1.102-1933	www.atis.org/ and www.ansi.org/

Installation

This section describes your EDT installation package and explains the installation process. To begin:

1. Install the Pcd driver software on the host computer, as instructed in the EDT installation package.

2. Plug your EDT board into the PCI / PCIe bus connector.

3. Connect the required cable to the 68-pin connector.

4. Connect the external device(s) to the cable as necessary.

About the Software and Firmware

Your EDT installation package includes these 16TE3-specific files:

`te3_16in.bit`	This VHDL FPGA configuration file, a subset of `te3_16io.bit,` is included for single-board loopback testing. It enables acquisition of unframed or raw data from the sixteen LIUs. The sixteen differential signals are treated as eight simple clock / data interfaces. Fifteen-bit pseudorandom (PRBS15) data is generated for each of the outputs.
`te3_16io.bit`	This VHDL FPGA configuration file enables input identical to the `te3_16in.bit.` Data for each output can be sourced by any of sixteen DMA channels that are not used for input.
`16te3_set_e3.c`	Configures the LIUs for E3 operation.
`16te3_set_t3.c`	Configures the LIUs for T3 operation.
`genprbs15.c`	Generates DMA data for a 15-bit pseudorandom bit sequence in the specified channels.
`chkprbs15.c`	Checks DMA data against a 15-bit pseudorandom bit sequence in the specified channels.

The PCI / PCIe FPGA on the main board requires one of these 16-channel FPGA configuration files:

`pciss16.bit`	The VHDL FPGA configuration file for the PCI / PCIe FPGA on a PCI SS board.
`pcigs16.bit`	The VHDL FPGA configuration file for the PCI / PCIe FPGA on a PCI GS board.

The subdirectory `pcd_config` includes sample configuration files for all board configurations. Files include:

`16te3.cfg`	Sample configuration file to configure the 16TE3. This is an editable text file that you can customize for your own applications.

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 16TE3. 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 at the factory. However, you'll need to load the required FPGA configuration file for the UI FPGA yourself.

The firmware files specific to your 16TE3 are listed at the beginning of this section. Instructions for loading them are provided in Configuring the 16TE3.

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 16TE3 are listed at the beginning of this section. Instructions for their use are provided in Configuring the 16TE3.

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`, send mail to `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 16TE3 to specific frequencies used by the UI FPGA for input and output.

Utility Files

`initpcd`	A utility for initializing and configuring the 16TE3.
`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 16TE3

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 configuration 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 16TE3

The utility `initpcd` loads the UI FPGA configuration files, programs the registers, sets the clocks (if necessary), and gets the 16TE3 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 16TE3, your application can concern itself solely with performing DMA and other application-specific operations; it will therefore omit 16TE3-specific operations and be portable to other EDT boards that peform DMA.

To configure the 16TE3, 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 te3_16io.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.

Checking For 16-channel Firmware

To verify that the PCI / PCIe FPGA on the main board is loaded with the correct 16-channel firmware:

1. Run `pciload` without arguments.

2. If the PROM ID includes the string `pcigs16` or `pciss16,` then 16-channel firmware is already loaded. If not, load the 16-channel firmware with the appropriate command – either:

    ```
    pciload -u unit number pciss16 or pciload -u unit number pcigs16
    ```

3. At the prompt, press Enter to confirm the loading operation.

Basic Testing

You can perform basic testing on the 16TE3 in three ways:

- The internal loopback test uses a loopback connector to test both input and output on the same board. It does so by connecting the transmits of the first eight channels to the receives of the second eight channels, and vice versa.

- The external loopback test also tests the connector pins and, with generated test data, the entire circuit.

- The board-to-board test uses two boards – one as output and one as input.

NOTE   If you wish to conduct the external loopback or board-to-board tests, contact tech@edt.com for information on constructing the loopback connecter and test cable.

Internal Loopback Test

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. For a discussion of the basic test files, see Basic Testing Files on page 5.

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 this test:

1. Leave the board in the host computer with the mezzanine board (if any) attached, but disconnect any external device and its cable.

2. In a command window, enter:

   ```
   sslooptest -u unit number
   ```

   The outcome will vary, depending on the main board and mezzanine board installed. Errors are redirected to the file `sslooptest.err` in the current directory; if no such file exists, the test was completed without errors.

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

   ```
   Total errs=0 bufs=4000; Channel errs(NNNNxNNNNNNNNNNN) bufs(YYYYxYYYYYYYYYY)
   ```

   `Total errs` shows the error count so far, while `bufs` shows the number of buffers in use. The sixteen characters after `Channel errs` show the absence (`N`) or presence (`Y`) of a data error in a specific channel (0–15). An `x` indicates that a channel is not in use.

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

3. After the test has completed, reconfigure the board using `initpcd` (or your own application) to disable loopback.

4. Reconnect the board to the external device.

External Loopback Test

To run the tests with the loopback connector:

1. Connect the 62-pin loopback connector to the board.

2. At a command prompt, enter:

```
initpcd -u unit number -f pcd_config/16te3.cfg
```

3. To test transmit from first 8 channels, receive on second 8 channels, at a command prompt, enter:

```
genprbs15 -u unit number -c 0 -n 8 -l 0
```

4. In a second command window, enter:

```
chkprbs15 -u unit number -c 8 -n 8 -l 0 -t -e ffff -N -I
```

5. To test transmit from second 8 channels, receive on first 8 channels, at a command prompt, enter:

```
genprbs15 -u unit number -c 8 -n 8 -l 0
```

6. In a second command window, enter:

```
chkprbs15 -u unit number -c 0 -n 8 -l 0 -t -e ffff -N -I
```

The output appears as shown in Internal Loopback Test on page 8.

7. Reconnect the board to the external device.

Board-to-Board Test With Cables

To test between two boards, connect a board-to-board cable between two 16TE3 boards. The boards can reside on different machines.

1. For each board, at a command prompt, enter:

```
initpcd -u unit number -f pcd_config/16te3.cfg
```

2. At a command prompt on the host for the output board, enter:

```
genprbs15 -u unit number -l 0
```

3. At a command prompt on the host for the input board, enter:

```
chkprbs15 -u unit number -l 0
```

The output appears as shown in Internal Loopback Test on page 8.

4. After the test has completed, reconnect the board to the external device.

Pinout

Table 1 shows connections made by the VHDL FPGA configuration file `te3_16io.bit`.

Table 1. Connector Pinout for E3 / T3 Signals

		E3 or T3 Channel Number:	
Positive Pin Number	**Negative Pin Number**	**Input (RX)**	**Output (TX)**
1	2	0	–
3	4	–	0
5	6	3	
7	8	–	3
9	10	6	–
11	12	–	6
13	14	9	
15	16		9
17	18	12	–
19	20	–	12
21	20	–	13
22	23	–	1
24	25	1	–
26	27	–	4
28	29	4	–
30	31	–	7
32	33	7	–
34	35	–	10
36	37	10	–
38	39	–	14
40	39	–	15
41	42	13	–
43	44	2	–
45	46	–	2
47	48	5	–
49	50	–	5
51	52	8	–
53	54	–	8
55	56	11	–
57	58	–	11
59	60	14	–
61	62	15	–

Registers

This section shows the 16TE3 registers. The following registers are implemented but not used:

- PLL 0, PLL 1, and PLL 3 Divider registers
- DMA Mapping registers

0x00 Command

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

Bit	Name	Description
7–4	[no name]	Not used.
3	CMD_EN	Set this bit, and enable the required channels in 0x10, 11 Channel Enable for DMA to occur. When clear, resets all channels, flushes the FIFOs, and clears 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, 17 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	Name	Description
7–4	[no name]	Not used.
3	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	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, 11 Channel Enable

Access / Notes: Two 8-bit read-write registers, accessible with one 16-bit write operation / SSD16_CHEN

Bit	Name	Description
15–0	CH_ENABLE	A value of one in a bit enables the corresponding channel for DMA. Channels correspond to register bits as shown in Table 2.

Table 2. How Channels Correspond to Bits in Registers

Register	Register with low address								Register with low address							
Bit number	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Channel number	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

0x12, 13 Channel Direction

Access / Notes: 16-bit read-only / SSD16_CHDIR

Bit	Name	Description
15–0	CH_DIR	A value of zero in a bit indicates input for the corresponding DMA channel; a value of one indicates output. Channels correspond to register bits as shown in Table 2.

0x16, 17 Least Significant Bit First

Access / Notes: 16-bit read-write / SSD16_LSB

Use this register and 00x0F Configuration to control the order of bits in a 32-bit word.

Bit	Name	Description
15–0	LSB_FIRST	When set for a channel, the least significant bit of the 32-bit data word is the first bit, and the most significant bit is the last. When clear for a channel, the most significant bit of a 32-bit word is the first bit.

0x18, 19 Underflow

Access / Notes: 16-bit read-only / SSD16_UNDER

Bit	Name	Description
15–0	UNDERFLOW	A value of 1 in a bit indicates that the corresponding channel's internal FIFO has underflowed since the previous CMD_EN or CHANNEL_ENABLE. Reset by first disabling, then re-enabling, the channel (see 0x10, 11 Channel Enable).

0x1A, 1B Overflow

Access / Notes: 16-bit read-only / SSD16_OVER

Bit	Name	Description
15–0	OVERFLOW	A value of 1 in a bit indicates that the corresponding channel's internal FIFO has overflowed since the previous CMD_EN or CHANNEL_ENABLE. Reset by first disabling, then re-enabling, the channel (see 0x10, 11 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	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.

0x21 Output Clock Select

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

Used to select the source for the transmit clock used by each interface.

Bit	Name	Description
7–3	[no name]	Not used.
2	T3E3_CLK	The source for the transmit clock on E3 / T3 interfaces 0–3. The default, 0, selects the source as PLL 2. A value of 1 turns off the transmit clock.
1–0	[no name]	Not used.

0x28, 29 PLL Divider

Access / Notes: 16-bit read-write / EDT_SS_PLL2_CLK and EDT_SS_PLL2_X

This register is set by `set_ss_vco`.
– For E3 operation, set PLL2 to 34.368 MHz.
– For T3 operation, set PLL2 to 44.74 MHz.

Bit	Name	Description
15–0	[no name]	A post-scalar divider used to achieve lower frequencies than those at which the PLLs can be programmed. After this division (if any), the clocks are divided by two for an even duty cycle — half the time high, and half low; the program `set_ss_vco` takes this into account.
		The program `sslooptest` uses PLL1 to set the output clock for the diagnostic PRBS15 test data generator.

0x45–48 E3 / T3 Status

Access / Notes: 32-bit read-only:

0x45 = 16TE3_STATUS_0 (addresses channels 0–3)
0x46 = 16TE3_STATUS_1 (addresses channels 4–7)
0x47 = 16TE3_STATUS_2 (addresses channels 8–11)
0x48 = 16TE3_STATUS_3 (addresses channels 12–15)

Bit	Name	Description
7–5	[no name]	Not used.
4	C3_SPO	Reads the status pin on the T3/E3 line interface unit, which is the LOS pin of the TDK 78P2344JAT LIU. Program this using the register 0x4C E3 / T3 Control.
3–0	T3E3_LOS	Reads the state of the TDK 78P2344JAT serial programming interface serial output.

0x4A, 4B E3 / T3 Enable

	Access / Notes:	16-bit read-write / 16TE3_TRANSMIT

Bit	Name	Description
15–0	EN_DATA	Set to 1 to enable transmit data for respective T3 or E3 channel.

0x4C E3 / T3 Control

	Access / Notes:	8-bit read-write / 16TE3_CONTROL

Used to set the input and output functions of the E3 / T3 LIU external pins. The TDK 78P2344JAT (for datasheet, see Related Resources on page 2) is programmed via a serial interface. Bits in this register are used by `16te3_set_e3.c` to set the LIU for E3 operation.

Bit	Name	Description
7	TE3_SCS	Controls the LIU programming chip select input.
6	TE3_SCK	Controls the LIU programming serial clock input.
5	TE3_SDI	Controls the LIU programming serial data input.
4	TE3_POR_L	Clear to place the TDK 78P2344JAT power on reset low (reset state); set to 1 to enable the LIU.
3–2	[no name]	Not used.
1–0	TDK_SEL	Determines which LIU chip to access:
		00 = LIU 0
		01 = LIU 1
		10 = LIU 2
		11 = LIU 3

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	Name	Description
7–4	[no name]	Used by `extbdid.exe`.
3–0	BOARD_ID	See EDT Board ID and Extended Board ID table (below).

Table 3. EDT Board ID and Extended Board ID (CPLD), part 1 of 2

Board ID Register, Bits 3–0	Extended Board ID	Board Name	Detail
0 0 0 0 0x0	–	RS422	–
0 0 0 1 0x1	–	LVDS	–
0 0 1 0 0x2	–	Reserved	For extended board IDs (below).
– – – – –	0x0A	SRXL	–
– – – – –	0x10	16TE3	–
– – – – –	0x11	OC192	–
– – – – –	0x12	3x3G	–
– – – – –	0x13	MSDV	–
– – – – –	0x14	SRXL2 (rev01 & 02)	Contact EDT to exchange for later revision.
– – – – –	0x15	Net10G	–
– – – – –	0x16	DRX	–

Table 3. EDT Board ID and Extended Board ID (CPLD), part 2 of 2

Board ID Register, Bits 3–0	Extended Board ID	Board Name	Detail
– – – – –	0x17	DDSP	–
– – – – –	0x18	SRXL2 (rev03+)	For the IDM + LBM option.
– – – – –	0x19	SRXL2 (rev03+)	For the IDM + IDM option.
– – – – –	0x1A	SRXL2 (rev03+)	For the IMM + IMM option.
– – – – –	0x1B	SRXL2 (rev03+)	For the IMM + LBM option.
– – – – –	0x1C	SRXL2 (rev03+)	For the IDM + IMM option.
– – – – –	0x1D	DRX16	For the IDX + IDX option.
– – – – –	0x1E	OCM2P7G	–
0 0 1 1 0x3	–	Reserved	–
0 1 0 0 0x4	–	SSE	–
0 1 0 1 0x5	–	HRC	For E4, STS3, STM1 / OC3 I/O.
0 1 1 0 0x6	–	OCM	–
0 1 1 1 0x7	–	Combo 2	For LVDS I/O.
1 0 0 0 0x8	–	ECL/LVDS-E/RS422-E	For ECL, LVDS, RS422, E1/T1 I/O.
1 0 0 1 0x9	–	TLK1501	–
1 0 1 0 0xA	–	Reserved	–
1 0 1 1 0xB	–	Combo 3	For RS422 I/O.
1 1 0 0 0xC	–	Combo 3	For LVDS I/O.
1 1 0 1 0xD	–	Combo 3	For ECL I/O.
1 1 1 0 0xE	–	Combo 2	For RS422 I/O.
1 1 1 1 0xF	–	Combo	For ECL I/O.

Revision Log

Below is a history of modifications to this guide.

Date	Rev	By	Pp	Detail
20100600	03	PH	i–2	• Updated title pages (new format). • Updated "Related Resources" (new format). • Added "FPGAs" and "Companion Products" sections.
20100600	03	PH	8	• Changed head "Testing Procedures" to "Basic Testing."
20100600	03	PH,MM, TL,BO	3-7	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	03	PH	All	• Text: Changed "Xilinx" to "FPGA" and "PCI" to "PCI / PCIe" where applicable.
20100600	03	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	03	PH	End	• Added Revision Log
2006-2007	00-02	LW	All	• Created new guide.