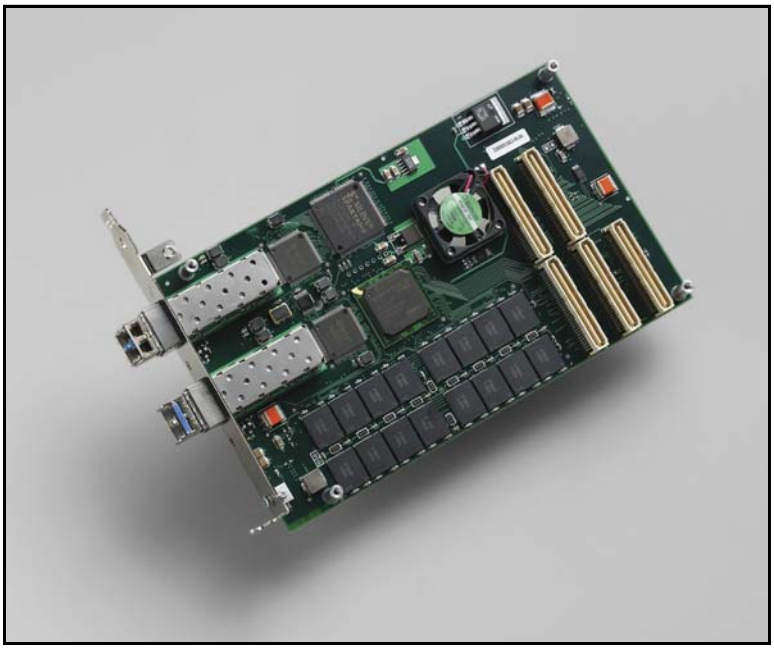
User's Guide

OCM/P

Mezzanine Board



Optical carrier multirate interface (with programmable oscillators) for use with an EDT main board

Doc. 008-02335-09b
Rev. 2013 October 28

EDT | Engineering Design Team, Inc.
3423 NE John Olsen Ave
Hillsboro, OR 97124
U.S.A.
Tel: +1-503-690-1234 | Toll free (in U.S.A.): 800-435-4320
Fax: +1-503-690-1243
www.edt.com

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

© 1997-2019 Engineering Design Team, Inc. All rights reserved.

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., 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, U.S.A.) 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

Overview 6
Physical Channels, DMA Channels, and Data Formats 6
Companion Products 7
FPGAs: Mezzanine Board + Main Board 7
Related Resources 8
Installation 9
Your EDT Installation Package 9
The PCD Device Driver 10
FPGA Configuration (.bit) Files 10
Software Initialization (.cfg) Files 10
Sample Applications and Utilities 11
Building or Rebuilding an Application 12
About ocm_snap and ocm_play 12
Configuring the OCM/P 13
Configuring the Ports 13
Loading the Firmware 14
Querying the Transceivers 15
Framing 15
OTN Operation 15
OC / STM Operation 15
Ethernet Operation 16
Basic Self-Test 16
Board Architecture 17
Initializing and Setup 17
Initializing the Clock Signals and Logic Circuits 18
Initializing Port 0 Memory 19
Other Application Setup 19
Enabling and Verifying the Input Signal 19
Initializing and Enabling the Data Paths 21
Registers 23
0x00 Command 23
0x02 Function 23
0x03 Status 23
0x05 Main Board FPGA Configuration File Organization 23
0x0F Configuration 24
0x10 Channel Enable 24
0x16 Least Significant Bit First 24
0x18 Underflow 25
0x1A Overflow 25
0x1A–1B Overflow – DE1 Firmware 25
0x20 Port 0 Configuration 0 26
0x21 Port 0 Configuration 1 26
0x22 Port 0 Status 27
0x23 Port 0 Transceiver 27
0x24 Port 0 Enable 28
0x30–34 Port 1 [See 0x20–24 Port 0] 28
0x40–43 FPGA Load 28
0x60–62 Extended Indirect Register Address 28
0x63 Extended Indirect Register Data 29
0x7C–7D Main Board FPGA Configuration File Design ID 29
0x7E Main Board FPGA Configuration File Version String 29
0x7F Board ID 30

0x80 Port 0 Receive Framing Control 31
0x81 Port 0 Transmit Framing Configuration 31
0x82 Port 0 Transmit National Byte 32
0x83 Port 0 Receive Filter Control 32
0x84–87 Port 0 Transmit Test Data 32
0x88–8A Port 0 B1 Error Count 32
0x8B Port 0 B1 Error Mask 33
0x8C–8F Port 0 B2 Error Count 33
0x90–92 Port 0 M1 Error Count Control 33
0x93 Port 0 Frame Statistics Count Control 33
0x94 Port 0 Receive Status 34
0x95 Port 0 Receive Frame Status 34
0x97 Port 0 Demux Bitmap 34
0x98 Port 0 Demux Bitmap Readback 35
0x99 Port 0 Transmit Status 35
0x9C–9D Port 0 Loss of Frame Count 36
0x9E–9F Port 0 Frame Pattern Error Count 36
0xA0 Port 0 Mezzanine FPGA Configuration File Version String 36
0xA1 Port 0 Mezzanine FPGA Configuration File Organization 36
0xA2–A3 Port 0 Mezzanine FPGA Configuration File Design ID 37
0xA4–A5 Port 0 False Frame Count 37
0xA6 Port 0 Programmable Clock Enable 37
0xA7 Port 0 Programmable Clock Status 37
0xA8 Port 0 Programmable Clock Read 38
0xA9 Port 0 Programmable Clock Register Address 38
0xAA Port 0 Programmable Clock Write 38
0xC0–EA Port 1 [See 0x80–AA Port 0] 38
Revision Log 39

OCM/P Mezzanine Board

Overview

The OCM/P is a two-port, multirate mezzanine board with programmable oscillators that supports 1GbE (electrical or optical) and SDH / SONET (STM1 / OC3 through STM16 / OC48).

The OCMP – an updated version of the OCM (as of rev30) – has these additional capabilities...

- Two programmable or settable oscillators (one per port), instead of one fixed oscillator;
- Ethernet receive-and-transmit capability on either port; and
- Optical transport network (OTN) receive-and-transmit capability on either port.

For details and signal standards, see Physical Channels, DMA Channels, and Data Formats (below) and Related Resources on page 8.

Physical Channels, DMA Channels, and Data Formats

The OCM/P has two ports, or physical channels (marked CH 0 and CH 1 on the board), each with one SFP that can support multiple data formats, and each physical channel is associated with two DMA channels – one to receive (RX) and the other to transmit (TX).

Table 1 shows the data formats supported by each port; other options may be available by request.

Table 1. Signals and SFP Options Supported on OCM/P

	RX	TX	Signals supported	Default	Optional	Notes
Port 0 ("CH 0" on board)	DMA channel 0	DMA channel 2	OTN	1310 nm	–	OTN standard*
			OC3 / STM1	1310 nm	–	–
			OC12 / STM4	1310 nm	–	–
			OC48 / STM16	1310 nm	–	Requires the optional 2 GB memory buffer if used with a PCI SS or PCI GS main board.
			1GbE (1000 BaseLX)	1310 nm	–	–
			1GbE (1000 BaseT)	–	SFP	Electrical SFP (Finisar FCMJ-8521-3)*
Port 1 ("CH 1" on board)	DMA channel 1	DMA channel 3	OTN	1310 nm	–	OTN standard*
			OC3 / STM1	1310 nm	–	–
			OC12 / STM4	1310 nm	–	–
			OC48 / STM16	–	1310 nm	Requires a PCIe8 LX main board.
			1GbE (1000 BaseLX)	1310 nm		–
			1GbE (1000 BaseT)	–	SFP	Electrical SFP (Finisar FCMJ-8521-3)*

* For details on these standards and parts, see Related Resources on page 8.

Companion Products

Some of the EDT products which are designed to work with the OCM/P mezzanine board include...

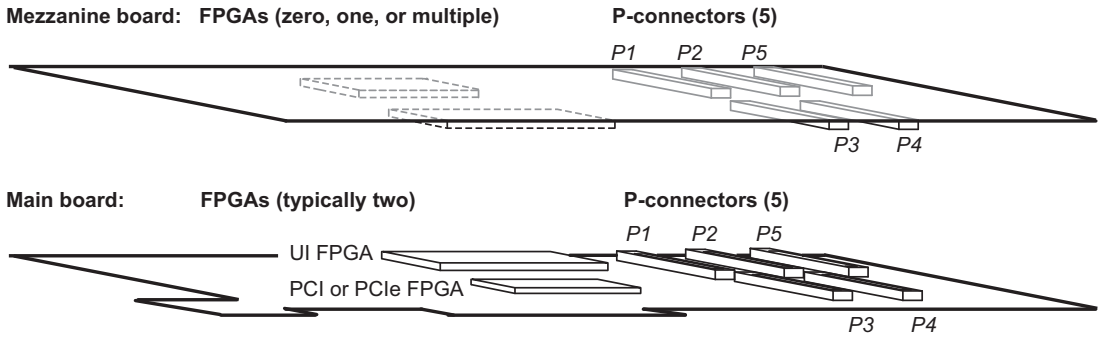
- The PCI/e Main Board (PCI SS / GS or PCIe8 LX / FX / SX) – for additional resources and DMA.
- The DE1 Configuration Package – for demultiplexing signals down to E1.
- The Time Distribution Auxiliary Board – for adding precise timestamping to your data.

The OCM/P is paired with an EDT main board (PCI/e Main Board) for high-speed DMA and other resources. The board pair provides the field-programmable gate arrays (FPGAs) below.

FPGAs: Mezzanine Board + Main Board

In general, an EDT board pair can have several 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 OCM/P board pair has the following FPGAs...

- The OCM/P mezzanine board has two user-configurable FPGAs: the port 0 FPGA and the port 1 FPGA. For loading instructions, see Installation on page 9.
- The PCI/e Main Board has two FPGAs...
  - The *user interface (UI) FPGA* links the FPGAs on the mezzanine board (OCM/P) to the main board's PCI or PCIe FPGA. Loading details are in the PCI/e Main Board user's guide (see Related Resources on page 8).
  - The *PCI/e 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").

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

Related Resources

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

EDT Resources

Description	*Detail*	*Web link*
• OCM/P specifications	Datasheet	www.edt.com/pciss_gs_ocm.html
• PCI/e Main Board documentation	Datasheet & user's guide	www.edt.com/manuals/PCD/main_boards.pdf
• DE1 Configuration Package	User's guide	
• Time Distribution board documentation	Datasheet & user's guide	www.edt.com/manuals/PCD/time-dist.pdf
• Application Programming Interface	HTML and PDF	www.edt.com/api
• Installation packages: Windows, Linux, Mac	Software & firmware downloads	www.edt.com/software.html

Standards

Description	*Pertains to*	*Documentation*	*Web link*
• International Telecom-munications Union (ITU)	Optical Transport Network (OTN)	G.709/Y.1331 03/2003	www.itu.int
• IEEE	Ethernet framing	IEEE 802-3	www.ieee802.org/3/

Parts

Description	*Part Number*	*Manufacturer*	*Web link*
• SFP (electrical)	FCMJ-8521-3	Finisar	www.finisar.com/optics/FCMJ-8521-3.php
• Line interface unit (LIU)	SLK2511	Texas Instruments	www-s.ti.com/sc/ds/slk2511.pdf

Installation

To install the OCM/P mezzanine board, follow the steps below while referring to Figure 2 on page 17.

1. First, note the position of each SFP on the mezzanine board. The two SFPs are not the same, so you need to know which one goes where.
2. Remove both SFPs to enable the board pair to be inserted into the host system.
3. Insert the board pair into the host system.
4. Working through the back panel on the host system, put each SFP back into its original position on the mezzanine board.

NOTE Before replacing transceivers, EDT recommends powering down the board, despite the manufacturer's claim that transceivers are hot-swappable.

Now you can review the resources included in the EDT installation package.

Your EDT Installation Package

Your EDT installation package provides a variety of resources, as explained throughout this section.

The subdirectory `bitfiles` includes two subdirectories...

`XC2VP4`	Contains configuration files for the port 0 FPGA.
`XC3S200`	Contains configuration files for the port 1 FPGA.

Each of those contains at least these FPGA configuration files...

`ocm12.bit`	Configures the appropriate port's FPGA for OC3 or OC12 operation.
`ethernet.bit`	Configures the appropriate port's FPGA for ethernet operation.
`ocm48.bit`	Configures the port 0 FPGA for OC48 operation.

The utility `mezzload` configures the specified port with the appropriate FPGA configuration file from either `XCV2P4` or `XC3S200`.

Your EDT installation package also includes files specifically for the OCM/P mezzanine board...

`ocm.bit`	Configures the UI FPGA on the main board to communicate with the OCM/P.
`ocmdual.bit`	Configures the UI FPGA on the main board to communicate with the OCM/P when using OC48 operation on port 1.
`ocm_snap`	Example application that captures data from the OCM/P and transfers it to disk for testing or verification.
`ocm_snap.c`	C source for `ocm_snap`.
`ocm_play`	Example application that outputs the data captured by `ocm_snap` from the disk for testing or verification.
`ocm_play.c`	C source for `ocm_play`.
`read_xfp_sfp`	Example application that queries the state of the transceiver modules. For details, see Querying the Transceivers on page 15.
`read_xfp_sfp.c`	C source for `read_xfp_sfp`.

`lib_xfp_sfp.c`	C library routines used by `read_xfp_sfp,` or available for you to use in your own application.
`lib_ocm.c`	C library routines that you can use in your OCM/P applications.
`edt_ocm.h`	Include file for the above C library routines.

Example configuration files for all board configurations are in the `pcd_config` subdirectory of your EDT installation package, including...

`ocm12.cfg`	File for `initpcd` to use to configure the OCM/P for OC3 or OC12 operation.
`ocm48.cfg`	File for `initpcd` to use to configure the OCM/P for OC48 operation.

NOTE Do not be distracted by variations of file name or path in your EDT installation package; instead, rely on the user's guide to indicate the correct file name, path, and argument for each process. If you follow the instructions in the user's guide, the automatically-loading firmware for your product will load correctly.

The PCD Device Driver

The PCD device driver is EDT software which runs on the host computer so that the host operating system can communicate with the OCM/P. The driver is loaded into the kernel upon installation, and thereafter runs as a kernel module. The driver name and subdirectory are specific to each operating system; the installation script automatically installs the correct device driver, in the correct way, for the specified operating system.

FPGA Configuration (.bit) Files

FPGA configuration (`.bit`) files define the firmware required for the FPGAs. The table below shows where these files typically are stored and loaded.

The FPGA configuration file for...	**...is stored in this subdirectory...**	**...and loaded this way...**
PCI/e FPGA functionality	`flash`	Automatically, from flash memory
UI FPGA functionality	`bitfiles/`*`FPGA_part_number`*	Manually, by the user
Any mezzanine board FPGA	`bitfiles/`*`FPGA_part_number`*	Manually, by the user

Each FPGA must be loaded with the correct firmware to ensure that it will be compatible with the other FPGAs and with your interface.

The FPGA configuration files specific to your OCM/P are listed at the beginning of this section. For instructions on loading them, see Configuring the OCM/P on page 13.

Software Initialization (.cfg) Files

Each manually-loaded FPGA configuration file is implemented by a matching software initialization (`.cfg`) file – an editable text file that runs like a script, configuring the EDT board to perform DMA.

The commands in a software initialization file are defined in the C application `initpcd.` When you invoke `initpcd,` you use the flag `-f` to specify which software initialization file to use.

Software initialization files are located in the `pcd_config` subdirectory. Those specific to your OCM/P are listed at the beginning of this section; for details on their use, see Configuring the OCM/P on page 13.

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:`). If you are using an EDT mezzanine board that has an FPGA, you can load the appropriate firmware on it by running the command `mezzload,` which determines the appropriate firmware for the specific 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 tell `initpcd` to use the 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, your EDT installation package includes application and utility files that you can use to initialize and configure the board, access the registers, and perform basic testing. In many cases, C source is provided so that you can use the files as starting points to write your own applications. The most commonly useful files 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 ring buffer input for multiple DMA channels.
`wr16`	Performs simple ring buffer output for multiple DMA channels.
`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 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`	Serves as 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 OCM/P to specific frequencies used by the UI FPGA for input and output.

Utilities

`initpcd`	A utility file for initializing and configuring the OCM/P.

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

Basic Self-Test Files

A variety of files – C source, executables, and FPGA configuration files – are available to help you perform self-tests on your EDT board (see the section Basic Testing in this guide). These files include at least...

`sslooptest` Verifies installation and basic functioning of most EDT PCI/e boards. Determines the board model and runs the appropriate loopback test for that model.

Building or Rebuilding an Application

To build or rebuild an application, use a compiler that works with your operating system, as indicated below.

- Linux users: You can use the `gcc` compiler that is typically included with the Linux installation.
- Windows users: Install a C compiler (such as Microsoft Visual C), or contact tech@edt.com to use `gcc`.
- Solaris users: Contact EDT.

To implement your build, run `make` in the top-level PCD directory, where the executables and PCD source files are located.

After your build is finished, use the `--help` command line option for a list of usage options and descriptions.

About ocm_snap and ocm_play

For an example of data capture and playback, you can invoke the command-line applications `ocm_snap` and `ocm_play` (and related C source code) with various options to customize the results.

For a Help message listing all usage options, add the `-h` flag. For example...

```
ocm_snap -h
```

The following example captures 2 GB of a STM1 / OC3 signal from DMA channel 0, port 0 (see Physical Channels, DMA Channels, and Data Formats on page 6) specifying a line rate of 1 (STM1 / OC3) and an output file size of 2 GB (2048 MB)...

```
ocm_snap -c 0 -r 1 -s 2048 -o output_file
```

...which takes approximately two minutes. When the run is completed, the companion invocation of `ocm_play` outputs the captured data to DMA channel 2, port 0 (see Physical Channels, DMA Channels, and Data Formats on page 6) at the OC3 line rate, with input from the specified file...

```
ocm_play -c 2 -r 1 -i input_file
```

...which requires a host disk system that can read and write a file at 20 MB per second. A comparable run for STM4 / OC12 data requires a disk system that can function at 80 MB per second.

The following points cover a few considerations not given in the Help message.

- To specify that the output be formatted in hexadecimal chunks of 32, 16, or 8 bits, use `ocm_snap` with the `-H, -Hw,` or `-Hb` flag, respectively. In all cases, the most significant bit is the first bit output (in time) and the leftmost bit of the chunk (in memory).
- To specify the final file size in megabytes, use `ocm_snap` with the `-s` flag. The application will terminate when the specified size has been reached.
- To change the default number or size of the ring buffers, use `ocm_snap` with the `-n` or `-b` flag, respectively. For performance reasons, the ring buffer size is always rounded to the nearest multiple of

4096. The application then checks to determine whether the requested size and number of ring buffers is reasonable for the line rate. If it is not, the application configures the ring buffers as requested, but emits a warning message. If you purchased the optional 2 GB memory buffer, use `ocm_snap` with `-M`. If you did not purchase this option, the `-M` flag will produce meaningless data.

- To skip automatic initialization (e.g., if another program has already performed initialization and you do not wish to run it again), add the `-I` flag to `ocm_snap` or `ocm_play`.

Configuring the OCM/P

Configuring the OCM/P requires these basic steps...

- Configuring the ports, and
- Loading the firmware that correlates to the main board you are using.

To implement these steps and conduct loopback testing (see Basic Self-Test on page 16), use this section. To begin data acquisition, further initialization is required (see Initializing and Setup on page 17).

Configuring the Ports

As shown below, use `mezzload` to load the FPGAs on the OCM/P mezzanine board, replacing the italicized text with the values required in your own case.

To configure port 0 for OC48 and port 1 for OC12 or OC3, enter...

```
mezzload -u unit_number
```

To configure both ports for OC12 or OC3, enter...

```
mezzload -u unit_number -n
```

To configure the board by specific unit number and port, specifying any firmware file you wish...

```
mezzload -u unit_number -c port_number filename
```

The `mezzload` program uses `pciss16test.bit` to detect and load the UI FPGA on the main board, if it is not loaded already. If it is loaded already and you want to reload it, use the `bitload` utility...

```
bitload -u unit_number ocm
```

Automatic Configuration

The `initpcd` utility takes, as an argument, a configuration file and then automatically runs the relevant command(s) from those discussed above. This utility loads the FPGA configuration files, programs the registers, sets the clocks (if needed), and prepares the OCM/P to perform DMA.

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

To configure the OCM/P, enter...

```
initpcd -u unit_number -f ocmx.cfg
```

...replacing *unit_number* with the number of the board (by default, 0), and *x* with the appropriate number (3, 12, or 48), depending on the desired configuration.

NOTE The provided configuration files are editable text files. If they do not meet your needs, copy and modify one that's similar to what you need, and then run `initpcd` with your new configuration file.

Loading the Firmware

To verify the loading of the correct FPGA configuration file for your main board (`pciss4`, `pcigs4`, or `pcie8`), follow the steps below.

1. Navigate to the directory in which you installed the driver. The default locations are...
   - For Windows, `\EDT\pcd`
   - For Linux, Solaris, or Mac: `/opt/EDTpcd`
2. At the prompt, enter...

   ```
   pciload verify
   ```

   ...to compare the current PCI/e FPGA configuration file in the package with what is in the flash memory on the OCM/P.

   If more than one board is installed on a system, specify the unit number following the `-u` option...

   ```
   pciload -u unit number verify
   ```

   If the dates and revision numbers of the flash memory and File ID are the same, there is no need for an upgrade. If they differ, proceed through the steps below to upgrade the flash memory.
3. At the prompt, enter...

   ```
   pciload update
   ```
4. Shut down the operating system; turn the host computer off, and then on again. The board reloads firmware from flash memory only during powerup. Thus, after running `pciload`, the new FPGA configuration file is not in the FPGA until the system has been power-cycled; simply rebooting is not adequate.

NOTE The provided configuration files are editable text files. If they do not meet your needs, copy and modify one that's similar to what you need, and then run `initpcd` with your new configuration file.

To see which boards are in the system, run `pciload` without any arguments...

```
pciload
```

To see other `pciload` options, run...

```
pciload help
```

Querying the Transceivers

The two transceiver modules have a two-wire serial interface that lets you query their state, including...

- The laser's transmit power, in decibels;
- The laser's receive power, in decibels; and
- The temperature of the transceiver module in degrees Celsius.

Other data also may be available; for details, see the datasheet for your transceiver module.

By default, the SFP transceiver module is disabled. To enable it, enter...

```
read_xfp_sfp -e 1
```

To disable the module, enter...

```
read_xfp_sfp -e 0
```

After the SFP transceiver module is enabled, to turn on the laser, enter...

```
read_xfp_sfp -l 1
```

To turn off the laser while leaving the module enabled, enter...

```
read_xfp_sfp -l 0
```

Framing

The OCM/P supports framing capability for multiple formats, including OTN, OC/STM, and ethernet.

OTN Operation

The OCM/P supports input and output of unformatted bits only, with no framing.

OC / STM Operation

When framing is enabled, the board searches and locks onto incoming SONET / SDH frames after detecting the presence of the A1 and A2 header patterns at 125-microsecond intervals. The algorithm sequence is...

1. *Search.* The board searches for the A1 and A2 header patterns until it sees a match; then it moves to Check.
2. *Check.* The board checks for three consecutive SONET / SDH frames at 125-microsecond intervals with the A1 and A2 header patterns in the proper position, before declaring Lock.
3. *Lock.* Once locked, incoming SONET / SDH frames are collected and forwarded to the host. The board continues to check for the A1 and A2 header patterns, and remains in this state until the A1 and A2 header patterns are lost. When the patterns are lost, it enters the Flywheel state.
4. *Flywheel.* If the A1 and A2 header patterns are not seen for three consecutive frames, the board returns to Search; if it finds them, it returns to Lock. SONET / SDH frames are collected and forwarded to the host in this state as well.

Framing headers are included in the data transferred during DMA.

Ethernet Operation

Framing conforms to the IEEE standard (for a link, see Related Resources on page 8). The data format is controlled by the status of the MAC_FILTER and DISABLE_8B10B bits in 0x80 Port 0 Receive Framing Control, as outlined below.

Table 2. Status of bits MAC_FILTER and DISABLE_8B10B

Status of MAC_FILTER	Status of DISABLE_8B10B	Result
Set	Clear	Four 8-bit symbols are packed into each 32-bit word. The control symbols for carrier extend, start of packet, and end of packet are not filtered out of the datastream. Unless `FORCE_ALL_DATA` (in 0x83 Port 0 Receive Filter Control) is set, idles are filtered out of the datastream.
Set	Set	Each 32-bit word is in one of two formats, and data is passed only on bad symbol decodes. The two formats are shown below.
Clear	Clear	Unaligned raw data bits are packed into the low 30 bits of each 32-bit word.
Clear	Set	Unaligned raw data bits are packed into all the bits of each 32-bit word.

Format 1

Bit	*Description*
31	Error (always 0)
30	Aligned (always 1)
29–28	(not used)
27–26	Alignment count
25–24	Symbol valid
23–8	Two decoded symbols
7–6	(not used)
5–4	Command symbol
3–2	Decode error
1–0	Disparity error

Format 2

Bit	*Description*
31	Error
30	Aligned
29–0	Three aligned 10-bit words

Basic Self-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 included test files, see Basic Self-Test Files on page 12.

NOTE The loopback test overwrites the FPGA configuration file in the UI FPGA. After completing the test, yo must reconfigure the board before using it again.

To perform the loopback 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 varies depending on your main-mezzanine board pair. Errors are directed 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 such lines as...

```
Total errs=0 bufs=4000; Channel errs(NNNNxxxxxxxxxxxx) bufs(YYYYxxxxxxxxxxxx)
```

`Total errs` shows the error count so far; `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 DMA channel (0–15), while an `x` indicates a DMA channel is not in use.

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

3. After the test is done, reconfigure the board using `initpcd` (or your own application) to disable loop-back.
4. Reconnect the board to the external device.

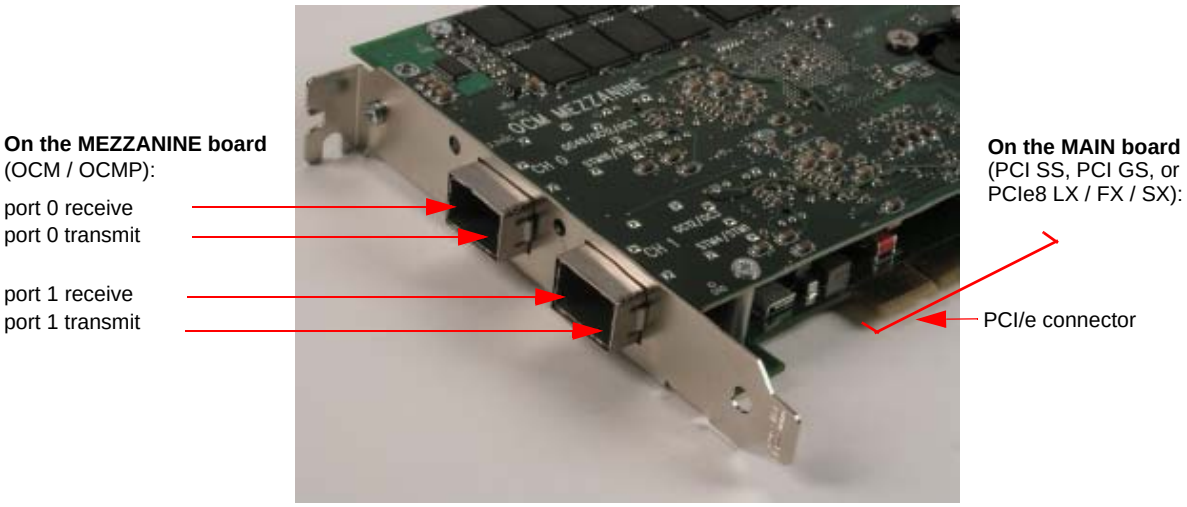
Board Architecture

The connectors on an OCM/P board pair are shown in Figure 2 below.

- The OCM/P mezzanine board (on the top) shows the SFPs and their port connections.
- The main board (on the bottom) shows the location of the PCI/e connector.

The SFP furthest from the PCI/e connector is port 0; the SFP nearest the PCI/e connector is port 1.

Figure 2. Board Architecture for OCM/P



Initializing and Setup

The example applications initialize the OCM/P by synchronizing the digital PLLs that produce the clock signals, initializing the port 0 memory (if any), enabling the LIUs, and enabling the data paths. if you're using

the example applications, this initialization is done for you. If, however, you are writing your own application for the OCM/P, the correct initialization sequence is given below.

A state transition diagram showing the initialization sequence appears in Figure 3 on page 22.

The C library routine names referenced in this procedure are prefixed with either `edt_ocx, edt_ocm,` or `edt_oc192`. In general, we recommend using the routines prefixed `edt_ocx`. These are general-purpose routines that determine whether the board installed in your host is an OCM/P or an OC192 and, when necessary, call the correct one of two parallel functions intended for the specific board model in use. If your application calls the functions prefixed by `edt_ocx,` it is more likely to be portable for use with the OC192 or future mezzanine board designs.

These routines are provided in the EDT installation package (in the main directory, in the file `lib_ocm.c` and its header file `edt_ocm.h`. They are documented in the EDT API (see Related Resources on page 8).

NOTE In the following procedure, as well as in the register descriptions, if a bit is *set*, then its value is 1; if it is *clear*, then its value is 0.

Initializing the Clock Signals and Logic Circuits

This procedure initializes basic board operations and communication between the OCM/P mezzanine board and the main board.

The OCM/P has two FPGAs — one for each port (0 and 1) — which must communicate with the UI FPGA on the main board. All communication between the boards is synchronized to the 100 MHz reference clock signal, which is initialized automatically upon main board startup but then must be distributed to all three FPGAs on the OCM/P board through the board's low-skew clock buffer.

Until these FPGAs all are synchronized to the reference clock signal, the main board cannot read or write any registers on the OCM/P mezzanine board (addresses 0x80 – 0xF0).

The C library routine `edt_ocx_base_init` determines the kind of board installed in your system, loads the main and mezzanine FPGA configuration files if necessary, and performs the steps below. It uses a `.cfg` structure to set bits 0 and 3 (SSWAP and BSWAP) in 0x0F Configuration as needed for your host, as well as to set other fields. For details on filling in the fields in this structure, consult the EDT API documentation (see Related Resources on page 8); for an example of its use, see the example application `ocm_snap.c`.

To initialize the clock signals and logic circuits...

1. Load the appropriate firmware as described in Configuring the OCM/P on page 13. Among other actions, this initializes the main board PPL that generates the 100 MHz reference clock.

NOTE If the firmware is loaded already and you do not wish to reload it, clear 0x00 Command and 0x24 Port 0 Enable by writing all zeroes.

2. To initialize the logic circuits in the UI FPGA on the main board, set bit 3 (CMD_EN) in 0x00 Command.
3. Each port on the OCM/P also has an FPGA that receives the 100 MHz reference clock signal. To synchronize incoming data, synchronize these FPGAs to the main board's UI FPGA reference clock by setting bit 2 (SYS_EN) in 0x24 Port 0 Enable.
4. To ensure that the FPGAs on the OCM/P mezzanine board have been synchronized, verify that bit 5 (SYS_LOCK) of 0x24 Port 0 Enable is set.
5. To ensure that the main board's UI FPGA has been synchronized, verify that bit 0 (LOCAL_SYS_LOCK) of 0x03 Status is set.

   At this point, insert a timeout in your application. (The library routine `edt_ocx_base_init` does so.) The bits in steps 4 and 5 should be set within approximately 10 ms of the bits in steps 2 and 3. If they are not, this probably indicates a fault in the board. In such a case, without a timeout, your application would hang. A wait of approximately half a second should be adequate.

If the bits in steps 4 and 5 still are not set, stop initialization and contact EDT.

The main board is now set up correctly; basic initialization is complete and does not need to be done again until the host computer is power-cycled.

The C library routine `edt_ocx_base_init` demonstrates initializing the clock signals and logic circuits. Steps 2 through 5 are accomplished by its calling `edt_ocx_lock_clocks`.

Initializing Port 0 Memory

If you ordered the OCM/P with optional memory for port 0, this memory also uses the 100 MHz reference clock and must be initialized after the PLLs above are synchronized. To initialize the memory...

1. Set bit 3 (RAM_EN) for port 0 in 0x24 Port 0 Enable.
2. Include a timeout for the RAM to finish initialization and for the refresh to stabilize – for example...

```
edt_msleep(10)
```

The memory is now available for use. This operation, too, need not be repeated until you restart your application.

The C library routine `edt_ocx_base_init` initializes the port 0 memory, if it is present.

Other Application Setup

At this point, your application must configure the ports for your expected data rate and reference clock. To do so, use the library routine `edt_ocx_channel_set_rate,` which writes the appropriate values to 0x20 Port 0 Configuration 0 and 0x21 Port 0 Configuration 1.

The ports can be set up independently; you can configure one while the other is acquiring data.

Before continuing initialization, now you can check for a signal coming into the fiberoptic transceiver modules. You need not do so, as the data path initialization procedure below also includes a check for an incoming signal; however, if you wish to check at this point, you can use the library routine `edt_ocx_channel_get_signal_detect,` or you can simply verify that the transceiver has detected an incoming signal by checking bit 7 (SIG_DET) in 0x22 Port 0 Status.

The application `read_xfp_sfp.c` includes an example of using these routines to open a port, access the transceiver module, and determine its status. Typical values for laser receive power are in the range of –40 dbm, indicating no signal, to –7 dbm, indicating an incoming signal.

The library routines and example applications are listed and discussed in Installation on page 9 and Configuring the OCM/P on page 13.

Enabling and Verifying the Input Signal

Each port has two digital phase-locked loops (PLLs) — one for the LIU and one for the FPGA — that must be locked to the incoming data. If no data is incoming, or if it has been interrupted, these PLLs must be reset.

The data paths can be initialized independently, so you can perform this procedure for one port while the other is acquiring data. To do so, use the library routines `edt_ocx_channel_setup` (which prepares a port to acquire data) and `edt_ocx_channel_lock_frontend` (which locks the PLLs).

As you can see in Figure 3, `edt_ocx_channel_setup` sets the local and remote loopback, enables the PRBS7 generator, and sets framing, scrambling, and demultiplexing.

Except where noted, `edt_ocx_channel_lock_frontend` performs the following procedure...

1. Clear bits 0 (SLK_EN) and 1 (PLL_EN) in the desired register (0x24 Port 0 Enable).
2. To enable the LIU, set bit 0 (SLK_EN) bit in the desired register (0x24 Port 0 Enable).
3. Wait approximately 100 ms.
4. Verify that the SFP transceiver module has detected an incoming signal by checking bit 7 (SIG_DET) in the desired register (0x22 Port 0 Status).
5. Loop until bit 7 (SIG_DET) is set, showing application status as necessary.
6. To ensure that the LIU PLL is locked, verify that bit 4 (LOL) in the desired register (0x22 Port 0 Status) is clear. If it is set, the LIU is not receiving a stable input signal in the range of the selected reference clock.

NOTE In certain cases, the LOL bit can be clear even if the PLL is not locked. If the phase difference between the incoming data and the reference clock differs by more than 500 parts per million, the LOL state is not valid (for details on the SLK2511 LIU, see Related Resources on page 8).

7. If the incoming signal is a framed SONET or SDH signal, you may wish to verify that bit 5 (LOS) in the desired register (0x22 Port 0 Status) is clear; if it is set, the LIU is not receiving a framed SONET or SDH signal. (This step is not performed by `edt_ocx_channel_lock_frontend`.)

NOTE In order for bit 5 (LOS) to be clear, the signal must be a framed SONET or SDH signal. This bit has no meaning for other signal types.

8. Enable the PLL in the FPGA that communicates with the LIU. Set the PLL_EN bit in the desired register (0x24 Port 0 Enable).
9. To verify that the receive clock PLL is locked to the receive clock of the LIU, verify that bit 6 (RX_LOCK) of the desired register (0x24 Port 0 Enable) is set. If it is clear, the board is not receiving an input signal. Restart the data path initialization procedure from the beginning. (Your code can implement the previous 9 steps as a loop.)
10. If you are receiving a framed SONET/SDH signal, you can verify that it is properly framed and gather statistics about the quality of signal framing. Bit 7 (LOCKED) in 0x95 Port 0 Receive Frame Status tells you if the signal is currently in frame.The library routine `edt_ocx_enable_framing_errors` enables the error counter registers, which track how many errors have occurred in signal framing and what type of errors they are. This routine sets bit 7 (EN_COUNTERS) in 0x93 Port 0 Frame Statistics Count Control.

    The framing error registers are...

    — 0x88–8A Port 0 B1 Error Count on page 32;

    — 0x8B Port 0 B1 Error Mask on page 33;

    — 0x8C–8F Port 0 B2 Error Count on page 33;

    — 0x90–92 Port 0 M1 Error Count Control on page 33;

    — 0x9C–9D Port 0 Loss of Frame Count on page 36;

    — 0x9E–9F Port 0 Frame Pattern Error Count on page 36; and

    — 0xA4–A5 Port 0 False Frame Count on page 37.

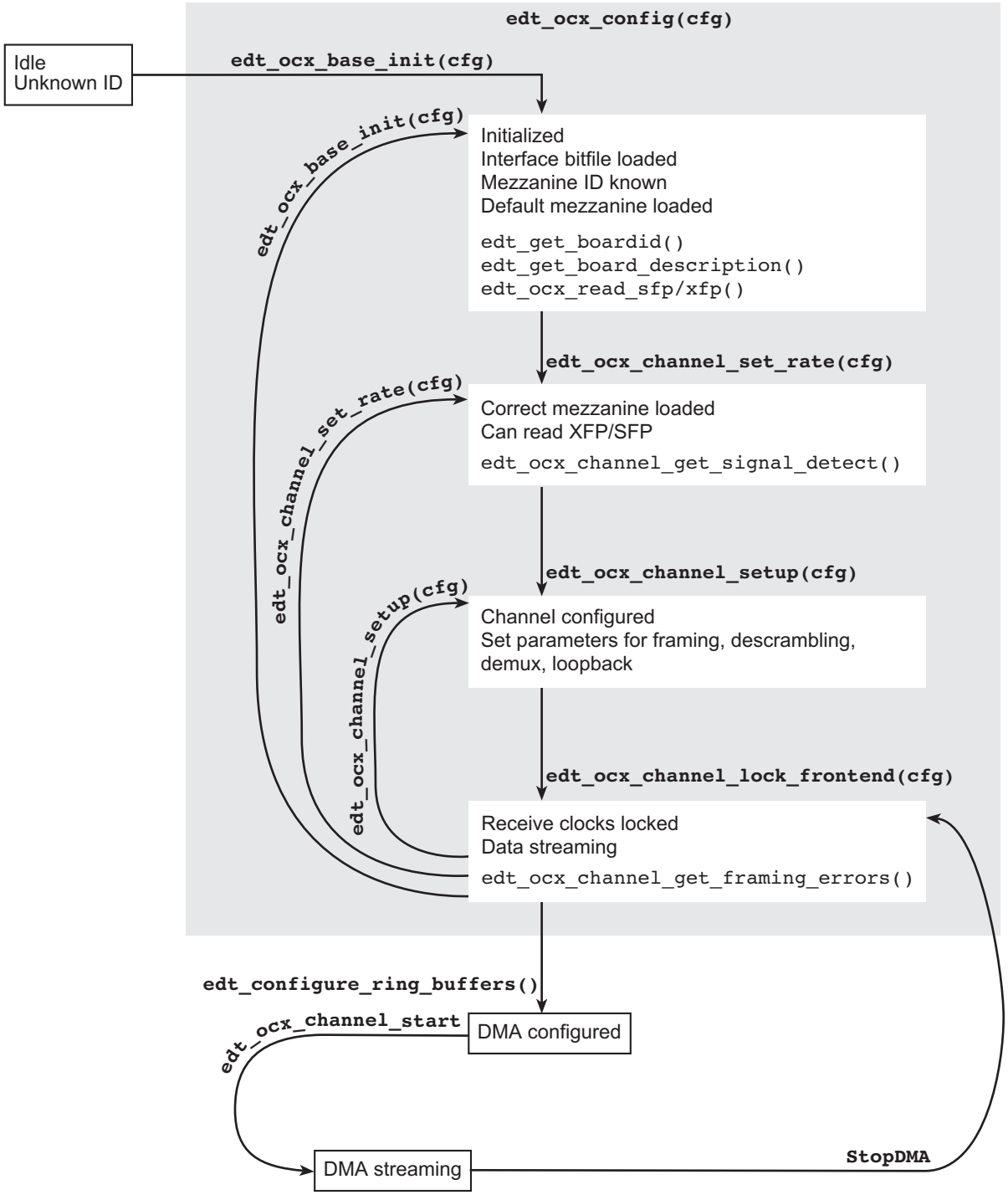
    The library routine `edt_ocx_channel_get_framing_error` accesses these registers and returns the information by filling in the structure `edt_ocx_frame_errors` defined in `edt_ocm.h`.

Initializing and Enabling the Data Paths

1. Configure DMA as required, using the library routine `edt_configure_ring_buffers`.
2. Set up DMA and enable the ports using the C library routine `edt_ocx_channel_start`.
3. Start DMA using the library routine `edt_start_ring_buffers`.
4. Your application can now start transmitting or receiving data by setting the appropriate bits (0–3, CH_ENABLE) in 0x10 Channel Enable.

Included in the C library routines (`lib_ocm.c`) is the file `edt_ocm_configure`, which performs all the steps up to `edt_ocx_channel_start;` browse it for an example that implements OCM/P initialization.

Figure 3. OCM/P Initialization State Transitions



Legend: Normal typeface = transition functions
`Monospace typeface = status functions`

Registers

The registers related to the OCM/P are shown below. The following legacy registers are implemented but not used: Data Path; Channel Direction; Channel Edge; Channel Framing Status.

0x00 Command

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

Bit	Name	Description
7–4	[no name]	Not used.
3	CMD_EN	Set to enable the required DMA channels in 0x10 Channel Enable. When clear, resets all DMA channels, flushes all FIFOs, and clears all under- and overflow bits.
2–0	[no name]	Not used.

0x02 Function

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

Bit	Name	Description
7–1	[no name]	Not used.
0	FUNCT[0]	Reserved for testing.

0x03 Status

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

Bit	Name	Description
7–1	[no name]	Not used.
0	LOCAL_SYS_LOCK	The 100 MHz clock used for data transfers between the main and mezzanine boards is locked in the main board's UI FPGA to the main board reference clock. This phase-locked loop is reset when ports 1 and 0 both are disabled; see 0x10 Channel Enable.

0x05 Main Board FPGA Configuration File Organization

Access / Notes: 8-bit read-only / PCD_ORG

Bit	Name	Description
7–0	[no name]	This byte specifies the organization that created the FPGA configuration file currently loaded in the UI FPGA on the main board. An EDT FPGA configuration file returns the value 0xFF.

0x0F Configuration

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

This register and 0x16 Least Significant Bit First both can affect how data is ordered. For details, see 0x16 Least Significant Bit First.

Bit	Name	Description
7–4	[no name]	Not used.
3	SSWAP	This is the short swap bit; it swaps the position of the two 16-bit short words in one 32-bit data word, so that *short 1*is transferred before *short 0*. It does not change the order of the bits within each short.
2–1	[no name]	Not used.
0	BSWAP	This is the byte swap bit; it swaps the position of bytes 0 and 1, and also bytes 2and 3, in a 32-bit data word, so that the bytes are positioned 1, 0, 3, 2. It does not change the position of the bits within each byte.

Below is the structure of a 32-bit data word, with no swapping in effect. With bit 3 set, short 0 appears before short 1. With bit 0 set, byte 2 appears before byte 3, and byte 0 before byte 1. With both set, the order of the bytes is 3, 2, 1, 0.

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							

0x10 Channel Enable

Access / Notes: 16-bit read-write / SSD16_CHEN

Bit	Name	Description
15–4	[no name]	Not used.
3–0	CH_ENABLE[3–0]	Set or clear the appropriate bit to enable or reset the corresponding I/O port... • Set bit 3 to enable transmit data for port 1; clear to reset. • Set bit 2 to enable transmit data for port 0; clear to reset. • Set bit 1 to enable received data from port 1; clear to reset. • Set bit 0 to enable received data from port 0; clear to reset.

0x16 Least Significant Bit First

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

This register and 0x0F Configuration both can affect how data is ordered. For details, see 0x0F Configuration.

Bit	Name	Description
15–2	[no name]	Not used.
1–0	LSB_FIRST[1–0]	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 port, the most significant bit of a 32-bit word is first.

0x18 Underflow

Access / Notes: 8-bit read-only / SSD16_UNDER

Bit	Name	Description
7–2	[no name]	Not used.
1–0	UNDERFLOW[1–0]	A value of 1 in a bit indicates that the corresponding DMA channel's internal FIFO has underflowed since the previous CMD_EN (register 0x00, bit 3) or CH_ENABLE (register 0x10, bits 3–0). Underflow causes the corresponding DMA channel to transmit the last valid byte repeatedly until it receives new DMA data. To reset, clear and reenable the DMA channel (see 0x10 Channel Enable).

0x1A Overflow

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

Bit	Name	Description
7–2	[no name]	Not used.
1–0	OVERFLOW[1–0]	A value of 1 in a bit indicates that the corresponding DMA channel's internal FIFO has overflowed since the previous CMD_EN (register 0x00, bit 3) or CH_ENABLE (register 0x10, bits 3–0). Data received while the FIFO is in overflow is discarded. To reset, clear and reenable the DMA channel (see 0x10 Channel Enable).

0x1A–1B Overflow – DE1 Firmware

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

Bit	Name	Description
15–0	OVERFLOW[15–0]	For DE1 firmware. A value of 1 in a bit indicates that the corresponding STM1 DMA channel's internal FIFO has overflowed since the previous overflow clear. Data received while the FIFO is in overflow is discarded. To clear, write a 1 to the bit position you wish to clear in this register.

0x20 Port 0 Configuration 0

Access / Notes: 8-bit read-write:

0x20 (port 0) = OCM_CH0_CONFIG0
0x30 (port 1) = OCM_CH1_CONFIG0

This register sets options in the LIU (for a datasheet, see Related Resources on page 8).

Bit	Name	Description
7	PRBS_EN	When set, the LIU transmits a full-band PRBS7 code. The LIU receiver checks the code and sets a bit in register 0x22 Port 0 Status (or 0x32 Port 1 Status) for the appropriate port. The received data is also forwarded for DMA, if DMA is enabled
6	LOCAL_LOOP	Sets local loopback. The port receiver is connected directly to the corresponding port transmitter.
5	REMOTE_LOOP	Sets remote loopback. Data received through the fiberoptic connector is immediately transmitted out the corresponding transmit connector.
4	AUTO_DETECT	When this bit is set and bits 3-2 (RX_SEL) are 00, the LIU automatically detects the incoming bit rate and reports the results in 0x22 Port 0 Status (or 0x32 Port 1 Status) for the appropriate port. Texas Instruments does not recommend using automatic detection.
3–2	RX_SEL	Set the expected receive bit rate as follows... 00 = OC48/STM16 01 = Gigabit ethernet 10 = OC12/STM4 11 = OC3/STM1
1	LOCK_REF	When set, the LIU receive clock is locked to the transmit clock internal to the LIU. For normal operation, keep this bit clear.
0		Reserved.

0x21 Port 0 Configuration 1

Access / Notes: 8-bit read-write:

0x21 (port 0) = OCM_CH0_CONFIG1
0x31 (port 1) = OCM_CH1_CONFIG1

Sets options in the TI SLK2511 LIU; see link under Related Resources on page 8.

Bit	Name	Description
7–5	[no name]	Reserved.
4	LOOPTIME	Set this bit to obtain transmit timing from the receive timing. Normal OCM/P operation is 0 (internal timing).
3–2	PRE2511	The de-emphasis level of the SLK2511 electrical interface can be programmed. For OCM/P operation, always set to 00.
1–0	CONFG2511	The SLK2511 LIU can be configured as follows... 00 = Full duplex transceiver, required for normal operation 01 = Transmit only 10 = Receive only 11 = Repeater mode For OCM/P operation, always set to 00 — full duplex transceiver.

0x22 Port 0 Status

Access / Notes: 8-bit read-write:

0x22 (port 0) = OCM_CH0_STATUS
0x32 (port 1) = OCM_CH1_STATUS

This register sets options in the LIU (for a datasheet, see Related Resources on page 8).

Bit	Name	Description
7	SIG_DET	Set when the SFP transceiver module has detected an incoming signal.
6	[no name]	Reserved; set to zero.
5	LOL	Set when the receive phase-locked loop cannot lock to the incoming data.
4	LOS	Set when the SLK2511 detects loss of signal. This bit is not reset until a SONET or SDH frame is detected.
3–2	RATE_DET	Bits set to the RX rate detected. Set for bit rates as follows... 00 = OC48/STM16 01 = Gigabit ethernet 10 = OC12/STM4 11 = OC3/STM1
1	PRBSPASS	The receiver is receiving a valid full-band PRBS7 code as transmitted by the SLK2511, when PRBS mode is enabled; see bit 7 (PRBS_EN) of 0x20 Port 0 Configuration 0. The PRBS mode must be enabled, although the code can come from any source. **NOTE:** Certain bit error testers use inverted PRBS7 code.
0	SPILL2511	SLK2511 Transmit FIFO overflow. This can occur in loop-timing mode only if the internal transmit data clock is not the same as the received bit rate.

0x23 Port 0 Transceiver

Access / Notes: 8-bit read-write:

0x23 (port 0) = OCM_CH0_TRANSCEIVER
0x33 (port 1) = OCM_CH1_TRANSCEIVER

This register enables the fiberoptic transceivers, reads their status, and allows you to access their internal registers. The defaults are adequate for normal operation; if you need to make modifications, contact EDT.

Bit	Name	Description
7	[no name]	Reserved.
6	XCVR_FLT	Set if the transceiver detects a fault. (Replace the transceiver if the error recurs.)
5	XCVR_PRES	Set when the transceiver is plugged in.
4	XCVR_RDATA	Serial data on COM port two-wire interface.
3	XCVR_TS	Serial data tristate (set to read transceiver data in).
2	XCVR_WDATA	Serial data out for two-wire transceiver comm port.
1	XCVR_SCL	Serial clock for two-wire transceiver comm port.
0	DISABLE_TX	Set to disable light transmitter, if your application is receive-only.

0x24 Port 0 Enable

Access / Notes: 8-bit read-write:

0x24 (port 0) = OCM_CH0_ENABLE
0x34 (port 1) = OCM_CH1_ENABLE

This register is used to initialize the different clock phase-locked loops in the correct order. For details, see Initializing and Setup on page 17.

Bit	Name	Description
7	[no name]	Reserved; reads zero.
6	RX_LOCKED	Set when system clock phase-locked loop is locked.
5	SYS_LOCKED	Set when system clock phase-locked loop is locked.
4		Reserved; reads zero.
3	RAM_EN	**Port 0:** When set, enables the RAM buffer state machines in the `ocm48` firmware. Clear to reset these state machines (for example, during board initialization). **Port 1:** Reserved. Reads last value, but not used.
2	SYS_EN	When set, enables the phase-locked loop of the 100 MHz system clock used to transfer data from the main to the mezzanine board.
1	PLL_EN	When set, the digital phase-locked loop in the OCM/P FPGA device is locked to the receive clock from the SLK2511 LIU. The digital phase-locked loop is held reset when this bit is clear.
0	SLK_EN	When set, the SLK2511 operates normally. Clear to reset the SLK2511 receive clock phase-locked loop.

0x30–34 Port 1 [See 0x20–24 Port 0]

0x40–43 FPGA Load

Access / Notes: 32-bit read-write / no access

Bit	Name	Description
31–0	[no name]	Reserved. Do not write this register space; it is used by `mezzload` to configure the FPGA parts on the OCM/P mezzanine board.

0x60–62 Extended Indirect Register Address

Access / Notes: 24-bit read-write / [no register access name]

These registers are implemented only if the DE1 Configuration Package is used (for details, see Related Resources on page 8).

0x60 = low 8 bits; 0x61 = middle 8 bits; 0x62 = high 8 bits

Bit	Name	Description
23–0	[no name]	This is the register address space for the extended indirect registers.

0x63 Extended Indirect Register Data

Access / Notes: 8-bit read-write / [no register access name]

These registers are implemented only if the DE1 Configuration Package is used (for details, see Related Resources on page 8).

Bit	Name	Description
7–0	[no name]	This is the register data for the extended indirect registers. — Writing this register writes data to the register addressed by 0x60–62. — Reading this register reads the data addressed by 0x60–62.

0x7C–7D Main Board FPGA Configuration File Design ID

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

Bit	Name	Description
15–0	[no name]	A sixteen-bit number assigned by the organization that produced the FPGA configuration file loaded in the main board UI FPGA. (EDT uses the top eight bits only.) The design ID for `ocm.bit` is 0x0400.

0x7E Main Board FPGA Configuration File Version String

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

To read the FPGA configuration file version string from ROM, write the ROM address to the register and read the ASCII data from the same register. The version string is a maximum of 64 bytes long, so only the first six bits of the address are significant.

Bit	Name	Description
7–0	ID_ADD_DATA	Write an address to read ROM contents. Result is... `mainBoard_mezzBoard_bitfileName version.revision mm/dd/yyyy (number of DMA channels used, number of DMA channels required by PCI/e FPGA)` The date given is the date the FPGA configuration file was created. Placeholders in italics are replaced by actual values — for example, `gs4_ocm_ocm 3.12 02/14/2007 (4,4).`

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 3. 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 / SX; GS = PCI GS; SS = PCI SS; AMC = AMC FX5

0x80 Port 0 Receive Framing Control

Access / Notes: 8-bit read-write:

0x80 (port 0) = OCM_CH0_RCV_FRAMING
0xC0 (port 1) = OCM_CH1_RCV_FRAMING

Bit	Name	Description
7	[no name]	Reserved.
6	EN_PAR_CNT	Set to enable count of parity errors; clear to reset the counter.
5	SUSPEND_AQ	Set to halt data acquisition if FIFO overflow occurs.
4	DISABLE_SCRAM	Set to disable descrambling on received framed data. You must set bit 0 (FRAME_EN) and be in frame before this bit has any effect.
	DISABLE_8B10B	For ethernet bit files only: Set to disable the 8b/10b decoder (see Ethernet Operation on page 16).
3–2	RX_DATA_SRC	00 disables the 2 GB onboard FIFO
		01 enables the 2 GB onboard FIFO
		1*x* test mode
1	FRAME_EN	Set to allow data acquisition only when the framer is locked to the incoming signal. Collected data is also descrambled.
	MAC_FILTER	Clear to acquire raw data without framing or descrambling.
		For ethernet bit files only: Set to align data. See also Ethernet Operation on page 16.
0	RESET_FRM	Set, then clear to cause the framing circuits to drop and then relock onto the framing pattern.

0x81 Port 0 Transmit Framing Configuration

Access / Notes: 8-bit read-write:

0x81 (port 0) = OCM_CH0_XMT_FRAMING
0xC1 (port 1) = OCM_CH1_XMT_FRAMING

Bit	Name	Description
7–6	CLOCK_SEL_MSK	Selects the oscillator frequency...
		00 = 125 MHz 01 = 155.52 MHz (OC / STM) 10 = 156.25 MHz (ethernet) 11 = 166.62857 MHz (OTN)
		NOTE Applies to OCM/P boards with programmable oscillators (rev 30 or later).
5–4	[no name]	Reserved.
3	EN_SCRAMBLE	Set to enable scrambling of output data if an OC / STM frame is detected.
2	TX_FRAME_EN	Set to enable local framing with BIP error feedback. Data from the PCI/e bus is payload only and aligned with the AU pointer 0.
1	TEST_DATA	Set to send test data; clear for normal operation.
0	RESET_PTR	Toggle high, then low, to send the AU pointer reset command.

0x82 Port 0 Transmit National Byte

Access / Notes: 8-bit read-write:

0x82 (port 0) = OCM_CH0_XMT_NATIONAL
0xC2 (port 1) = OCM_CH1_XMT_NATIONAL

Bit	Name	Description
7–0	NAT_BYTE	Define byte to be sent in the National Byte field of the frame.

0x83 Port 0 Receive Filter Control

Access / Notes: 8-bit read-write:

0x83 (port 0) = OCM_CH0_RCV_FILTER
0xC3 (port 1) = OCM_CH1_RCV_FILTER

Bit	Name	Description
7–4	[no name]	Reserved.
3	FORCE_ALL_DATA	For ethernet bit files only: Set to force all idles and other command symbols. See also Ethernet Operation on page 16.
2–1	[no name]	Reserved.
0	OVERHEAD_ONLY	Set to acquire SONET / SDH frame overhead only; the payload is discarded.

0x84–87 Port 0 Transmit Test Data

Access / Notes: 32-bit read-write – each access code addresses all 32 bits:

0x84–7 (port 0) = OCM_CH0_XMT_TEST_DATA
0xC4–7 (port 1) = OCM_CH1_XMT_TEST_DATA

Bit	Name	Description
31–0	TEST_DATA	Bytes of a 32-bit transmit data test pattern.

0x88–8A Port 0 B1 Error Count

Access / Notes: 24-bit read-only:

0x88–8A (port 0) = OCM_CH0_B1_ERROR_CNT
0xC8–CA (port 1) = OCM_CH1_B1_ERROR_CNT

Bit	Name	Description
23–0	[no name]	The number of B1 bits found to be in error since the counter was last reset.

0x8B Port 0 B1 Error Mask

Access / Notes: 8-bit read-only:

0x8B (port 0) = OCM_CH0_B1_ERROR_MASK
0xCB (port 1) = OCM_CH1_B2_ERROR_MASK

Bit	Name	Description
7–0	[no name]	The value of the error mask for the last B1 error. A B1 error mask is one byte resulting from a logical operation on the previous frame. Errors occurring regularly in the same bit of the byte can indicate a problem in the sender's or receiver's equipment. If errors are occurring on the fiber, the B1bits in error are more likely to be randomly located in the byte.

0x8C–8F Port 0 B2 Error Count

Access / Notes: 32-bit read-only:

0x8C–8F (port 0) = OCM_CH0_B2_ERROR_CNT
0xCC–CF (port 1) = OCM_CH1_B2_ERROR_CNT

Bit	Name	Description
31–0	[no name]	The number of B2 bits found to be in error since the counter was last reset.

0x90–92 Port 0 M1 Error Count Control

Access / Notes: 24-bit read-only:

0x90–92 (port 0) = OCM_CH0_M1_ERROR_CNT
0xD0–D2 (port 1) = OCM_CH1_M1_ERROR_CNT

Bit	Name	Description
23–0	[no name]	The number of M1 bits found to be in error since the counter was last reset. The M1 byte is sent from the remote receiver of the signal, if that remote receiver has detected a B1 error. In that case, the B1 error mask is copied and sent back as the M1 byte.

0x93 Port 0 Frame Statistics Count Control

Access / Notes: 8-bit read-write:

0x93 (port 0) = OCM_CH0_CNT_CTRL
0xD3 (port 1) = OCM_CH1_CNT_CTRL

Bit	Name	Description
7	EN_COUNTERS	Set to enable framing error counters; clear to reset the counters
6–1	[no name]	Not used.
0	COUNT_HOLD	Set to hold framing error counters so that they can be read without updating; clear to update counters continuously.

0x94 Port 0 Receive Status

Access / Notes: 8-bit read-only:

0x94 (port 0) = OCM_CH0_RCV_STATUS
0xD4 (port 1) = OCM_CH1_RCV_STATUS

Bit	Name	Description
7–1	[no name]	Reserved; always reads zero.
0	FRAMED	Set when incoming data is framed.

0x95 Port 0 Receive Frame Status

Access / Notes: 8-bit read-only:

0x95 (port 0) = OCM_CH0_RCV_FRAME_STATUS
0xD5 (port 1) = OCM_CH1_RCV_FRAME_STATUS

Bit	Name	Description
7	LOCKED	Set when frame is locked – same as bit 0 (FRAMED) in 0x94 Port 0 Receive Status.
6	FOUND	Set when frame pattern is found.
5–4	DROP_CNT	The number of pattern mismatches in framer state machine.
3–2	MATCH_CNT	The number of pattern matches in framer state machine.
1	BYTE_SYNC	Set when the byte synchronization framing pattern is found.
0	BIT_SYNC	Set when the bit synchronization framing pattern is found.

0x97 Port 0 Demux Bitmap

Access / Notes: 8-bit read-write:

0x97 (port 0) = OCM_CH0_DEMUX_BITMAP
0xD7 (port 1) = OCM_CH1_DEMUX_BITMAP

This register addresses a 48-bit mask register that is divided into twelve 4-bit writes and can be written four bits at a time. The top four bits address the desired 4-bit mask register, while the bottom four bits are the value written.

Bit 3 of address 0 disables the first byte of each 48-byte OC48/STM16 multiplexed group, and bit 0 of address B disables the last byte of each multiplexed group. To enable all bytes of the group, all bits are zero (the default).

For STM1 / OC3, use only bits 3–1. The first STM0 or STS1 is demultiplexed by writing 0x07. Bit 0 of the first 4-bit address is not used.

For STM4 / OC12, use only the first twelve bits. These bits function in the same manner, with each matching a column modulo 12. For example, to enable STM1 (1,0), write these three bytes to this register: 0x07, 0x17, 0x27.

Bit	Name	Description
7–4	MASK_ADDR	Bit register address 0x00–0B.
3–0	DEMUX_MASK	The bitmask. A value of one masks the corresponding byte of each 48-byte multiplexed group.

STM1	STM4	STM16
Only bits 3–1 of the mask are used. **Example:** To select AU-3(1), the first AU-3, use the `pdb` command...] : w 97 07	The mask is written 4 bits at a time; only bits 0–11 of the mask are used. **Example:** To select AU-4(1,0), the first STM1, use the `pdb` commands... : w 97 07 : w 97 17 : w 97 27	The mask is written 4 bits at a time; only bits 0–47 of the mask are used. **Example:** To select AU-4(1,1,0), the first STM1, use the `pdb` commands... : w 97 07 : w 97 1F : w 97 2F : w 97 3F : w 97 47 : w 97 5F : w 97 6F : w 97 7F : w 97 87 : w 97 9F : w 97 AF : w 97 BF

0x98 Port 0 Demux Bitmap Readback

Access / Notes: 8-bit read-write – applies only when the firmware file `ocm48.bit` is loaded:

0x98 (port 0) = OCM_CH0_DEMUX_BITMAP_READ
0xD8 (port 1) = OCM_CH1_DEMUX_BITMAP_READ

This register reads back the demultiplexing bitmask. Write this register with the address of the 4-bit mask you wish to read. Read the stored bit pattern in the bottom four bits.

Bit	Name	Description
7–4	MASK_ADDR	Bit register address 0x00 – 0x0B.
3–0	DEMUX_MASK	The bitmask. A value of one masks the corresponding byte of each 48-byte multiplexed group.

0x99 Port 0 Transmit Status

Access / Notes: 8-bit read-only:

0x99 (port 0) = OCM_CH0_TX_STATUS
0xD9 (port 1) = OCM_CH1_TX_STATUS

Bit	Name	Description
7–1	[no name]	Reserved (may not always read zero).
0	TX_FRAMED	Set when a frame is detected in transmitted data. If scrambling is enabled in bit 3 of 0x81 Port 0 Transmit Framing Configuration, then the data is being scrambled.

0x9C–9D Port 0 Loss of Frame Count

Access / Notes: 16-bit read-only:

0x9C–9D (port 0) = OCM_CH0_LOF_CNT
0xDC–DD (port 1) = OCM_CH1_LOF_CNT

Bit	Name	Description
15–0	[no name]	The number of times framing was lost since the counter was last reset. This equals the number of times that the LOCKED bit or the FRAMED bit has gone clear (bit 7 in 0x95 Port 0 Receive Frame Status or bit 0 in 0x94 Port 0 Receive Status). Framing is lost when four consecutive bad framing patterns are detected.

0x9E–9F Port 0 Frame Pattern Error Count

Access / Notes: 16-bit read-only:

0x9E–9F (port 0) = OCM_CH0_FRM_PAT_CNT
0xDE–DF (port 1) = OCM_CH1_FRM_PAT_CNT

Bit	Name	Description
15–0	[no name]	The number of times that the framing pattern was not correct, after data has been in frame. Because framing is not lost until the framing pattern has been incorrect four consecutive times, an incorrect framing pattern does not necessarily mean that framing was lost.

0xA0 Port 0 Mezzanine FPGA Configuration File Version String

Access / Notes: 8-bit read-write:

0xA0 (port 0) = MEZZ_CH0_BITFILE_VERSION
0xE0 (port 1) = MEZZ_CH1_BITFILE_VERSION

Use this register to read the FPGA configuration file version string from ROM. Write the ROM address to the register and read the ASCII data from the same register. The version string is a maximum of 64 bytes long, so only the first six bits of the address are significant

Bit	Name	Description
7–0	ID_ADD_DATA	Write an address to read ROM contents. Result is *bitfileName version.revision mm/dd/yyyy* (the date given is the date the FPGA configuration file was created). The italicized placeholders are replaced by actual values (e.g., *ocm48 3.1 02/14/2007*).

0xA1 Port 0 Mezzanine FPGA Configuration File Organization

Access / Notes: 8-bit read-only:

0xA1 (port 0) = PCD_MEZZ_CH0_ORG
0xE1 (port 1) = PCD_MEZZ_CH1_ORG

Bit	Name	Description
7–0	[no name]	This byte specifies the organization that created the FPGA configuration file currently loaded in the FPGA for the specified port on the mezzanine board. An EDT FPGA configuration file returns the value 0xFF.

0xA2–A3 Port 0 Mezzanine FPGA Configuration File Design ID

Access / Notes: 16-bit read-only:

0xA2–A3 (port 0) = MEZZ_CH0_DESIGN_ID
0xE2–E3 (port 1) = MEZZ_CH1_DESIGN_ID

Bit	Name	Description
15–0	[no name]	A sixteen-bit number assigned by the organization that produced the FPGA configuration file loaded in the specified port of the mezzanine board FPGA. (EDT uses the top eight bits only.) The design ID for `ocm12.bit` is 0x0500; for `ocm48.bit`, it's 0x0600.

0xA4–A5 Port 0 False Frame Count

Access / Notes: 16-bit read-only:

0xA4–A5 (port 0) = OCM_CH0_FALSE_FRM_CNT
0xE4–E5 (port 1) = OCM_CH1_FALSE_FRM_CNT

Bit	Name	Description
15–0	[no name]	When searching for frame, the number of times that a possible frame pattern was detected but the signal was not framed. This can be useful for distinguishing whether the signal sometimes appears to be framed, or whether it always appears to be unframed, and therefore possibly gibberish.

0xA6 Port 0 Programmable Clock Enable

Access / Notes: 8-bit read-write – valid only when the clock programming interface is enabled:

0xA6 (port 0) = OCM_CH0_PROG_CLK
0xE6 (port 1) = OCM_CH1_PROG_CLK

Use this register to determine whether the clocks on your mezzanine board are programmable and, if so, to enable the clock programming interface.

Bit	Name	Description
7–6	IS_OCMP	Read to determine if the board has the programmable clock. 00 = If OCM/P board is rev. 30 or higher, then the clocks are not programmable. 01 = If OCM/P board is rev. 30 or higher, then the clocks are programmable.
5–1	[no name]	Reserved.
0	EN_PROG_INTFC	Set to enable programmable interface for the SI570; clear to use classic clock select. See 0x81 Port 0 Transmit Framing Configuration.

0xA7 Port 0 Programmable Clock Status

Access / Notes: 8-bit read-write – valid only when the clock programming interface is enabled:

0xA7 (port 0) = OCM_CH0_PROG_CLK_DEV_ADDR
0xE7 (port 1) = OCM_CH1_PROG_CLK_DEV_ADDR

Bit	Name	Description
7	SER_DEV_BSY	Read only. When set, the SI570 is busy.
6	SER_DEV_ACK_FAIL	Read only. When set, the SI570 failed to respond to the last command.
5–0	SER_DEV_ADDR	Not used.

0xA8 Port 0 Programmable Clock Read

Access / Notes: 8-bit read-write – valid only when the clock programming interface is enabled:

0xA8 (port 0) = OCM_CH0_PROG_CLK_RD
0xE8 (port 1) = OCM_CH1_PROG_CLK_RD

Bit	Name	Description
7–0	[no name]	Write the register address on the SI570 that you wish to read. Read the data returned from address.

0xA9 Port 0 Programmable Clock Register Address

Access / Notes: 8-bit read-write – valid only when the clock programming interface is enabled:

0xA9 (port 0) = OCM_CH0_PROG_CLK_REG_ADDR
0xE9 (port 1) = OCM_CH1_PROG_CLK_REG_ADDR

Bit	Name	Description
7–0	[no name]	Write the register address on the SI570 that you wish to read. Read the previously accessed SI570 register address.

0xAA Port 0 Programmable Clock Write

Access / Notes: 8-bit read-write – valid only when the clock programming interface is enabled:

0xAA (port 0) = OCM_CH0_PROG_CLK_WR
0xEA (port 1) = OCM_CH1_PROG_CLK_WR

Bit	Name	Description
7–0	[no name]	Write data to the register address in register 0xA9 (for port 0) or 0xE9 (for port 1).

0xC0–EA Port 1 [See 0x80–AA Port 0]

Revision Log

Below is a history of modifications to this guide.

Date	Rev	By	Pp	Detail
20131028	09b	PH,SB	Various	p.7 - Under Related Resources, deleted reference to no-longer-supported Solaris. All pgs: Changed “channel” to “port” when it means physical connector, not DMA.
20110103	09a	PH,SB	18-35	For duplicated registers, separated one combined title at the first hex location into distinct titles at all hex locations (i.e., “0x81, C1” becomes “0x81” and “0xC1”), with each title linking back to the first hex location.
20101116	09	PH	All	Updated product name to clarify that OCMP is an updated OCM.
20100921	08	PH,SB	26	Updated register 0x80, C0.
20100521	08	PH,TL, MM	6	Deleted the deprecated “xtest.”
20100521	08	PH	i	Updated template for title page.
20091012	07	PH	All	Updated template (numbered registers, wider margins, “Related Resources,” etc.)
20060000	00	LW	All	Created guide for this new product.