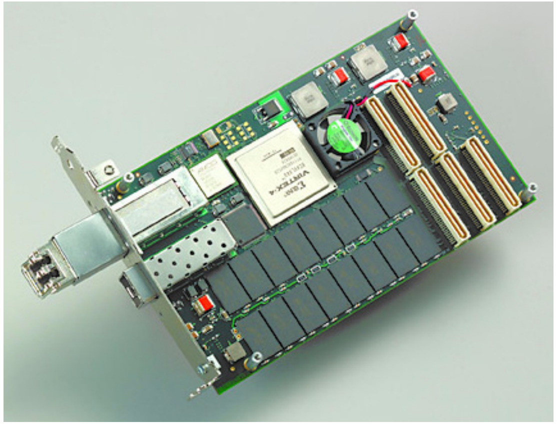
PCI GS OC192

Optical Carrier Interface Mezzanine Board

for use with PCI GS Main Board



May 14, 2007

008-02699-02

a HEICO company

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

Contents

The OC192 Optical Carrier Interface Mezzanine Board

The OC192 Optical Carrier Interface mezzanine board connects to the PCI GS main board. The OC192 supports synchronous optical network (SONET) OC-3, OC-12, OC-48 and OC-192, and synchronous data hierarchy (SDH) STM-1, STM-4, STM-16, and STM-64 telecommunications standards. It has framing and demultiplexing capabilities.

The mezzanine board has two fiber optic LC connectors. One fiber optic connector supports the OC-3 (STM-1), OC-12 (STM-4) or OC-48 (STM-16) standard bit rates, while the other supports the OC-192 (STM-64) only. Throughout this manual, OC-3 refers also to STM-1; OC-12 to STM-4; OC-48 to STM-16; and OC-192 to STM-64.

NOTE If you wish to use channel 1 for OC-192 operation, or channel 0 for OC-48 operation, you must order the optional 4 GB memory buffer.

Ethernet and Optical Transport Network Capability

Boards of Revision 10 or later have additional capabilities:

- a programmable oscillator for each channel (as opposed to earlier boards which have one fixed oscillator for both channels);
- one-gigabit Ethernet receive and transmit capability on channel 0;
- ten-gigabit Ethernet receive and transmit capability on channel 1;
- optical transport network (OTN OTU-1) receive and transmit capability on channel 0; and
- optical transport network (OTN OTU-2) receive and transmit capability on channel 1.

For additional information on OTN, see ITU-T Interfaces for the Optical Transport Network G.709/Y.1331 03/2003 and the International Telecommunications Union website.

To determine the revision of a board, see the revision number on the white sticker on the back of the board, beside the fan.

Related Manuals

Detailed documentation on EDT's C software library routines, helpful for writing your applications, is available on EDT's website in either HTML or PDF form. The *PCI SS/GS Main Board User's Guide* is available in PDF form.

Manual	URL
EDT DMA Software Library (HTML)	www.edt.com/api
EDT DMA Software Library (PDF)	www.edt.com/manuals/misc/api.pdf
PCI SS/GS Main Board User's Guide	www.edt.com/manuals/PCD/pciss_gs.pdf

About the DMA Interface

The OC192 implements the DMA interface using two field-programmable gate arrays (FPGAs), referred to as the PCI FPGA and the UI (user interface) FPGA:

- The *PCI FPGA* communicates with the host computer over the PCI Bus. It implements the DMA engine, which transfers data between the board and the host computer, and loads its firmware on powerup from flash ROM located on the main board.
- The *UI FPGA* transfers data between the user device and the PCI FPGA; in some instances, it also sends the data to the mezzanine board. The UI FPGA or mezzanine board may also process the data in some manner, depending on the application.

When data comes in from the user device, the UI FPGA sends it to input and output FIFO buffers, which smooth data transfer between the user device and the PCI Bus, as well as accommodating data during the transition from one DMA to the next. Host DMA transfers are queued in hardware, minimizing the amount of FIFO required.

To ensure maximum throughput, EDT's DMA library, the DMA driver, and the FPGA configuration files all support pipelining.

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

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

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

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

Installation

Install the OC192 by fitting the fiber optic connectors through the host back panel and then plugging into the PCI connector. The fiber optic connector furthest from the PCI connector is channel 0, and the closest is channel 1; see Figure 1 for details.

When channel 1 is used for OC-192 operation, or channel 0 for OC-48 operation, you must also install the optional 4 GB memory buffer.

NOTE If channel 0 is used for OC-48, OC-12, or OC-3 operation, see the Registers section of the Optical Carrier Multirate Mezzanine Board manual for register descriptions.

About the Software and Firmware

The following OC192-specific files are included in the distribution directory:

`oc192.bit`	Configures the user interface Xilinx® on the main board to communicate with the OC192 mezzanine board.
`oc192m.bit`	Configures the mezzanine Xilinx for OC-192 operation.
`oc48m.bit`	Configures the mezzanine Xilinx for OC-48 operation,.
`oc12m.bit`	Configures the mezzanine Xilinx for OC-12 or OC-3 operation.
`oc192_set`	Utility application that configures the AMCC line interface unit.
`oc192_set.c`	C source for `ocm_set`.
`ocm_snap`	Example application that captures a frame of data from the OC192 board and transfers it to disk for testing or verification.
`ocm_snap.c`	C source for `ocm_snap`.
`read_xfp_sfp`	Example application that queries the state of the transceiver modules. For details, see Querying the Transceivers on page 9.
`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.
`snap10g`	Example script (on UNIX-based systems) or batch file (on Windows systems) that calls (in order):`otuload`, `oc192_set`, and `oc192_snap`.

Sample configuration files for all board configurations are in the `pcd_config` subdirectory of the distribution directory, including:

`oc192_oc3.cfg`	Configuration file for `initpcd` to use to configure the OC192 for OC-3 operation.
`oc192_oc12.cfg`	Configuration file for `initpcd` to use to configure the OC192 for OC-12 operation.
`oc192_oc48.cfg`	Configuration file for `initpcd` to use to configure the OC192 for OC-48 operation.

oc192_oc192.cfg Configuration file for initpcd to use to configure the OC192 for OC-192 operation.

The file names you see in the EDT distribution 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 OC192. 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 EDT top-level distribution directory. UI FPGA firmware files are in the bitfiles subdirectory of the EDT top-level distribution 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 OC192 are listed at the beginning of this section. Instructions for loading them are provided in Configuring the OC192.

Software Initialization Files

Software initialization files (having the extension .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 EDT top-level distribution directory. The software initialization files specific to your OC192 are listed at the beginning of this section. Instructions for their use are provided in Configuring the OC192.

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 Software is updated regularly; the latest versions are available on our website at www.edt.com/software.html. We encourage you to use the latest versions for new installations. For existing applications, upgrade only if you have a specific reason to do so.

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`	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 OC192 to specific frequencies used by the UI FPGA for input and output.

Utility Files

`initpcd`	A utility for initializing and configuring the OC192.

`pdb`	Utility application that enables interactive reading and writing of the PCI SS/GS UI FPGA registers.

Testing Files

A variety of files — C source, executables, and FPGA configuration files — are available to test the boards. Their uses are described in the documents listed under the heading Testing Procedures. They include at least:

`sslooptest`	Tests most PCI SS- and PCI GS-based boards. Determines the board model and selects the loopback test to run, then runs it.
`xtest`	Tests the PCI CD and CDa boards, and the single-channel DMA interface for the PCI SS and PCI GS main boards.

Building Applications

Executable and PCD source files are at the top level of the EDT PCD driver distribution directory. If you need to 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.

About ocm_snap

The application `ocm_snap` and its accompanying C source code provides an example of capturing data. It is a command-line applicatios and can be invoked with a number of options to customize its behavior. For a Help message listing all usage options, invoke these applications with the flag `-h`. For example:

```
ocm_snap -h
```

The example application initializes the board and begins capturing data, filling memory as it proceeds; data can be read back as fast as the host computer can do so.

The following example captures 2 GB of an OC-192 signal from channel 1 (the only channel possible for the OC192), specifying a line rate of 64 (OC-192 or STM-64) and an output file size of 2 GB (2048 MB):

```
ocm_snap -c 1 -r 64 -s 2048 -o output_file
```

(You can also specify a line rate of 192, which is equivalent.)

The example application `ocm_snap` allows you to specify that the output be formatted in hexadecimal chunks of 32, 16, or 8 bits, using the flags `-H`, `-Hw`, or `-Hb`, respectively. In all cases, the most significant bit is the first bit output (in time) and the leftmost bit of the chunk (in memory).

The flag `-s` to `ocm_snap` specifies the final file size in megabytes. The application will terminate when the specified size has been reached.

`ocm_snap` allows you to change the default number and size of the ring buffers using the flags `-n` and `-b`. 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.

Configuring the OC192

Configuring the OC192 mezzanine board requires:

1. configuring the channels with `otuload`,
2. configuring the line interface unit with `oc192_set`, and
3. loading the correct firmware in the PCI Xilinx on the main board with `pciload`.
4. Optionally, you may also wish to verify correct with the example application `oc192_snap`.

Configuring the Channels

In the instructions below, placeholders appear in italics; replace these with the values you require.

To configure channel 1 for OC-192 operation, enter:

```
otuload -u unit_number
```

To configure channel 0 for OC-48 operation, enter:

```
otuload -u unit_number -b oc48m
```

To configure channel 0 for OC-12 or OC-3 operation, enter:

```
otuload -u unit_number -b oc12m
```

The `otuload` program detects and loads the main board user interface Xilinx with `oc192.bit` if it is not already loaded. If it is already loaded and you want to reload it, use the `bitload` utility:

```
bitload -u unit_number oc192
```

Setting Up the Line Interface Unit

The utility application `oc192_set` sets the registers as required, configuring the line interface unit for default operation. This must be done after the channels have been configured, but before you can test the board or use it for normal operation.

Automatic Configuration

The utility `initpcd` takes, as an argument, a software initialization file, and then automatically runs the pertinent command (or commands) of those discussed above. This utility loads the FPGA configuration files, programs the registers, sets the clocks (if necessary), and gets the OC192 mezzanine board ready to perform DMA:

1. It calls `otuload` with the appropriate arguments to configure the channels.
2. It calls `oc192_set` to set up the registers to configure the line interface unit.

If you use `initpcd` to configure the OC192, your application can concern itself solely with performing DMA and other application-specific operations; it will therefore omit OC192-specific operations and be portable to other EDT boards that peform DMA.

To configure the OC192, enter:

```
initpcd -f filename
```

replacing *filename* with one of the configuration files; for example:

```
initpcd -f oc192_oc192.cfg
```

NOTE The software initialization files are editable text files. If the files provided don't meet your needs, copy and modify the one that's closest to your required configuration, then run `initpcd` with your new initialization file.

Finally, the utility application `snap10g` is also available as a convenience. It takes one optional argument, the unit number (by default, 0) and then:

1. It calls `otuload` with the appropriate arguments to configure the channels.
2. It calls `oc192_set` to set up the registers to configure the line interface unit.
3. It calls `ocm_snap` to capture a frame of data from the OC192 and transfer it to disk for testing or verification.

Loading the PCI Xilinx Firmware

For the OC192 to operate correctly, the PCI Xilinx needs to be loaded with the `pcigs4` FPGA configuration file. To check, or to load the correct FPGA configuration file:

1. Navigate to the directory in which you installed the driver (by default, for UNIX-based systems, `/opt/EDTpcd`; for Windows-based systems, `\EDT\pcd`).
2. At the prompt, enter:

```
pciload verify
```

This compares the current PCI Xilinx file in the package with what is currently on the board's PROM.

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

```
pciload -u unit_number verify
```

Dates and revision numbers of the PROM and File ID are displayed. If these numbers match, there is no need for a field upgrade. If they differ, upgrade the flash PROM as follows:

3. At the prompt, enter:

```
pciload update
```

4. Shut down the operating system and turn the host computer off and then back on again. The board reloads firm-ware from flash ROM only during power-up. Therefore, after running pciload, the new FPGA configuration file is not in the Xilinx until the system has been power-cycled; simply rebooting is not adequate.

To see what boards are in the system, run `pciload` without any arguments:

```
pciload
```

To see other `pciload` options, run:

```
pciload help
```

Querying the Transceivers

The OC192's two transceiver modules have a two-wire serial interface allowing you to query their state, including:

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

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

By default, the XFP transceiver module is off. To enable it, enter:

```
read_xfp_sfp -e 1
```

To disable the module, enter:

```
read_xfp_sfp -e 0
```

After the XFP 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
```

Testing

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

NOTE The loopback test overwrites the FPGA configuration file in the user interface Xilinx. 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 test outcome varies 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 completed without errors.

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

```
Total errs=0 bufs=4000; Channel errs(xNxxxxxxxxxxxxxx) bufs(xYxxxxxxxxxxxxxx)
```

   `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 channel (0–15); an `x` indicates a channel is not in use.

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

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

Connector Pinout

The fiber optic transceivers connect to the channels as shown below:

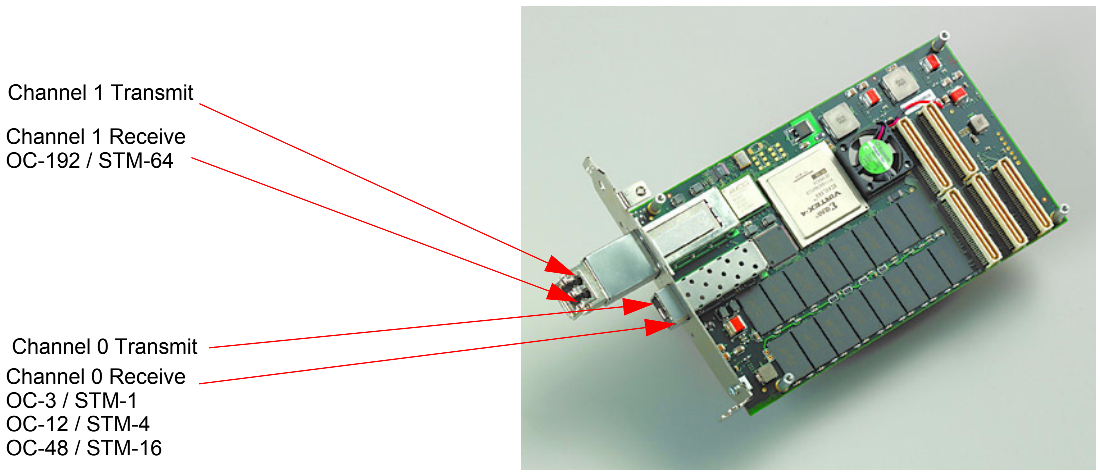


Figure 1. Connector Pinout

Registers

This section describes general-purpose registers and those for Channel 1. Channel 0 registers are described in the OCM Optical Carrier Multirate Manual.

The following legacy registers are implemented but not used:

- Data Path
- Function
- Status
- Channel Framing Status

Command Register

Size	8-bit
I/O	read-write
Address	0x00
Access	PCD_CMD

Bit	Name	Description
7–4		not used
3	CMD_EN	Set this bit, and enable the required channels in the Channel Enable Register, for DMA to occur. When clear, resets all channels, flushes the FIFOs, and clears all under- and overflow bits.
2–0		not used

Status Register

Size	8-bit
I/O	read-write
Address	0x03
Access	PCD_STAT

Bit	Name	Description
7–2		not used
1	MEZZ_SYS_LOCK	The 100 MHz clock used for data transfers between the main and mezzanine boards is locked in the mezzanine board's Xilinx to the main board reference clock.
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 Xilinx to the main board reference clock.

Configuration Register

Size	8-bit
I/O	read-write
Address	0x0F
Access	PCD_CONFIG

Bit	Name	Description
7–4		not used
3	SSWAP	Swaps the position of the two 16-bit short words in one 32-bit data word, so that *short 2* is transferred before *short 1*. Does not change the order of the bits within each short. See Figure 2.
2–1		not used
0	BSWAP	Swaps the position of bytes 0 and 1, and also bytes 3 and 4, in a 32-bit data word, so that the bytes are positioned 1, 0, 3, 2. Does not change the position of the bits within each byte. See Figure 2.

NOTE The Least Significant Bit First Register can also affect the way in which data is ordered.

Figure 2 shows the structure of a 32-bit data word.

Figure 2. Data Word Structure Without Swapping

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

Channel Enable Register

Size	8-bit
I/O	read-write
Address	0x10
Access	SSD16_CHEN

Bit	Name	Description
7–4		not used
3–0	CH_ENABLE[3–0]	Set to 1 to enable the corresponding I/O channel. The I/O channels are assigned as follows: • Bit 0 enables received data from channel 0. • Bit 1 enables received data from channel 1. • Bit 2 enables transmit data for channel 0. • Bit 3 enables transmit data for channel 1. Clear a bit to reset the respective channel.

Least Significant Bit First Register

Size	16-bit
I/O	read-write
Address	0x16
Access	SSD16_LSB

Bit	Name	Description
7–4		not used
3–0	LSB_FIRST[3–0]	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. • Bit 0 changes bit order of received data from channel 0. • Bit 1 changes bit order of received data from channel 1. • Bit 2 changes bit order of transmit data for channel 0. • Bit 3 changes bit order of transmit data for channel 1.

NOTE Byte Swap and Short Swap in the Status Register can also affect the order of bits in a 32-bit word.

Reference Clock Register

Size	8-bit
I/O	read-write
Address	0x2F
Access	OC192_REF_CLOCK

Bit	Name	Description
7–3		not used
2–0	FREQ_SEL	Sets the reference clock frequency. *value frequency (MHz) protocol* 000 155.52 OC / STM 001 167.33165 OTN 010 161.1328 10 Gb Ethernet with 64/66B encoded 011 173.37075 10 Gb Ethernet with 64/66B encoded and FEC 100 125 101 156.25 Ethernet 110 155.52 OC / STM 111 166.62857 OTN

Channel 1 Transceiver Com Port Register

Size	8-bit
I/O	read-write
Address	0x31
Access	OC192_XCVR_COM_PORT
Comment	This register reads and writes the two-wire serial interface COM port to the 10 Gb fiber optic transceiver for Channel 1.

Bit	Name	Description
7	SDA_IN	Read the state of the data pin in the serial interface.
6	SCL_IN	Read the state of the clock pin in the serial interface.
5–4		not used
3	SCL_TRI	Tristate control for serial output clock. Clear this bit to drive the clock pin.
2	SCL_OUT	Serial output clock.
1	SDA_TRI	Tristate control for serial write data out. Clear this bit to drive the data pin.
0	SDA_OUT	Serial interface write data out to transceiver.

Channel 1 Transceiver Control and Status Register

Size	8-bit
I/O	read-write
Address	0x32
Access	OC192_XCVR_CTL_STAT
Comment	This register reads and writes control and status information for the 10 Gb fiber optic transceiver for Channel 1.

Bit	Name	Description
7	INTERRUPT	Indicates the presence of an interrupt; read the condition that caused it over the serial two-wire interface, using the Channel 1 Transceiver Com Port Register.
6	MOD_ABS	This bit is set when the transceiver is absent.
5	MOD_NR	This bit is set when the transceiver is not ready.
4	LOS	When this bit is set, indicates loss of signal.
3		not used
2	POWER_UP	Set to power up the transceiver; clear to power down.
1		not used
0	TX_ENABLE	Set to turn on the transceiver's laser transmitter.

Channel 1 Line Interface Unit MDIO Bus Register

Size	8-bit
I/O	read-write
Address	0x33
Access	OC192_LIU_MDIO_BUS
Comment	This register reads and writes the S19235 line interface unit registers through the MDIO bus. For more information, see the S19235 data sheet.

Bit	Name	Description
7–5		not used
4	SDATA_IN	Serial data received from the line interface unit.
3	SDATA_TRI	Tristate control for serial data input/output buffer — set for writing to the bus; clear for reading.
2	SCLK_TRI	Tristate control for serial clock — set for writing to the bus; clear for reading.
1	SCLK	Serial clock out to the line interface unit.
0	SDATA_OUT	Serial data out to the line interface unit.

Channel 1 Enable Register

Size	8-bit
I/O	read-write
Address	0x34
Access	OC192_ENABLE
Comment	Used to initialize parts of the OC192 circuit. See the configuration file for the correct initialization sequence and values. Initialize the device whenever the receive rate has changed.

Bit	Name	Description
7	RX_LOCKED	When set, indicates the receive clock phase-locked loop is locked.
6	TX_LOCKED	When set, indicates the transmit clock phase-locked loop is locked.
5	SYS_LOCKED	When set, indicates the system clock phase-locked loop is locked.
4	RPLL_EN	When set, the Xilinx receive digital clock manager (DCM) operates normally. When clear, the receive DCM is reset.
3	RAM_EN	When set, enables the RAM buffer state machines in the `oc192m` firmware. Clear to reset these state machines (for example, during board initialization).
2	INIT_EN	When set, enables the DDR2 memory initialization sequence.
1	TPLL_EN	When set, the Xilinx transmit digital clock manager (DCM) operates normally. When clear, the transmit DCM is reset.
0	AMCC_RST	Clear for normal operation. Set to reset the S19235 line interface unit.

Channel 1 Status Register

Size	8-bit
I/O	read only
Address	0x35
Access	OC192_LIU_STATUS
Comment	Reports the status of the S19235 line interface unit.

Bit	Name	Description
7–2		not used, reads zero
1	TX_LOCKDET	When set, the S19235 line interface unit's transmit phase-locked loop is locked.
0	RX_LOCKDET	When set, the S19235 line interface unit's receive phase-locked loop is locked.

Channel 1 Framing Register

Size	8-bit
I/O	read-write
Address	0x37
Access	OC192_FRAMING
Comment	Used to detect SONET / SDH frames.

Bit	Name	Description
7	FRAME_LOCK	Set when the framing engine has locked onto the incoming SONET / SDH data stream.
6	BYTE_SYNC	Set when the byte synchronization framing pattern is found.
5	BIT_SYNC	Set when the bit synchronization framing pattern is found.
4	DIS_SCRAM	Set to disable scrambling on framed data. Bit 1, FRAME_EN, must be set before the scrambler can be disabled.
3–2		not used; reads 0
1	FRAME_EN	Set this bit to allow data collection only when the framer is locked to the incoming signal. Collected data is also descrambled. Clear to collect raw data without framing or descrambling.
0	SEARCH	Set and then clear to cause the framing circuits to drop and then relock onto the OC192 / STM-64 framing pattern.

Channel 1 Output Data Select Register

Size	8-bit
I/O	read-write
Address	0x3A
Access	OC192_OUTPUT_DATA_SEL
Comment	For testing, to select different data sources for the OC192 output channel.

Bit	Name	Description
7–5		not used
4	HEADER_ON	When set, and when PRBS_EN is also set, adds three values to the top of the PRBS frame: 0xF6, 0x28, 0xAA.
3	PRBS_EN	When set, enables PRBS data.
2–1		not used
0	PRBS7	When set, PRBS data is in PRBS7 format. When clear, the data is in PRBS15 format.

OC192 Xilinx Load Registers

Size	four 8-bit registers
I/O	read-write
Address	0x40 through 0x43
Comment	Used by `otuload` to configure the Xilinx on the OC192.
NOTE	Do not write these registers.

OC192 Select Register

Size	8-bit
I/O	read-write
Address	0x4A
Access	OC192_CHAN_SELECT
Comment	Used to switch between channel 0 (OC-48, OC-12, or OC-3 operation) and channel 1 (OC-192 operation).

Bit	Name	Description
7–1		not used, reads zero
0	OC192_SEL	When set, selects OC-192 mode, using fiber optic connector 1. When clear, selects OC-48, OC-12, or OC-3 mode, using fiber optic connector 0. (Channel 0 and channel 1 share a single data path from the OC192 Xilinx to the user interface Xilinx on the PCI SS or PCI GS.)

Frame Statistics Count Control Register

Size	8-bit
I/O	read-write
Address	0x93
Access	OC192_CNT_CTRL

Bit	Name	Description
7	EN_COUNTERS	Set to enable framing error counters; clear to reset the counters.
6–1		not used
0	COUNT_HOLD	Set to hold framing error counters so that they can be read without updating; clear to update counters continuously.

Channel 1 Receive Filter Control Low Register

Size	8-bit
I/O	read-write
Address	0x95
Access	OC192_RCV_FILTER

Bit	Name	Description
7–1		reserved
0	OVERHEAD_ONLY	When set, and framing is enabled, acquires SONET / SDH frame overhead only; discards payload.

Channel 1 Receive Filter Control High Register

Size	8-bit
I/O	read-write
Address	0x96
Access	OC192_RCV_FILTER

Bit	Name	Description
7–0		reserved

Channel 1 Demux Bitmap Registers

These three registers address a 192-bit mask register than can be written 16 bits at a time. The mask register is divided into twelve 16-bit writes. The lower four bits of register 0x99 address the desired 16-bit mask register, while the sixteen bits at 0x97 and 0x98 are the value written.

Bit 15 of address 0 disables the first byte of each 192-byte OC-192/STM-64 multiplexed group, and bit 0 of address 0x0B disables the last byte of each multiplexed group. Bit 7 of register 0x99 must be set to write and clear to read. To enable all bytes of the group, all bits are zero (the default).

For example, to enable STM-1-(1,1,0) using `pdb` (described on page 6), write:

```
>>pdb -u unit number
>>: w 97 FF
>>: w 98 7F
>>: w 99 80
>>: w 97 FF
>>: w 98 FF
>>: w 99 81
>>: w 99 82
>>: w 99 83
>>: w 99 84
>>: w 99 82
>>: w 99 85
>>: w 99 86
>>: w 99 87
>>: w 99 88
>>: w 99 89
>>: w 99 8A
>>: w 99 8B
```

Size	8-bit
I/O	read-write
Address	0x97
Access	OC192_DEMUX_BITMASK

Bit	Name	Description
7–0	DEMUX_MASK[7–0]	A bit pattern. A value of 1 masks the corresponding byte of each 192-byte multiplexed group.

Size	8-bit
I/O	read-write
Address	0x98
Access	OC192_DEMUX_BITMASK

Bit	Name	Description
7–0	DEMUX_MASK[15–8]	A bit pattern. A value of 1 masks the corresponding byte of each 192-byte multiplexed group.

Size	8-bit
I/O	read-write
Address	0x99
Access	OC192_DEMUX_MASK_ADDR

Bit	Name	Description
7	WRITE_STROBE	A value of one to write and zero to read.
6–4		not used
3–0	MASK_ADDR	Bitmask register address 0x00–0x0B.

Channel 1 Demux Bitmap Readback Register

This register reads back the demultiplexing bitmask. To do so, write register 0x99 with bit seven clear to zero. Read the stored bit pattern in these two registers.

Size	two 8-bit registers
I/O	read-write
Address	0x9A, 0x9B
Access	OC192_DEMUX_BITMASK_RD

Bit	Name	Description
7–0	DEMUX_MASK_READ[7–0] for 0x9A DEMUX_MASK_READ[15–8] for 0x9B	A bit pattern. A value of one masks the corresponding byte of each 192-byte multiplexed group.

Loss of Frame Count Register

Size	16-bit
I/O	read only
Address	0x9C, 0x9D
Access	OC192_LOF_CNT

Bit	Description
15–0	The number of times framing was lost since the counter was last reset. This equals the number of times that the FRAME_LOCK bit has gone clear (see the Channel 1 Framing Register on page 16). Framing is lost when four consecutive bad framing patterns are detected.

Frame Pattern Error Count Register

Size	16-bit
I/O	read only
Address	0x9E, 0x9F
Access	OC192_FRM_PAT_CNT

Bit	Description
15–0	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.

Main Board FPGA Configuration File Design ID Register

Size	16-bit
I/O	read only
Address	0x7C, 0x7D
Access	PCD_DESIGN_ID

Bit	Description
15–0	A sixteen-bit number assigned by the organization that produced the FPGA configuration file loaded in the main board UI Xilinx. (EDT uses the top eight bits only.) The design ID for `oc192.bit` is 0x0900.

Main Board Configuration File Version String Register

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.

Size	8-bit
I/O	read-write
Address	0x7E
Access	MAIN_BITFILE_VERSION

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 the PCI Xilinx*). The date given is the date the FPGA configuration file was created. Placeholders in italics are replaced by actual values — for example, `gs4_oc192_oc192 3.12 02/14/2007 (4,4)`.

Board ID Register

Size	8-bit
I/O	read-write
Address	0x7F
Access	EDT_BOARDID
Comment	Returns a unique four-bit code corresponding to the mezzanine board installed. A value of 2 indicates an extended board ID. To read an extended board ID code, use the application `extbdid.exe` or the EDT DMA library routine `edt_get_boardID`.

Bit	Name	Description
7–5		used by `extbdid.exe`
4		not used; always set
3–0	BOARD_ID	The ID code of the installed mezzanine board: 12 3x3G 11 OC192 10 16TE3 F Combo I/O, ECL E Combo II I/O, RS-422 D Combo III I/O, ECL C Combo III I/O, LVDS B Combo III I/O, RS-422 A SRXL (with Graychips) 9 TLK1501 I/O 8 ECL I/O 7 Combo II I/O, LVDS 6 OCM 5 HRC for E4, STM-1, OC3 4–2 reserved 1 LVDS I/O 0 RS-422 I/O

Mezzanine Configuration File Version String Register

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.

Size	8-bit
I/O	read-write
Address	0xE0
Access	MEZZ_BITFILE_VERSION

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. Placeholders in italics are replaced by actual values — for example, `oc192m 3.1 02/14/2007`.

Mezzanine FPGA Configuration File Organization Register

Size	8-bit
I/O	read only
Address	0xE1
Access	PCD_MEZZ_ORG

Bit	Description
7–0	A byte specifying the organization that created the FPGA configuration file currently loaded in the Xilinx for the specified channel on the mezzanine board. An FPGA configuration file produced by EDT returns the value 0xFF.

Mezzanine FPGA Configuration File Design ID Register

Size	16-bit
I/O	read only
Address	0xE2, 0xE3
Access	MEZZ_DESIGN_ID

Bit	Description
15–0	A sixteen-bit number assigned by the organization that produced the FPGA configuration file loaded in the specified channel of the mezzanine board Xilinx. (EDT uses the top eight bits only.) The design ID for `oc192.bit` is 0x0900; for `oc192m.bit`, it's 0x0A00.

False Frame Count Register

Size	16-bit
I/O	read only
Address	0xA4, 0xA5
Access	OC192_FALSE_FRM_CNT

Bit	Description
15–0	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.