

SSE

Synchronous Serial ECL Mezzanine Board

for use with a PCI SS, PCI GS, or PCIe LX main board



December 18, 2008

008-02382-02



a HEICO company

© 1997-2008 Engineering Design Team, Inc. All rights reserved.

EDT and Engineering Design Team are trademarks of Engineering Design Team, Inc.

Xilinx is a registered trademark of Xilinx, Inc.

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 device driver binary software, as well as all source code for 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 all forms, both physical and electronic ("Documentation").

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

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

Limited 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 from defects in material and workmanship for a period of 12 months from date of shipment to the initial Buyer. This warranty does not apply to any products which are 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. Sellers Plant, Beaverton, Oregon) any goods which are found to be non-conforming or defective as specified by Buyer within 30 days of discovery of any defect. All installation and transportation expenses, and all other incidental expenses and damages, shall be borne by the Buyer. *This warranty is expressly in lieu of all other warranties, express or implied, including but not limited to any warranties of merchantability or fitness for any particular purpose.*

Limitation of Liability. *In no event shall seller be liable for any type of special consequential, incidental, or penal damages, whether such damages arise out of 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 revenues, loss of use of the goods or associated equipment, costs of substitute goods, equipment, 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

About the SSE Synchronous Serial ECL Board	1
Related Manuals	1
About the Software and Firmware	2
FPGA Configuration Files For Receiving and Transmitting Raw Data	2
FPGA Configuration Files For Receiving Reed-Solomon-encoded Data	2
FPGA Configuration Files For the Main Board PCI Xilinx	2
Software Initialization Files	3
Utility Files	3
The PCD Device Driver	3
FPGA Configuration Files	3
Software Initialization Files	3
Sample Applications and Utilities	4
Sample Applications	4
Utility Files	5
Testing Files	5
Building Applications	5
Configuring the SSE	5
Checking the PCI FPGA Firmware	6
Loading the UI FPGA Firmware and Configuring the SSE	7
Using Custom FPGA Configuration Files	7
Initializing With sseload	7
Using the Reed-Solomon Decoder	8
Testing	8
Single-board Test	9
Board-to-board Test	9
Pinouts	10
Registers	11
Command Register	11
Configuration Register	11
Channel Enable Register	12
Least Significant Bit First Register	12
Underflow Register	12
Overflow Register	13
Main Board PLL Programming Register	13
Board ID Register	14
Mezzanine Board PLL Programming Register	15
Mezzanine Board Clock Control Register	15

Reed-Solomon Encoder/Decoder Registers	16
Main Board PLL 0 Divider Register	16
Frame Length Register	16
Virtual Fill Register	16
Reed-Solomon Decoder Registers	17
Decoder Synchronization Pattern Register	17
Mask Register	17
Check / Flywheel Count Register	17
Decoder Control Register	17
Reed-Solomon Encoder Registers	19
Encoder Synchronization Pattern Register	19
Encoder Control Register	19

About the SSE Synchronous Serial ECL Board

The PCI SS/GS Synchronous Serial ECL mezzanine board (SSE) connects to the PCI SS/GS main boards to enable the transfer of data at speeds up to 400 MHz (or as limited by PCI bus bandwidth) using three independent serial channels of one bit each. Channels 0 and 1 are input channels; channel 2 is an output channel. Through these channels, ten SSE samples data on the rising edge of the clock signal and stores it in host memory through the PCI SS/GS DMA engine.

Each input channel accepts a differential data signal and a differential clock at standard ECL signal levels. The output channel sends one differential data and one differential clock signal. All input signals are terminated through 50 Ω to -2 V.

NOTE The data and clock signals for the output channel are not terminated on the SSE; therefore, in order to receive signals from the output channel, the data path must be terminated at the receiver.

The SSE mezzanine board has a Xilinx field-programmable gate array that works with the two Xilinx field-programmable gate arrays on the PCI SS/GS main board. For proper operation, compatible firmware must be loaded into all three of these Xilinxes. Two combinations are available:

- One combination of firmware files enables the transfer of raw data.
- Another firmware combination allows decoding of Reed-Solomon-encoded input data.

An additional firmware file is included for testing. A list of the firmware files provided is available in [About the Software and Firmware on page 2](#), and instructions for loading them are provided in [Configuring the SSE on page 5](#).

Related Manuals

Detailed documentation on EDT's C software library routines, helpful for writing your applications, is available in either HTML or PDF formats:

Manual	URL
EDT DMA Software Library (HTML)	www.edt.com/api
EDT DMA Software Library (PDF)	www.edt.com/manuals/misc/api.pdf

The PCI SS/GS main board is described in:

PCI SS/GS Main Board User's Guide www.edt.com/manuals/PCD/pciss_gs.pdf

The following manuals describe the FPGA configuration files available for the SSE:

`eclopt_sse.bit` www.edt.com/manuals/bitfiles/eclopt_sse.pdf

describes the FPGA configuration files for the transfer of raw data: `eclopt_sse.bit` for the main board UI Xilinx and its companion `sseio.bit` for the mezzanine board Xilinx. This also describes the test configuration file `ssein.bit`.

`sse_rs_decoder.bit` www.edt.com/manuals/bitfiles/sse_rs_decoder.pdf

describes the FPGA configuration files for decoding Reed-Solomon-encoded input data: `sse_rs_decoder.bit` for the main board UI Xilinx and its companion `sseio_asm.bit` for the mezzanine board Xilinx.

In addition, those using the Reed-Solomon decoder may find the following document useful:

Consultative Committee for Space Data Systems (CCSDS) 101.0-B-6. Telemetry Channel Coding at: <http://public.ccsds.org/publications/archive/101x0b6s.pdf>

About the Software and Firmware

The SSE comes with firmware files to configure the two Xilinxes (having the extension `.bit`), a variety of utility applications, a firmware file to use for testing the board, and software initialization files (having the extension `.cfg`) to use to initialize the board for each configuration.

The PCI Xilinx firmware files are in the `flash` subdirectory of the EDT top-level distribution directory. UI Xilinx firmware files are in the `bitfiles` subdirectory of the EDT top-level distribution directory.

The SSE ships with the following SSE-specific software:

FPGA Configuration Files For Receiving and Transmitting Raw Data

`eclopt_sse.bit` The FPGA configuration file for the UI Xilinx on the PCI SS/GS main board.
`sseio.bit` The FPGA configuration file for the Xilinx on the SSE mezzanine board.
`ssein.bit` The FPGA configuration file for the Xilinx on the SSE mezzanine board, for testing only.

FPGA Configuration Files For Receiving Reed-Solomon-encoded Data

`sse_rs_decoder.bit` The FPGA configuration file for the UI Xilinx on PCI SS/GS main board.
`sseio_asm.bit` The FPGA configuration file for the Xilinx on the SSE mezzanine board.

FPGA Configuration Files For the Main Board PCI Xilinx

Compatible 4-channel bitfiles must be loaded in the PCI Xilinx on the main boards. In all cases, these are:

`pciss4.bit` The FPGA configuration file for the PCI Xilinx on PCI SS boards.
`pcigs4.bit` The FPGA configuration file for the PCI Xilinx on PCI GS boards.

Software Initialization Files

Sample software initialization files for all board configurations are in the `pcd_config` subdirectory of the distribution directory. Software initialization files are editable text files that you can customize for your own applications. The SSE comes with:

<code>sse.cfg</code>	Configures the SSE for the transfer of raw data.
<code>sse_rs.cfg</code>	Configures the SSE to receive and decode Reed-Solomon-encoded input data.

Utility Files

<code>mezzload</code>	A utility for initializing and configuring the SSE.
<code>rfile</code>	A utility for reading data from an input DMA channel and writing it to a file.
<code>wfile</code>	A utility for reading data from a file and writing it to the output DMA channel.

The firmware 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 SSE. 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 SSE are listed at the beginning of this section. Instructions for loading them are provided in [Configuring the SSE](#).

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 SSE are listed at the beginning of this section. Instructions for their use are provided in [Configuring the SSE](#).

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 utilities, 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

<code>rd16</code>	Performs simple multichannel ring buffer input.
<code>wr16</code>	Performs simple multichannel ring buffer output.
<code>simple_read</code>	Performs DMA input without using ring buffers. Data is therefore subject to interruptions, depending on system performance.
<code>simple_write</code>	Performs DMA output without using ring buffers. Data is therefore subject to interruptions, depending on system performance.
<code>simple_getdata</code>	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.

<code>simple_putdata</code>	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.
<code>test_timeout</code>	Under normal operation, timeouts cancel DMA transfers. This application exemplifies giving notification when a timeout occurs, without canceling DMA
<code>set_ss_vco</code>	A utility for programming the output clock or clocks on the SSE to specific frequencies used by the UI FPGA for input and output.

Utility Files

<code>initpcd</code>	A utility for initializing and configuring the SSE.
<code>pdb</code>	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:

<code>sslooptest</code>	Tests most PCI SS- and PCI GS-based boards. Determines the board model and selects the loopback test to run, then runs it.
<code>xtest</code>	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.

Configuring the SSE

For the SSE to operate as you require, it must be loaded with the appropriate FPGA configuration files for both FPGAs. The PCI FPGA is loaded from flash ROM, which is shipped from the factory already loaded with the appropriate FPGA configuration file; however, you must load the UI FPGA yourself.

Before loading the UI FPGA, however, you may wish to check the firmware in the PCI FPGA to ensure that it is correct and up-to-date.

Checking the PCI FPGA Firmware

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

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

NOTE The presence of a newer version of the firmware with a new driver doesn't necessarily mean that the firmware must be updated; if a package contains a mandatory upgrade, it is prominently stated in the README file.

On UNIX systems, `pciload` is an application in the installation directory `/opt/EDTpcd`.

On Windows systems, double-click the Pcd Utilities icon to bring up a command shell in the installation directory `\EDT\Pcd`.

On Macintosh systems, `pciload` is an application in the installation directory `/Applications/EDT/pcd`.

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

```
pciload
```

The program outputs the date and revision number of the firmware in the PROM.

To compare the PCI FPGA firmware in the package with the one already loaded on the board, enter:

```
pciload verify
```

The program compares the firmware in the PROM against the firmware file in the installation directory. If they match, there's no need to upgrade the firmware. If they differ, you'll see error messages. This does not necessarily indicate a problem; if your application is operating correctly, you may not need to upgrade the firmware.

If you wish to update the standard firmware, enter:

```
pciload update
```

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

```
pciload firmware_filename
```

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

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

```
pciload -u unit_number filename
```

2. At the prompt, press **Enter** to confirm the loading operation. (If the file date is older than the PROM ID date, you may need to press **Enter** twice.)

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

NOTE Updating the firmware requires cycling power, not simply rebooting.

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

```
pciload --help
```

Loading the UI FPGA Firmware and Configuring the SSE

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

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

To configure the SSE, enter:

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

replacing `unit_number` with the number of the board (by default, 0), and replacing `filename` with one of the initialization files listed in [About the Software and Firmware](#); for example:

```
initpcd -f sse.cfg
```

NOTE 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 file.

Using Custom FPGA Configuration Files

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

Initializing With `sseload`

Replace the placeholders in italics with the values required in your own case.

To configure the SSE to receive and transmit raw data, enter:

```
sseload
```

This loads the UI Xilinx on the main board with the firmware `eclopt_sse.bit` and the mezzanine board Xilinx with `sseio.bit`. If you have more than one EDT board in your host, you can specify a unit number with the `-u` option:

```
sseload -u unit_number
```

The default unit number is 0, for the first or only board in a system.

To configure the SSE to receive Reed-Solomon-encoded data, enter:

```
sseload -R
```

This loads the UI Xilinx on the main board with the firmware `sse_rs_decoder.bit` and the mezzanine board Xilinx with `sseio_asm.bit`. Again, add a `-u` and a unit number if necessary.

To configure the board for testing, enter:

```
sseload -b ssein
```

This loads the UI Xilinx on the main board with the firmware `eclopt_sse.bit` and the mezzanine board Xilinx with `ssein.bit`. It also sets the mezzanine PLL to 100 MHz.

To set the mezzanine PLL to a different frequency, enter:

```
sseload -F frequency
```

Frequency is in MHz, so the `sseload` value must be in millions. For example, to select 250 MHz, enter:

```
sseload -F 250
```

Older boards default to the mezzanine PLL clock, and frequency must be 50–800 MHz.

Newer boards have an SI570 clock, which is used by default, and frequency must be 10–800 MHz.

Using the Reed-Solomon Decoder

The Reed-Solomon decoder conforms to [Consultative Committee for Space Data Systems \(CCSDS\) 101.0-B-6](#) (255,223) with a five-way interleave. The Reed-Solomon decoder is embedded in input channel 0 only. It is not available for data using input channel 1.

NOTE The FPGA configuration file that implements the decoder — `sse_rs_decoder.bit` along with its companion configuration files — does not also implement an encoder for output. If you wish to output Reed-Solomon-encoded data, contact [EDT](#).

You can input data from the SSE to the host in any of three ways:

- Send raw data to the host from input channels 0 and 1 — the default mode of operation.
- Detect synchronization, then send raw data to the host in multiples of the frame size, from input channels 0 and 1.
- Detect synchronization, decode the data, then send the decoded data frames to the host, from input channel 0 only.

To program the SSE for Reed-Solomon decoding, see the instructions in [Initializing With sseload on page 7](#).

The Reed-Solomon decoder uses PLL 0 on the main board as the decoding clock. The decoder has a processing delay of 660 symbol clock cycles. Therefore, to keep up with a 200 MHz serial stream, PLL 0 must be set to 66 MHz. To calculate the correct PLL 0 frequency for your data stream, the formula is:

$$\text{PLL0 freq} / (\text{serial_freq}/8) \text{ must be greater than } 660/225$$

You can set the PLL 0 frequency using the utility `set_ss_vco`. For example, to set PLL 0 to 66 MHz, enter:

```
set_ss_vco -F 66000000 0
```

The argument after the `-F` specifies the frequency; the final argument specifies which of the four PLLs you're setting — in this example, PLL 0. (To specify a board other than the default first board in a system (unit 0), add a `-u` and a unit number if necessary.)

Testing

You can test one SSE by itself, or you can connect two of them and test the data path between them.

Single-board Test

The loopback test determines the board configuration, loads the appropriate firmware, 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 2](#) for the complete list.

NOTE The loopback test overwrites the firmware in the UI 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.

Test output for a functional board contains lines such as:

```
Total errs=0 bufs=4000; Channel errs(NNNxxxxxxxxxxxxxx) bufs(YYYxxxxxxxxxxxxxx)
```

`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`, `sseload`, or your own application to disable loopback.
4. Reconnect the board to the external device.

Board-to-board Test

To test two boards:

1. Run `sseload` on both, thus:

```
sseload -u 0
sseload -u 1
```

This loads the UI Xilinx on both main boards with the firmware `eclopt_sse.bit` and the Xilinx on both mezzanine boards with `sseio.bit`. It also sets both mezzanine PLLs to 100 MHz.

2. Generate PRBS15 test data on the first board:

```
genprbs15 -u 0 -c 2 -n 1 -l 0
```

The `-c 2` indicates the channel on which to begin producing output — in this case, on channel 2, the SSE output channel. The `-n 1` indicates the number of channels on which to produce output — in this case, only one. The `-l 0` indicates the number of times to loop through the data generating code — a value of 0 causes the software to loop indefinitely, until terminated by an operating system-level halt such as pressing Control-C. (Generating the code indefinitely allows you to start checking on the second board at any time, without worrying that the test data will terminate before the second board can start checking it.)

3. Check the test data on the second board:

```
chkprbs15 -u 0 -c 0 -n 1 -l 1000
```

The `-c 0` indicates the channel on which to begin checking input — in this case, on channel 0; though you can change this to `-c 1` if you wish to receive the input on channel 1 instead. The `-n 1` indicates the number of channels on which to receive input — in this case, only one. The `-l 0` indicates the number of times to loop through the data generating code — in this case, we're checking the test pattern 1000 times.

4. To stop the `genprbs15` process, use your operating system halt command, such as Control-C.

Test output for a functional board contains lines such as:

```
Total errs=0 bufs=4000; Channel errs(NNNxxxxxxxxxxxxxxxx) bufs(YYYxxxxxxxxxxxxxxxx)
```

`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.

Pinouts

The SSE connects your device to the PCI SS/GS main board using the 15-pin D connector as shown in [Table 1](#).

Table 1. SSE Connector Pinout

Pin	Signal	Input / Output
1	DAT0+	input
2	DAT0–	input
3	CLK0+	input
4	CLK0–	input
5	DAT2–	output
6	spare	input
7	DAT1–	input
8	CLK1–	input
9	ground	
10	DAT2+	output
11	spare	input
12	DAT1+	input
13	CLK1+	input
14	CLK2–	output
15	CLK2+	output

Registers

The following registers are implemented in the FPGA configuration files `eclopt_see.bit` and `sse_rs_decoder.bit`.

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

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 order of the two 16-bit short words in one 32-bit data word, so that <i>short 2</i> is transferred before <i>short 1</i> . Does not change the order of the bits within each short. See Figure 1 for the details of data word structure.
2–1		not used
0	BSWAP	Swaps the order of bytes 1 and 2, and also bytes 3 and 4, in a 32-bit data word, so that the bytes are transferred in the order 2, 1, 4, 3. Does not change the order of the bits within each byte. See Figure 1 for the details of data word structure.

NOTE The [Least Significant Bit First Register](#) can also affect the order in which data is transferred.

[Figure 1](#) shows the structure of a 32-bit data word, with no swapping in effect. With SSWAP set, short 0 appears before short 1. With BSWAP set, byte 2 appears before byte 3, and byte 0 before byte 1. With both set, byte 0 appears first, followed by byte 1, byte 2, and finally byte 3.

Figure 1. Data Word Structure

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
7–3		not used
2–0	CH_ENABLE	A value of one in a bit enables the corresponding channel for DMA.

Least Significant Bit First Register

Size	8-bit
I/O	read-write
Address	0x16
Access	SSD16_LSB

Bit	Name	Description
7–3		not used
2–0	LSB_FIRST	When set for a channel, the least significant bit of each 8-bit data byte is the first bit, and the most significant bit is the last. When clear for a channel, the most significant bit of the byte is the first bit.

NOTE BSWAP and SSWAP in the [Configuration Register](#) can also affect the order of bits in a 32-bit word. A combination of these bits allow the data to be formatted correctly for your host computer and application.

Underflow Register

Size	8-bit
I/O	read only
Address	0x18
Access	SSD16_UNDER

Bit	Name	Description
7–3		not used
2	UNDERFLOW	A value of 1 in a bit indicates that the output channel's internal FIFO has underflowed since the previous CMD_EN or CHANNEL_ENABLE. Reset by first disabling, then re-enabling, the channel (see the Channel Enable Register).
1–0		not used

Overflow Register

Size	8-bit
I/O	read only
Address	0x1A
Access	SSD16_OVER

Bit	Name	Description
7–2		not used
1–0	OVERFLOW	A value of 1 in a bit indicates that the corresponding input channel's internal FIFO has overflowed since the previous CMD_EN or CHANNEL_ENABLE. Reset by first disabling, then re-enabling, the channel (see the Channel Enable Register).

Main Board PLL Programming Register

Size	8-bit
I/O	read-write
Address	0x20
Access	EDT_SS_PLL_CTL
Comment	The program <code>set_ss_vco</code> uses this register to program the serial interface of the four PLLs on the PCI SS/GS main board.

Bit	Name	Description
7	PLL_SCLK	Connected to all four PLL serial clock inputs.
6	PLL_DATA	Connected to all four PLL serial data inputs.
5–4		not used
3–0	PLL_STROBE	Connected to the strobe inputs of PLL 3–0, respectively.

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 <code>extbdid.exe</code> or the EDT DMA library routine <code>edt_get_boardID</code> .

Bit	Name	Description
7–5		used by <code>extbdid.exe</code>
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 Board PLL Programming Register

Size	8-bit
I/O	read-write
Address	0x80
Access	SSE_PLL_CTL
Comment	On older boards that do not have the SI570 clock, the program <code>sseload</code> uses this register to program the high-speed PLL on the mezzanine board. The PLL reference clock input comes from a 16 MHz oscillator.

Bit	Name	Description
7–3		not used
2	SCLK	Connected to the PLL serial clock input.
1	SDATA	Connected to the PLL serial data input.
0	SLOAD	Connected to the PLL S_LOAD input.

Mezzanine Board Clock Control Register

Size	8-bit
I/O	read-write
Address	0x84
Access	SSE_CLK_CTL
Comment	Selects the clock source for the output channel.

Bit	Name	Description
7–3		not used
2	SI570_CLK_SELECT	Set to use SI570 clock for output timing (10–800 MHz).
1	CLK_SELECT	Clear to use the high-speed PLL (50–800 MHz). To select a frequency below 50 MHz, set to use PLL1 from the main board.
0	CLK_DISABLE	Set to disable the output clock.

Reed-Solomon Encoder/Decoder Registers

All Reed-Solomon registers (for both encoding and decoding) are implemented in the FPGA configuration file `sse_rs_decoder.bit` only.

Main Board PLL 0 Divider Register

Size	16-bit
I/O	read-write
Address	0x24, 0x25
Access	EDT_SS_PLL0_CLK
Comment	This register is set by <code>set_ss_vco</code> .

Bit	Name	Description
15–0	PLL0_DIV	<p>A post-scalar divider used to achieve lower frequencies than those at which the main board PLLs can be programmed. After this division (if any), the clocks are divided by two for an even duty cycle — half the time high, and half low. <code>set_ss_vco</code> takes this into account.</p> <p>PLL0 sets the output clock for the diagnostic PRBS15 test data generator. It's also used as the clock for the Reed-Solomon decoder. As such, to keep up with a data stream:</p> <p><i>PLL0 freq / (serial_freq/8) must be greater than 660/225</i></p>

Frame Length Register

Size	16-bit
I/O	read-write
Address	0x94, 0x95
Access	RS_FRAME_LEN

Bit	Name	Description
15–0	FRAMELEN	<p>Sets the frame length. If frame synchronization is on but Reed-Solomon decoding is not, as set in the Decoder Control Register. (If Reed-Solomon decoding is on, frame length is fixed.)</p> <p>A value of 0 specifies a frame length of 65,536.</p>

Virtual Fill Register

Size	8-bit
I/O	read-write
Address	0x70
Access	VFILL

Bit	Name	Description
7-2		not used
1-0	VFILL	Sets the number of virtual fill bytes (0, 1, 2 or 3) for both the encoder and decoder.

Reed-Solomon Decoder Registers

The following registers are implemented in the FPGA configuration file `sse_rs_decoder.bit` only.

Decoder Synchronization Pattern Register

Size	32-bit
I/O	read-write
Address	0x86, 0x87, 0x88, 0x89
Access	RS_DSYNC_PAT

Bit	Name	Description
31–0	DSYNC	The 32-bit synchronization pattern for the decoder.

Mask Register

Size	32-bit
I/O	read-write
Address	0x90, 0x91, 0x92, 0x93
Access	RS_MASK

Bit	Name	Description
31–0	MASK	The 32-bit synchronization pattern mask, permitting some bits of the synchronization pattern to be ignored.

Check / Flywheel Count Register

Size	8-bit
I/O	read only
Address	0x98
Access	RS_CHECK_FLY

Bit	Name	Description
7–4	CHKCNT	Check frame count.
3–0	FLYCNT	Flywheel frame count

Decoder Control Register

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

Bit	Name	Description
7–4		not used
3	STAT_EMBED	Set to append status information to the end of data frames. Status information tells the current state of the frame synchronization engine, Reed-Solomon error statistics, or both.
2		not used

Bit	Name	Description
1	RS_EN	Set to enable the Reed-Solomon decoder. Applies to channel 0 only. Incoming Reed-Solomon-encoded data is assumed to conform to CCSDS (255,223) with a five-way interleave.
0	FS_DET	Set to enable frame synchronization.

These bits interact as shown in [Table 2](#), in which the character — means “has no effect”:

Table 2. Control Register Bits

RS_EN	FS_DET	STAT_EMBED	Description
1	—	1	Reed-Solomon decoding is on. Frame length is fixed at $255 \times 5 = 1275$. Two status bytes are appended to each decoded data frame; see Table 3 for status byte details.
1	—	0	Reed-Solomon decoding is on; frame length is fixed at $255 \times 5 = 1275$; no status information is appended to frames.
0	1	1	Frame synchronization is on without Reed-Solomon decoding. Raw data is sent to the host in multiples of the frame length, as set by the Check / Flywheel Count Register . The first status byte only is appended to each decoded data frame; see Table 3 for status byte details.
0	1	0	Frame synchronization is on without Reed-Solomon decoding. Raw data is sent to the host in multiples of the frame length, as set by the Check / Flywheel Count Register . No status information is appended to frames.
0	0	—	Raw data only is sent to the host, without framing, decoding, or status information.

[Table 3](#) shows the format of the two status bytes:

Table 3. Status Byte Format

Byte	Bit	Description
second: error statistics	7–5	not used
	4	error statistics for fifth block in frame; 0=no errors, 1=bad block
	3	error statistics for fourth block in frame; 0=no errors, 1=bad block
	2	error statistics for third block in frame; 0=no errors, 1=bad block
	1	error statistics for second block in frame; 0=no errors, 1=bad block
	0	error statistics for first block in frame; 0=no errors, 1=bad block
first: state of framer	7–4	If in flywheel, current flywheel count.
	3–1	not used
	0	0 = flywheel, 1 = lock

Reed-Solomon Encoder Registers

The following registers are implemented in the FPGA configuration file `sse_rs_decoder.bit` only.

Encoder Synchronization Pattern Register

Size	32-bit
I/O	read-write
Address	0x60, 0x61, 0x62, 0x63
Access	RS_ESYNC_PAT

Bit	Name	Description
31-0	ESYNC[31-0]	The 32-bit synchronization pattern for the decoder.

Encoder Control Register

Size	8-bit
I/O	read-write
Address	0x6C
Access	RS_ECTRL

Bit	Name	Description
7	BITSWAP	Set to bitswap the byte data before encoding.
6-5		not used
4	RND_ON	Set to turn on the randomizer
3-2		not used
1	RS_ENC_ON	Set to turn on the Reed Solomon encoder.
0		not used