



User's Guide

SRXL2

Mezzanine Board



**Signal receiver and processor
for L-band and IF, v.2
for use with an EDT main board**

Rev. 2010 April 09

Engineering Design Team (EDT), Inc.

1400 NW Compton Drive, Suite 315

Beaverton, OR 97006

p 503-690-1234 / 800-435-4320

f 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†.

© 1997-2010 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., 20230, U.S.A.

Limitation of Rights. Seller grants Buyer a royalty-free right to modify, reproduce, and distribute executable files using the Seller's Software and Firmware, provided that: a) the source code and executable files will be used only with Seller's Hardware; b) Buyer agrees to indemnify, hold harmless, and defend Seller from and against any claims or lawsuits, including attorneys' fees, that arise or result from the use or distribution of Buyer's products containing Seller's Products. Seller's Hardware may not be copied or recreated in any form or by any means without Seller's express written consent.

No Liability for Consequential Damages. In no event will Seller, its directors, officers, employees, or agents be liable to Buyer for any consequential, incidental, or indirect damages (including damages for business interruptions, loss of business profits or information, and the like) arising out of the use or inability to use the Products, even if Seller has been advised of the possibility of such damages. Because some jurisdictions do not allow the exclusion or limitation of liability for consequential or incidental damages, the above limitations may not apply to Buyer. Seller's liability to Buyer for actual damages for any cause whatsoever, and regardless of the form of the action (whether in contract, product liability, tort including negligence, or otherwise) will be limited to fifty U.S. dollars (\$50.00).

Limited Hardware Warranty. Seller warrants that the Hardware it manufactures and sells shall be free of defects in materials and workmanship for a period of 12 months from date of shipment to initial Buyer. This warranty does not apply to any product that is misused, abused, repaired, or otherwise modified by Buyer or others. Seller's sole obligation for breach of this warranty shall be to repair or replace (F.O.B. Seller's plant, Beaverton, Oregon, USA) any goods that are found to be non-conforming or defective as specified by Buyer within 30 days of discovery of any defect. Buyer shall bear all installation and transportation expenses, and all other incidental expenses and damages.

Limitation of Liability. *In no event shall Seller be liable for any type of special consequential, incidental, or penal damages, whether such damages arise from, or are a result of, breach of contract, warranty, tort (including negligence), strict liability, or otherwise.* All references to damages herein shall include, but not be limited to: loss of profit or revenue; loss of use of the goods or associated equipment; costs of substitute goods, equipment, or facilities; downtime costs; or claims for damages. Seller shall not be liable for any loss, claim, expense, or damage caused by, contributed to, or arising out of the acts or omissions of Buyer, whether negligent or otherwise.

No Other Warranties. Seller makes no other warranties, express or implied, including without limitation the implied warranties of merchantability and fitness for a particular purpose, regarding Seller's Products or Documentation. Seller does not warrant, guarantee, or make any representations regarding the use or the results of the use of the Products or Documentation or their correctness, accuracy, reliability, currentness, or otherwise. All risk related to the results and performance of the Products and Documentation is assumed by Buyer. The exclusion of implied warranties is not permitted by some jurisdictions. The above exclusion may not apply to Buyer.

Disclaimer. Seller's Products and Documentation, including this document, are subject to change without notice. Documentation does not represent a commitment from Seller.

Contents

Overview	1
FPGAs: Mezzanine Board + Main Board	1
Related Resources	2
Installation	3
Included Files	3
Loading the Firmware	6
Getting the Board ID	7
Board Architecture	7
IF Direct Module (IDM)	8
IF Mixer Module (IMM)	9
L-band Module (LBM)	10
General Board Control	11
Power	12
Reference Clock	12
Time Code	12
Programming the SRXL2 Devices	12
Sample Clock	13
Oscillators (IMM and LBM Only)	13
Gain	15
Graychips	15
Acquiring a Signal	16
PCIe8 LX	16
PCI GS	16
PRBS Test Pattern	17
Registers	18
0x00 Command	18
0x10 Channel Enable Low – DMA Channels 7–0	18
0x11 Channel Enable High – DMA Channels 15–8	18
0x50 Sample Clock Control	18
0x51 Sample Clock Read – Channel 0	19
0x52 Sample Clock Write Address – Channel 0	19
0x53 Sample Clock Write Data – Channel 0	19
0x54 Serial Control	19
0x55 FIFO Overflow Low – DMA Channels 7–0	19
0x56 FIFO Overflow High – DMA Channels 15–8	19
0x57 FIFO Control	20
0x58 DAC A Low – Channel 1	20
0x59 DAC A High – Channel	20
0x5A DAC B Low – Channel 0	20
0x5B DAC B High – Channel 0	21
0x5C SPI Data	21
0x5D SPI Status and Control	21
0x5E SPI Strobe	21
0x5F Board Control	22
0x60 LED Control	22
0x61 FIFO Underflow Low – DMA Channels 7–0	22
0x62 FIFO Underflow High – DMA Channels 15–8	22
0x63 Capture Control	23
0x64 Graychip Address	23
0x65 Graychip Data	23
0x66 Power Control	23

0x67 Oscillator Control..... 24
0x68 – 0x6F Oscillator Frequency 1 – 8 24
0x71 Sample Clock Read – Channel 1 24
0x72 Sample Clock Write Address – Channel 1 25
0x73 Sample Clock Write Data – Channel 1 25
Connectors and Pins..... 26
Pinouts 27
Revision Log 38

SRXL2

Overview

The SRXL2, an enhanced version of the SRXL, is a mezzanine board that digitizes and captures IF and L-band radio frequency (RF) signals. For specifications, see [Related Resources on page 2](#).

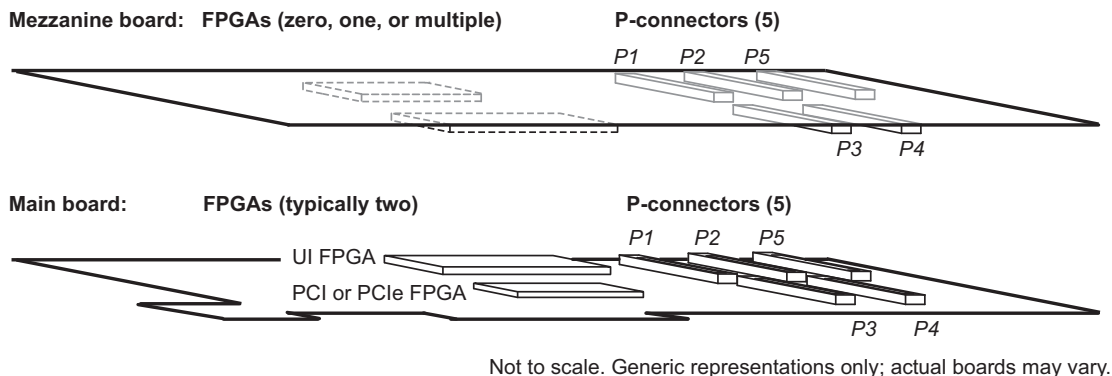
NOTE The SRXL2 was revised significantly in revision 03. If your board has an earlier revision number (see white sticker on back of board), contact EDT Sales to exchange it for a newer one.

The SRXL2 is paired with an EDT main board (PCIe8 LX / FX or PCI GS) 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 has FPGAs and P-connectors (to link the two boards), as in [Figure 1](#).

Figure 1. Generic EDT Board Pair



In specific, your SRXL2 board pair has the following FPGAs.

- Your SRXL2 mezzanine board has one FPGA (called the *SRXL2 FPGA*). To load this FPGA, see [Installation on page 3](#).
- Your PCIe8 LX / FX or PCI GS main board has two FPGAs:
 - The *PCI FPGA* links the PCI / PCIe bus on the host computer to the SRXL2 FPGA.
At powerup, the PCI FPGA automatically loads the correct firmware from the main board's flash ROM and implements DMA between the board and the host.
 - The *UI FPGA* links the user interface (UI) on the user device to the PCI FPGA.
To load the UI FPGA, see [the EDT Main Board User's Guide \(Related Resources, below\)](#).

Related Resources

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

<i>Resource</i>	<i>EDT webpage</i>
SRXL2 specifications (datasheet)	www.edt.com (go to SRXL2 page)
EDT Main Board User's Guide	www.edt.com/manuals/PCD/main_boards.pdf
Application Programming Interface	www.edt.com/api
Installation packages (software downloads):	www.edt.com/software.html
Windows, Linux, Solaris, Mac OS	

The third-party resources below can provide specifications for parts used on the SRXL2.

<i>Manufacturer</i>	<i>Part Number</i>	<i>Manufacturer's website</i>
Analog Devices	ADL5330	www.analog.com
"	ADF4360-8	"
"	ADF4360-7	"
Linear Technologies	LTC2242	www.linear.com
Silicon Labs	Si5326	www.silabs.com
Synergy Microwave	LFSW150320-50	www.synergymwave.com
Texas Instruments	GC4016	www.ti.com
"	MSP430F7222	"
"	TLV5617A	"

Installation

EDT provides installation packages for all supported operating systems (Windows, Linux, Solaris, Mac OS). These packages are provided on the EDT installation disk that ships with every EDT product.

However, to prevent installation package version issues, EDT recommends going to the EDT website and doing one of the following:

- For a new application, download the latest package.
- For an existing application, use the same package that was used to build it (from your own or EDT's archives), or recompile / relink the application with the latest installation package download.

In either case, to find the installation package you need (either the latest one or an archived version), see [Related Resources on page 2](#).

Included Files

When installation is complete, open the top-level EDT/PCD directory to find the subdirectories and other resources that pertain to the SRXL2.

- Relevant subdirectories include `bitfiles`, `flash`, `pci_config`, and `srxl`.
- Relevant other resources include applications and utilities.

bitfiles

This subdirectory contains one FPGA configuration file for each user-configurable FPGA available on your board pair (SRXL2 + PCIe8 LX / FX or PCI GS). For proper FPGA configuration, find and load the appropriate files as explained below.

SRXL2

For the SRXL2 mezzanine board: In `bitfiles`, find the `xc4vsx55` subdirectory. In that subdirectory, find and load the file below that correlates to the module option and main board you ordered.

<code>srxl2_top.bit</code>	This legacy file is not used with SRXL2 boards of rev03 and later. If your board is older than that, contact EDT Sales to exchange it for a newer version.
<code>srxl2_top_idmidm.bit</code>	Configures the SRXL2 with the IDM/IDM option for use with the PCI GS board.
<code>srxl2_top_idlmbm.bit</code>	Configures the SRXL2 with the IDM/LBM option for use with the PCI GS board.
<code>srxl2_top_immimm.bit</code>	Configures the SRXL2 with the IMM/IMM option for use with the PCI GS board.
<code>srxl2_top_immlbm.bit</code>	Configures the SRXL2 with the IMM/LBM option for use with the PCI GS board.
<code>srxl2_top_lx2_idmidm.bit</code>	Configures the SRXL2 with the IDM/IDM option for use with the PCIe8 LX board.
<code>srxl2_top_lx2_idlmbm.bit</code>	Configures the SRXL2 with the IDM/LBM option for use with the PCIe8 LX board.

`srxl2_top_lx2_immimm.bit`

Configures the SRXL2 with the IMM/IMM option for use with the PCIe8 LX board.

`srxl2_top_lx2_immlbm.bit`

Configures the SRXL2 with the IMM/LBM option for use with the PCIe8 LX board.

PCIe8 LX

For a PCIe8 LX main board: In `bitfiles`, find the correct subdirectory for the FPGA you ordered (XC5VLX110T / 220T / 330T). In that subdirectory, find and load the file below.

`srxl2_2in.bit`

Configures the PCIe8 LX UI FPGA for use with the SRXL2.

PCI GS

For a PCI GS main board: In `bitfiles`, find the correct subdirectory for the FPGA you ordered (XC2VP50 / XC2VP70). In that subdirectory, find and load the file below.

`srxl2_16in.bit`

Configures the PCI GS UI FPGA for use with the SRXL2; requires a 16-channel DMA interface (see [Loading the Firmware on page 6](#)).

flash

This subdirectory relates to the PCI FPGA on your main board, which loads automatically on powerup. When you receive your main board from EDT, it should come with the correct configuration file below already preloaded into flash ROM.

PCIe8 LX

In `flash`, find the `ep2sgx30d` subdirectory. In that subdirectory, find and load the file below.

`pe8lx16.bit`

Configures the PCI FPGA on the PCIe8 LX main board.

PCI GS

In `flash`, find the `xc2s200` subdirectory. In that subdirectory, find and load the appropriate file below.

NOTE For a 3.3- or 5.0-volt PCI bus, the correct firmware file is chosen automatically when you run `pciload`. Therefore, `pcigs16_classic.bit` is the correct argument for loading the firmware.

`pcigs16_classic_3v.bit`

Configures the PCI FPGA on a PCI GS main board installed in a 3.3-volt slot.

`pcigs16_classic_5v.bit`

Configures the PCI FPGA on a PCI GS main board installed in a 5.0-volt slot.

pci_config

This subdirectory contains `.cfg` files for software initialization and other purposes.

NOTE EDT does not currently provide `.cfg` files for the SRXL2. However, the following discussion of these files may prove helpful to you.

The `.cfg` files are editable text files that run like scripts to configure EDT boards and prepare the boards to perform DMA. The commands in these files are defined in a C application named `initpcd`. When you invoke `initpcd`, you specify which `.cfg` file to use with the `-f` flag.

A typical `.cfg` file loads an FPGA configuration file into the UI FPGA on the main board, and then sets up various registers to prepare for DMA transfers. Some `.cfg` files may load an FPGA configuration file into an FPGA residing on the mezzanine board.

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: srxl2_2in.bit
command_reg: 0x08
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 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`, contact EDT.

srxl

This subdirectory contains source files for the SRXL2 library functions and debugger application.

<code>lib_srxl2.c</code>	C source for routines called by the example applications. These routines in turn call routines in <code>libedt.c</code> , EDT's API library (see Related Resources on page 2).
<code>lib_srxl2.h</code>	A header file for <code>lib_srxl2.c</code> .
<code>srxl2_debugger.c</code>	C source for the <code>srxl2_debugger</code> application.
<code>srxl2_fft.c</code>	The C source for the <code>fft</code> application.

Applications and Utilities

In addition to the above subdirectories, the top-level `EDT/PCD` directory contains applications and utilities that you can use for board initialization and configuration, register access, and testing (in many cases, C source is included to give you a starting point for writing your own applications). Some commonly useful applications and utilities are listed below; for the complete list, see the `README` file.

<code>bitload</code>	Loads the configuration files for the UI FPGA on the main board.
<code>chkprbs15</code>	Checks DMA data against a 15-bit pseudorandom bit sequence in the specified channels.
<code>extbdid</code>	Displays the ID and revision number of the mezzanine board installed.
<code>initpcd</code>	Initializes and configures the mezzanine board.
<code>mezzload</code>	Loads the configuration files for the main board UI FPGA and the mezzanine board FPGA.
<code>pciload</code>	Shows currently installed EDT boards, outputs the date and revision number of the firmware in the PROM, and can be used to update the firmware in the PROM.
<code>pdb</code>	Enables interactive reading and writing of the registers on the main board UI FPGA and the mezzanine board FPGA.
<code>simple_getdata</code>	Provides an example of several DMA-related operations, including reading data from the connector interface, writing the data to a file, and measuring input rate.

<code>srxl2_debugger</code>	<p>Exercises most of the mezzanine board functions and provides debugging aids. The example application <code>srxl2_debugger</code> provides a way to:</p> <ul style="list-style-type: none"> - run a diagnostic <code>checksum</code> test on the Graychips; - read and write the SRXL2 and UI FPGA registers; - access and set values on the SRXL2 devices; - capture data from channel 0 or channel 1; and - perform complex Fast Fourier Transforms (FFT) on the captured data. <p>To run the example application, at the Pcd Utilities prompt, enter:</p> <pre>srxl2_debugger</pre> <p>At the SRXL2 debugger prompt, enter the <code>h</code> command for a list of all the usage options and their descriptions.</p>
<code>srxl2_fft.c</code>	Performs a Fast Fourier Transform (FFT) on both real and imaginary components of the captured data.
<code>srxl2_gc_diag</code>	Performs internal <code>checksum</code> tests on individual Graychips.

Building Applications

Executables and PCD source files are located in the `EDT_PCD` directory. Therefore, if you need to rebuild an application, run `make` in this top-level directory.

[Table 1](#) shows platform-specific details (namely, the recommended compiler and the location of the library and the debugger application) for each major platform.

Table 1. SRXL2 platform-specific details

	Recommended compiler	Library & debugger location
Windows	Install Microsoft Visual C compiler (or contact EDT to use <code>gcc</code>)	<code>srxl</code> directory
Solaris	Install Sun Workshop C compiler (or contact EDT to use <code>gcc</code>)	<code>./srxl/</code> subdirectory
Linux	Use <code>gcc</code> compiler (typically included with Linux installation)	<code>./srxl/</code> subdirectory

After building an application, enter the `--help` command line option to see a list of usage options and descriptions.

Loading the Firmware

For FPGA configuration, follow the instructions below, replacing *italic terms* (such as `unit_number`) with the appropriate values.

1. Run `pciload` to determine the unit number of the PCIe8 LX / FX or PCI GS main board (by default, 0), and to verify that the host detects the main board and that the main board is loaded with the appropriate firmware.
2. If the PROM ID includes the string `pcigs16` or `pcie8LX16`, then 16-channel firmware is already loaded. If not, load the appropriate firmware with the appropriate command:

```
pciload -u unit_number firmware_file
(where firmware_file is either pcigs16_classic or pe8lx16)
```

3. At the prompt, press Enter to confirm the loading operation.
4. Power-cycle the system.

5. Load the correct configuration files for the main board UI FPGA and the mezzanine board FPGA:

```
mezzload -u unit_number -b mezzanine_board_bitfile -B main_board_UI_FPGA
```

For example, a PCIe8 LX main board with an SRXL2 IDM/LBM option loaded in unit 0 would be:

```
mezzload -u 0 -b srxl2_top_lx_idmlbm.bit -B srxl2_2in.bit
```

6. Watch for the onscreen message which verifies that the files are loaded.

Getting the Board ID

After you install the SRXL2 and load the firmware, you'll need to get the extended board ID and revision information. To do so, see [Connectors and Pins on page 26](#).

Board Architecture

The SRXL2 has two independent channels; the module options for each channel are shown below.

- Channel 0 = IF direct module (IDM) **or** IF mixer module (IMM)
- Channel 1 = IF direct module (IDM) **or** IF mixer module (IMM) **or** L-band module (LBM)

Each module can receive, sample, and digitize either IF or L-band signals. The signals are then captured in the SRXL2 FPGA. From there, the digitized data can be routed as follows:

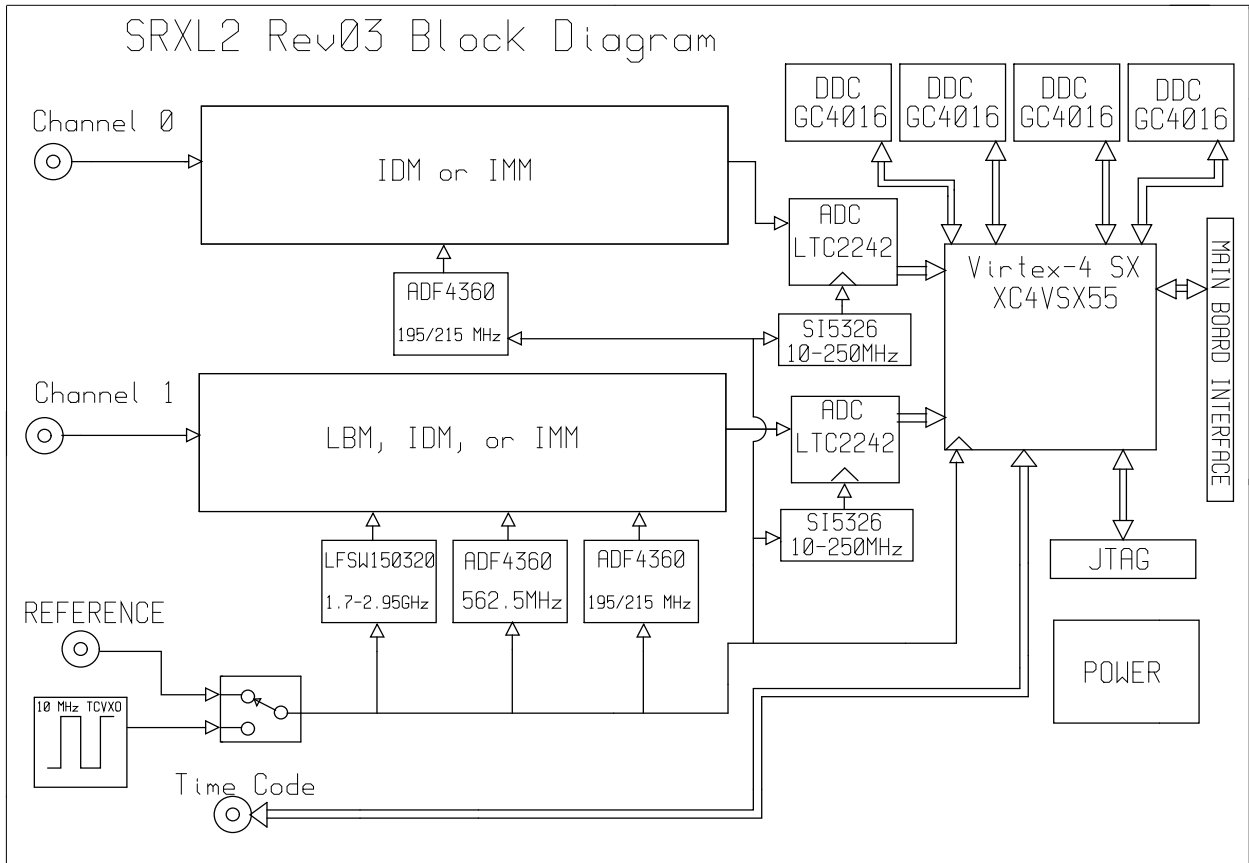
- to four optional Graychips for a total of sixteen channels of digital down-conversion (DDC); or
- to the UI FPGA on the main board for more processing or for DMA transfer to the host computer.

Resources in both the SRXL2 FPGA and the main board UI FPGA are fully user-programmable.

The analog-to-digital converters (ADC) on the SRXL2 are 12-bit, and the sample clocks for each channel can be individually programmed from 10 to 250 MHz. The board provides an internal 10 MHz TCXO reference clock as the timebase, but also has an external input so that you can provide your own if desired. There is also a time code input that accepts IRIG-B or one-pulse-per-second (1pps) format for time stamping acquired data.

The SRXL2 architecture is shown in [Figure 2](#).

Figure 2. SRXL2 (rev03)



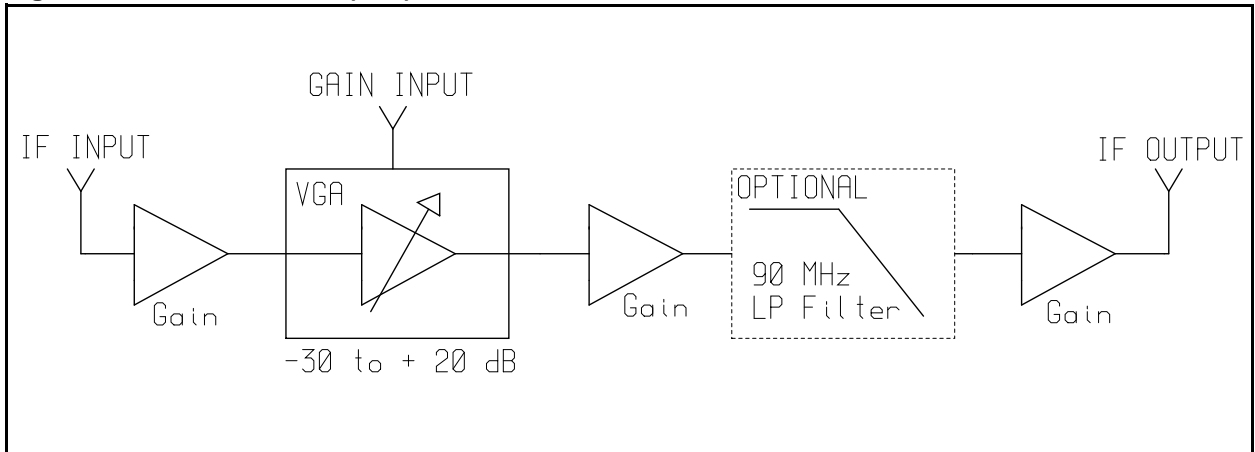
IF Direct Module (IDM)

The standard IDM configuration accepts IF frequencies from 10 to 200MHz and has 75-ohm input signaling. This configuration has a 300MHz low-pass filter to eliminate high frequency signals from aliasing into the desired bandwidth. Optionally, 50-ohm input signaling can be ordered, as well as a 90MHz low-pass antialias filter. The bandwidth of a particular IF depends on external filtering (not provided). The IDM output signal is sampled directly by the ADC on the SRXL2.

The dynamic range on the input is about 45dB, accepting input power levels of -20 to -65dBm. A digital-to-analog converter (DAC) on the SRXL2 is used to provide a control input to a variable gain amplifier (VGA). This VGA is used to adjust the drive level to the ADC.

The sample clock for this module can be programmed from 10 to 250 MHz.

Figure 3 (below) shows a block diagram of the IDM.

Figure 3. IF Direct Module (IDM)

If you ordered this module with the 90 MHz filter, you can use it for IFs below 90 MHz.

If you ordered this module without the 90 MHz filter, it ships with a 300 MHz lowpass filter in place of the 90 MHz filter. The lowpass filter can be set for IFs of 70, 140, or 160, or programmed for other IFs from 1 to 200 MHz.

To set up this module:

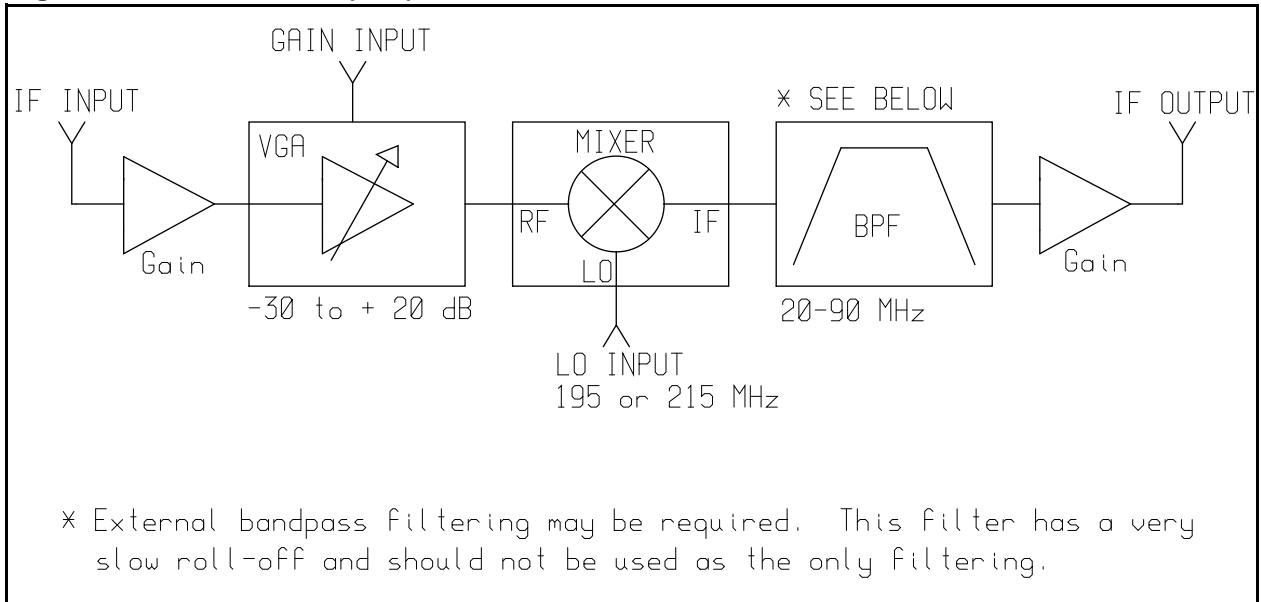
1. Turn on channel power; see the [0x66 Power Control](#).
2. Select either the onboard 10-MHz TCXO reference or an external reference; see the [0x5F Board Control](#).
3. Program the sample clock; see [Sample Clock on page 13](#).
4. Program the gain; see [Gain on page 15](#).
5. Acquire a signal; see [Acquiring a Signal on page 16](#).

IF Mixer Module (IMM)

The IMM accepts IF frequencies of 140 or 160 MHz. The IF is mixed with a local oscillator (either 195 or 215MHz) to create a 55MHz signal. The IF bandwidth is determined by external filtering (not provided) – typically around 70MHz or less. The 55MHz signal is what gets digitized using the ADC on the SRXL2.

This module accepts input power levels of -20 to -65dBm; the dynamic range on the input is about 45dB. A digital-to-analog converter (DAC) on the SRXL2 provides a control input to a variable gain amplifier (VGA). This VGA is used to adjust the drive level to the ADC. The sample clock for this module typically is set to 250MHz, but it may be programmed to lower frequencies if necessary.

[Figure 4](#) (below) shows a block diagram of the IMM.

Figure 4. IF Mixer Module (IMM)

To set up this module:

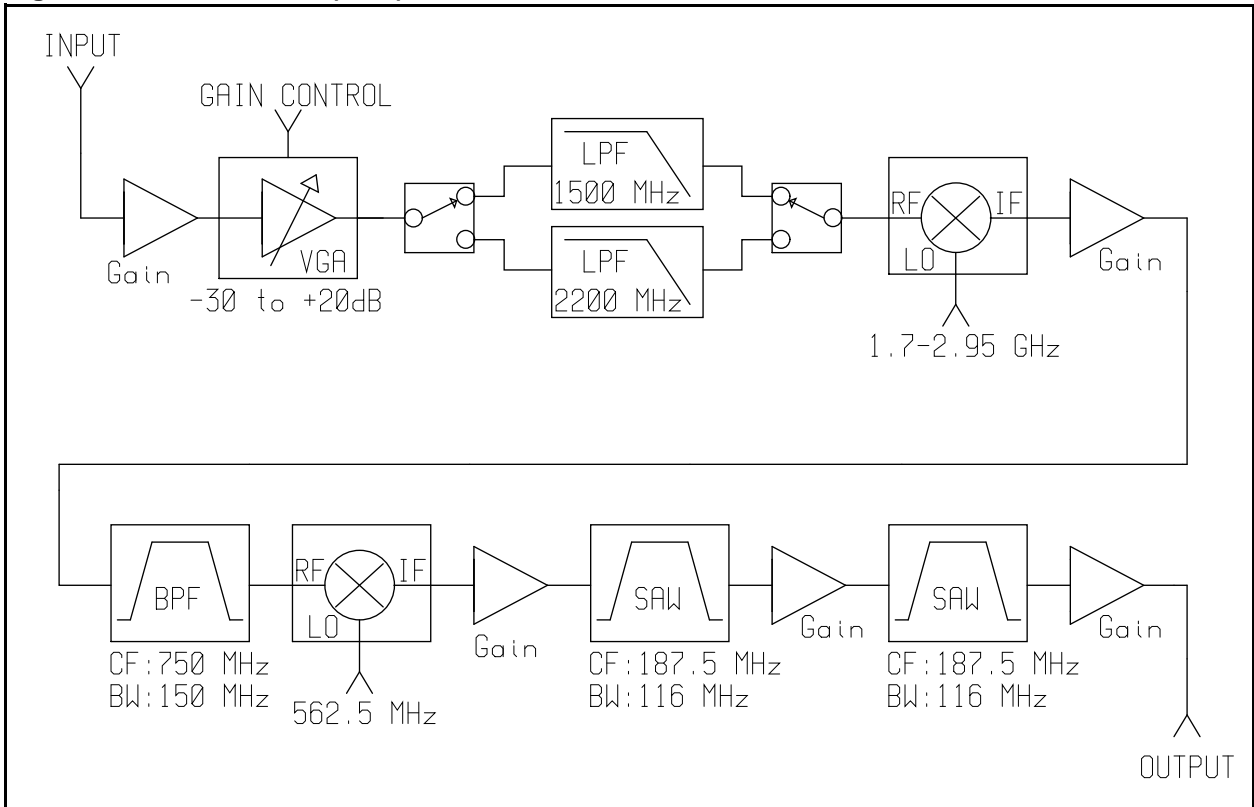
1. Turn on channel power; see the [0x66 Power Control](#).
2. Select either the onboard 10-MHz TCXO reference or an external reference; see the [0x5F Board Control](#).
3. Program the sample clock; see [Sample Clock on page 13](#).
4. Program the gain; see [Gain on page 15](#).
5. Set the input IF to a frequency of 140 or 160MHz, update the local oscillator, and make sure the oscillator has locked; see the [0x67 Oscillator Control](#).
6. Acquire a signal; see [Acquiring a Signal on page 16](#).

L-band Module (LBM)

The LBM (see [Figure 5](#) below) receives signals in the range of 900 to 2200 MHz and down-converts the desired center frequency to a 116-MHz band centered at 187.5 MHz. Variable gain for input power levels of -20 to -65 dBm is provided through a DAC, which the user must program and adjust until the ADC is driven to full scale.

The L-band path has two mixers, each with its own local oscillator (LBM1 and LBM2):

- The LBM1 oscillator (range = 1700 to 2950 MHz) must be set to a frequency that is 750 MHz higher than the desired input center frequency; for instance, to digitize the 116 MHz band centered at 1000 MHz, set this oscillator to 1750 MHz. There also is a selectable filter which must be set by the user.
- The LBM2 oscillator must be set to a fixed frequency of 562.5 MHz. The EDT firmware automatically does this for you when the FPGA configuration file is loaded on the SRXL2.

Figure 5. L-band Module (LBM)

To set up this module:

1. Turn on channel power; see the [0x63 Capture Control](#).
2. In the [0x67 Oscillator Control](#), verify that bit 5 (CH1_LO2_LOCK) is set. If not, toggle bit 0 (SERIAL_RESET) in the [0x54 Serial Control](#) and check again, repeating until lock is achieved.
3. Select either the onboard 10-MHz reference oscillator or the external reference input; see the [0x5F Board Control](#).
4. Set the sample clock to 250 MHz (the EDT-provided firmware does this for you when the FPGA configuration file is loaded on the SRXL2); see [Sample Clock on page 13](#).
5. Set the oscillators; see [Oscillators \(IMM and LBM only\) on page 13](#).
6. Select the appropriate filters for the signal path and the LBM1 oscillator, depending on the desired frequency; see [General Board Control on page 11](#).
7. Set the gain; see [Gain on page 15](#).
8. Acquire a signal; see [Acquiring a Signal on page 16](#).

General Board Control

On the SRXL2, you can control capabilities related to power, reference clock, and time code. The location of the components that support these capabilities are shown in [Figure 6 on page 26](#).

Power

On the SRXL2, when both channels and all other components are fully powered up and running, the power requirements may exceed PCI/PCIe specifications for power per slot. If you have this problem, try these possible solutions:

- Connect the auxiliary power cable on the board to any 12-volt power source to boost power.
- If you're using only one module channel, power down the other module channel; see the [0x66 Power Control](#).

Reference Clock

The SRXL2 requires a 10 MHz signal from a reference clock, to which all other clocks on the board are phase-locked. For this reference clock, you can select either the internal TCXO or an external reference via bit 7 (EXT_REF_SEL) of the [0x5F Board Control](#).

Time Code

The SRXL2 accepts time code input (IRIG-B or 1 pps) for timestamping acquired data, and the time code circuitry on the SRXL2 is the same as that found on the EDT Time Distribution board.

Therefore, for details on the time code circuitry and the 7-pin Lemo pinout, see the three registers in the Time Distribution User's Guide. Those three registers also appear in this user's guide, but with different addresses, as shown below.

Table 2. SPI Registers

	SRXL2 mezzanine board	Time Distribution auxiliary board
SPI Data Register	0x5C	0x60
SPI Status and Control Register	0x5D	0x61
SPI Strobe Register	0x5E	0x62

Programming the SRXL2 Devices

For each channel (channel 0 and channel 1) on the SRXL2, you'll need to program certain components by reading and writing the appropriate registers in the FPGA. These components are:

- the sample clock synthesizer;
- the IF and L-band oscillators;
- the gain adjustments.

Also, you'll need to program the Graychips if you ordered them.

The firmware serializes your instructions and sends the resulting data or instructions to the target device. These serial transfers are slower than the register programming interface.

Therefore, to prevent errors in the serial datastream, verify that bit 7 (XFER_BUSY) of the [0x54 Serial Control](#) is clear before you start a new register access.

Sample Clock

For each channel, the module's output is sampled and digitized using an LTC2242 12-bit ADC.

For each ADC, the sample clock rate is generated by an Si5326 low-jitter precision clock multiplier. For parts information, see [Related Resources on page 2](#).

On each module channel, the sample clock is programmable as shown below.

Table 3. SRXL2 sample clock programming (by channel and module)

Channel	Module	Sample clock parameters
0	IDM	Programmable from 10 to 250 MHz
	IMM	Programmable from 10 to 250 MHz
1	IDM	Programmable from 10 to 250 MHz
	IMM	Programmable from 10 to 250 MHz
	LMB	Must be set to 250 MHz

NOTE Sampling rates below 10 MHz or above 250 MHz are not possible in any configuration.

For each channel you are using, write the appropriate Si5326 register to program the sample clock:

1. Write the address of the Si5326 target register to either the [0x52 Sample Clock Write Address – Channel 0](#) or the [0x72 Sample Clock Write Address – Channel 1](#).
2. Write the desired value to either the [0x53 Sample Clock Write Data – Channel 0](#) or the [0x7F Board ID](#).

NOTE Writing to register 0x53 or register 0x73 will cause the hardware to serialize the data and send it to the Si5326. During this data transfer, bit 7 (XFER_BUSY) in the [0x54 Serial Control](#) is set. Monitor this bit until you see that it is clear before starting any new register access.

For each channel you are using, read from the appropriate Si5326 register:

1. Write the address of the Si5326 target register to either the [0x51 Sample Clock Read – Channel 0](#) or the [0x71 Sample Clock Read – Channel 1](#).
2. Monitor bit 7 (XFER_BUSY) in the [0x54 Serial Control](#) until you see that it is clear.
3. Read the value in either the [0x51 Sample Clock Read – Channel 0](#) or the [0x71 Sample Clock Read – Channel 1](#).

Oscillators (IMM and LBM only)

The IDM option does not have oscillators, but the IMM and LBM options do.

To program these oscillators, follow the steps below.

IMM oscillator (channel 0 or 1)

This oscillator is part of the IMM option, which can be ordered on channel 0, channel 1, or both.

- If the IF is 140 MHz, this oscillator must be set to 195 MHz.
- If the IF is 160 MHz, this oscillator must be set to 215 MHz.

To program the IMM, use the [0x67 Oscillator Control](#) and follow the steps below.

Channel 0

1. Program the frequency:
 - For 140 MHz, set bit 2 (CH0_LO_FREQ_SEL).
 - For 160 MHz, clear bit 2 (CH0_LO_FREQ_SEL).
2. To update, set bit 0 (CH0_LO_UPDATE); this bit clears itself after the update is completed.
3. Check bit 3 (CH0_LO_LOCK); if it is set, the oscillator is locked to the new frequency.

Channel 1

1. Program the frequency:
 - For 140 MHz, set bit 6 (CH1_LO_FREQ_SEL).
 - For 160 MHz, clear bit 6 (CH1_LO_FREQ_SEL).
2. To update, set bit 4 (CH1_LO_UPDATE); this bit clears itself after the update is completed.
3. Check bit 7 (CH1_LO_LOCK); if it is set, the oscillator is locked to the new frequency.

LBM1 oscillator – channel 1 only

For L-band, the first oscillator (LBM1) uses an LFSW150320-50 synthesizer, programmable from 1.5 to 3 GHz in 500 KHz steps (for parts information, see [Related Resources on page 2](#)). Set its frequency to 750 MHz higher than the desired input center frequency, as outlined in [Board Architecture on page 7](#).

1. To determine the desired frequency word: Convert the desired frequency in KHz to hexadecimal (if necessary, pad with zeroes until you have eight characters) and convert each character to its ASCII hexadecimal value, as shown in [Table 4](#).

Table 4. Hexadecimal to ASCII Conversion

Hexadecimal	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ASCII	30	31	32	33	34	35	36	37	38	39	41	42	43	44	45	46

2. Write the desired frequency as a 64-bit word to the 8-bit oscillator frequency registers ([0x68 – 0x6F Oscillator Frequency 1 – 8](#)). Write 0x6F last, because writes to that register will cause the hardware to serialize the data in all eight registers and send it to the oscillator specified in [step 1](#).
3. To verify that the oscillator was updated, monitor bit 7 (CH1_LO_LOCK) of the [0x67 Oscillator Control](#). When this bit is set, the tuner is locked to the programmed frequency.

For example, if the desired frequency is 2 GHz, follow the steps below.

1. Convert 2.75 GHz to kHz to get 2750000 kHz.
2. Convert 2750000 kHz to hexadecimal to get 0x29F630.
3. Since the result is only six characters but eight characters are required, pad with zeroes to get 0x0029F630.
4. Convert each character to its ASCII equivalent to get 30 30 32 39 46 36 33 30.
5. Write the [0x68 – 0x6F Oscillator Frequency 1 – 8](#) as follows: 30 to 0x68 (frequency 1), 30 to 0x69 (frequency 2), 32 to 0x6A (frequency 3), and so on through all the oscillator frequency registers until you write 30 to 0x6F (frequency 8) last. Writes to 0x6F will trigger the serialization and transfer of all eight register values to the selected oscillator.

LBM2 oscillator (channel 1 only)

The SRXL2 also includes an interface to a second-stage L-band oscillator, which you must set to 562.5 MHz if you write your own VHDL. When you toggle bit 0 (SERIAL_RESET) in the [0x54 Serial Control](#), the EDT VHDL automatically sets the oscillator to that frequency. To verify that this oscillator is locked, check bit 5 (CH1_LO2_LOCK) in the [0x67 Oscillator Control](#).

Gain

Each channel provides a variable-gain amplifier (VGA), with 60 dB gain control range, to protect its ADC from overrange conditions. The gain control is set via analog voltage provided by a 10-bit DAC.

- The A channel sets the gain on channel 1.
- The B channel sets the gain on channel 0.

Each DAC channel accepts 12-bit data but ignores the two least significant bits, providing 10-bit resolution. Each DAC output provides a gain control signal adjustable from 0–1.4 V.

If you wish to write your own application to set the gain, the formula to do so is:

$$\text{code} = (V/3) * 1024$$

where code is the numeric value to input, and V is the desired voltage. The numeric value must then be converted to hexadecimal. For example, for a gain voltage of 0.75 V:

1. Divide 0.75 by 3 = 0.25.
2. Multiply 0.25 by 1024 = 256.
3. Convert decimal 256 to hexadecimal = 0x100.

Therefore, for a gain of 0.75 V, write a 0x00 and a 0x01 to the two registers for DAC A or DAC B, depending on which gain you wish to set.

You write the gain-control DAC channels through four byte-wide registers: DACA_LOW, DACA_HIGH, DACB_LOW, and DACB_HIGH, described below. After either the DACA_HIGH or DACB_HIGH register is loaded, both DAC channels update with the current contents of their respective registers.

Graychips

The SRXL2 has four Texas Instruments [GC4016](#) Graychips for digital down-conversion. They are connected to the FPGA, so your software can access them in whatever manner required. Each must be programmed separately by writing to 8-bit control registers. To do so:

1. Write the address of the desired control register into the GC_ADDR register.
2. Write the 8-bit data into the GC_DATA register. To ensure that data goes to the correct register, write the address register first; then write to the Graychip occurs immediately after the GC_DATA register has been written.

To read the data from one of the registers:

1. Specify the register to read by writing its address to GC_ADDR. The most significant bit (GC_SELECT) determines which Graychip you read.
2. Read the GC_DATA register.

For local Graychip register descriptions, see the link under [Related Resources on page 2](#).

NOTE The SRXL2 debugger application allows you to run a diagnostic `checksum` on the Graychips.

Acquiring a Signal

For acquiring a signal, the EDT FPGA configuration files work differently depending on the main board.

- For the PCI GS main board, EDT's configuration files provide a 500 KB snapshot of the data, which is then transferred via DMA to the host computer.
- For the PCIe8 LX main board, EDT's configuration files enable continuous capture and transfer of data, via DMA, to the host computer.

For example application code that you can use to acquire data, see the `srxl2_debugger` source file.

PCIe8 LX

If you are using a PCIe8 LX main board with your SRXL2, follow these steps to start acquiring a signal:

1. Clear and then set bit 3 (CMD_EN) of the [0x10 Channel Enable Low – DMA Channels 7–0](#) to reset and enable DMA FIFOs in the UI FPGA.
2. Make the appropriate EDT driver call(s) to start DMA on the main board.
3. Set the appropriate bit CH_ENABLE (bit 0 or bit 1) of the [0x10 Channel Enable Low – DMA Channels 7–0](#) to enable DMA from either the channel 0 module or the channel 1 module.
4. For each channel you are acquiring from, check the appropriate overrange flag:
 - a. If you are acquiring from channel 0, verify that bit 5 (CH0_OTR) is clear. If it is set, you must lower the gain of the module and start this acquisition process over; see [Gain on page 15](#).
 - b. If you are acquiring from channel 1, verify that bit 6 (CH1_OTR), is clear. If it is set, you must lower the gain of the module and start this acquisition process over; see [Gain on page 15](#).

PCI GS

If you are using the PCI GS main board with your SRXL2, follow these steps to start acquiring a signal:

1. Clear and then set bit 3 (CMD_EN) of the [0x10 Channel Enable Low – DMA Channels 7–0](#) to reset and enable DMA FIFOs in the UI FPGA.
2. Set bit 1 (CH_ENABLE) of the [0x10 Channel Enable Low – DMA Channels 7–0](#) to enable DMA channel 0.
3. Toggle bit 1 (FIFO_RESET) of the [0x57 FIFO Control](#) to reset 500kB snapshot FIFO.
4. Set the INPUT_SEL (bit 0) of the [0x63 Capture Control](#) to acquire data from the module on channel 1; clear for the module on channel 0.
5. Set bit 7 (CAPTURE) of the [0x63 Capture Control](#) to initiate snapshot capture. This bit will clear itself when the snapshot is complete; wait for the bit to clear before proceeding.
6. Check bit 4 (FULL) of the [0x57 FIFO Control](#) to make sure the FIFO is full.
7. For each channel you are acquiring from, check the appropriate overrange flag:
 - a. If you are acquiring from channel 0, verify that bit 5 (CH0_OTR) is clear. If it is set, you must lower the gain of the module and start this acquisition process over; see [Gain on page 15](#).
 - b. If you are acquiring from channel 1, verify that bit 6 (CH1_OTR), is clear. If it is set, you must lower the gain of the module and start this acquisition process over; see [Gain on page 15](#).
8. Immediately make the appropriate EDT driver call(s) to start DMA on the main board.

PRBS Test Pattern

To test DMA transfer, the SRXL2 can automatically generate a PRBS15 test pattern, which in turn can be used with the `checkprbs15` application software to verify the DMA is working.

To enable the PRBS15 test pattern, set bit 1 (PRBS_EN) in the [0x63 Capture Control](#).

- For the PCI GS main board, the PRBS test pattern is on DMA channel 1.
- For the PCIe8 LX main board, the PRBS test pattern is on DMA channel 0 – so when the test pattern is enabled, you cannot acquire data from the module on channel 0 of the SRXL2.

Registers

This section shows the SRXL2 registers, arranged by hex address.

0x00 Command

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

Bits	Name	Description
7-4	[no name]	not used
3	CMD_EN	Set to enable the required DMA channels in the 0x10 Channel Enable Low – DMA Channels 7–0 and 0x11 Channel Enable High – DMA Channels 15–8 . Clear to reset all DMA channels, flush the FIFOs, and clear all under- and overflow bits.
2-0	[no name]	not used

0x10 Channel Enable Low – DMA Channels 7–0

Access / Notes: SSD16_CHENL / 8-bit read-write

Bits	Name	Description
7-0	CH_ENABLE	Set to enable corresponding I/O channel; clear to reset.

0x11 Channel Enable High – DMA Channels 15–8

Access / Notes: SSD16_CHENH / 8-bit read-write

Bits	Name	Description
15-8	CH_ENABLE	Set to enable corresponding I/O channel; clear to reset.

0x50 Sample Clock Control

Access / Notes: SRXL2_SCLK_CTRL / 8-bit read-write

Bits	Name	Description
7	CH1_LOL	For channel 1. Read only; if set, the Si5326 has not locked to the sample clock.
6	CH1_LOS	For channel 1. Read only; if set, the Si5326 is not receiving a signal from the reference clock.
5	[no name]	not used
4	CH1_RESET	For channel 1. Toggle to reset the Si5326.
3	CH0_LOL	For channel 0. Read only; if set, the Si5326 has not locked to the sample clock.
2	CH0_LOS	For channel 0. Read only; if set, the Si5326 is not receiving a signal from the reference clock.
1	[no name]	not used
0	CH0_RESET	For channel 0. Toggle to reset the Si5326.

0x51 Sample Clock Read – Channel 0

Access / Notes: SRXL2_CH0_SCLK_READ / 8-bit read-write

For channel 0. Write the address of the desired Si5326 register. When bit 7 (XFER_BUSY) in the [0x54 Serial Control](#) is clear, read the data from the requested register.

0x52 Sample Clock Write Address – Channel 0

Access / Notes: SRXL2_CH0_SCLK_ADDR / 8-bit read-write

For channel 0. The address of the Si5326 register to which you wish to write.

0x53 Sample Clock Write Data – Channel 0

Access / Notes: SRXL2_CH0_SCLK_DATA / 8-bit read-write

For channel 0. The data that you wish to write to the Si5326 register specified by the address in the [0x52 Sample Clock Write Address – Channel 0](#).

0x54 Serial Control

Access / Notes: SRXL2_SERIAL_CTRL / 8-bit read-write

Control and status bits for functions accessed by serial interface.

Bits	Name	Description
7	XFER_BUSY	Read only. The firmware sets this bit when any of the serial interfaces are busy; wait until it is clear before starting any new read or write to any SRXL2 device.
6–1	[no name]	not used
0	SERIAL_RESET	Toggle to reset all SRXL2 serial devices. This bit is toggled each time the FPGA configuration file is loaded.

0x55 FIFO Overflow Low – DMA Channels 7–0

Access / Notes: SRXL2_FIFO_OVERFLOW_LOW / 8-bit read-write

Applies only to the PCIe8 LX main board.

Bits	Name	Description
7–0	CH_OVFL	Read only. Overflow flags for DMA channels 7–0.

0x56 FIFO Overflow High – DMA Channels 15–8

Access / Notes: SRXL2_FIFO_OVERFLOW_HIGH / 8-bit read-write

Applies only when using the PCIe8 LX main board.

Bits	Name	Description
15–8	CH_OVFL	Read only. Overflow flags for DMA channels 15–8.

0x57 FIFO Control

Access / Notes: SRXL2_AD_FIFO_CTRL / 8-bit read-write

Applies only to the PCI GS main board.

Bits	Name	Description
7	UNDERFLOW	Read only. When set, indicates that a FIFO underflow has occurred. Cleared when FIFO is reset (bit 1).
6	OVERFLOW	Read only. When set, indicates that a FIFO overflow has occurred. Cleared when FIFO is reset (bit 1).
5	EMPTY	Read only. When set, indicates that FIFO is empty.
4	FULL	Read only. When set, indicates that FIFO is full.
3–2	[no name]	not used
1	FIFO_RESET	Toggle to reset the FIFO.
0	[no name]	not used

0x58 DAC A Low – Channel 1

Access / Notes: SRXL2_DACA_LOW / 8-bit read-write

Control and status bits for functions accessed by serial interface.

Bits	Name	Description
7–2	DACA_LOW	For channel 1. Least significant component of voltage specification for the signal gain.
1–0	[no name]	not used

0x59 DAC A High – Channel 1

Access / Notes: SRXL2_DACA_HIGH / 8-bit read-write

After either the DAC A High or the DAC B High register is loaded, both A and B channels are updated with the current contents of their respective register.

Bits	Name	Description
7–4	[no name]	not used
3–0	DACA_HIGH	For channel 1. Most significant component of voltage specification for the signal gain.

0x5A DAC B Low – Channel 0

Access / Notes: SRXL2_DACB_LOW / 8-bit read-write

After either the DAC A High or the DAC B High register is loaded, both A and B channels are updated with the current contents of their respective register.

Bits	Name	Description
7–2	DACB_LOW	For channel 0. Least significant component of voltage specification for the signal gain.
1–0	[no name]	not used

0x5B DAC B High – Channel 0

Access / Notes: SRXL2_DACB_HIGH / 8-bit read-write

After either the DAC A High or the DAC B High register is loaded, both A and B channels are updated with the current contents of their respective register.

Bits	Name	Description
7–4	[no name]	not used
3–0	DACB_HIGH	For channel 0. Most significant component of voltage specification for the signal gain.

0x5C SPI Data

Access / Notes: SRXL2_SPI_DATA / 8-bit read-write

When read, bits read from the input FIFO; when written, bits write to the output FIFO.

0x5D SPI Status and Control

Access / Notes: SRXL2_SPI_STAT_CTRL / 8-bit read-write

Bits	Name	Description
7	CHIP_ENABLE	Detects which main board is connected to the SRXL2 master header. A value of one indicates the master header; otherwise reads zero.
6	IRIGB_PULSE_IN	When set, indicates that the one-pulse-per-second signal has been detected from the incoming IRIG signal, and it has been processed by, and passed through, the MSP430.
5	MODEM_PULSE_IN	When set, indicates that the one-pulse-per-second signal has been detected from the satellite modem and passed directly through, without any processing by the MSP430.
4	[no name]	not used; reads as zero
3	FIFO_OUT_EMPTY	When set, indicates the output FIFO is empty.
2	FIFO_OUT_FULL	When set, indicates the output FIFO is full.
1	FIFO_IN_EMPTY	When set, indicates the input FIFO is empty.
0	When written: RESET When read: FIFO_IN_OV	On write: toggle this bit to reset the SPI data path. On read: when set, indicates the input FIFO has overflowed. Data may be lost.

0x5E SPI Strobe

Access / Notes: SRXL2_SPI_STROBE / 8-bit read-write

Write (any value) to this register to advance the input FIFO.

0x5F Board Control

Access / Notes: SRXL2_AD_BOARD_CTRL / 8-bit read-write

Bits	Name	Description
7	EXT_REF_SEL	0: select the 10 MHz internal reference clock. 1: select an external 10 MHz reference clock.
6	[no name]	not used
5	DCM_LOCKED	Read only; if set, the internal DCM is locked.
4	DCM_RESET	Resets internal digital clock manager (DCM).
3–2	LB_SEL	01: 900-1500 MHz low-pass filtering for L-band oscillator. 10: 1500-2200 MHz low-pass filtering for L-band oscillator.
1–0	LBLO_SEL	01: 2000-2950 MHz low-pass filtering for L-band oscillator. 10: 1700-2000 MHz low-pass filtering for L-band oscillator.

0x60 LED Control

Access / Notes: SRXL2_LED_CTRL / 8-bit read-write

Bits	Name	Description
7–2	[no name]	not used
1–0	LED_ON	Controls the LEDs, useful for debugging 00 Both LEDs off 01 LED1 on (1 Hz blink) 10 LED2 on 11 Both LEDs on

0x61 FIFO Underflow Low – DMA Channels 7–0

Access / Notes: SRXL2_FIFO_UNDERFLOW_LOW / 8-bit read-write

Applies only to the PCIe8 LX main board.

Bits	Name	Description
7–0	CH_UNF	Read only. Underflow flags for DMA channels 7–0.

0x62 FIFO Underflow High – DMA Channels 15–8

Access / Notes: SRXL2_FIFO_UNDERFLOW_HIGH / 8-bit read-write

Applies only to the PCIe8 LX main board.

Bits	Name	Description
15–8	CH_UNFL	Read only. Underflow flags for DMA channels 15–8.

0x63 Capture Control

Access / Notes: SRXL2_AD_CAPTURE_CTRL / 8-bit read-write		
Bits	Name	Description
7	CAPTURE	Set to start capturing 1 MB of data into the FIFO. Clears itself when the FIFO is full.
6	CH1_OTR	For channel 1. Read-only. Set if this ADC reported an overrange on any data acquired since the last FIFO reset (bit 0 of the 0x63 Capture Control), meaning the input signal exceeded the range of the analog-to-digital converter. Remains set until the FIFO is reset.
5	CH0_OTR	For channel 0. Read-only. Set if this ADC reported an overrange on any data acquired since the last FIFO reset (bit 0 of the 0x63 Capture Control), meaning the input signal exceeded the range of the analog-to-digital converter. Remains set until the FIFO is reset.
4–2	[no name]	not used
1	PRBS_EN	For PCIe8 LX main board: - Set to enable PRBS data on DMA channel 0. - Clear to enable digitized data from IDM or IMM on DMA channel 0. For PCI GS main board: - Set to enable PRBS data on DMA channel 1.
0	INPUT_SEL	Set to capture data from the channel 1 input. Clear to capture data from the channel 0 input.

0x64 Graychip Address

Access / Notes: SRXL2_GC_ADDR / 8-bit read-write		
Bits	Name	Description
7–6	GC_SELECT	00 selects Graychip number 1. 01 selects Graychip number 2. 10 selects Graychip number 3. 11 selects Graychip number 4.
5	[no name]	not used
4–0	A	Address of register to read from or write to. Most significant bit is bit 4; least significant is bit 0.

0x65 Graychip Data

Access / Notes: SRXL2_GC_DATA / 8-bit read-write		
Bits	Name	Description
7–0	GC_DATA	Contains the data to write to the register specified in the Graychip Address register, or the data read from it

0x66 Power Control

Access / Notes: SRXL2_PWR_REG / 8-bit read-write		
Bits	Name	Description
7–2	[no name]	not used
1	CH1_PWRDOWN	For channel 1. Set to power down.
0	CH0_PWRDOWN	For channel 0. Set to power down.

0x67 Oscillator Control

Access / Notes: SRXL2_LO_CTRL / 8-bit read-write

For channel 1, bits 7–4 have dual meanings, depending on whether you ordered IMM or LBM in that channel.

For channel 0, bits 3–0 are applicable only if you ordered IMM on that channel

Bits	Name	Description
7	CH1_LO_LOCK	For channel 1 (LBM). Read only. If set, L-band oscillator 1 is locked.
6	CH1_LO_ERROR	For channel 1 (LBM). Read only. If set, L-band oscillator 1 has an internal error: loss of lock, invalid frequency, or hardware error.
5	CH1_LO_FREQ_SEL	For channel 1 (IMM). Set to program this oscillator for 140 MHz IF; clear for 160 MHz IF.
4	CH1_LO2_LOCK	For channel 1 (LBM). Read only. If set, L-band oscillator 2 is locked.
3	CH1_LO_PWRDOWN	For channel 1 (IMM). Set to disable this oscillator; the bit clears itself after powerdown.
2	CH1_LO_RST	For channel 1 (LBM). Toggle to reset L-band oscillator 1.
1	CH1_LO_UPDATE	For channel 1 (IMM). Set to update this oscillator to the correct frequency, depending on the status of bit 6 (above); the bit clears itself after the update begins.
0	CH0_LO_LOCK	For channel 0 (IMM). Read only. If set, the oscillator is locked.

0x68 – 0x6F Oscillator Frequency 1 – 8

Access / Notes: See below. This is a 64-bit word used to set the oscillator frequency.

Bits	Name	Description
7–0	[no name]	0x68 (access = SRXL2_LO_FREQ1). The least significant 8 bits of the 64-bit word.
15–8	[no name]	0x69 (access = SRXL2_LO_FREQ2).
23–16	[no name]	0x6A (access = SRXL2_LO_FREQ3).
31–24	[no name]	0x6B (access = SRXL2_LO_FREQ4).
39–32	[no name]	0x6C (access = SRXL2_LO_FREQ5).
47–40	[no name]	0x6D (access = SRXL2_LO_FREQ6).
55–48	[no name]	0x6E (access = SRXL2_LO_FREQ7).
63–56	[no name]	0x6F (access = SRXL2_LO_FREQ8). The most significant 8 bits of the 64-bit word.

Write bits 63–56 last, because a write to this register causes the hardware to program the selected synthesizer with the frequency specified in all eight oscillator frequency registers.

0x71 Sample Clock Read – Channel 1

Access / Notes: SRXL2_CH1_SCLK_READ / 8-bit read-write

For channel 1. Write the address of the desired Si5326 register. When bit 7 (XFER_BUSY) in the [0x54 Serial Control](#) is clear, read the data from the requested register.

0x72 Sample Clock Write Address – Channel 1

Access / Notes: SRXL2_CH1_SCLK_ADDR / 8-bit read-write

For channel 1. The address of the Si5326 register to which you wish to write.

0x73 Sample Clock Write Data – Channel 1

Access / Notes: SRXL2_CH1_SCLK_DATA / 8-bit read-write

For channel 1. The data that you wish to write to the Si5326 register specified by the address in the [0x72 Sample Clock Write Address – Channel 1](#).

0x7F Board ID

Access / Notes: EDT_BOARDID / 8-bit

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	RW	Name	Description
7-4	RW	[no name]	Used by <code>extbdid.exe</code> .
3-0	RW	BOARD_ID	See EDT Board ID and Extended Board ID table (below).

Table 5. EDT Board ID and Extended Board ID (CPLD) – part 1 of 2

Board ID Register, Bits 3–0	Extended Board ID	Board Name	Detail
0 0 0 0	0x0	RS422	–
0 0 0 1	0x1	LVDS	–
0 0 1 0	0x2	Reserved	For extended board IDs (below).
– – – –	0x0A	SRXL	–
– – – –	0x10	16TE3	–
– – – –	0x11	OC192	–
– – – –	0x12	3x3G	–
– – – –	0x13	MSDV	–
– – – –	0x14	SRXL2 (rev01 & 02)	Contact EDT to exchange for later revision.
– – – –	0x15	Net10G	–
– – – –	0x16	DRX	–
– – – –	0x17	DDSP	–
– – – –	0x18	SRXL2 (rev03+)	For the IDM + LBM option.
– – – –	0x19	SRXL2 (rev03+)	For the IDM + IMM option.
– – – –	0x1A	SRXL2 (rev03+)	For the IMM + IMM option.
– – – –	0x1B	SRXL2 (rev03+)	For the IMM + LBM option.
– – – –	0x1C	SRXL2 (rev03+)	For the IDM + IMM option.
– – – –	0x1D	DRX16	For the IDX + IDX option.
– – – –	0x1E	OCM2P7G	–
0 0 1 1	0x3	Reserved	–
0 1 0 0	0x4	SSE	–
0 1 0 1	0x5	HRC	For E4, STS3, STM1 / OC3 I/O.
0 1 1 0	0x6	OCM	–
0 1 1 1	0x7	Combo 2	For LVDS I/O.
1 0 0 0	0x8	ECL/LVDS-E/RS422-E	For ECL, LVDS, RS422, E1/T1 I/O.
1 0 0 1	0x9	TLK1501	–
1 0 1 0	0xA	Reserved	–
1 0 1 1	0xB	Combo 3	For RS422 I/O.

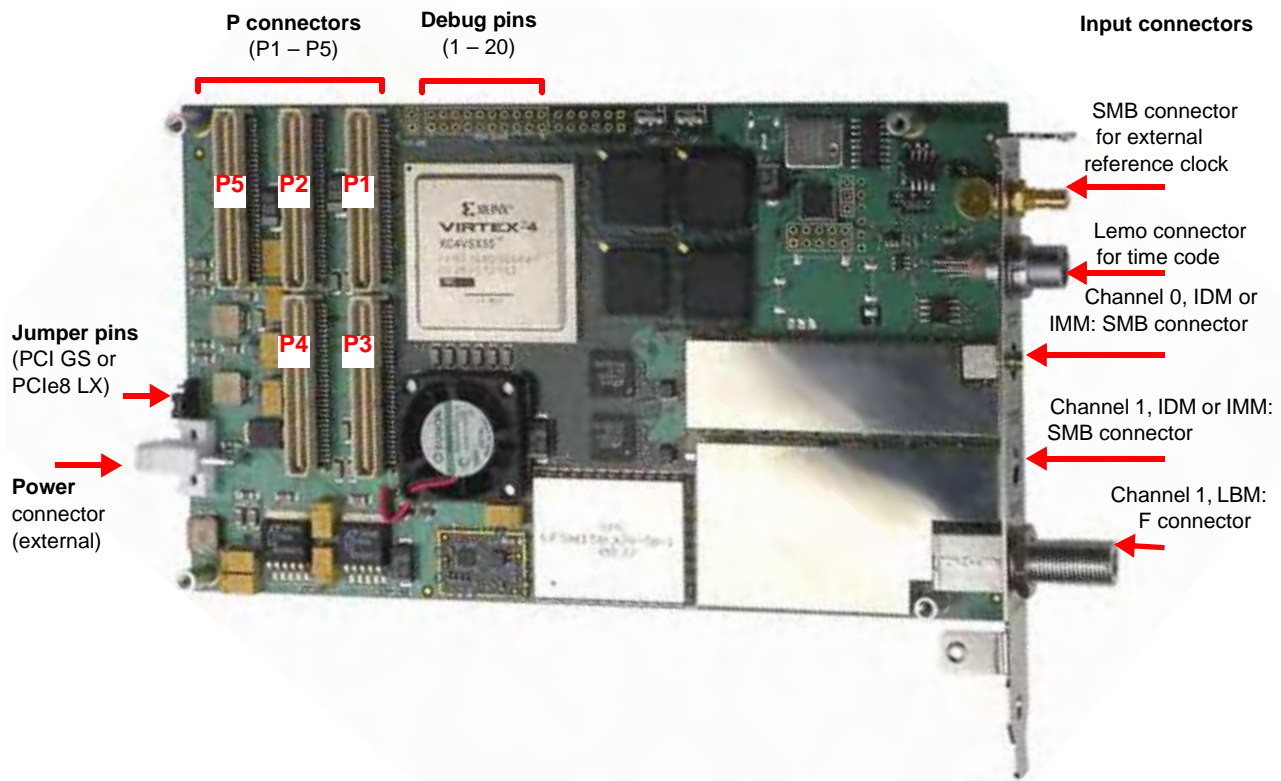
Table 5. EDT Board ID and Extended Board ID (CPLD) – part 2 of 2

Board ID Register, Bits 3–0	Extended Board ID	Board Name	Detail
1 1 0 0 0xC	–	Combo 3	For LVDS I/O.
1 1 0 1 0xD	–	Combo 3	For ECL I/O.
1 1 1 0 0xE	–	Combo 2	For RS422 I/O.
1 1 1 1 0xF	–	Combo	For ECL I/O.

Connectors and Pins

Figure 6, with its accompanying notes, identifies the connectors and pins on the SRXL2.

Figure 6. SRXL2 Connectors and Pins



- 5 P-connectors These 64-pin CMC-type connectors (P1 – P5) link the SRXL2 to the main board.
- 20 debug pins The ten pins farthest from the edge of the board are odd-numbered (1 – 19).
The ten pins closest to the edge of the board are even-numbered (2 – 20).
- 5 input connectors There are five input connectors (the last one is an “either/or” option):
 - One SMB connector for external reference clock.
 - One Lemo connector for time code.
 - One SMB connector for IDM or IMM on channel 0.
 - One SMB connector for IDM or IMM, or one F connector for LBM, on channel 1.
- 3 jumper pins The jumper configuration depends on which main board you are using:
 - For PCI GS, put the jumper on the two pins closest to the P5 connector.
 - For PCIe8 LX, put the jumper on the two pins farthest from the P5 connector.
- 1 power connector This connector is for the external power source.

Pinouts

This section contains a series of tables showing the pinouts from the SRXL2 FPGA to other devices.

The first five tables cover the pinouts from the SRXL2 FPGA to the P-connectors. For these tables, the meaning of each function is shown below.

If the function says...	...the meaning is...
+5 V	These pins are the 5-volt supply.
+12 V	These pins are the 12-volt supply.
BD IDx[0-3]	These pins are for mezzanine board identification (the lowest four bits of 0x67 Oscillator Control); use with Connectors and Pins on page 26 .
extbidid...	These pins are the interface for the extended board ID.
free	These pins have wires connecting the SRXL2 FPGA to the UI FPGA on the main board, so this signal can be accessed by your firmware.
FPGA...	These pins are dedicated for programming the SRXL2 FPGA.
ground	These pins are the ground connection.
unused	These pins do not have wires connecting the SRXL2 FPGA to any outside device, so the signals cannot be accessed.

The remaining tables show the pinouts from the SRXL2 FPGA to the other devices on the board.

Table 6. SRXL2 FPGA to P1 Connector

P1 Pin	SRXL2 FPGA Pin	SRXL2 Signal	P1 Pin	SRXL2 FPGA Pin	SRXL2 Signal
P1.1	U22	FPGA PROG_B	P1.33	G2	free
P1.2		unused	P1.34		ground
P1.3		ground	P1.35		ground
P1.4	A9	free	P1.36	J4	free
P1.5	A8	free	P1.37	G1	free
P1.6	A6	free	P1.38		+5 V
P1.7		BD ID0	P1.39		ground
P1.8		+5 V	P1.40	L8	free
P1.9	B6	free	P1.41	H2	free
P1.10	A5	free	P1.42	M7	free
P1.11		ground	P1.43	J1	free
P1.12	B5	free	P1.44		ground
P1.13	A4	free	P1.45		unused
P1.14		ground	P1.46	M8	free
P1.15		ground	P1.47	J2	free
P1.16	A3	free	P1.48	N9	free
P1.17	B3	free	P1.49	K1	free
P1.18		+5 V	P1.50		+5 V
P1.19		unused	P1.51		ground
P1.20	B2	free	P1.52	P10	free
P1.21	D2	free	P1.53	L1	free
P1.22	E4	free	P1.54	P11	free
P1.23	D1	free	P1.55	L3	free
P1.24		ground	P1.56		ground
P1.25		ground	P1.57		unused
P1.26	E2	free	P1.58	N12	free
P1.27	E1	free	P1.59	M1	free
P1.28	F4	free	P1.60	N13	free
P1.29	F1	free	P1.61	M2	free
P1.30		+5 V	P1.62		+5 V
P1.31		unused	P1.63		ground
P1.32	H4	free	P1.64	N15	free
P1.32	H4	free	P1.64	N15	free

Table 7. SRXL2 FPGA to P2 Connector

P2 Pin	SRXL2 FPGA Pin	SRXL2 Signal	P2 Pin	SRXL2 FPGA Pin	SRXL2 Signal
P2.1		unused	P2.33		ground
P2.2	B11	free	P2.34	F6	free
P2.3	A11	free	P2.35	K3	free
P2.4	A10	free	P2.36		unused
P2.5	B10	free	P2.37		ground
P2.6		ground	P2.38	H5	free
P2.7		ground	P2.39	K2	free
P2.8	C9	free	P2.40		ground
P2.9	C8	free	P2.41		unused
P2.10	B8	free	P2.42	J5	free
P2.11		BD ID1	P2.43	K4	free
P2.12		unused	P2.44		ground
P2.13	B7	free	P2.45	U21	FPGA INIT
P2.14		BD ID2	P2.46	M16	free
P2.15		unused	P2.47		ground
P2.16		BD ID3	P2.48	L5	free
P2.17	C5	free	P2.49	L4	free
P2.18		ground	P2.50		unused
P2.19	C4	free	P2.51	M5	free
P2.20	C3	free	P2.52	M6	free
P2.21		ground	P2.53		unused
P2.22	C2	free	P2.54	N5	free
P2.23	E3	free	P2.55	N7	free
P2.24		unused	P2.56		ground
P2.25	F3	free	P2.57	P5	free
P2.26	D9	free	P2.58	P6	free
P2.27		unused	P2.59		ground
P2.28	D7	free	P2.60	P7	free
P2.29	G3	free	P2.61	N10	free
P2.30		ground	P2.62		unused
P2.31	H3	free	P2.63		ground
P2.32	D4	free	P2.64	M10	free

Table 8. SRXL2 FPGA to P3 Connector

P3 Pin	SRXL2 FPGA Pin	SRXL2 Signal	P3 Pin	SRXL2 FPGA Pin	SRXL2 Signal
P3.1	AC2	free	P3.33		ground
P3.2		ground	P3.34	AG3	free
P3.3		ground	P3.35	AK1	free
P3.4	AC5	free	P3.36	AH2	free
P3.5	AD1	free	P3.37	AB17	free
P3.6	AC3	free	P3.38		ground
P3.7	AD2	free	P3.39		unused
P3.8		ground	P3.40	AH3	free
P3.9		unused	P3.41	AL1	free
P3.10	AC4	free	P3.42	AJ2	free
P3.11	AE1	free	P3.43	AM1	free
P3.12	AD5	free	P3.44		ground
P3.13	AE2	free	P3.45		ground
P3.14		ground	P3.46	AK2	free
P3.15		ground	P3.47	AB18	free
P3.16	R17	FPGA CCLK	P3.48	AK3	free
P3.17	AF1	free	P3.49	AL14	free
P3.18	AD4	free	P3.50		ground
P3.19	AG1	free	P3.51		ground
P3.20		ground	P3.52	AL3	free
P3.21		unused	P3.53	AL15	free
P3.22	AE4	free	P3.54	AH14	free
P3.23	AM2	free	P3.55	AM15	free
P3.24	AE3	free	P3.56		ground
P3.25	AM3	free	P3.57		unused
P3.26		ground	P3.58	AJ14	free
P3.27		ground	P3.59	AN15	free
P3.28	AF3	free	P3.60	AJ15	free
P3.29	AJ1	free	P3.61	AP15	free
P3.30	AG2	free	P3.62		ground
P3.31	AC17	free	P3.63		ground
P3.32		ground	P3.64	AK14	free

Table 9. SRXL2 FPGA to P4 Connector

P4 Pin	SRXL2 FPGA Pin	SRXL2 Signal	P4 Pin	SRXL2 FPGA Pin	SRXL2 Signal
P4.1		unused	P4.33	AB6	free
P4.2	P9	free	P4.34	AA11	free
P4.3	R9	free	P4.35	AC7	free
P4.4	T10	free	P4.36		ground
P4.5	T11	free	P4.37	U15	FPGA DONE
P4.6		ground	P4.38	AB8	free
P4.7	R11	free	P4.39	AD6	free
P4.8	P12	free	P4.40	AC8	free
P4.9	AA15	free	P4.41		ground
P4.10	Y16	free	P4.42	AD7	free
P4.11		ground	P4.43	AE6	free
P4.12	AA13	free	P4.44	AE7	free
P4.13	AB13	free	P4.45	AF4	free
P4.14	AB10	free	P4.46		ground
P4.15	AB12	free	P4.47	AF5	free
P4.16		ground	P4.48	AF8	free
P4.17	AC9	free	P4.49	AH4	free
P4.18	AE8	free	P4.50	AF6	free
P4.19	Y11	free	P4.51		ground
P4.20	AC10	free	P4.52	AD16	free
P4.21		ground	P4.53	AH5	free
P4.22	AB15	free	P4.54	AD17	free
P4.23	Y12	free	P4.55	AG5	free
P4.24	AC15	free	P4.56		ground
P4.25	AA8	free	P4.57	AG6	free
P4.26		ground	P4.58	AG13	free
P4.27	AA9	free	P4.59	AG7	free
P4.28	AB16	free	P4.60	AF14	free
P4.29	AB5	free	P4.61		ground
P4.30	Y14	free	P4.62	AF15	free
P4.31		ground	P4.63	AG8	free
P4.32	Y13	free	P4.64	AG15	free

Table 10. SRXL2 FPGA to P5 Connector

P5 Pin	SRXL2 FPGA Pin	SRXL2 Signal	P5 Pin	SRXL2 FPGA Pin	SRXL2 Signal
P5.1		unused	P5.33	F5	free
P5.2	D12	free	P5.34	H12	free
P5.3	D14	free	P5.35	G5	free
P5.4	E14	free	P5.36		ground
P5.5	C14	free	P5.37	F10	free
P5.6		ground	P5.38	G8	free
P5.7	C13	free	P5.39	G7	free
P5.8	E13	free	P5.40		extbdid CLK
P5.9	C7	free	P5.41		ground
P5.10	E12	free	P5.42		extbdid DATA
P5.11		ground	P5.43	H8	free
P5.12	D11	free	P5.44	H9	free
P5.13	D6	free	P5.45	J9	free
P5.14	E11	free	P5.46		ground
P5.15	D5	free	P5.47	H10	free
P5.16		ground	P5.48	J11	free
P5.17	B13	free	P5.49	E6	free
P5.18	D10	free	P5.50	F8	free
P5.19	B12	free	P5.51		ground
P5.20	E9	free	P5.52	G6	free
P5.21		ground	P5.53	J6	free
P5.22	F13	free	P5.54	H7	free
P5.23	C12	free	P5.55	J7	free
P5.24	F11	free	P5.56		ground
P5.25	C10	free	P5.57	K6	free
P5.26		ground	P5.58	M3	free
P5.27	E8	free	P5.59	T16	FPGA DIN
P5.28	G11	free	P5.60	L6	free
P5.29	E7	free	P5.61		ground
P5.30	G10	free	P5.62	K8	free
P5.31		ground	P5.63	L9	free
P5.32	G12	free	P5.64	K9	free

Table 11. SRXL2 FPGA to Graychips (GCs) – part 1 of 3

GC Pin Name	GC Pin	SRXL2 FPGA Pin to GC1	SRXL2 FPGA Pin to GC2	SRXL2 FPGA Pin to GC3	SRXL2 FPGA Pin to GC4
AIN[0]	D14	B30	*	*	*
AIN[1]	E13	B21	*	*	*
AIN[2]	E12	B22	*	*	*
AIN[3]	E14	A21	*	*	*
AIN[4]	F13	B25	*	*	*
AIN[5]	F12	A23	*	*	*
AIN[6]	F14	A24	*	*	*
AIN[7]	G13	B26	*	*	*
AIN[8]	G14	A25	*	*	*
AIN[9]	H12	C27	*	*	*
AIN[10]	H11	A26	*	*	*
AIN[11]	H14	F28	*	*	*
AIN[12]	H13	K28	*	*	*
AIN[13]	J12	A28	*	*	*
BIN[0]	J14	G27	*	*	*
BIN[1]	J13	K27	*	*	*
BIN[2]	K12	C30	*	*	*
BIN[3]	K14	N23	*	*	*
BIN[4]	K13	J24	*	*	*
BIN[5]	L14	N22	*	*	*
BIN[6]	L12	K24	*	*	*
BIN[7]	M11	D31	*	*	*
BIN[8]	P11	E31	*	*	*
BIN[9]	L10	M26	*	*	*
BIN[10]	M10	M25	*	*	*
BIN[11]	M9	M27	*	*	*
BIN[12]	P9	J29	*	*	*
BIN[13]	L8	K34	*	*	*
CIN[0]	N7	AD30	*	*	*
CIN[1]	L6	AM32	*	*	*
CIN[2]	P6	AD29	*	*	*
CIN[3]	N6	AC29	*	*	*
CIN[4]	N5	AL31	*	*	*
CIN[5]	L4	AJ32	*	*	*
CIN[6]	M4	AK32	*	*	*
CIN[7]	N4	AM33	*	*	*
CIN[8]	L2	AL33	*	*	*
CIN[9]	L3	AH32	*	*	*
CIN[10]	L1	AL34	*	*	*
CIN[11]	K2	AK33	*	*	*
CIN[12]	K3	AG31	*	*	*
CIN[13]	K1	AK34	*	*	*
DIN[0]	J2	AJ34	*	*	*
DIN[1]	J3	AG32	*	*	*
DIN[2]	J1	AH33	*	*	*
DIN[3]	H2	AH34	*	*	*
DIN[4]	H1	AG33	*	*	*
DIN[5]	G3	AD32	*	*	*
DIN[6]	G4	AE32	*	*	*
DIN[7]	G1	AF33	*	*	*

Table 11. SRXL2 FPGA to Graychips (GCs) – part 2 of 3

GC Pin Name	GC Pin	SRXL2 FPGA Pin to GC1	SRXL2 FPGA Pin to GC2	SRXL2 FPGA Pin to GC3	SRXL2 FPGA Pin to GC4
DIN[8]	G2	AF34	*	*	*
DIN[9]	F3	AC32	*	*	*
DIN[10]	F1	AE33	*	*	*
DIN[11]	F2	AE34	*	*	*
DIN[12]	E3	AC30	*	*	*
DIN[13]	E1	AD34	*	*	*
C[0]	B9	F21	*	*	*
C[1]	A9	F23	*	*	*
C[2]	C9	E21	*	*	*
C[3]	B8	F26	*	*	*
C[4]	C8	G23	*	*	*
C[5]	A7	F24	*	*	*
C[6]	B7	G25	*	*	*
C[7]	A6	F25	*	*	*
P[0]	B11	C32	C20	M32	F31
P[1]	C11	B32	B20	N30	L34
P[2]	A11	B33	D21	L31	L33
P[3]	B10	C34	D22	M31	L29
P[4]	D9	D32	D24	N29	L28
P[5]	C7	D34	D25	AA26	G31
P[6]	D7	E32	C25	AA28	H32
P[7]	D5	E34	D27	AA29	K31
P[8]	D4	F33	C28	AB28	K32
P[9]	F4	H33	C29	AD31	Y26
P[10]	H3	H34	D30	AE31	AB23
P[11]	H4	J32	D29	AF31	P20
P[12]	J4	J34	E29	AJ31	P22
P[13]	K4	K33	F29	AK31	R21
P[14]	M6	L26	H29	AM31	AC28
P[15]	M7	M28	G28	AJ30	AC27
P[16]	N8	G30	J27	AF30	AD27
P[17]	N9	J30	E24	AG30	AC25
P[18]	N10	K29	E23	AH30	AB25
P[19]	L11	L25	E26	R23	AA24
P[20]	K11	B28	E27	R24	AA25
P[21]	J11	B27	E28	P24	AB22
P[22]	F11	C24	C22	P26	R19
P[23]	E11	B23	C23	P27	AA23
A[0]	C6	H24	*	*	*
A[1]	B6	H25	*	*	*
A[2]	A5	H27	*	*	*
A[3]	C5	J20	*	*	*
A[4]	B5	H22	*	*	*
RD	A4	J25	*	*	*
WR	C4	J21	*	*	*
CE	B4	E33	D26	AA30	J31

Table 11. SRXL2 FPGA to Graychips (GCs) – part 3 of 3

GC Pin Name	GC Pin	SRXL2 FPGA Pin to GC1	SRXL2 FPGA Pin to GC2	SRXL2 FPGA Pin to GC3	SRXL2 FPGA Pin to GC4
CK	P7	F30	K26	AE29	AB26
SCK	D12	B31	D20	N25	Y24
SFS	D13	A31	D19	N27	M33
RDY	A10	C33	A20	M30	L30
SIB	D1	F34	H28	AC34	R22
SIA	D3	G32	A29	AB30	W24
SO	E2	G33	A30	AC33	W25
DVAL	P8	H30	E22	AF29	AD26

* same as previous column

Table 12. SRXL2 FPGA to ADC

Pin	Channel 1	Pin	Channel 0
AM12	Channel 1: data in 0+	AM30	Channel 0: data in 0+
AP11	Channel 1: data in 1+	AP31	Channel 0: data in 1+
AP9	Channel 1: data in 2+	AP30	Channel 0: data in 2+
AN10	Channel 1: data in 3+	AP29	Channel 0: data in 3+
AN14	Channel 1: data in 4+	AP27	Channel 0: data in 4+
AN12	Channel 1: data in 5+	AP25	Channel 0: data in 5+
AN8	Channel 1: data in 6+	AM26	Channel 0: data in 6+
AM6	Channel 1: data in 7+	AP21	Channel 0: data in 7+
AP5	Channel 1: data in 8+	AN25	Channel 0: data in 8+
AP4	Channel 1: data in 9+	AL28	Channel 0: data in 9+
AK13	Channel 1: data in 10+	AP24	Channel 0: data in 10+
AN7	Channel 1: data in 11+	AL26	Channel 0: data in 11+
AM11	Channel 1: data in 0-	AL30	Channel 0: data in 0-
AP10	Channel 1: data in 1-	AP32	Channel 0: data in 1-
AN9	Channel 1: data in 2-	AN30	Channel 0: data in 2-
AM10	Channel 1: data in 3-	AN29	Channel 0: data in 3-
AP14	Channel 1: data in 4-	AN27	Channel 0: data in 4-
AP12	Channel 1: data in 5-	AP26	Channel 0: data in 5-
AM8	Channel 1: data in 6-	AM27	Channel 0: data in 6-
AM5	Channel 1: data in 7-	AP22	Channel 0: data in 7-
AN5	Channel 1: data in 8-	AM25	Channel 0: data in 8-
AN4	Channel 1: data in 9-	AL29	Channel 0: data in 9-
AL13	Channel 1: data in 10-	AN24	Channel 0: data in 10-
AM7	Channel 1: data in 11-	AK26	Channel 0: data in 11-
AP7	Channel 1: data clock +	AN28	Channel 0: data clock +
AP6	Channel 1: data clock -	AM28	Channel 0: data clock -
AN3	Channel 1: overrange +	AL24	Channel 0: overrange +
AN2	Channel 1: overrange -	AL25	Channel 0: overrange -

Table 13. SRXL2 FPGA to Other Local Devices

Pin	Signal Description	Pin	Signal Description
Reference Clock		LED	
E19	Reference clock into FPGA	A13	LED 1
E18	External or internal reference clock selector	A14	LED 0
Sample Clock		Board Control	
AE17	Loss of 10 MHz signal (channel 0)	AH28	LBM front-end filter select, bit 1
AE18	Loss of 10 MHz signal (channel 1)	AG25	LBM front-end filter select, bit 0
AF18	SPI serial data out	AH24	LBM oscillator 1 filter select, bit 1
AG16	SPI clock	AH25	LBM oscillator 1 filter select, bit 0
AG17	SPI serial data in		
AF16	SPI select (channel 0)	Time Code	
AH17	SPI select (channel 1)	AJ22	SPI serial data out to MSP430
AE16	Reset signal (channel 0)	AF21	SPI serial data in from MSP430
AH18	Reset signal (channel 1)	AJ20	SPI serial data clock to MSP430
AJ17	Loss of lock (channel 0)	AJ21	SPI serial enable to MSP430
AK18	Loss of lock (channel 1)	AM20	One-pulse-per-second input to FPGA
		AE21	IRIG-B comparator input to FPGA
DAC		Debug Pins	
AP20	Chip select out to gain DAC	B15	Pin 0
AL19	Serial data out to gain DAC	G13	Pin 1
AN20	Clock out to gain DAC	A15	Pin 2
		F14	Pin 3
Channel 0 Oscillator: IMM		C17	Pin 4
AD19	Programming clock	G15	Pin 5
AD20	Serial data	C18	Pin 6
AC19	Latch enable	C15	Pin 7
AE19	Lock status	C19	Pin 8
		D16	Pin 9
Channel 1 Oscillator: IMM or LBM2		D17	Pin 10
G16	Programming clock	H13	Pin 11
G18	Serial data	H15	Pin 12
H19	Latch enable	H14	Pin 13
J17	Lock status	K16	Pin 14
		J15	Pin 15
Channel 1 Oscillator: LBM1		K14	Pin 16
AM16	Programming clock	J14	Pin 17
AM17	Serial data	unused	Pin 18
AM18	Latch enable	ground	Pin 19
AL20	Lock status	ground	Pin 20
AL18	Error flag		

Revision Log

Below is a history of modifications to this guide.

Date	By	Rv	Pp	Detail
20100408	PH, BO		26	Swapped descriptions for jumper pins so they now read correctly as: - For PCI GS, put the jumper on the two pins closest to the P5 connector. - For PCIe8 LX, put the jumper on the two pins farthest from the P5 connector.
20100216	PH, BO	-00	All	Added RW column to registers and updated styles.