



User's Guide

SRXL

Mezzanine Board



**Signal receiver and processor
for L-band and IF
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
Building Applications	6
Loading the Firmware	6
Getting the Board ID	7
Board Architecture	7
Programming the SRXL Devices	7
Sample Clock (via AD9951)	9
Channel 0: IF Oscillator (via LMX2364)	10
Channel 1: L-band Tuner (via MAX2118)	11
Gain (via TLV5617A)	13
Graychips (GC4016)	14
Acquiring a Signal	14
DMA Mode	14
Test Mode	15
PRBS Test Pattern	15
Registers	16
0x00 Command Register	16
0x10 Channel Enable Low – DMA Channels 7–0 Register	16
0x11 Channel Enable High – DMA Channels 15–8 Register	16
0x50 DDS Control Register	16
0x51 DDS Data Register	16
0x54 Serial Control Register	17
0x55 Maxim Address Register	17
0x56 Maxim Data Register	17
0x58 DAC A Low Register	18
0x59 DAC A High Register	18
0x5A DAC B Low Register	18
0x5B DAC B High Register	18
0x5C PLL Low Register	18
0x5D PLL Mid Register	19
0x5E PLL High Register	19
0x5F Sample Clock Control Register	19
0x60 Read Data Register	19
0x62 PRBS Enable Register	19
0x63 Capture Control Register	20
0x64 Graychip Address Register	20
0x65 Graychip Data Register	20
0x7F Board ID Register	20
Connectors and Pins	21
Pinouts	22
Revision Log	31

SRXL

Overview

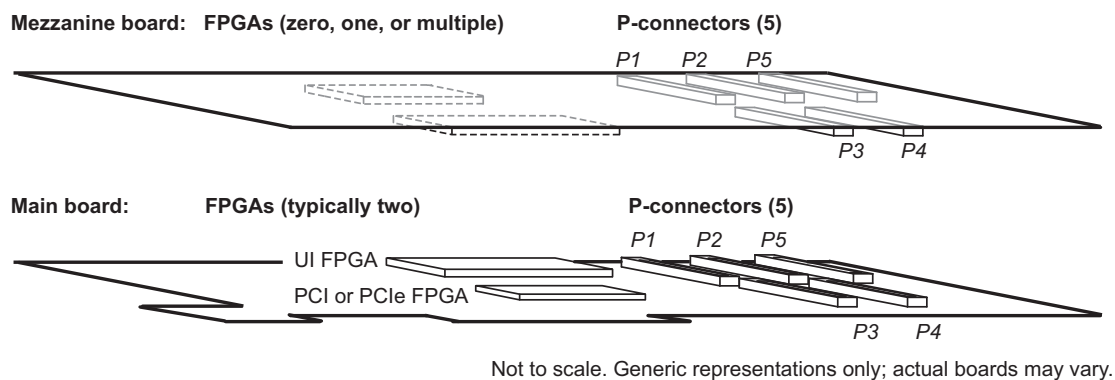
The SRXL is an EDT mezzanine board that digitizes and captures IF and L-band radio frequency (RF) signals. For specifications, see [Related Resources on page 2](#).

The SRXL is paired with an EDT main board (PCI SS/GS or PCIe8 LX/FX) 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 SRXL board pair has the following FPGAs.

- The mezzanine board (SRXL) has one user-programmable FPGA – the *SRXL FPGA*. To load it, see [Installation on page 3](#).
- The main board (PCI SS/GS or PCIe8 LX/FX) has two FPGAs:
 - The *UI FPGA* links the SRXL FPGA to the PCI or PCIe FPGA. To load the UI FPGA, use the EDT Main Board User's Guide (see link under [Related Resources](#), below).
 - The *PCI or PCIe FPGA* communicates with the host computer over the PCI or PCIe bus and implements the DMA engine, which transfers data between the board and the host. At powerup, this FPGA automatically loads with the correct firmware from the main board's flash ROM.

NOTE All FPGAs must be loaded with correct firmware to work with each other and with your interface.

Related Resources

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

<i>Resource</i>	<i>EDT webpage</i>
SRXL specifications (datasheet)	www.edt.com (go to SRXL 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):	
Windows, Linux, Solaris, Mac OS	www.edt.com/software.html

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

<i>Manufacturer</i>	<i>Part Number</i>	<i>Manufacturer's website</i>
Analog Devices	AD9283	www.analog.com
"	AD9951	"
Linear Technologies	LT5546	www.linear.com
Maxim	MAX2118	www.maxim-ic.com
National Semiconductor	LMX2364	www.national.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 that you go to the EDT website and do 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 download the latest package and recompile / relink the application.

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

Included Files

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

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

Subdirectory: bitfiles

This subdirectory contains one FPGA configuration file for each user-configurable FPGA available on your board pair (SRXL + main board). For proper FPGA configuration, find and load the appropriate files as explained below.

SRXL

For the SRXL mezzanine board: In `bitfiles`, find the `XC3S1500` subdirectory. In that subdirectory, find and load the file below that correlates to the module option and main board you ordered.

<code>srxl_top.bit</code>	Configures the SRXL for use with a PCI SS or PCI GS main board.
<code>srxl_top.lx2.bit</code>	Configures the SRXL for use with a PCIe8 LX main board.

PCIe8 LX

For a PCIe8 LX main board: In `bitfiles`, find the correct subdirectory for the FPGA you ordered (`XC5VLX110T` or `XC5VLX220T` or `XC5VLX330T`). In that subdirectory, find and load the file below.

<code>srxl_2in.bit</code>	Configures the UI FPGA on a PCIe8 LX main board.
---------------------------	--

PCI GS

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

<code>srxl_16io.bit</code>	Configures the UI FPGA on a PCI GS main board with a 16-channel DMA interface.
<code>srxl_4io.bit</code>	Configures the UI FPGA on a PCI GS main board with a four-channel DMA interface.

PCI SS

For a PCI SS main board: In `bitfiles`, find the correct subdirectory for the FPGA you ordered (`XCV1000E` or `XCV2000E` or `XCV600E`). In that subdirectory, find and load the file below.

`srxl_4io.bit` Configures the UI FPGA on a PCI SS main board with a four-channel DMA interface.

Subdirectory: flash

This subdirectory relates to the PCI FPGA on your main board. The correct configuration file for this FPGA is preloaded into flash ROM and thus loads automatically on powerup.

PCIe8 LX

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

`pe8lx16.bit` Configures the PCI FPGA on a PCIe8 LX main board.

PCI GS

In `flash`, from the `xc2s200` 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 SS

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, `pciss16_classic.bit` is the correct argument for loading the firmware.

`pciss16_classic_3v.bit`

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

`pciss16_classic_5v.bit`

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

Subdirectory: pci_config

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

NOTE Currently, EDT does not provide `.cfg` files for the SRXL. However, the following discussion of these files may be helpful if you want to create your own.

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: srxl_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'd like us to include your new software initialization commands in subsequent releases of `initpcd`, email us at tech@edt.com.

Subdirectory: srxl

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

<code>halfbandfft.c</code>	C source for the <code>halfbandfft</code> application.
<code>lib_srxl.c</code>	C source for routines called by the example applications. These routines in turn call routines in <code>libedt.c</code> , the EDT DMA library.
<code>lib_srxl.h</code>	A header file for <code>lib_srxl.c</code> .
<code>srxl_debugger.c</code>	C source for the <code>srxl_debugger</code> application.
<code>fft.c</code>	C source for the <code>fft</code> application.
<code>srxl_gc_diag.c</code>	C source for the <code>srxl_gc_diag</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 `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>halfbandfft</code>	An example application that performs a Fast Fourier Transform on both real and imaginary components of the captured data and then performs image rejection to minimize the reflected signal as best it can.
<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>srxl_debugger</code>	<p>Exercises most of the mezzanine board functions and provides debugging aids. The example application <code>srxl_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 SRXL and user interface FPGA registers; - access and set values on the SRXL 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>srxl_debugger</pre> <p>At the SRXL debugger prompt, enter the <code>h</code> command for a list of all the usage options and their descriptions.</p>
<code>fft.c</code>	Performs a Fast Fourier Transform on both real and imaginary components of the captured data.
<code>srxl_gc_diag</code>	Performs internal checksum 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.

The recommended compiler, as well as the location of the library and the debugger application, for each supported platform is shown below.

	Recommended compiler	Library & debugger
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 `--help` at the `PCD` command line to display a list of usage options and descriptions.

Loading the Firmware

For FPGA configuration, follow the steps below, replacing italicized terms (such as *unit_number*) with the appropriate values.

1. Run `pciload` to find the unit number of the main board (by default, 0) to verify that the host is detecting the main board, and that the main board is loaded with the appropriate firmware.
2. Load the appropriate firmware with the following command:


```
pciload -u unit_number firmware_file
```
3. For example, *firmware_file* could be `pcigs16_classic` or `pe8lx16`.
4. At the prompt, press Enter to confirm the loading operation.
5. Power-cycle the system.
6. 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
```
7. For example, a PCIe8 LX main board with the SRXL loaded in unit 0 would be:


```
mezzload -u 0 -b srxl_top_lx.bit -B srxl_2in.bit
```
8. Watch for the onscreen message which verifies that the files are loaded.

Getting the Board ID

The SRXL board ID is accessed through the [0x7F Board ID Register](#). After the SRXL is installed and loaded with firmware, you can verify the board ID and revision data by running `extbdid`.

Board Architecture

The SRXL has two independent channels:

- Channel 0 = IF
- Channel 1 = L-band

On each channel, the signal is received, converted to quadrature baseband (I and Q), digitized, and then captured in the FPGA on the SRXL board.

From there, the digitized data is routed in one of these two ways:

- to two optional Graychips for a total of eight 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 SRXL FPGA and the main board UI FPGA are fully user-programmable.

Each channel has a dual 12-bit analog-to-digital converter (ADC) that digitizes both the I and Q. For the format of the digital data, you can select either twos-complement or offset binary by using bit 5 (AD_DFS) of the [0x5F Sample Clock Control Register](#).

Both channels share the same sample clock, programmable from 1 to 65 MHz. This sample clock is routed to the ADCs through the FPGA on the SRXL. Therefore, internal FPGA digital clock managers could be used to generate a separate sample clock for each channel.

The board also has one reference clock, which serves as the timebase for all clocks on the board.

To select this clock, use bit 7 (EXT_REF_SEL) of the [0x5F Sample Clock Control Register](#):

- Clear the bit to select the internal 10 MHz TCXO provided; or
- Set the bit to select your own external reference.

For a diagram and a photo of SRXL devices, see [Figure 2 on page 8](#) and [Figure 3 on page 21](#).

Programming the SRXL Devices

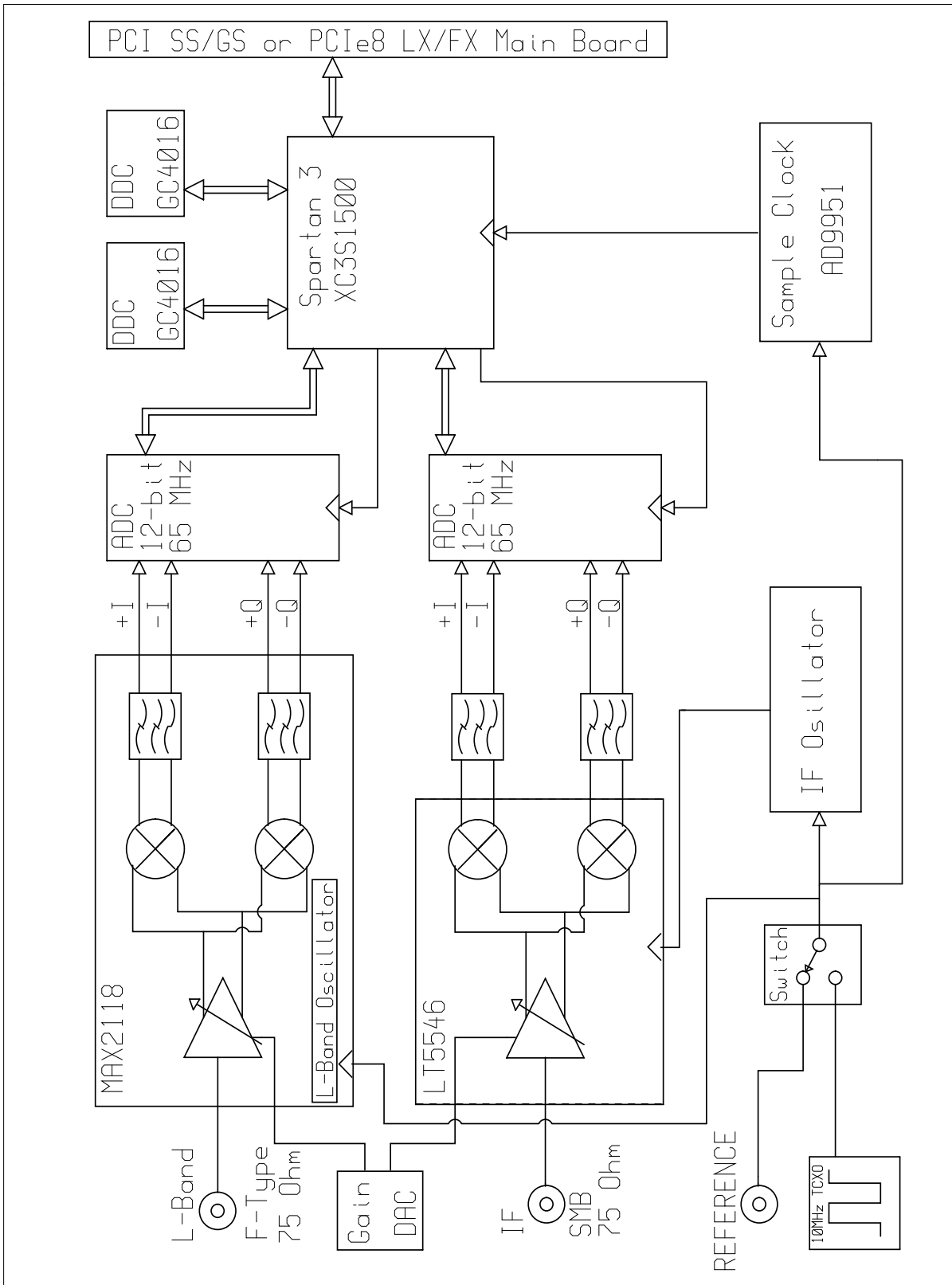
All programmable SRXL devices are programmed through correlating registers on the SRXL FPGA.

On the SRXL, you can program:

- the sample clock (shared by both channels);
- the IF oscillator (channel 0);
- the L-band tuner (channel 1);
- the gain (can be programmed separately for each channel); and
- the Graychips (if you ordered them).

NOTE To prevent datastream errors, always verify that bit 7 (XFER_BUSY) of the [0x54 Serial Control Register](#) is clear before you write a new register, since the firmware serializes your instructions (and sends them to the target device) at a rate that is slower than the register interface.

Figure 2. SRXLBlock Diagram



Sample Clock (via AD9951)

The sample clock (shared by channel 0 and channel 1) is generated by an Analog Devices AD9951 direct digital synthesizer (DDS), using the reference timebase as its input clock. Valid sample clock frequencies are 1–65 MHz.

NOTE The AD9951 has six programmable registers (programmed through the SRXL FPGA) of varied sizes. To find an AD9951 datasheet, use the manufacturer's link under [Related Resources on page 2](#).

To write an AD9951 register

1. Write the address of the desired AD9951 register to the [0x51 DDS Data Register](#).
2. Wait for bit 7 (XFER_BUSY) in [0x54 Serial Control Register](#) to clear.
3. Write the data one byte at a time, from most to least significant; after each byte, wait for bit 7 (XFER_BUSY) in [0x54 Serial Control Register](#) to clear before going on.
4. Set and then clear bit 0 (DDS_UPDATE) in the [0x50 DDS Control Register](#).

To program the sample clock

1. Program the duty clock stabilizer via bit 4 (AD_DCS) of the [0x5F Sample Clock Control Register](#):
 - Set the bit for rates greater than or equal to 40 MHz; or
 - Clear the bit for rates lower than 40 MHz.
2. Set and then clear bit 1 (DDS_RESET) in the [0x50 DDS Control Register](#) to reset the AD9951.
3. Write the AD9951 registers with the values in [Table 1](#) below.

Table 1. AD9951 registers

	Bit	Description
0x0	31–24	0x00
	23–16	0x00
	15–8	0x00
	7–0	0x00
0x1	23–16	0x00
	15–8	0x00
	7–0	0xA0
0x2	15–8	0x00
	7–0	0x00
0x3	7–0	0x00
0x4	31–24	Bits 31–24 of the 32-bit tuning word*
	23–16	Bits 23–16 of the 32-bit tuning word*
	15–8	Bits 15–8 of the 32-bit tuning word*
	7–0	Bits 7–0 of the 32-bit tuning word*
0x5	15–8	0x00
	7–0	0x00

* In AD9951 register 0x4, the equation for the tuning word is: $(232 * \text{sample clock frequency}) / 200$.

Channel 0: IF Oscillator (via LMX2364)

On channel 0, the IF center frequency is determined by the IF oscillator frequency, which must be set to twice the desired IF center frequency. Valid center frequencies are 63–112 MHz (low range); or 125–225 MHz (high range). Thus, valid IF oscillator frequencies are 126–224 MHz or 250–450 MHz.

The IF oscillator is generated using a phase-locked loop (PLL), which contains a 250–450 MHz VCO and a synthesizer (National Instruments LMX2364) with a tuning resolution of 1 MHz. For the low range IF center frequencies, a divide-by-two prescaler must be enabled.

The LMX2364 uses the reference timebase that you selected as the input clock signal.

NOTE The LMX2364 has seven programmable registers (programmed through the SRXL FPGA). To find an LMX2364 datasheet, use the manufacturer's link under [Related Resources on page 2](#).

To write an LMX2364 register

Write the three SRXL FPGA registers below to make one 24-bit programming word.

1. Write [0x5C PLL Low Register](#) with bits 7–0 of the desired LMX2364 register.
2. Write [0x5D PLL Mid Register](#) with bits 15–8 of the desired LMX2364 register.
3. Write [0x5E PLL High Register](#) with bits 23–16 of the desired LMX2364 register.
4. Wait for bit 7 (XFER_BUSY) in [0x54 Serial Control Register](#) to clear.

To program the IF oscillator

1. Each time you power up the SRXL, follow the steps above under [To write an LMX2364 register](#) to write LMX2364 registers 0x0 and 0x3–0x6, using the values in [Table 2](#) below.
2. To change the IF oscillator frequency, write LMX2364 registers 0x1 and 0x2 in [Table 2](#).
3. To ensure that the PLL is locked, wait 100 ms for the PLL to stabilize; then read bit 5 (IF_FTEST) of the [0x54 Serial Control Register](#).

Table 2. LMX2364 registers – part 1 of 2

0x	Bit	Description	Notes
0	23–0	0x0C0028	SRXL FPGA 0x5C PLL Low Register = 28 SRXL FPGA 0x5D PLL Mid Register = 00 SRXL FPGA 0x5E PLL High Register = 0C
1	23–20	0000	–
	19–3	These bits are the frequency word: Bits 19-11 = 000000000 Bits 10-3 = <i>frequency_word</i>	For IF center frequency of 125–225 MHz: Set frequency word to the desired center frequency in MHz. For IF center frequency of 63–112 MHz: Set frequency word to <i>twice</i> the desired center frequency in MHz.
	2–0	001	–
2	23–5	0000000000000000	–
	4	Controls the divide-by-two prescaler.	For IF center frequency of 125–225 MHz: Clear this bit to disable. For IF center frequency of 63–112 MHz: Set this bit to enable.
	3–0	1010	–
3	23–0	0x07FFFFB	SRXL FPGA 0x5C PLL Low Register = FB SRXL FPGA 0x5D PLL Mid Register = FF SRXL FPGA 0x5E PLL High Register = 07

Table 2. LMX2364 registers – part 2 of 2

0x	Bit	Description	Notes
4	23–0	0xFFFFFC	SRXL FPGA 0x5C PLL Low Register = FC SRXL FPGA 0x5D PLL Mid Register = FF SRXL FPGA 0x5E PLL High Register = FF
5	23–0	0x00000D	SRXL FPGA 0x5C PLL Low Register = OD SRXL FPGA 0x5D PLL Mid Register = 00 SRXL FPGA 0x5E PLL High Register = 00
6	23–0	0x000046	SRXL FPGA 0x5C PLL Low Register = 46 SRXL FPGA 0x5D PLL Mid Register = 00 SRXL FPGA 0x5E PLL High Register = 00

Channel 1: L-band Tuner (via MAX2118)

On channel 1, the L-band tuner is provided by a Maxim direct-conversion tuner (MAX2118), which controls the center frequency, baseband bandwidth, and gain.

NOTE The MAX2118 has six programmable registers (programmed through the SRXL FPGA). To find a MAX2118 datasheet, use the link to the manufacturer under [Related Resources on page 2](#).

To write a MAX2118 register

1. Clear bit 0 (I2C_READ) of the [0x54 Serial Control Register](#) to indicate a write.
2. Write [0x55 Maxim Address Register](#) with the desired MAX2118 register number.
3. Write [0x56 Maxim Data Register](#) with the desired data byte.
4. Wait for bit 7 (XFER_BUSY) in [0x54 Serial Control Register](#) to clear.

To read MAX2118 status

1. Set bit 0 (I2C_READ) of the [0x54 Serial Control Register](#) to indicate a read.
2. Write a byte (the value is irrelevant) to the [0x56 Maxim Data Register](#) to trigger read cycle.
3. Wait for bit 7 (XFER_BUSY) in [0x54 Serial Control Register](#) to clear.
4. Read the status byte from the [0x56 Maxim Data Register](#).

To program the L-band tuner

To program the L-band tuner, follow the steps below, referring to [Table 3](#).

1. To program the baseband gain, write MAX2118 register 0x5. Of the thirty-two baseband gain settings, higher values represent a lower gain; the total adjustment range is approximately 24 dB.
2. To program the baseband bandwidth from 4–33 MHz, write MAX2118 registers 0x3 and 0x4.
3. To program the center frequency, write MAX2118 registers 0x0 and 0x1. The equation to program the N-counter is:

$$N \text{ counter} = (\text{desired center frequency}) / 1.25$$

4. To make your initial selection from the eight VCOs in the MAX2118, write MAX2118 register 0x2, bits 2–0 (OSC2–0).

NOTE Although the approximate tuning range of each VCO is known, due to production tolerances you must use the “trial and error” method to find the best VCO for the current configuration.

5. Wait 100 milliseconds for the PLL to stabilize.
6. Set bit 6 (ADE) in MAX2118 register 0x4.
7. Wait for bit 7 (XFER_BUSY) in [0x54 Serial Control Register](#) to clear.

8. Set bit 7 (ADL) in MAX2118 register 0x4.
9. Read the status; see [To program the L-band tuner](#), above.
10. Clear bit 7 (ADL) in MAX2118 register 0x4.
11. If the tuning voltage is 111, select the next highest numbered VCO; if it is 000, select the next lowest numbered VCO. If the tuning voltage is any other value, the correct VCO is already selected.
12. If you changed the selected VCO in [step 11](#), repeat [step 4](#) through [step 11](#) to read the current VCO voltage once more.
13. Set the charge pump current to the recommended value for tuning voltage that you read with the correct VCO; see [0x56 Maxim Data Register](#) and MAX2118 register 0x2 ([Table 3](#)).
14. Clear bit 6 (ADE) of MAX2118 register 0x4 ([Table 3](#)).

Table 3. MAX2118 registers – part 1 of 2

0x	Bit	Name	Description
0	7	DIV2	0 for desired center frequencies lower than 1125 MHz. 1 for desired center frequencies of 1125 MHz or above.
	6–0	N14–8	The N-counter (bits 14–8)
1	7–0	N7–0	The N-counter (bits 7–0)
2	7–5	R2–0	010 (reference divider)
	4–3	CP1–0	Tuning voltage is bits 4–2 of MAX2118 status. Charge pump current is: 01 for tuning voltage of 001 01 for tuning voltage of 010 10 for tuning voltage 011 10 for tuning voltage of 100 11 for tuning voltage of 101 11 for tuning voltage of 110
	2–0	OSC2–0	For desired center frequencies below 1125 (in register 0x0 above, bit 7 (DIV2) = 0): 100 = 925–931 MHz 101 = 932–1035 MHz 110 = 1036–1123 MHz 111 = 1124–1125 MHz For desired center frequencies of 1125 or above (in register 0x0 above, bit 7 (DIV2) = 1): 000 = 1126–1216 001 = 1217–1355 010 = 1356–1512 011 = 1513–1670 100 = 1671–1863 101 = 1864–2071 110 = 2072–217
3	7	(none)	0
	6–0	F6–0	Write these bits using the equations below (rounding the results to the nearest integer): If M-counter value is 01010: $(desired\ baseband\ bandwidth\ in\ MHz - 4) \times 6.9$ If M-counter value is 00101: $(desired\ baseband\ bandwidth\ in\ MHz - 8) \times 3.45$
4	7	ADL	0 = disable, 1 = enable (ADC latch)
	6	ADE	0 = disable, 1 = enable (tuning voltage DAC read)
	5	DL	1 (differential output drive level)

Graychips (GC4016)

The SRXL has two onboard Graychips (Texas Instruments GC4016) for digital down-conversion.

NOTE Each GC4016 has many programmable registers (programmed through the SRXL FPGA). To find a GC4016 datasheet, use the link to the manufacturer under [Related Resources on page 2](#).

To write a GC4016 register:

1. Write the address of the desired GC4016 register into the [0x64 Graychip Address Register](#); bit 7 (GC_SELECT) determines which Graychip you write.
2. Write the data intended for the address in step 1 to the [0x65 Graychip Data Register](#).

To read a GC4016 register:

1. Write the address of the desired GC4016 register to [0x64 Graychip Address Register](#); bit 7 (GC_SELECT) determines which Graychip you read.
2. Read the GC4016 register data in the [0x65 Graychip Data Register](#).

Acquiring a Signal

You can acquire a signal in DMA mode (normal operation) or test mode, using the [0x63 Capture Control Register](#).

- In DMA mode, the digitized data is transferred continuously to the host via DMA; the size of the transfer is determined by software.
- In test mode, a FIFO is used to store a “snapshot” of the digitized data, which is read one byte at a time using register I/O.

DMA Mode

For DMA mode, follow the steps below to start the data transfer.

1. Clear bit 1 (test mode) of the [0x63 Capture Control Register](#).
2. Set and then clear bit 2 (FIFO reset) of the [0x63 Capture Control Register](#).
3. Set the appropriate bit(s) in the [0x10 Channel Enable Low – DMA Channels 7–0 Register](#).
 - For IF, set bit 0 (CH_ENABLE) to enable DMA channel 0.
 - For L-band, set bit 1 (CH_ENABLE) to enable DMA channel 1.
 - For both IF and L-band, set bits 0 and 1 to enable both DMA channels.
4. To enable the DMA interface in the FPGA on the main board, set bit 3 (CMD_EN) in the [0x00 Command Register](#) on the main board.
5. Set bit 7 (Capture) of the [0x63 Capture Control Register](#) to start acquiring data in the input FIFOs.
6. Immediately make the appropriate EDT driver call(s) to start DMA on the main board FPGA.

Test Mode

The test mode captures 2048 32-bit data samples (each of which includes both L-band and IF data) to the input data FIFOs. This data is read using programmed I/O rather than DMA. Each data sample is four bytes, with the data represented as shown below.

bits 31–28	bits 27–16	bits 15–12	bits 11–0
0000	Q channel data	0000	I channel data

The least significant byte of each word reads back first. For each byte, bit 0 (Channel select) of the [0x63 Capture Control Register](#) selects whether L-band or IF data will be read from the [0x60 Read Data Register](#).

To capture and read data in test mode:

1. Set and then clear bit 2 (FIFO reset) of the [0x63 Capture Control Register](#).
2. Set bit 1 (Test mode) and bit 7 (Capture) of the [0x63 Capture Control Register](#).
3. Wait for bit 7 (XFER_BUSY) in [0x54 Serial Control Register](#) to clear.
4. Read back captured data one byte at a time from the [0x60 Read Data Register](#). After each read, advance the internal FIFO pointers to the next byte by writing any value to that same register.

PRBS Test Pattern

To test DMA transfer, the SRXL 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 0 (PRBS_EN) of the [0x62 PRBS Enable Register](#).

Registers

The SRXL registers are shown below.

0x00 Command Register

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 Channel Enable Registers (0x10 Channel Enable Low – DMA Channels 7–0 Register and 0x11 Channel Enable High – DMA Channels 15–8 Register) that start on page 16 . 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 Register

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 Register

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 DDS Control Register

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

Bits	Name	Description
7–2	[no name]	Not used.
1	DDS_RESET	Set, then clear to reset the AD9951 device; then reprogram the AD9951 before operation.
0	DDS_UPDATE	Set to transfer all bytes written to its internal buffers to its internal registers.

0x51 DDS Data Register

Access / Notes: DDS_DATA / 8-bit read-write.

Bits	Name	Description
7–0	DDS_DATA	Write data to be written to the AD9951.

0x54 Serial Control Register

Access / Notes: SERIAL_CTRL / 8-bit read-write
Control and status bits for functions accessed by serial interface.

Bits	Name	Description
7	XFER_BUSY	Firmware sets this bit when any of the serial interfaces are busy; wait until it is clear before starting a new read or write to any SRXL device.
6	I2C_ERROR	Firmware sets this bit after any I ² C data transfer for which an ACK was not received, indicating that the Maxim2118 is not responding. Firmware clears before starting a new I2C data transfer.
5	IF_FTEST	Status bit from the IF local oscillator PLL synthesizer. By default, if set, the PLL loop is locked; if clear, it is not locked.
4–1	[no name]	Not used.
0	I2C_READ	Set when reading the Maxim2118 status register. Clear when writing to any Maxim2118 register.

0x55 Maxim Address Register

Access / Notes: LBAND_ADDR / 8-bit read-write
Control and status bits for functions accessed by serial interface.

Bits	Name	Description
7–0	[no name]	Address of MAX2118 register to which to write the data word in Maxim Data register.

0x56 Maxim Data Register

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

READ STATUS

Bits	Name	Description
7	[no name]	Not used (0).
6	PWR	Set for the first read after powerup.
5	[no name]	Not used (0).
4–2	ADC	3-bit value from the VCO tuning voltage analog-to-digital converter inside the Maxim device, used to search for the correct VCO and verify that the VCO loop is locked.
1–0	[no name]	Not used (0).

WRITE

Bits	Name	Description
7–0	[no name]	8-bit data word to write.

0x58 DAC A Low Register

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

Bits	Name	Description
7–2	DACA_LOW	Least significant component of voltage specification for L-band signal gain.
1–0	[no name]	Not used.

0x59 DAC A High Register

Access / Notes: 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 registers.

Bits	Name	Description
7–4	[no name]	Not used.
3–0	DACA_HIGH	Most significant component of voltage specification for L-band signal gain.

0x5A DAC B Low Register

Access / Notes: 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 registers.

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

0x5B DAC B High Register

Access / Notes: 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 registers.

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

0x5C PLL Low Register

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

Bits	Name	Description
7–3	PLL_LOW	Bits 7–3 of the PLL internal register.
2–0	LMX_REG_ADDR	The register address in the LMX2364.

0x5D PLL Mid Register

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

Bits	Name	Description
7–0	PLL_MID	Bits 15–8 of the PLL internal register.

0x5E PLL High Register

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

Bits	Name	Description
7–0	PLL_HIGH	Bits 23–16 of the PLL internal register.

0x5F Sample Clock Control Register

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

Bits	Name	Description
7	EXT_REF_SEL	Setting this bit selects the external reference input as the source for the 10 MHz reference clock. For proper operation, be sure a 10 MHz clock signal source is connected to the SRXL. When clear, selects the onboard TCXO for the 10 MHz reference clock.
6	LED_ON	1 = sets the LED to a rapid blink; 0 = sets the LED to a slow blink.
5	AD_DFS	1 = selects twos-complement output format for the A/D converters; 0 = selects offset-binary format.
4	AD_DCS	The analog-to-digital clock stabilizer bit. Set for sample clocks of 40 MHz and above; clear for frequencies below 40 MHz.
3–0	[no name]	Not used.

0x60 Read Data Register

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

Bits	Name	Description
7–0	[no name]	For test mode only. Write to advance the FIFO pointers; read one byte at a time from the IF or L-band FIFO.

0x62 PRBS Enable Register

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

Bits	Name	Description
7–1	[no name]	Not used.
0	PRBS_EN	Set to generate PRBS15 test pattern; clear to stop generating test pattern. Use <code>checkprbs15</code> application to check the pattern and detect any errors.

0x63 Capture Control Register

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

Bits	Name	Description
7	Capture	When set, A/D data is being acquired, in either test or DMA mode. When clear, no data is being acquired.
6	L-band OTR	Read only. Set if the L-band ADC reported an overrange on any data acquired since the last reset by bit 2 (FIFO reset), meaning the input signal exceeded the range of the ADC.
5	IF OTR	Read only. When set, the IF ADC reported an overrange on any data acquired since the last reset by bit 2 (FIFO reset), meaning the input signal exceeded the range of the ADC.
4	L-band FIFO overrun	Read only. When set, L-band input data FIFO filled during an acquisition and lost data since the last reset of bit 2 (FIFO reset).
3	IF FIFO overrun	Read only. When set, IF input data FIFO filled during an acquisition and lost data since the last reset of bit 2 (FIFO reset)
2	FIFO reset	Set and then clear this bit to clear ADC input FIFO before starting data acquisition. Setting this bit also clears bits 6–3 (OTR and flag overrun).
1	Test mode	Set for test mode; clear for DMA mode.
0	Channel select	Set to read back L-band data, or clear to read back IF data, in test mode.

0x64 Graychip Address Register

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

Bits	Name	Description
7	GC_SELECT	When clear, selects Graychip number 1. When set, selects Graychip number 2.
6–5	[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 Register

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

Bits	Name	Description
7–0	GC_DATA	Contains the data to write to the register specified in 0x64 register, or the data read from it.

0x7F Board ID Register

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

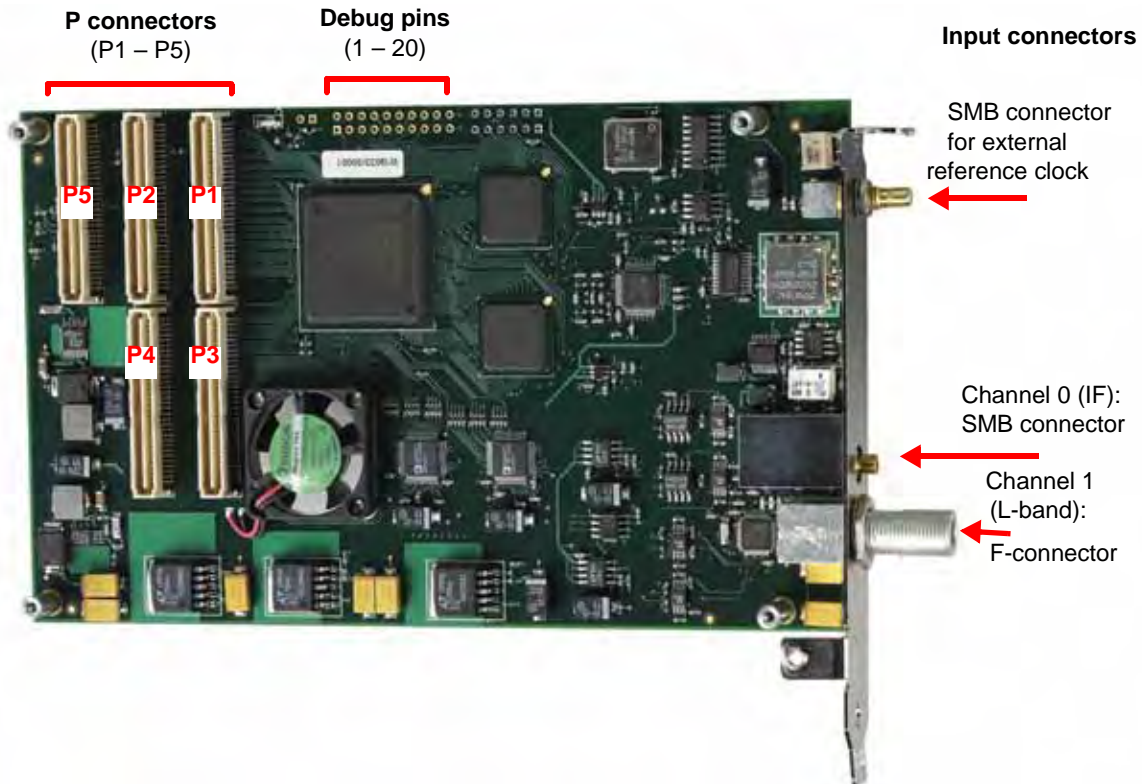
Used to identify EDT mezzanine boards. A value of 0x2 in the lowest four bits indicates an extended board ID, which is hard-wired into a nonvolatile complex programmable logic device (CPLD).

Bits	Name	Description
7–4	[no name]	Used by <code>extbdid.exe</code> .
3–0	BOARD_ID	SRXL: 0010 (CPLD extended board ID = 0x0A) Other board IDs: See the user's guide for hardware interfaces on the EDT website.

Connectors and Pins

Figure 3, with its accompanying notes, identifies the connectors and pins on the SRXL.

Figure 3. SRXL Connectors and Pins



5 P-connectors

These 64-pin CMC-type connectors (P1 – P5) link the SRXL 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).

3 input connectors

There are three input connectors:
- One SMB connector for external reference clock.
- One SMB connector for IF input.
- One SMB connector for L-band input.

Pinouts

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

The first five tables cover the pinouts from the SRXL 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.
BD IDx	A board identification signal, where x is replaced with a value (see 0x7F Board ID Register on page 20).
extbidid...	These pins are the interface for the extended board ID.
free	These pins have wires connecting the SRXL 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 SRXL FPGA.
ground	These pins are the ground connection.
unused	These pins do not have wires connecting the SRXL FPGA to any outside device, so the signals cannot be accessed.

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

Table 5. SRXL FPGA to P1 Connector

P1 Pin	SRXL FPGA Pin	SRXL Signal	P1 Pin	SRXL FPGA Pin	SRXL Signal
P1.1	D3	FPGA PROG_B	P1.33	AE7	free
P1.2		unused	P1.34		ground
P1.3		ground	P1.35		ground
P1.4	W2	free	P1.36	AF7	free
P1.5	W1	free	P1.37	AE8	free
P1.6	Y2	free	P1.38		+5 V
P1.7		BD ID0	P1.39		ground
P1.8		+5 V	P1.40	AF8	free
P1.9	AA2	free	P1.41	AE10	free
P1.10	Y1	free	P1.42	AE9	free
P1.11		ground	P1.43	AF10	free
P1.12	AA1	free	P1.44		ground
P1.13	AB2	free	P1.45		unused
P1.14		ground	P1.46	AE11	free
P1.15		ground	P1.47	AE12	free
P1.16	AB1	free	P1.48	AF11	free
P1.17	AC2	free	P1.49	AF12	free
P1.18		+5 V	P1.50		+5 V
P1.19		unused	P1.51		ground
P1.20	AC1	free	P1.52	AF13	free
P1.21	AD1	free	P1.53	AF15	free
P1.22	AD2	free	P1.54	AE15	free
P1.23	AE4	free	P1.55	AE16	free
P1.24		ground	P1.56		ground
P1.25		ground	P1.57		unused
P1.26	AF4	free	P1.58	AF16	free
P1.27	AF5	free	P1.59	AF17	free
P1.28	AE5	free	P1.60	AE17	free
P1.29	AE6	free	P1.61	AE18	free
P1.30		+5 V	P1.62		+5 V
P1.31		unused	P1.63		ground
P1.32	AF6	free	P1.64	AD14	free

Table 6. SRXL FPGA to P2 Connector

P2 Pin	SRXL FPGA Pin	SRXL Signal	P2 Pin	SRXL FPGA Pin	SRXL Signal
P2.1		unused	P2.33		ground
P2.2	W3	free	P2.34	AC10	free
P2.3	Y4	free	P2.35	AC11	free
P2.4	W4	free	P2.36		unused
P2.5	AA3	free	P2.37		ground
P2.6		ground	P2.38	AD12	free
P2.7		ground	P2.39	AB12	free
P2.8	AA4	free	P2.40		ground
P2.9	AB4	free	P2.41		ground
P2.10	AB3	free	P2.42	AC13	free
P2.11		BD ID1	P2.43	AB13	free
P2.12		unused	P2.44		ground
P2.13	AD4	free	P2.45	AC14	FPGA INIT
P2.14		BD ID2	P2.46	AF14	free
P2.15		unused	P2.47		ground
P2.16		BD ID3	P2.48	AD15	free
P2.17	AC5	free	P2.49	AC16	free
P2.18		ground	P2.50		unused
P2.19	AC6	free	P2.51	AC17	free
P2.20	AD5	free	P2.52	AB16	free
P2.21		ground	P2.53		unused
P2.22	AD6	free	P2.54	AD17	free
P2.23	AB7	free	P2.55	W16	free
P2.24		unused	P2.56		ground
P2.25	AD8	free	P2.57	Y16	free
P2.26	AC7	free	P2.58	AA16	free
P2.27		unused	P2.59		ground
P2.28	AC8	free	P2.60	Y17	free
P2.29	AD9	free	P2.61	AA17	free
P2.30		ground	P2.62		unused
P2.31	AD10	free	P2.63		ground
P2.32	AC9	free	P2.64	AB17	free

Table 7. SRXL FPGA to P3 Connector

P3 Pin	SRXL FPGA Pin	SRXL Signal	P3 Pin	SRXL FPGA Pin	SRXL Signal
P3.1	AE20	free	P3.33		ground
P3.2		ground	P3.34	AF20	free
P3.3		ground	P3.35	W26	free
P3.4	AE21	free	P3.36	AF19	free
P3.5	AF21	free	P3.37	V25	free
P3.6	AE22	free	P3.38		ground
P3.7	AF22	free	P3.39		unused
P3.8		ground	P3.40	AE19	free
P3.9		unused	P3.41	U26	free
P3.10	AE23	free	P3.42	V20	free
P3.11	AE24	free	P3.43	U25	free
P3.12	AF24	free	P3.44		ground
P3.13	AD25	free	P3.45		ground
P3.14		ground	P3.46	W23	free
P3.15		ground	P3.47	T26	free
P3.16	AD26	FPGA CCLK	P3.48	V24	free
P3.17	AC25	free	P3.49	T25	free
P3.18	AC26	free	P3.50		ground
P3.19	AB25	free	P3.51		ground
P3.20		ground	P3.52	U23	free
P3.21		unused	P3.53	R26	free
P3.22	AB26	free	P3.54	R24	free
P3.23	AA25	free	P3.55	R25	free
P3.24	AA26	free	P3.56		ground
P3.25	Y25	free	P3.57		unused
P3.26		ground	P3.58	AB20	free
P3.27		ground	P3.59	P26	free
P3.28	Y26	free	P3.60	P23	free
P3.29	Y21	free	P3.61	P25	free
P3.30	AF23	free	P3.62		ground
P3.31	W25	free	P3.63		ground
P3.32		ground	P3.64	Y20	free

Table 8. SRXL FPGA to P4 Connector

P4 Pin	SRXL FPGA Pin	SRXL Signal	P4 Pin	SRXL FPGA Pin	SRXL Signal
P4.1		unused	P4.33	V21	free
P4.2	AC18	free	P4.34	U21	free
P4.3	AB18	free	P4.35	T21	free
P4.4	AD18	free	P4.36		ground
P4.5	AA18	free	P4.37	AC24	FPGA DONE
P4.6		ground	P4.38	R21	free
P4.7	Y18	free	P4.39	AB24	free
P4.8	AC19	free	P4.40	W20	free
P4.9	AD19	free	P4.41		ground
P4.10	AB19	free	P4.42	U20	free
P4.11		ground	P4.43	AB23	free
P4.12	AA19	free	P4.44	T20	free
P4.13	Y19	free	P4.45	AA24	free
P4.14	AD23	free	P4.46		ground
P4.15	AD22	free	P4.47	AA23	free
P4.16		ground	P4.48	R20	free
P4.17	AB22	free	P4.49	Y23	free
P4.18	AC22	free	P4.50	W24	free
P4.19	Y22	free	P4.51		ground
P4.20	AA22	free	P4.52	U24	free
P4.21		ground	P4.53	T23	free
P4.22	W22	free	P4.54	R22	free
P4.23	AD21	free	P4.55	P24	free
P4.24	AC21	free	P4.56		ground
P4.25	V22	free	P4.57	AC20	free
P4.26		ground	P4.58	AA20	free
P4.27	U22	free	P4.59		unused
P4.28	AB21	free	P4.60		unused
P4.29	T22	free	P4.61		ground
P4.30	AA21	free	P4.62		unused
P4.31		ground	P4.63		unused
P4.32	W21	free	P4.64		unused

Table 9. SRXL FPGA to P5 Connector

P5 Pin	SRXL FPGA Pin	SRXL Signal	P5 Pin	SRXL FPGA Pin	SRXL Signal
P5.1		unused	P5.33	Y12	free
P5.2	W5	free	P5.34		unused
P5.3	Y5	free	P5.35	W12	free
P5.4	W6	free	P5.36		ground
P5.5	Y6	free	P5.37	AA13	free
P5.6		ground	P5.38		unused
P5.7	AB5	free	P5.39	Y13	free
P5.8	Y7	free	P5.40		extbdid CLK
P5.9	AA6	free	P5.41		ground
P5.10	AB6	free	P5.42		extbdid DATA
P5.11		ground	P5.43	W13	free
P5.12	AA7	free	P5.44		unused
P5.13	AA8	free	P5.45	AB14	free
P5.14	AB8	free	P5.46		ground
P5.15	Y8	free	P5.47	AA14	free
P5.16		ground	P5.48		unused
P5.17	AA9	free	P5.49	Y14	free
P5.18	AB9	free	P5.50		unused
P5.19	AA10	free	P5.51		ground
P5.20	Y9	free	P5.52		unused
P5.21		ground	P5.53	W14	free
P5.22	AB10	free	P5.54		unused
P5.23	AB11	free	P5.55	AB15	free
P5.24	Y10	free	P5.56		ground
P5.25	AA11	free	P5.57	AA15	free
P5.26		ground	P5.58		unused
P5.27	W11	free	P5.59	Y15	FPGA DIN
P5.28	Y11	free	P5.60		unused
P5.29	AA12	free	P5.61		ground
P5.30		unused	P5.62		unused
P5.31		ground	P5.63	W15	free
P5.32		unused	P5.64		unused

Table 10. SRXL FPGA to Graychips (GCs) – part 1 of 2

GC Pin Name	GC Pin #	SRXL FPGA Pin to GC 1	SRXL FPGA Pin to GC 2	GC Pin Name	GC Pin #	SRXL FPGA Pin to GC 1	SRXL FPGA Pin to GC 2
AIN[0]	D14	B10	G22	DIN[10]	F1	K1	A15
AIN[1]	E13	P4	F23	DIN[11]	F2	K2	F15
AIN[2]	E12	P3	H26	DIN[12]	E3	L7	F14
AIN[3]	E14	A10	G23	DIN[13]	E1	L2	B15
AIN[4]	F13	E9	G26	C[0]	B9	P6	J20
AIN[5]	F12	C9	E23	C[1]	A9	N1	J21
AIN[6]	F14	B9	H25	C[2]	C9	P7	H21
AIN[7]	G13	F8	F26	C[3]	B8	N2	H12
AIN[8]	G14	G8	G25	C[4]	C8	N5	E11
AIN[9]	H12	G6	E21	C[5]	A7	N4	A11
AIN[10]	H11	F6	D20	C[6]	B7	N3	E12
AIN[11]	H14	G7	F25	C[7]	A6	M1	B11
AIN[12]	H13	D7	E26	P[0]	B11	R7	H23
AIN[13]	J12	E6	D21	P[1]	C11	R5	H22
BIN[0]	J14	C8	E25	P[2]	A11	R6	J23
BIN[1]	J13	E7	D26	P[3]	B10	P1	H24
BIN[2]	K12	D6	C21	P[4]	D9	P8	D22
BIN[3]	K14	D8	D25	P[5]	C7	N6	F12
BIN[4]	K13	F7	C26	P[6]	D7	N7	G12
BIN[5]	L14	E8	C25	P[7]	D5	L6	G13
BIN[6]	L12	K7	C23	P[8]	D4	R8	H14
BIN[7]	M11	G5	B23	P[9]	F4	K4	D14
BIN[8]	P11	R2	A23	P[10]	H3	E1	G16
BIN[9]	L10	C5	F20	P[11]	H4	E2	C15
BIN[10]	M10	D5	G20	P[12]	J4	H7	E16
BIN[11]	M9	F4	C19	P[13]	K4	H6	F17
BIN[12]	P9	A6	A22	P[14]	M6	E3	D18
BIN[13]	L8	E4	D19	P[15]	M7	C4	E19
CIN[0]	N7	B4	B19	P[16]	N8	B5	B20
CIN[1]	L6	F3	C17	P[17]	N9	B6	B21
CIN[2]	P6	A3	A19	P[18]	N10	E5	B22
CIN[3]	N6	B3	C18	P[19]	L11	G4	E20
CIN[4]	N5	D2	F19	P[20]	K11	F5	F21
CIN[5]	L4	H5	G18	P[21]	J11	C6	G21
CIN[6]	M4	F2	E18	P[22]	F11	D9	E22
CIN[7]	N4	D1	G19	P[23]	E11	P5	C22
CIN[8]	L2	H1	H20	A[0]	C6	N8	C12
CIN[9]	L3	G2	F18	A[1]	B6	M2	H13
CIN[10]	L1	J7	B18	A[2]	A5	M3	A12
CIN[11]	K2	H2	D17	A[3]	C5	L5	F13
CIN[12]	K3	G1	E17	A[4]	B5	L4	E13
CIN[13]	K1	J6	A17	RD	A4	M7	B12

Table 10. SRXL FPGA to Graychips (GCs) – part 2 of 2

GC Pin Name	GC Pin #	SRXL FPGA Pin to GC 1	SRXL FPGA Pin to GC 2	GC Pin Name	GC Pin #	SRXL FPGA Pin to GC 1	SRXL FPGA Pin to GC 2
DIN[0]	J2	H3	G17	WR	C4	M5	C13
DIN[1]	J3	F1	D16	CE	B4	M6	D13
DIN[2]	J1	J5	B17	CK	P7	A4	A20
DIN[3]	H2	H4	F16	SCK	D12	R3	E24
DIN[4]	H1	J4	A16	SFS	D13	R1	F24
DIN[5]	G3	K6	E15	RDY	A10	P2	J22
DIN[6]	G4	J3	H15	SIB	D1	L1	A14
DIN[7]	G1	J2	B16	SIA	D3	M8	G14
DIN[8]	G2	K5	H16	SO	E2	L8	E14
DIN[9]	F3	K3	G15	DVAL	P8	A5	A21

Table 11. SRXL FPGA to Other Devices

	Pin	Signal Description	Pin	Signal Description
Reference Clock	B14	Reference clock into FPGA	B7	External or internal reference selector
Sample Clock	F9	AD9951 clock programming	H11	AD9951 update
	C10	AD9951 data programming	E10	Positive differential sample clock
	G9	AD9951 reset	F10	Negative differential sample clock
Channel 0: IF Oscillator	A8	LMX2364 clock programming	A7	LMX2364 load enable
	B8	LMX2364 data programming	G10	LMX2364 lock
Channel 1: L-band Tuner	P19	MAX2118 clock I ² C	N26	MAX2118 data I ² C
ADC	L26	Bits 0 (L-band digital data inputs)	K22	Bit 3
	M19	Bit 1	K23	Bit 4
	M20	Bit 2	K24	Bit 5
	M21	Bit 3	K25	Bit 6
	M22	Bit 4	K26	Bit 7
	M24	Bit 5	L19	Bit 8
	M25	Bit 6	L20	Bit 9
	M26	Bit 7	L21	Bit 10
	N19	Bit 8	L22	Bit 11
	N20	Bit 9	L23	IF ADC overflow
	N21	Bit 10	N23	L-band ADC overflow
	N22	Bit 11	N24	ADC clock stabilizer
	J25	Bit 0 (IF digital data input)	N25	ADC format selector
	K20	Bit 1	L25	sample clock to L-band ADC
	K21	Bit 2	J24	sample clock to IF ADC
DAC	P21	DAC clock programming	P22	DAC chip select
	P20	DAC data programming		
Debug	V2	Pin 1	U5	Pin 11
	V3	Pin 2	U6	Pin 12
	V4	Pin 3	U7	Pin 13
	V5	Pin 4	T1	Pin 14
	V6	Pin 5	T2	Pin 15
	V7	Pin 6	T4	Pin 16
	U1	Pin 7	T5	Pin 17
	U2	Pin 8	T6	Pin 18
	U3	Pin 9	ground	Pin 19
	U4	Pin 10	ground	Pin 20
LED Signal	W7	LED fast blink		

Revision Log

Below is a history of modifications to this guide.

Date	By	Pp	Detail
20100409	PH, BO	24-35	Deleted duplicate 0x64 register; corrected pinouts.
20100409	PH	All	Updated formatting and standardized text.