Publication No. REF-320-002-R01

Channel Adapter and Channel Adapter Plus Software Reference Manual

Red Rapids

797 North Grove Rd, Suite 101
Richardson, TX 75081
Phone: (972) 671-9570
www.redrapids.com

Table of Contents

1.0 Introduction 1
1.1 Contents and Structure 1
1.2 Supporting Documents 1
1.3 Conventions 1
1.4 Manual Compatibility 2
1.5 Revision History 2
2.0 Application Program Interface 3
2.1 General API Usage 3
2.2 API C Library Versions 3
2.2.1 Visual Studio 6 (x86-32) 3
2.2.2 Linux, x86-32 with gcc3 3
2.3 API Functions 4
2.3.1 Handle 4
2.3.2 Access Functions 5
2.4 Interrupt Service Routine 6
3.0 Hardware Initialization and Operation 8
3.1 Card Initialization 9
3.1.1 Clock Initialization 9
3.1.2 ADC Input Voltage Range 9
3.1.3 FPGA Program Status 9
3.1.4 Hardware Specific Control Registers 9
3.2 DMA Transfers 9
3.3 Application Specific Configuration 11
3.4 Interrupt Processing 11
3.5 Terminate Processing 11
4.0 Application Software 12
4.1 Diagnostic Software (diagnostic.exe) 12
4.2 Local Bus Interface (localbus.c) 12
4.3 FPGA and Configuration PROM Programming (load_XSVF.c) 13
5.0 Memory Map 14
5.1 BAR0 Controller Memory Map 15
5.2 BAR2 DMA Memory Map 30

List of Figures

Figure 3-1 Initialization and Operation Flowchart.. 8
Figure 3-2 Format of a DMA Transfer 10
Figure 3-3 Scatter-Gather DMA Sequence 10
Figure 4-1 Local Bus DMA Transfer Datapath 12

List of Tables

Table 2-1 s_ChannelAdapter Handle Structure Members 4

1.0 Introduction

1.1 Contents and Structure

This manual describes the Channel Adapter and Channel Adapter Plus software features. This publication supplements the supporting documents listed in section 1.2 to provide a complete description of the capabilities and operation of the product. The focus of this manual is the software API, example code, and configuration register descriptions. Other Channel Adapter manuals focus on the hardware and FPGA programming.

Section	**Description**
Section 1	Introductory information about the manual.
Section 2	Overview of functions provided in the application programming interface.
Section 3	Procedures for initializing the hardware and activating features through software.
Section 4	Application examples supplied with sample code.
Section 5	Summary of the memory map and detailed register descriptions.

The latest product documentation and software is available for download from the Red Rapids web site (www.redrapids.com) by following the Technical Support link.

1.2 Supporting Documents

Author	**Number**	**Title**
Red Rapids	REF-314-000	Model 314 Hardware Reference Manual
Red Rapids	REF-316-000	Model 316 Hardware Reference Manual
Red Rapids	REF-320-000	Model 320 Hardware Reference Manual
Red Rapids	REF-321-000	Model 321 Hardware Reference Manual
Red Rapids	REF-314-001	Model 314 Installation Guide
Red Rapids	REF-316-001	Model 316 Installation Guide
Red Rapids	REF-320-001	Model 320 Installation Guide
Red Rapids	REF-321-001	Model 321 Installation Guide
Red Rapids	REF-320-003	Channel Adapter FPGA Core Reference Manual
PCI SIG	PCI Rev 2.2	PCI Local Bus Specification

1.3 Conventions

This manual uses the following conventions:

- Hexadecimal numbers are prefixed by “0x” (e.g. 0x00058C).
- *Italic* font is used for names of registers.
- **Bold** font is used for names of directories, files and OS commands.
- Palatino font is used to designate source code.
- Active low signals are followed by ‘#’, For example, TRST#.

☞ Text in this format highlights useful or important information.

!	Text shown in this format is a warning. It describes a situation that could potentially damage your equipment. Please read each warning carefully.

The following are some of the acronyms used in this manual.

• **ADC**	Analog to Digital Converter
• **API**	Application Program Interface
• **BAR**	Base Address Register
• **DCM**	Digital Clock Manager
• **DMA**	Direct Memory Access
• **GPIO**	General Purpose Input/Output
• **HAL**	Hardware Abstraction Layer
• **ISR**	Interrupt Service Routine
• **MSPS**	Mega Samples per Second
• **PCI**	Peripheral Component Interconnect

1.4 Manual Compatibility

The applicable hardware part numbers are defined as follows:

• Model 314-XXX	(Channel Adapter 14/125)
• Model 316-XXX	(Channel Adapter Plus 8/1500)
• Model 320-XXX	(Channel Adapter 12/213)
• Model 321-XXX	(Channel Adapter Plus 16/130)

1.5 Revision History

Version	**Date**	**Description**
R00	10/17/06	Initial release.

2.0 Application Program Interface

The Channel Adapter application program interface (API) provides C functions to access all hardware resources from the host. These functions are the basis for sample programs provided by Red Rapids that can be used as a foundation for custom software development.

- Features of the API include:
- Unified Windows and Linux driver structure.
- Support for up to eight cards in a single system.
- Support for DMA transfers.
- Interrupt processing for error and status indicators.

2.1 General API Usage

A hardware abstraction layer (HAL) link must be established prior to accessing the hardware using the API functions. The HAL link is created by successfully calling the Adapter_Open() function. The Adapter_Close() function must be called prior to exiting any program that has issued a Adapter_Open() command to de-allocate system memory assigned to the hardware. Failure to do so will result in memory leakage and eventual system crash. Be sure to include the file **channeladapter.h** in any application that calls an Adapter_ function. This file contains the prototypes for all functions in the API.

2.2 API C Library Versions

For each supported development environment and host platform a specific version of the C library containing the API is provided in the **channeladapterlib** directory. The supported platforms and library locations are detailed below.

Development Platform	
Visual Studio 6 (x86–32)	**win32_vs6/channeladapter.lib**
Linux, x86-32, gcc3	**linux_x86-32_gcc3/libchanneladapter.a**

2.2.1 Visual Studio 6 (x86-32)

Add **c:\<extract_path>\channeladapterlib\RXX\win32_vs6\channeladapter.lib** into the project and add **c:\<extract_path>\channeladapterlib\RXX\** to the include path. The project must be set to generate multithreaded code since the interrupts are handled in a separate thread.

!	Programs will crash if multithreaded code generation is not turned on.

2.2.2 Linux, x86-32 with gcc3

Link the application to the static file:

</extract_path>/channeladapterlib/RXX/linux_x86-32_gcc3/libchanneladapter.a

The project must be compiled with the pthread library since the interrupts run in a separate thread from the main program.

The following should be added to the include path and the symbol LINUX should be defined for any object that includes the *channeladapter.h* file:

</extract_path>/channeladapterlib/RXX/

2.3 API Functions

The API supplied with Channel Adapter consists of a memory map to all local registers, a handle to access the primary hardware features, and software functions to initialize the card and perform data transfers.

2.3.1 Handle

The s_ChannelAdapter handle provides the mechanism to access the Channel Adapter hardware.

Table 2-1 s_ChannelAdapter Handle Structure Members

Member Name	Data Type	Definition
DevNum	integer	Detection order for multiple Channel Adapter boards. Starts at zero.
UnitsFound	integer	Number of Channel Adapter boards in the system.
RevCode	DWORD	Hardware revision of the PCI bridge.
HwConfig	DWORD	Hardware description of the board.
DMA [2]	s_DMAConfig	Structure that contains DMA configuration information.
IntrMask	DWORD	Stores a copy of the interrupt mask for re-enabling interrupts after processing.
DevStatus	DWORD	Status flags for the s_ChannelAdapter handle.
ReturnStatus	DWORD	Stores function return codes.
FILE * debugFP	FILE *	Output location for messages in debug enabled version of the API

There are functions defined in the API for initializing the handle. These functions are described in the following sections.

Prototype	DWORD Adapter Zero(s_ChannelAdapter *pCA);
Arguments	s_ChannelAdapter * pCA – Pointer to a s_ChannelAdapter handle.
Description	Zeros the members of the s_ChannelAdapter handle.
Return	0 if OK Non-zero if error

Prototype	char * Adapter_GetAPIVerInfo(DWORD *Version);
Arguments	s_ChannelAdapter * pCA – Pointer to a s_ChannelAdapter handle. DWORD * - Returns a 32bit API revision level in the form: [31:24] Major Rev [23:16] Minor Rev [15:0] Sub Rev
Description	Returns text and numeric values providing the API revision level and target platform
Return	char * - Pointer to text string.

Prototype	char * Adapter_GetHwDescStr(s_ChannelAdapter *pCA);
Arguments	s_ChannelAdapter * pCA – Pointer to a s_ChannelAdapter handle.
Description	Returns a text string with the FPGA and PROM description.
Return	char * - Pointer to text string.

2.3.2 Access Functions

The API includes software functions to access the Channel Adapter. The functions described in the following sections are included in the **channeladapter.h** header file.

Prototype	DWORD Adapter_Open(s_ChannelAdapter *pCA);
Arguments	s_ChannelAdapter * pCA – Pointer to a s_ChannelAdapter handle.
Description	This function opens an instance of the Channel Adapter. It returns an unsigned integer based on the mapping status. It stores information about the mapped unit in the s_ChannelAdapter handle.
Return	0 if OK Non-zero if error

Prototype	DWORD Adapter_Close(s_ChannelAdapter *pCA);
Arguments	s_ChannelAdapter * pCA – Pointer to a s_ChannelAdapter handle.
Description	This function closes an open Channel Adapter.
Return	0 if OK Non-zero if error

Prototype	DWORD Adapter_Read32(s_ChannelAdapter *pCA, BAR Bar, DWORD Offset, DWORD *pData);
	DWORD Adapter_Read64(s_ChannelAdapter *pCA, BAR Bar, DWORD Offset, QWORD *pData);

Arguments	s_ChannelAdapter * pCA – Pointer to a s_ChannelAdapter handle. BAR – Memory area to read from, BRIDGE or V4. DWORD Offset – Address offset in the memory area. DWORD * pData – Returns data for 32-bit read. QWORD * pData – Returns data for 64-bit read.
Description	This function performs a bus read to the specified BAR and offset.
Return	0 if OK Non-zero if error

Prototype	DWORD Adapter_Write32(s_ChannelAdapter *pCA, BAR Bar, DWORD Offset, DWORD Data);
	DWORD Adapter_Write64(s_ChannelAdapter *pCA, BAR Bar, DWORD Offset, QWORD Data);

Arguments	s_ChannelAdapter * pCA – Pointer to a s_ChannelAdapter handle. BAR – Memory area to read from, BRIDGE or V4. DWORD Offset – Address offset in the memory area. DWORD Data – 32-bit value to write to offset. QWORD Data – 64-bit value to write to offset.
Description	This function performs a bus write to the specified BAR and offset.
Return	0 if OK Non-zero if error

Prototype	DWORD Adapter_SampleClkSelect(s_ChannelAdapter *pCA, CLKSRC ClkSrc, s_ClkSettings *pClkSettings);
Arguments	s_ChannelAdapter * pCA – Pointer to a s_ChannelAdapter handle. CLKSRC ClkSrc – OFF, SYNTH, EXT S_ClkSettings * pClkSettings – Pointer to clock configuration structure, currently reserved.
Description	Configures the A/D and digital clock source. Use SYNTH to specify the on board synthesizer, driven either by the local 10 MHz reference or an external reference. Use EXT to provide the sample clock directly from the front panel.
Return	0 if OK Non-zero if error

2.4 Interrupt Service Routine

The Channel Adapter ISR function is called void Adapter_ISR(s_ChannelAdapter *pCA). It must be included in all C programs using the Channel Adapter API, but does not need to have a body if interrupts are not used.

When the API detects a possible interrupt from the Channel Adapter hardware, it disables interrupts from the PCI Controller and checks a flag to determine whether the application code had interrupts enabled. If interrupts were enabled, the API reads a status register in the PCI Controller to determine whether the FPGA issued an interrupt. The Adapter_ISR() is called only if the FPGA was the source of the interrupt, otherwise no action is taken.

The user application code must service any FPGA interrupt status registers and interrupt mask registers from within Adapter_ISR(). A typical procedure might involve masking bits associated with persistent error conditions (i.e. ADC overflow), reading the status register, taking the appropriate action, then unmasking bits that have been cleared. The application must finally re-enable interrupts by a writing to the *QL interrupt mask* register after all interrupt have been handled.

3.0 Hardware Initialization and Operation

The operation of the Channel Adapter hardware is controlled and monitored through a series of command and status registers. These registers configure the board and control all of the operations. Figure 3-1 illustrates the minimum sequence of events that must occur to operate the card.

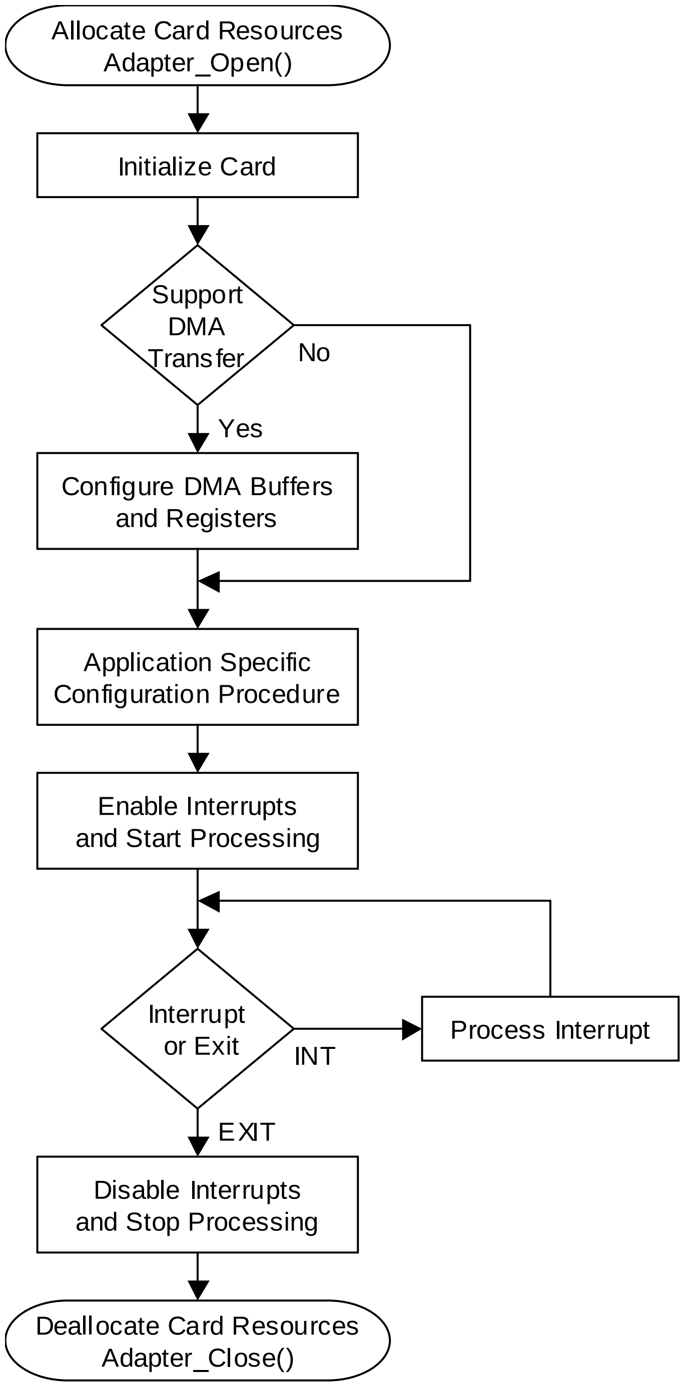
Allocate Card Resources
Adapter_Open()
Initialize Card
Support DMA Transfer
No
Yes
Configure DMA Buffers and Registers
Application Specific Configuration Procedure
Enable Interrupts and Start Processing
Interrupt or Exit
INT
Process Interrupt
EXIT
Disable Interrupts and Stop Processing
Deallocate Card Resources
Adapter_Close()

Figure 3-1 Initialization and Operation Flowchart

3.1 Card Initialization

The following initialization steps must be performed every time a Channel Adapter is powered on.

- Initialize the ADC sample clock.
- Set the ADC input voltage range.
- Check the FPGA program status.
- Set hardware specific control registers.

This initialization procedure is included in the Local Bus Interface example described in Section 4.2. The source code contains procedures for all variants of the Channel Adapter product family.

3.1.1 Clock Initialization

The Channel Adapter clock distribution network includes programmable devices that must be initialized to the correct operating frequency. The API includes a clock configuration structure to pass the required variables to these devices. This

3.1.2 ADC Input Voltage Range

The ADC input voltage range must be set to a level that is compatible with the expected signal amplitude. Each Channel Adapter product supports a different range of input voltages depending on the features of the ADC. Please consult the appropriate Channel Adapter Hardware Reference Manual for specific options.

!	The Model 320 is offered with an optional analog amplifier/attenuator circuit at the input to each channel. It is extremely important to set the analog attenuators to a level that guarantees the input cannot overdrive the ADC. Exceeding the maximum input power may result in unexpected behavior and can even damage the hardware. The default (power-on) state is full attenuation and the amplifier power turned off.

3.1.3 FPGA Program Status

Verify that the FPGA has been loaded with a valid bitstream before attempting any read or write operations to the FPGA address space. The Program Done bit in the Controller *FPGA Status* register will return a logic one if the part has been successfully loaded. The computer will hang if a PCI bus transaction is attempted to a blank FPGA.

3.1.4 Hardware Specific Control Registers

The Channel Adapter products are designed for maximum flexibility. The user has access to several programmable hardware features that may be useful, but are not required for normal operation. Consult the appropriate Hardware Reference Manual for the configuration options offered by each specific model number. Section 5.1 of this document lists all of the available configuration registers outside of the FPGA.

3.2 DMA Transfers

Bus Master DMA is the preferred method of transferring data between the Channel Adapter and host memory. Using this method, the hardware will automatically DMA data to/from a predetermined address space as it becomes available. The user application logic in the FPGA will initiate the transfer, usually when a FIFO requires service.

The *RX DMA Block Size* and *RX DMA Block Count* registers form the foundation of the scatter-gather DMA technique. As shown in Figure 3-2, these registers define the number of 64-bit (8-byte) transfers that occur in a data "group." Each "block" can transfer up to 512 kbytes and each group can transfer up to 64 kblocks, for a maximum payload of 32 Gbytes per group.

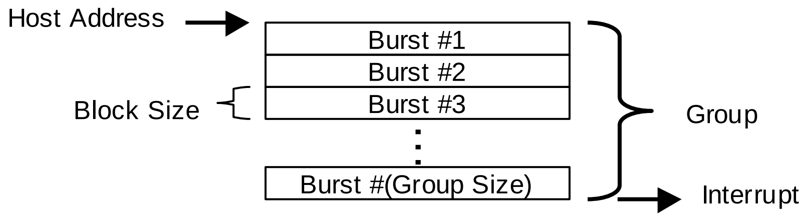

Figure 3-2 Format of a DMA Transfer

A set of 2048 *RX DMA Group Starting Address* registers are available in the DMA engine to support the scatter-gather operation. The number of available addresses is evenly distributed over the number of DMA channels. There is typically one DMA channel assigned to each signal input, but this is not required. Additional DMA channels may be desired if the application produces multiple output streams.

The addresses are arranged in 32 pages of 64 addresses. Pages are selected with the *Group Address Page Index* register. These addresses correspond to locations in host memory where sequential groups of sample data will be written as data becomes available. An interrupt is generated upon completion of the number of groups specified in the *RX DMA Interrupt Group Count* register to alert the application program of new data as shown in Figure 3-3.

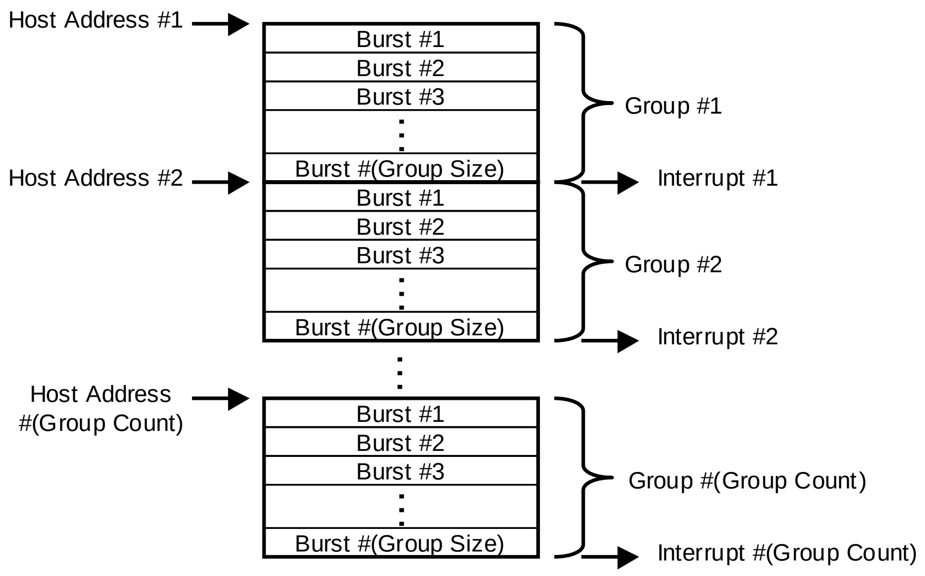

Figure 3-3 Scatter-Gather DMA Sequence

The scatter-gather DMA process continuously sequences through all of the available addresses, creating a circular buffer in host memory. The block count and number of groups used to create the circular buffer can be adjusted to accommodate the degree of interrupt latency imposed by the operating system. The theoretical maximum size of the circular buffer is 32 Tbytes.

The *RX DMA Current Group* register is used to keep track of completed transfers. This register contains the group number of the data transfer currently in progress with possible values ranging from 0 to 31. The number of completed groups can be calculated by subtracting the values obtained from two sequential read operations to this register.

The API provided with the Channel Adapter contains a handle to set up bus master DMA transfers with a minimal amount of input. All that is needed is a block count in bytes, the number of groups, and the number of blocks. The remaining values are calculated from these entries.

3.3 Application Specific Configuration

The ability to create custom FPGA code for the Channel Adapter will result in the creation of registers that are specific to an application. These registers can be accessed through the same controller interface logic that is used for default configuration registers.

3.4 Interrupt Processing

The Adapter_ISR() function must perform the following steps to process interrupts:

1. Read the *DMA Interrupt Status* and *Local Interrupt Status* registers to clear any pending interrupts.
2. Process the interrupts based on the status registers read in steps 1 and 2
3. Re-enable interrupts by writing to the DMA *Interrupt Mask* and *Local Interrupt Mask* registers.

3.5 Terminate Processing

The following sequence will terminate processing on the Channel Adapter:

1. Disable DMA activity in the *Control* register.
2. Disable Interrupts by writing 0x0 to the *DMA Interrupt Mask* and *Local Interrupt Mask* registers.
3. Read the *DMA Interrupt Status* and *Local Interrupt Status* registers to clear any undesired interrupts.
4. Release any host memory that was allocated to DMA buffer space with the *Adapter_FreeDMABuffers()* function.

4.0 Application Software

There are three application software modules supplied with the Channel Adapter hardware:

- Diagnostic software to capture a snapshot of digitized signal data (diagnostic.exe).
- Demonstration of DMA transfers from the FPGA to host memory (localbus.c).
- Utility to program the Configuration PROM or FPGA over the PCI bus (load_XSVF.c).

4.1 Diagnostic Software (diagnostic.exe)

The Channel Adapter product ships with a default FPGA bitstream preloaded in the PROM that can be used to run a quick diagnostic on the hardware. The diagnostic software performs a snapshot signal capture through the receiver and reports the magnitude and frequency of the signal captured. Please refer to the Installation Guide supplied with your product for further details.

4.2 Local Bus Interface (localbus.c)

The Channel Adapter product is distributed with VHDL and C source code to a reference design that exercises the DMA function of the local bus interface. This combined hardware and software example demonstrates the recommended technique for moving data from the Channel Adapter to host memory. The binary counters shown in the datapath illustration of Figure 4-1 are used to emulate data sources typically present in the FPGA application logic.

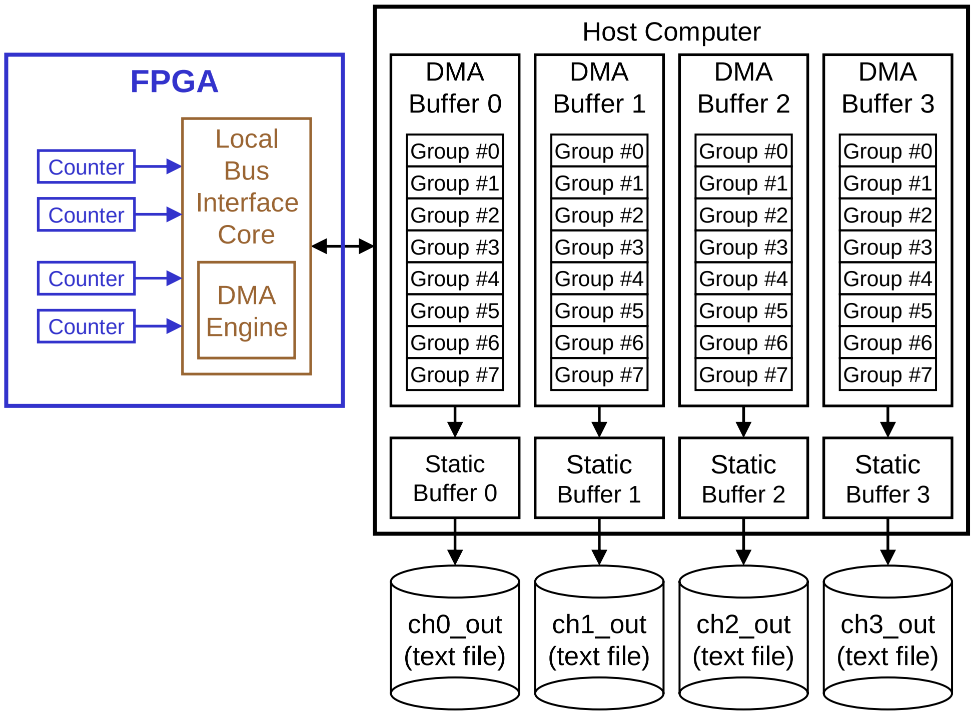

Figure 4-1 Local Bus DMA Transfer Datapath

The software also includes the initialization functions necessary to configure the digitizer, even though it is not used in the FPGA design. This allows the code to be used as a template for other applications by simply intercepting the data at that point it is copied from the DMA buffers to the static buffers.

Binary counters are used as a synthetic data source to produce a predictable sequence that can be easily parsed for errors. The bus interface is behaving properly if there are no missing or duplicated samples in the output.

The counters are connected to four DMA channels that continuously transfer data to host memory at whatever rate can be sustained by the PCI bus. The counter is incremented only when another sample is written to the core. An arbiter built into the FPGA cycles through the four DMA channel in sequence, allowing each to transfer one block of data before relinquishing the bus to the next channel.

There is one DMA buffer assigned to each channel. Each DMA buffer is partitioned into 8 groups, each group is further partitioned into 16 blocks, and the size of each block is set to 8 kbytes. Every group has a unique starting address in host memory that is associated with 128 kbytes (16 x 8 kbytes) of contiguous storage. The data is stored in 32-bit format, so there are 32 ksamples (128k bytes ÷ 4) stored in each group. The unsigned binary count is stored in the lower 27 bits of each sample and the DMA channel number is recorded in the upper five bits.

The Channel Adapter hardware will issue an interrupt when the last block in each group has been updated. The application code will service this interrupt by appending the entire group of data to a large static buffer. The size of the static buffer is specified as a command line option when the program is executed.

The collection process will terminate when all of the static buffers have been filled. The data from each static buffer is written to a text file on disk before the program exits. The text files are named chN_out.txt, where N is the channel number.

The local bus example is executed from a command line as follows:

```
localbus.exe -asy <m> -groups <g> -v <n>
```

where

m = Product model number (M314, M316, M320, or M321).

g = Number of groups to capture.

n = Verbosity level expressed as an integer from 0 (none) to 5 (maximum reporting).

The switches can be omitted from the command line. The default number of groups is 10 and the default verbosity setting is 0. The model number will default to M320, but this setting can be applied to any Channel Adapter hardware since none of the model specific configuration registers will impact the operation of the local bus.

4.3 FPGA and Configuration PROM Programming (load_XSVF.c)

The load_XSVF.c sample code demonstrates the procedure for loading a new design to the Channel Adapter from a Xilinx Serial Vector Format (XSVF) file. The new bitstream can be targeted to either the FPGA or Configuration PROM by selecting the appropriate command line switch (-FPGA or –PROM).

The diagnostic FPGA bitstream can be reloaded with this utility by downloading the XSVF file from the Red Rapids website.

5.0 Memory Map

The Channel Adapter card includes configuration registers, status registers, and data ports that are memory mapped to the address space of the host computer. The memory map of each card is referenced to two different base address registers (BARs) that separate the hardware control functions from the application controls.

BAR0 is assigned to the command/status registers that control the various hardware functions on the card. Most of these registers are initialized at start-up or when the card is opened by an application. The BAR0 memory map never changes, even as new application bit streams are loaded in the FPGA.

BAR2 is assigned to the configuration registers of the DMA engine. The DMA engine is provided as a core that should be instantiated in every FPGA design. Consequently, the DMA registers should occupy addresses 0x000 to 0x7FC in all applications.

BAR1 and BAR3 are reserved for systems capable of 64-bit addressing.

The memory map is not contiguous, any memory location not called out in the memory map should be considered reserved and should not be accessed. Writing to locations not called out in the memory map may cause erroneous operation.

The following convention is used to describe valid register operations:

(W)	Write
(R)	Read
(cl)	Read clears contents

5.1 BAR0 Controller Memory Map

The controller contains several registers that must be initialized either at system power-up or when a software application is opened. Most of the register settings remain static, but there may be cases when it is desirable to update a value during normal operation. Some of the controller functions also report hardware status through these registers. A check mark signifies the relevance of each register to a corresponding product number (e.g. Model 314).

BAR0 Controller Memory Map						
Byte Address Offset	**R/W/cl**	**314**	**316**	**320**	**321**	**Register (Group) Name**
0x000 - 0x088		Reserved				PCI Bridge Registers
0x084	cl	✓	✓	✓	✓	Global Interrupt Status
0x08C	R/W	✓	✓	✓	✓	Global Interrupt Mask
0x090 - 0x0E4		Reserved				PCI Bridge Registers
0x0E8	R	✓	✓	✓	✓	Hardware Reset
0x0EC	R/W	✓	✓	✓	✓	FPGA Reset
0x0F0 - 0x0FC		Reserved				PCI Bridge Registers
0x100	R	✓	✓	✓	✓	PCI Bridge Revision Code
0x104	R	✓	✓	✓	✓	FPGA Status
0x108 - 0x118		Reserved				Not Defined
0x11C	R/W	✓	✓	✓	✓	FPGA Programming
0x120	R/W	✓	✓	✓	✓	Temperature Sensor Control
0x124	R/W	✓	✓	✓	✓	USER Clock Control
0x128		Reserved				Not Defined
0x12C	R/W	✓		✓	✓	ADC Sample Clock Control
	R/W		✓			ADC Self Calibration
0x130	R/W	✓		✓	✓	ADC Sample Clock Serial Programming Interface
	R/W		✓			ADC Serial Programming Interface
0x134	R/W	✓	✓	✓	✓	Test Register
0x138 - 0x1FC		Reserved				Not Defined
0x200	R/W	✓				Model 314 Configuration Settings
	R/W			✓		Model 320 Configuration Settings
	R/W				✓	Model 321 Configuration Settings
0x204	R/W	✓		✓		LED Control
0x208	R/W	✓				GPIO
	R/W			✓		Channel A Attenuator Setting
0x210				✓		FPGA Attenuator Interface Enable
0x214	R/W	✓	✓	✓	✓	Synthesizer Serial Programming Interface
0x20C	R/W			✓		Channel B Attenuator Setting
0x210 - 0x3FC		Reserved				Not Defined

Address	**Register Name**
0x084	Global Interrupt Status

Product Relevance (checked box)
☒ Model 314 ☒ Model 316 ☒ Model 320 ☒ Model 321

Bit(s)	**R/W/cl**	**POR**	**Function**	**Description**
31:25			Reserved	
24	cl	0x0	Global Interrupt Status	Logic one indicates that the Channel Adapter has issued an interrupt to the host.
23:0			Reserved	

Address	**Register Name**
0x08C	Global Interrupt Mask

Product Relevance (checked box)
☒ Model 314 ☒ Model 316 ☒ Model 320 ☒ Model 321

Bit(s)	**R/W/cl**	**POR**	**Function**	**Description**
31:25			Reserved	
24	R/W	0x0	Global Interrupt Mask	Logic one enables the Channel Adapter to issue PCI interrupts, logic zero disables all interrupts.
23:0			Reserved	

Address	**Register Name**
0x0E8	Hardware Reset

Product Relevance (checked box)
☒ Model 314 ☒ Model 316 ☒ Model 320 ☒ Model 321

Bit(s)	**R/W/cl**	**POR**	**Function**	**Description**
31:1			Reserved	
0	W	0x0	Hardware Reset	Logic one performs a hardware reset of both the PCI Controller and FPGA. This bit self clears.

Address	**Register Name**
0x0EC	FPGA Reset

Product Relevance (checked box)
☒ Model 314 ☒ Model 316 ☒ Model 320 ☒ Model 321

Bit(s)	**R/W/cl**	**POR**	**Function**	**Description**
31:0			Reserved	
1	R/W	0x0	FPGA Reset	This bit is tied to the reset input of the FPGA on the local bus. A logic one places the FPGA in a reset condition, a logic zero releases reset.

Address	**Register Name**
0x100	PCI Bridge Revision Code

Product Relevance (checked box)
☒ Model 314 ☒ Model 316 ☒ Model 320 ☒ Model 321

Bit(s)	**R/W/cl**	**POR**	**Function**	**Description**
31:0	R	Rev	Revision Code	Hard-coded constant that indicates the revision level of the PCI bridge chip. Format: MM/DD/YYYY (i.e. 07/01/2006)

Address	**Register Name**
0x104	FPGA Status

Product Relevance (checked box)
☒ Model 314 ☒ Model 316 ☒ Model 320 ☒ Model 321

Bit(s)	**R/W/cl**	**POR**	**Function**	**Description**
31:1			Reserved	
0	R	0x0	FPGA Program Done	Logic one indicates that the FPGA has been successfully programmed. Note: Do not attempt to access registers in the FPGA until the firmware load is complete.

Address	**Register Name**
0x11C	FPGA Programming

Product Relevance (checked box)
☒ Model 314 ☒ Model 316 ☒ Model 320 ☒ Model 321

Bit(s)	**R/W/cl**	**POR**	**Function**	**Description**
31:16	R/W	0x0	JTAG Configuration Unlock	Holds the JTAG pins in tristate when not in use. Write 0xFEDC to program the FPGA or Configuration PROM over the PCI bus. Write 0x0000 to access the chain through the JTAG header.
15:7			Reserved	
6	R/W	0x0	Enable PROM Revision Selection	Logic one enables selection of one of four FPGA programming files stored in the Configuration PROM
5:4	R/W	0x0	PROM Revision Select	Two bit field indicating which programming file stored in the Configuration PROM should be used to load the FPGA.
3	R/W	0x0	TCK	Used to toggle the JTAG TCK line. Write logic one and logic zero to toggle TCK.
2	R/W	0x0	TMS	Used to toggle the JTAG TMS line.
1	R/W	0x0	TDI	Used to toggle the JTAG TDI line.
0	R/W	0x0	TDO	Used to read the JTAG TDO line.

Address	Register Name
0x120	Temperature Sensor Control

Product Relevance (checked box)
☒ Model 314 ☒ Model 316 ☒ Model 320 ☒ Model 321

Bit(s)	R/W/cl	POR	Function	Description
31:19	R	0x0	Temperature Status	Current temperature in 0.0625 degree C increments in 2's complement form.
18:16	R	0x0	Reserved	Always 0x7.
15:3	R/W	0xFFF	Temperature Threshold	Threshold that will set the FPGA temperature flag bit to a logic one. Uses the same format as the Temperature Status.
2:0	R/W	0x7	Reserved	Always write 0x7.

Address	Register Name
0x124	USER Clock Control

Product Relevance (checked box)
☒ Model 314 ☒ Model 316 ☒ Model 320 ☒ Model 321

Bit(s)	R/W/cl	POR	Function	Description
31:6		0x0	Reserved	
5	R/W	0x1	USER Clock Output Enable	Logic one tristates the USER clock output so that the two connector pins can be used as FPGA inputs or outputs.
4	R/W	0x0	Sample Clock Distribution Chip Enable	Logic zero disables the sample clock distribution chip. The true outputs drive to logic zero and the complementary output drives logic high in this state.
3:0		0x0	Reserved	

Address	Register Name
0x12C	ADC Sample Clock Control
Product Relevance (checked box)	
☒ Model 314 ☐ Model 316 ☒ Model 320 ☒ Model 321	

Bit(s)	R/W/cl	POR	Function	Description
31:1		0x0	Reserved	
0	R/W	0x0	Sync/Reset	Logic zero resets the ADC sample clock distribution chip. This bit must be returned to a logic one to release reset.

Address	Register Name
0x12C	ADC Self Calibration
Product Relevance (checked box)	
☐ Model 314 ☒ Model 316 ☐ Model 320 ☐ Model 321	

Bit(s)	R/W/cl	POR	Function	Description
31:1		0x0	Reserved	
0	R/W	0x0	ADC Calibration	A minimum 80 sample clock cycles of logic zero followed by a minimum 80 sample clock cycles of logic one initiates the ADC self-calibration sequence.

Address	**Register Name**
0x130	ADC Sample Clock Serial Programming Interface

Product Relevance (checked box)
☒ Model 314 ☐ Model 316 ☒ Model 320 ☒ Model 321

Bit(s)	**R/W/cl**	**POR**	**Function**	**Description**
31:4		0x0	Reserved	
3	R/W	0x0	Serial Data Clock	Synthetic clock generated by writing alternating logic zero and one values to toggle the signal.
2	R/W	0x0	Serial Data Chip Select	Set to logic zero to initiate serial programming
1	R/W	0x0	Serial Data Output	Serial data to the ADC sample clock distribution chip.
0	R	0x0	Serial Data Input	Serial data from the ADC sample clock distribution chip.

Address	**Register Name**
0x130	ADC Serial Programming Interface

Product Relevance (checked box)
☐ Model 314 ☒ Model 316 ☐ Model 320 ☐ Model 321

Bit(s)	**R/W/cl**	**POR**	**Function**	**Description**
31:4		0x0	Reserved	
3	R/W	0x0	Serial Data Clock	Synthetic clock generated by writing alternating logic zero and one values to toggle the signal.
2	R/W	0x0	Serial Data Chip Select	Set to logic zero to initiate serial programming.
1		0x0	Serial Data	Serial configuration data to the ADC.
0	R/W	0x0	Reserved	

Address	**Register Name**
0x134	Test Register

Product Relevance (checked box)
☒ Model 314 ☒ Model 316 ☒ Model 320 ☒ Model 321

Bit(s)	**R/W/cl**	**POR**	**Function**	**Description**
31:0	R/W	0x0	Test Register	This 32-bit register does not perform any function other than to provide a location to write and read back an arbitrary bit pattern as a test of the PCI interface.

Address	Register Name
0x200	Mode 314 Configuration Settings

Product Relevance (checked box)

☒ Model 314 ☐ Model 316 ☐ Model 320 ☐ Model 321

Bit(s)	R/W/cl	POR	Function	Description
31:5			Reserved	
4	R/W	0x0	ADC Voltage Range	Logic zero selects the low input voltage range to all four ADCs and logic one selects the high range.
3:2		0x0	Reserved	
1	R/W	0x0	Sample Clock Source Select	Logic zero selects the external sample clock input and disables the on-board synthesizer.
0		0x0	Reserved	

Address	Register Name
0x200	Model 320 Configuration Settings

Product Relevance (checked box)

☐ Model 314 ☐ Model 316 ☒ Model 320 ☐ Model 321

Bit(s)	R/W/cl	POR	Function	Description
31:6			Reserved	
5	R/W	0x0	ADC B Voltage Range	Logic zero selects the low input voltage range to ADC B and logic one selects the high range.
4	R/W	0x0	ADC A Voltage Range	Logic zero selects the low input voltage range to ADC A and logic one selects the high range.
3	R/W	0x0	GPIO Enable	Logic zero closes a FET switch between the GPIO connector and all GPIO bits (0 through 7). Logic one effectively tristates the GPIO connector pins.
2:1		0x0	Reserved	
0	R/W	0x0	Amplifier Power Enable	Logic one applies power to the analog amplifiers.

Address	**Register Name**
0x200	Model 321 Configuration Settings

Product Relevance (checked box)
☐ Model 314 ☐ Model 316 ☐ Model 320 ☒ Model 321

Bit(s)	**R/W/cl**	**POR**	**Function**	**Description**
31:6			Reserved	
5	R/W	0x0	ADC B Voltage Range	Logic zero selects the low input voltage range to ADC B and logic one selects the high range.
4	R/W	0x0	ADC A Voltage Range	Logic zero selects the low input voltage range to ADC A and logic one selects the high range.
3	R/W	0x0	ADC B Randomizer	Logic zero results in normal operation. Logic one causes ADC B output bits D1-D15 to be EXCLUSIVE-ORed with bit D0.
2	R/W	0x0	ADC A Randomizer	Logic zero results in normal operation. Logic one causes ADC A output bits D1-D15 to be EXCLUSIVE-ORed with bit D0.
1	R/W	0x0	ADC B Dither	Logic zero disables internal dither to ADC B, logic one enables internal dither.
0	R/W	0x0	ADC A Dither	Logic zero disables internal dither to ADC A, logic one enables internal dither.

Address	**Register Name**
0x204	LED Control

Product Relevance (checked box)
☒ Model 314 ☐ Model 316 ☒ Model 320 ☐ Model 321

Bit(s)	**R/W/cl**	**POR**	**Function**	**Description**
31:2	R	0x0	Reserved	
1	R/W	0x0	LED B (Green)	Model 320 - Logic zero illuminates the green LED (D4).
0	R/W	0x0	LED A (Yellow)	Model 314 - Logic zero illuminates the yellow LED (D5). Model 320 - Logic zero illuminates the yellow LED (D3).

Address	**Register Name**
0x208	GPIO

Product Relevance (checked box)
☒ Model 314 ☐ Model 316 ☐ Model 320 ☐ Model 321

Bit(s)	**R/W/cl**	**POR**	**Function**	**Description**
31:5		0x0	Reserved	
4	R/W	0x1	GPIO Enable	Logic zero closes a FET switch between the GPIO connector and all GPIO bits (0 through 5). Logic one effectively tristates the GPIO connector pins.
3	R/W	0x1	GPIO 5	GPIO output #5 (cannot be used as an input).
2	R/W	0x1	GPIO 4	GPIO output #4 (cannot be used as an input).
1:0		0x0	Reserved	

Address	**Register Name**
0x208	Channel A Attenuator Setting

Product Relevance (checked box)
☐ Model 314 ☐ Model 316 ☒ Model 320 ☐ Model 321

Bit(s)	**R/W/cl**	**POR**	**Function**	**Description**
31:5		0x0	Reserved	
4:0	R/W	0x1F	Channel A Attenuator Setting	Five bit value indicating the level of attenuation on the Channel A input. Values 0x0 to 0x1F correspond to the range of 0 dB to 31 dB of attenuation.

Address	**Register Name**
0x210	FPGA Attenuator Interface Enable

Product Relevance (checked box)
☐ Model 314 ☐ Model 316 ☒ Model 320 ☐ Model 321

Bit(s)	**R/W/cl**	**POR**	**Function**	**Description**
31:1		0x0	Reserved	
0	R/W	0x0	Interface Enable	Logic one allows the FPGA to load the attenuator setting registers on a Model 320 through a dedicated hardware interface to the PCI Controller. This bit must remain logic zero on all other models.

Address	Register Name
0x214	Synthesizer Serial Programming Interface

Product Relevance (checked box)
☒ Model 314 ☒ Model 316 ☒ Model 320 ☒ Model 321

Bit(s)	R/W/cl	POR	Function	Description
31:4		0x0	Reserved	
3	R/W	0x0	Interface Enable	Logic one activates the programming interface.
2	R/W	0x0	Serial Data Enable	Set to logic TBD to initiate serial programming.
1	R/W	0x0	Serial Data Clock	Synthetic clock generated by writing alternating logic zero and one values to toggle the signal.
0	R	0x0	Serial Data	Serial configuration data to the synthesizer.

Address	**Register Name**
0x20C	Channel B Attenuator Setting

Product Relevance (checked box)
☐ Model 314 ☐ Model 316 ☒ Model 320 ☐ Model 321

Bit(s)	**R/W/cl**	**POR**	**Function**	**Description**
31:5		0x0	Reserved	
4:0	R/W	0x1F	Channel B Attenuator Setting	Five bit value indicating the level of attenuation on the Channel B input. Values 0x0 to 0x1F correspond to the range of 0 dB to 31 dB of attenuation.

5.2 BAR2 DMA Memory Map

The DMA registers define the size and destination addresses for DMA transfers from the Channel Adapter to host memory. These registers are typically initialized by software at that start of an application and simply report DMA status during normal operation. All Channel Adapter products share a common BAR2 memory map.

BAR2 DMA Memory Map		
Byte Address Offset	**R/W/cl**	**Register (Group) Name**
0x000	R	DMA Core Revision
0x004		Reserved for Future Expansion
0x008	R/W	DMA Interrupt Mask
0x00C		Reserved for Future Expansion
0x010	R/W	DMA Interrupt Status
0x014		Reserved for Future Expansion
0x018-0x02C		Reserved for Future Expansion
0x030	R/W	Group Address Page Index
0x034-0x07c		Reserved for Future Expansion
0x080-0x84	R/W	RX DMA Block Size
0x088-0x0FC		Reserved for Future Expansion
0x100-0x104	R/W	RX DMA Block Count
0x108-0x17C		Reserved for Future Expansion
0x180-0x184	R/W	RX DMA Group Count
0x188-0x1FC		Reserved for Future Expansion
0x200-0x204	R/W	RX DMA Current Block
0x208-0x27C		Reserved for Future Expansion
0x280-0x284	R/W	RX DMA Current Group
0x288-0x2FC		Reserved for Future Expansion
0x300-0x304	R/W	RX DMA Interrupt Group Count
0x308-0x37C		Reserved for Future Expansion
0x400-0x404	R	RX DMA Current Address
0x408-0x4FC		Reserved for Future Expansion
0x500-0x5FC	R/W	RX DMA Group Starting Addresses
0x600-0x7FC		Reserved

Address	Register Name
0x000	DMA Core Revision

Bit(s)	R/W/cl	POR	Function	Description
31:0	R	Rev	DMA Core Rev Code	Hard-coded constant that indicates the revision level of the DMA Interface core. Format: MM/DD/YYYY (i.e. 07/01/2006)

Address	Register Name
0x008	DMA Interrupt Mask

Bit(s)	R/W/cl	POR	Function	Description
31:2			Reserved	
1	R/W	0x0	Channel B Group Done	A logic zero disables the interrupts that are generated when a DMA operation completes on Channel B.
0	R/W	0x0	Channel A Group Done	A logic zero disables the interrupts that are generated when a DMA operation completes on Channel A.

Address	Register Name
0x010	DMA Interrupt Status

Bit(s)	R/W/cl	POR	Function	Description
31:2			Reserved	
1	R/cl	0x0	Channel B DMA Complete	Logic one indicates that a Channel B DMA operation has completed. A DMA operation consists of transferring the number of groups specified in the RX DMA Group Count register.
0	R/cl	0x0	Channel A DMA Complete	Logic one indicates that a Channel A DMA operation has completed. A DMA operation consists of transferring the number of groups specified in the RX DMA Group Count register.

Address	Register Name
0x030	Group Address Page Index

Bit(s)	R/W/cl	POR	Function	Description
31:5			Reserved	
4:0	R/W	0x0	Page Index	There are 32 pages assigned to the storage of physical addresses associated with DMA buffers in host memory. Each page can hold 64 addresses as defined in the RX DMA Group Starting Address registers. This value specifies the index into each of the 32 pages for indirect addressing. Pages 0-15 correspond to Channel A and pages 16-31 correspond to Channel B.

Start Address	End Address	Register Group Name
0x080	0x084	RX DMA Block Size

Address	Register Name
0x080	Channel A DMA Block Size
0x084	Channel B DMA Block Size

Bit(s)	R/W/cl	POR	Function	Description
31:19			Reserved	
18:0	R/W	0x0	Block Size	Indicates the size of the DMA block in bytes.

Start Address	End Address	Register Group Name
0x100	0x104	RX DMA Block Count

Address	Register Name
0x100	Channel A DMA Block Count
0x104	Channel B DMA Block Count

Bit(s)	R/W/cl	POR	Function	Description
31:16			Reserved	
15:0	R/W	0x0	Block Count	Indicates how many bus master DMA blocks are in a group. The number written is one less than the number of blocks in a DMA group (i.e. 0=1 block/group, 1=2 blocks/group, 2=3 blocks/group, etc.).

Start Address	End Address	Register Group Name
0x180	0x184	RX DMA Group Count

Address	Register Name
0x180	Channel A DMA Group Count
0x184	Channel B DMA Group Count

Bit(s)	R/W/cl	POR	Function	Description
31:11			Reserved	
10:0	R/W	0x0	Group Count	Indicates how many groups will transfer before returning to the original starting address. Valid settings range from 0 to 511. The number written is one less than the total number of groups (i.e. 0=1 group, 1=2 groups, etc.).

Start Address	End Address	Register Group Name
0x200	0x204	RX DMA Current Block

Address	Register Name
0x200	Channel A DMA Current Block
0x204	Channel B DMA Current Block

Bit(s)	R/W/cl	POR	Function	Description
31:16			Reserved	
15:0	R	0x0	Current Block	Indicates which DMA block in the current group is being transferred.

Start Address	End Address	Register Group Name
0x280	0x284	RX DMA Current Group

Address	Register Name
0x280	Channel A DMA Current Group
0x284	Channel B DMA Current Group

Bit(s)	R/W/cl	POR	Function	Description
31:11			Reserved	
10:0	R	0x0	Current Group	Indicates the current group being transferred. Valid read values range from 0 to 511.

Start Address	End Address	Register Group Name
0x300	0x304	RX DMA Interrupt Group Count

Address	Register Name
0x300	Channel A DMA Interrupt Group Count
0x304	Channel B DMA Interrupt Group Count

Bit(s)	R/W/cl	POR	Function	Description
31:10			reserved	
9:0	R/W	0x0	Interrupt Count	The value written to this register indicates the number of DMA groups that must complete before an interrupt is generated.

Start Address	End Address	Register Group Name
0x400	0x404	RX DMA Current Address

Address	Register Name
0x400	Channel A DMA Current Address
0x404	Channel B DMA Current Address

Bit(s)	R/W/cl	POR	Function	Description
31:0	R	0x0	Current Address	Indicates the starting address of the current DMA. After a reset, the value read is set by the first write to the group #0 Starting Address for each channel.

Start Address	End Address	Register Group Name
0x500	0x57C	RX DMA Group Starting Address

Address	Register Name
0x500	RX Group #0 Starting Address
0x504	RX Group #1 Starting Address
0x508	RX Group #2 Starting Address
0x50C	RX Group #3 Starting Address
0x510	RX Group #4 Starting Address
0x514	RX Group #5 Starting Address
0x518	RX Group #6 Starting Address
0x51C	RX Group #7 Starting Address
0x520	RX Group #8 Starting Address
0x524	RX Group #9 Starting Address
0x528	RX Group #10 Starting Address
0x52C	RX Group #11 Starting Address
0x530	RX Group #12 Starting Address
0x534	RX Group #13 Starting Address
0x538	RX Group #14 Starting Address
0x53C	RX Group #15 Starting Address
0x540	RX Group #16 Starting Address
0x544	RX Group #17 Starting Address
0x548	RX Group #18 Starting Address
0x54C	RX Group #19 Starting Address
0x550	RX Group #20 Starting Address
0x554	RX Group #21 Starting Address
0x558	RX Group #22 Starting Address
0x55C	RX Group #23 Starting Address
0x560	RX Group #24 Starting Address
0x564	RX Group #25 Starting Address
0x568	RX Group #26 Starting Address
0x56C	RX Group #27 Starting Address
0x570	RX Group #28 Starting Address
0x574	RX Group #29 Starting Address
0x578	RX Group #30 Starting Address
0x57C	RX Group #31 Starting Address

Bit(s)	R/W/cl	POR	Function	Description
31:0	R/W	0x0	Address	Specifies the 64 group addresses assigned to the page selected in the Group Address Page Index register. Each group starting address corresponds to a physical address in host memory that represents the location of a DMA buffer. Indirect addressing is used to store a total of 2048 addresses in the 32 available pages (32 * 64 = 2048).

Start Address	End Address	Register Group Name
0x580	0x5FC	RX DMA Group Starting Address

Address	Register Name
0x580	RX Group #32 Starting Address
0x584	RX Group #33 Starting Address
0x588	RX Group #34 Starting Address
0x58C	RX Group #35 Starting Address
0x590	RX Group #36 Starting Address
0x594	RX Group #37 Starting Address
0x598	RX Group #38 Starting Address
0x59C	RX Group #39 Starting Address
0x5A0	RX Group #40 Starting Address
0x5A4	RX Group #41 Starting Address
0x5A8	RX Group #42 Starting Address
0x5AC	RX Group #43 Starting Address
0x5B0	RX Group #44 Starting Address
0x5B4	RX Group #45 Starting Address
0x5B8	RX Group #46 Starting Address
0x5BC	RX Group #47 Starting Address
0x5C0	RX Group #48 Starting Address
0x5C4	RX Group #49 Starting Address
0x5C8	RX Group #50 Starting Address
0x5CC	RX Group #51 Starting Address
0x5D0	RX Group #52 Starting Address
0x5D4	RX Group #53 Starting Address
0x5D8	RX Group #54 Starting Address
0x5DC	RX Group #55 Starting Address
0x5E0	RX Group #56 Starting Address
0x5E4	RX Group #57 Starting Address
0x5E8	RX Group #58 Starting Address
0x5EC	RX Group #59 Starting Address
0x5F0	RX Group #60 Starting Address
0x5F4	RX Group #61 Starting Address
0x5F8	RX Group #62 Starting Address
0x5FC	RX Group #63 Starting Address

Bit(s)	R/W/cl	POR	Function	Description
31:0	R/W	0x0	Address	Specifies the 64 group addresses assigned to the page selected in the Group Address Page Index register. Each group starting address corresponds to a physical address in host memory that represents the location of a DMA buffer. Indirect addressing is used to store a total of 2048 addresses in the 32 available pages (32 * 64 = 2048).