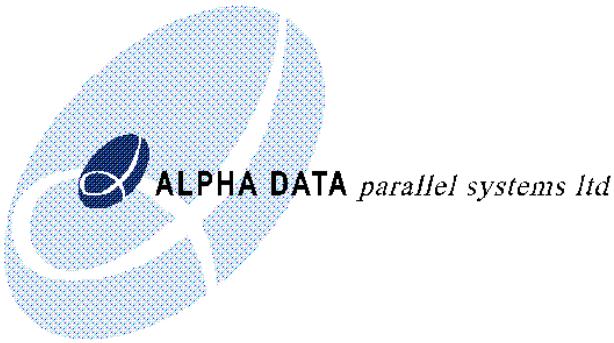
Alpha Data XRC Matlab Toolbox v1.6



Alpha Data Parallel Systems Ltd.
58 Timber Bush
Edinburgh EH6 6QH
Scotland
UK

Phone: +44 (0) 131 555 0303
Fax: +44 (0) 131 555 0728
Email: support@alphadata.co.uk

1 Introduction

The Matlab Toolbox allows many aspects of the ADM-XRC cards to be controlled easily from within the Matlab environment. Functions are provided to allow configuration and data transfer between the Matlab environment and an application running on an FPGA. These allow the powerful mathematical and graphical tools of Matlab to be used to implement a friendly user interface to an FPGA application.
Supported ADM-XRC cards:
ADM-XRC, ADM-XRC-P, ADM-XRC-II-L, ADM-XRC-II, ADP-WRC-II, ADM-XRC-IIPro-L(XPL), ADM-XRC-IIPro(XP)

2 Installation Instructions

The XRC Matlab Toolbox is provided as a zip file and a Matlab installation script:

xrc_matlab.zip
setup_xrc_matlab.m

1) Open Matlab

2) Change Directory to where xrc_matlab.zip and setup_xrc_matlab.m have been downloaded.

3) Type setup_xrc_matlab

4) Quit Matlab

3 ADMXRC Matlab Toolbox Tutorial

3.1 Basic functions

To perform any action using an ADMXRC card, you must first obtain a handle for it. This gives your program exclusive control over the card. The function `admxrc_init` is used to obtain a handle:

Type
```
>> handle=admxrc_init
```
to obtain the handle of the first card installed in your computer..

You can also use
```
>> handle=admxrc_init(id)
```
to open the card with the specified id
or
```
>> handle=admxrc_init([],n)
```
to open the nth card installed in the machine.

N.B. Handle is of type int32.

Having obtained a handle, you can now perform other operations on the card, such as obtaining information about the type of card you are accessing:
Type
```
>> [id,sn,bt,ft]=admxrc_getinfo(handle);
```

This will return the id of the card, the serial number (sn), the board type (bt) and the FPGA type (ft) .

Other information about the card is also returned by admxrc_getinfo if you supply more parameters on the left hand side.
Type
```
>> help admxrc_getinfo
```
to get the full list.

The functions `admxrc_getspaceinfo` and `admxrc_getbankinfo` provide further information about the memory mapping of the FPGA and the SRAM banks on the card.

After using a card, you should call the function `admxrc_close` to release the card and allow other processes to access it.
Type
```
>> admxrc_close(handle)
```

If you do not close the card, the next call to `admxrc_init` will fail.

3.2 Simple Example

In the sub directory examples, there are a number of demonstration programs. These use the FPGA bitstreams provided in the ADMXRC SDK for Windows. The Environment Variable ADMXRC_SDK4 must be set for the example programs to find the correct bit streams.

The first example is simple.m

To run this example, type:

```
>> cd c:\matlab\toolbox\admxrc\examples
>> simple
```

This will load a bitstream onto the FPGA. You will be invited to type in hexadecimal numbers up to 8 digits in length. These are then sent to the FPGA, which nibble reverses them. And then the values are read back. Enter 55AA to exit.

Type
>> edit simple
To look at the source code for simple.m
As you can see, the code obtains the handle for a card.

The function getFPGAfilename is used to get the correct filename for the bitstream file in the ADMXRC SDK. This function reads the environment variable ADMXRC_SDK4 to find the root directory. Since the SDK contains bit files for all boards and FPGA combinations, the function also converts the string 'simple' into the format 'simple-board-FPGA.bit'. e.g. simple-xrc-v1000e.bit

Having got the correct bit file for the card being used, the FPGA is configured using admxrc_config(handle,filename).

Two other functions are used in the simple.m example.

admxrc_write32 - writes data of type int32 to the card.
The elements of d are written to successive longword addresses starting at offset.
In this case only one element is written.

admxrc_read32 - reads data of type int32 to the card
The third parameter specifies the number of int32 elements to be read from the card.
In this case it is again 1.

3.3 Memory Test Example

The example memory test uses the zbt bitstream file used in the SDK memtest example. The main purpose of this example is to measure the data transfer speed between Matlab and the FPGA. 4 tests are performed. Large data transfers are initially performed using programmed I/O (the functions admxrc_write32 and admxrc_read32). The same data transfers are then performed using slave mode DMA. The functions admxrc_write32_dma and admxrc_read32_dma have the same

parameters as the programmed I/O functions, but use the PLX chip on the ADMXRC to perform a DMA data transfer.

Type
>> memtest
to run the rate test example.

3.4 I/O Examples

The example scripts frontio.m and reario.m run the frontio and reario test examples from the SDK. More details on these can be found the in SDK documentation.

4 ADMXRC Matlab Toolbox Reference

4.1 Basic Functions

4.1.1 admxrc_close

Closes an ADM-XRC card (*handle*):

USAGE:
```
>> admxrc_close(handle)
```

4.1.2 admxrc_config

Configures the FPGA on the ADM-XRC card (*handle*) with bitstream stored in *filename*

USAGE:
```
>> admxrc_config(handle,filename)
```

4.1.3 admxrc_getbankinfo

Returns information about a memory bank (*bank_no*) on the ADM-XRC card (*handle*)

USAGE:
```
>> [type,width,size,fitted] =
admxrc_getbankinfo(card,bank_no);
```

type - indicates if bank can operate in flowthrough and pipelined modes
width - bit width of the bank
size - no of words in the bank
fitted - boolean description of whether memory bank is fitted.

4.1.4 admxrc_getinfo

Returns information about an ADM-XRC card (*handle*)

USAGE:
```
>> [CardID,SerialNumber, BoardType,FPGAType,
    NumClock,NumDMAChan,NumRAMBank,NumSpace,
    RAMBanksFitted,BoardRevision,LogicRevision]
    =admxrc_getinfo(handle)
```

4.1.5 admxrc_getspaceinfo

Returns information about a memory space (*space_no*) on the ADM-XRC card (*handle*)

USAGE:

```
>> [virtual_base,virtual_size,physical_base,
    local_base,local_size] =
    admxrc_getspaceinfo(card,space_no)
```

space_no indicates which PCI space:
0 - the FPGA base address
1 - the PLX base address

4.1.6 admxrc_init

Returns a handle to an ADM-XRC card.

USAGE:

```
>> handle=admxrc_init;
>> handle=admxrc_init(id);
>> handle=admxrc_init([],index);
```

4.1.7 admxrc_setclockrate

Sets clock *clock_index* on card *handle* to frequency *frequency* specified in MHz.

USAGE:

```
>> actual_frequency =
admxrc_setclockrate(handle,clock_index,frequency);
```

actual_frequency is returned in Hz.

4.2 Data Transfer

4.2.1 admxrc_read32, admxrc_read16, admxrc_read8

Reads *length* 32,16 or 8 bit values from ADM-XRC card *handle* into int32 array *data* starting at FPGA local address *offset*. If flag is specified as 1, *data* will be of type uint32.

USAGE:

```
>> data = admxrc_read32(handle,offset,length)
>> data = admxrc_read32(handle,offset,length,flag)
```

4.2.2 admxrc_read32_dma

Reads *length* 32,16 or 8 bit values from ADM-XRC card *handle* into array *data* starting at FPGA local address *offset* using slave mode DMA.

USAGE:

```
>> data = admxrc_read32_dma(handle,offset,length)
>> data = admxrc_read32_dma(handle,offset,length,
dmachannel)
>> data = admxrc_read32_dma(handle,offset,length,
dmachannel,element_size)
```

Optional parameter *dmachannel* can be set to 0 or 1, to select the appropriate DMA channel on the ADM-XRC. Optional parameter *element_size* can be set to 1,2 or 4 to select the type of data returned: INT8, INT16 or INT32, -1,-2 or -4 to select the type of data returned as UINT8, UINT16 or UINT32, or 8 to return an array of doubles. By default, *admxrc_read32_dma* returns *data* of type INT32.

4.2.3 admxrc_write32, admxrc_write16, admxrc_write8

Writes elements of array *data* (of 32, 16 or 8 bit type) to ADM-XRC card *handle* at FPGA local address *offset*.

USAGE:

```
>> admxrc_write32(handle,offset,data)
```

4.2.4 admxrc_write32_dma

Writes elements of array *data* to ADM-XRC card *handle* at FPGA local address *offset* using slave mode DMA.

USAGE:

```
>> admxrc_write32_dma(handle,offset,data)
>> admxrc_write32_dma(handle,offset,data,dmachannel)
```