

Perseus

Programmer's Reference Guide

May 2016

Revision history

Revision	Date	Comments
1.0	October 2015	First edition.
1.1	May 2016	Removed functional examples section

© Nutaq All rights reserved.

No part of this document may be reproduced or used in any form or by any means—graphical, electronic, or mechanical (which includes photocopying, recording, taping, and information storage/retrieval systems)—without the express written permission of Nutaq.

To ensure the accuracy of the information contained herein, particular attention was given to usage in preparing this document. It corresponds to the product version manufactured prior to the date appearing on the title page. There may be differences between the document and the product, if the product was modified after the production of the document.

Nutaq reserves itself the right to make changes and improvements to the product described in this document at any time and without notice.

Trademarks

Acrobat, Adobe, and Reader are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries. IBM is a registered trademark of International Business Machines Corporation in the United States, other countries, or both. Intel and Pentium are registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries. Microsoft, MS-DOS, Windows, Windows NT, and the Windows logo are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. MATLAB, Simulink, and Real-Time Workshop are registered trademarks of The MathWorks, Inc. Xilinx, Spartan, and Virtex are registered trademarks of Xilinx, Inc. Texas Instruments, Code Composer Studio, C62x, C64x, and C67x are trademarks of Texas Instruments Incorporated. All other product names are trademarks or registered trademarks of their respective holders.

The ™ and ® marks have been omitted from the text.

WARNING

Do not use Nutaq products in conjunction with life-monitoring or life-critical equipment. Failure to observe this warning relieves Nutaq of any and all responsibility.

FCC WARNING

This equipment is intended for use in a controlled environment only. It generates, uses, and can radiate radio frequency energy and has not been tested for compliance with the limits of personal computers and peripherals pursuant to subpart J of part 15 of the FCC rules. These rules are designed to provide reasonable protection against radio frequency interference. Operating this equipment in other environments may cause interference with radio communications, in which case the user must, at his/her expense, take whatever measures are required to correct this interference.

This page was left intentionally blank.

Table of Contents

1	Introduction	1
1.1	Conventions	1
1.2	Glossary	1
1.3	Technical Support	2
2	Product Description	3
2.1.1	Perseus601X Outstanding Features.....	3
2.1.2	Perseus611X Outstanding Features.....	3
3	Perseus Functional Setup.....	4
3.1	Perseus CCE Communication	4
3.1.1	Default UART Parameters.....	4
3.1.2	Configuring the UART on Windows 7	4
3.1.3	Configuring the UART on Linux.....	5
3.2	Network Configurations.....	6
3.2.1	Configuring a Windows 7 Network.....	6
3.2.2	Configuring a Linux Network	7
3.3	Programming an FPGA Bitstream into Flash Memory	10
4	FPGA Development	12
4.1	Reference Design Block Diagram	12
4.2	Perseus FPGA Address Map	13
4.2.1	User Register Space	13
4.2.2	System Peripheral Space	13
4.3	Module List	14
4.3.1	MicroBlaze	14
4.3.2	Processor Boot Memory	14
4.3.3	DDR3 Processor Memory.....	14
4.3.4	AXI Lite Bus	14
4.3.5	Real-Time Timer.....	14
4.3.6	UART Controller	14
4.3.7	Watchdog Timer	15
4.3.8	Processor Debugging Module (mdm)	15
4.3.9	FMC I2C.....	15
4.3.10	Flash Memory Controller.....	15
4.3.11	Interrupt Controller	16
4.3.12	System Monitor	16
4.3.13	Platform Registers	16
4.3.14	Ethernet Controller and DMA.....	16
4.3.15	FMC Board	16
4.3.16	RTDEx Cores.....	17
4.3.17	Record Playback and the DDR3 SODIMM	17
4.3.18	Other Nutaq Cores.....	18
5	Perseus601X Platform Register Core Description.....	19
5.1	Core Level Features	19

5.2	Core Ports	19
5.3	Configuration Registers.....	21
5.3.1	Registers Map	21
5.3.2	Registers Description	21
6	Perseus 611X Platform Register Core Description.....	26
6.1	Core Level Features.....	26
6.2	Core Ports	26
6.3	Configuration Registers.....	29
6.3.1	Registers Map	29
6.3.2	Registers Description	29
7	Low-Level and Host Programming.....	34
7.1	Perseus601X Driver	34
7.2	Perseus611X Driver	35
7.3	Carrier Driver	36
7.4	Carrier EAPI Library	38
7.5	Carrier Command Line Interface (CLI) Commands.....	38

List of Figures and Tables

Figure 3-1 Configuration menu.....	5
Figure 3-2 UART parameters.....	5
Figure 3-3 Determining the MAC address	7
Figure 3-4 IP address of the host Ethernet card	8
Figure 3-5 Invalid jumbo frame size.....	8
Figure 3-6 Valid jumbo frame size	9
Figure 3-7 Installing ethtool.....	9
Figure 3-8 ethtool	10
Figure 3-9 Flow control.....	10
Figure 3-10 FPGA Flash Information	10
Figure 4-1 Perseus FPGA reference design block diagram	12
Figure 4-2 Perseus MicroBlaze address map	13
Table 1 Glossary.....	2
Table 2 FMC address map.....	17
Table 3 Perseus 601X Platform Register core port list.....	21
Table 4 Perseus 601X Registers map	21
Table 5 Perseus 601X INFO register	22
Table 6 Perseus 601X INFO descriptions.....	22
Table 7 Perseus 601X MODULE_CTRL_STATUS register	22
Table 8 Perseus 601X MODULE_CTRL_STATUS descriptions.....	23
Table 9 Perseus 601X CLK_CTRL register	23
Table 10 Perseus 601X CLK_CTRL descriptions.....	24
Table 11 Perseus 601X VADJ_RESET_CTRL registers	24
Table 12 Perseus 601X VADJ_RESET_CTRL descriptions.....	24
Table 13 Perseus 601X LED_CTRL register	24
Table 14 Perseus 601X LED_CTRL descriptions.....	24
Table 15 Perseus 601X MMC_CTRL register	25
Table 16 Perseus 601X MMC_CTRL descriptions.....	25
Table 17 Perseus 601X CUSTOM_REGISTER register	25
Table 18 Perseus 611X Platform Register core port list.....	28
Table 19 Perseus 611X Registers map	29
Table 20 Perseus 611X INFO register	29
Table 21 Perseus 611X INFO descriptions.....	29
Table 22 Perseus 611X MODULE_CTRL_STATUS register	30
Table 23 Perseus 611X MODULE_CTRL_STATUS descriptions.....	30
Table 24 Perseus 611X CLK_CTRL register	31
Table 25 Perseus 611X CLK_CTRL descriptions.....	32
Table 26 Perseus 611X VADJ_RESET_CTRL registers	32
Table 27 Perseus 611X VADJ_RESET_CTRL descriptions.....	32
Table 28 Perseus 611X LED_CTRL register	32
Table 29 Perseus 611X LED_CTRL descriptions.....	32
Table 30 Perseus 611X I2C_CTRL register	33
Table 31 Perseus 611X I2C_CTRL descriptions.....	33
Table 32 Perseus 611X CUSTOM_REGISTER register	33
Table 33 Perseus 601X driver functions.....	35

Table 34 Perseus 611X driver functions.....	36
Table 35 Carrier driver functions	38
Table 36 Carrier EAPI functions	38
Table 37 Carrier CLI function.....	38

1 Introduction

Congratulations on the purchase of a Perseus carrier!

This document contains all the information necessary to understand the software available with the Perseus carriers of both versions, the Perseus601x or the Perseus611x. It should be read carefully before using the card and stored in a handy location for future reference.

Unless specified by the use of the specific names Perseus601x and Perseus611x, the use of the name Perseus refers to both versions of the product.

1.1 Conventions

In a procedure containing several steps, the operations are numbered (1, 2, 3...). The diamond (♦) is used to indicate a procedure containing only one step, or secondary steps. Lowercase letters (a, b, c...) can also be used to indicate secondary steps in a complex procedure.

The abbreviation NC is used to indicate no connection.

Capitals are used to identify any term marked as is on an instrument, such as the names of connectors, buttons, indicator lights, etc. Capitals are also used to identify key names of the computer keyboard.

All terms used in software, such as the names of menus, commands, dialog boxes, text boxes, and options, are presented in bold font style.

The abbreviation N/A is used to indicate something that is not applicable or not available at the time of press.

Note:

The screen captures in this document are taken from the software version available at the time of press. For this reason, they may differ slightly from what appears on your screen, depending on the software version that you are using. Furthermore, the screen captures may differ from what appears on your screen if you use different appearance settings.

1.2 Glossary

This section presents a list of terms used throughout this document and their definition.

Term	Definition
Application programming interface (API)	An application programming interface is the interface that a computer system, library, or application provides to allow requests for services to be made of it by other computer programs or to allow data to be exchanged between them.
Base design	Empty design or template that is incapable of data processing and is not instantiated in the custom logic of the board's FPGA.
Board software development kit	Abbreviated BSDK, this kit gives users the possibility to quickly become fully functional developing C/C++ or assembly code for the DSP and HDL code for the FPGA through an understanding of all Nutaq boards major interfaces.
Chassis	Refers to the rigid framework onto which the CPU board, Nutaq development platforms, and other equipment are mounted. It also supports the shell-like case—the housing that protects all the vital internal equipment from dust, moisture, and tampering.

Term	Definition
Computer communication development	Refers to developing custom communication applications to communicate with Nutaq boards.
Default design	Design loaded by default on Nutaq boards used for FPGA design.
Digital signal processing	Digital signal processing is the study of signals in a digital representation and the processing methods of these signals. The algorithms required for DSP are sometimes performed using specialized devices that use specialized microprocessors called digital signal processors (DSP).
Digital signal processor (DSP)	A digital signal processor is a specialized microprocessor designed specifically for digital signal processing, generally in real time.
Example	Refers to examples used to demonstrate functions or applications supplied with the board software development kit. For this reason, examples come in two flavors: application examples and functional examples.
HDL	Stands for hardware description language.
Host	A host is defined as the device that configures and controls a Nutaq board. The host may be a standard computer or the CPU board of the cPCI chassis system where the Nutaq board is installed. You can develop applications on the host for Nutaq boards through the use of an application programming interface (API) that comprises protocols and functions necessary to build software applications. These API are supplied with the Nutaq board.
Model-based design	Refers to all the Nutaq board-specific tools and software used for development with the boards in MATLAB and Simulink and the Nutaq model-based design kits.
Reception	Any data received by the referent is a reception. Abbreviated RX.
Reference design	Blueprint of an FPGA system implanted on Nutaq boards. It is intended for others to copy and contains the essential elements of a working system (in other words, it is capable of data processing), but third parties may enhance or modify the design as necessary.
RTDEx	Refers to real-time data exchanges.
Software development	Refers to development performed with and for the board with a software development kit. Software development for a board comes in three flavors: host software development, DSP software development, and FPGA software development.
Transmission	Any data transmitted by the referent is a transmission. Abbreviated TX.
VHDL	Stands for VHSIC hardware description language.
VHSIC	Stands for very-high-speed integrated circuit.

Table 1 Glossary

1.3 Technical Support

Nutaq is firmly committed to providing the highest level of customer service and product support. If you experience any difficulties using our products or if it fails to operate as described, first refer to the documentation accompanying the product. If you find yourself still in need of assistance, visit the technical support page in the Support section of our Web site at www.nutaq.com.

2 Product Description

The Perseus advanced mezzanine card (AMC or AdvancedMC) are designed around the powerful Virtex-6 FPGA, combining unsurpassed fabric flexibility and a colossal external memory, as well as benefiting from multiple high-pin-count, modular, add-on FMC-based I/O cards.

The Perseus are intended for high-performance, high-bandwidth, low-latency processing applications. The cards also take full advantage of the Virtex-6 FPGA power, which, when combined with Nutaq's advanced software development tools, makes the Perseus perfect for reducing size, complexity, risks and costs associated to leading-edge telecommunications, networking, industrial, defense and medical applications. On top this, the Perseus' FMC expansion site offers almost endless I/O possibilities.

2.1.1 Perseus601X Outstanding Features

- Mid-size AMC for μ TCA and AdvancedTCA platforms
- Choice of powerful LXT and SXT Virtex-6 FPGAs
- One high-pin-count VITA 57.1 FMC expansion site for I/Os
- DDR3 SODIMM interface to upgrade system memory
- Support for AMC R2.0 and R1.0 through onboard clock switch
- Available GTX base clocks — 100 MHz, 125 MHz, 156.25 MHz (PCIe/GigE/XAUI/SRIO)
- Fabric clock — RX or TX (100 MHz PCIe, default)
- IPMI controller (based on the AVR version of the Pigeon Point AdvancedMC MMC)
- FPGA and IPMI JTAGs on the Mestor interface



2.1.2 Perseus611X Outstanding Features

- Double-width mid-size AMC card
- Choice of powerful LXT and SXT Virtex-6 FPGAs
- Two high-pin-count VITA 57.1 FMC expansion site for I/Os
- DDR3 SODIMM interface to upgrade system memory
- Support for AMC R2.0 and R1.0 through onboard clock switch
- Available GTX base clocks — 100 MHz, 125 MHz, 156.25 MHz (PCIe/GigE/XAUI/SRIO)
- Fabric clock — RX or TX (100 MHz PCIe, default)
- IPMI controller (based on the AVR version of the Pigeon Point AdvancedMC MMC)
- FPGA and IPMI JTAGs on the Mestor interface

3 Perseus Functional Setup

3.1 Perseus CCE Communication

Under the default Perseus FPGA configuration, there is a software processor called the MicroBlaze that runs the central communication engine (CCE) under runs a Linux distribution (PetaLinux). The CCE handles all the behavior of the Perseus—module initialization, clock management, as well as other tasks. You can access the Linux distribution through a console to configure your system. It allows you to copy files, change initialization files, and configure Ethernet addresses, for example. To access all this, you must connect your host device to the Linux console as described below.

3.1.1 Default UART Parameters

- Baud rate: 115200 bps
- Data bits: 8
- Parity: none
- Stop bits: 1
- Flow control: none

3.1.2 Configuring the UART on Windows 7

To configure the UART on Windows 7:

1. On a standalone Perseus in a uTCA chassis, connect a mini-USB cable between your computer and the Mestor adapter.
On a PicoSDR or a PicoDigitizer, connect a USB a mini-USB cable between your computer and the PicoSDR/Digitizer backplane.
2. Open **Control Panel**.
3. Click **System**.
4. In the **System Properties** dialog box, on the **Hardware** tab, click **Device Manager**.
5. In the **Device Manager** window, expand **Ports (COM & LPT)**.
6. Verify that your device is correctly installed by finding a device corresponding to **Silicon Labs CP210x USB to UART Bridge (COM#)**.
The # is replaced by the COM port number. If your card is not automatically detected, install the Silicon Labs CP210x USB to UART Bridge driver. You can download it from www.silabs.com.
7. Start a serial console program.
8. Create a new connection with the default parameters defined above and connect to the Perseus.
9. To verify the communication link, program the Perseus FPGA with a valid bitstream designed for PetaLinux.
To program the FPGA, see FPGA development with VHDL. Once the FPGA is programmed, messages appear in the console window.
10. In the console window, type root as the perseustst login and root as the password.
You are now logged onto your Perseus Linux. Feel free to browse your file system.

3.1.3 Configuring the UART on Linux

Identifying a UART device

Under Fedora, the USB UART identifies itself as a ttyUSB device. In a console, type `ls /dev/ttyUSB*` to list all the available devices. Normally, only the Perseus should appear in the list, but if there are more devices, read your kernel (dmesg) log to identify the device(s) added after connecting the Perseus USB cable.

Installing a terminal

To communicate with the Perseus you need to configure a terminal. For this Nutaq, uses gtkterm, a simple terminal emulator. To install it, proceed as follows in the root of your Fedora system.

11. To install the software, type **yum install gtkterm**.
12. To run the emulator, type **gtkterm**.

Configuring gtkterm

To configure the UART on Linux for gtkterm:

13. On the **Configuration** menu, click **Port**.

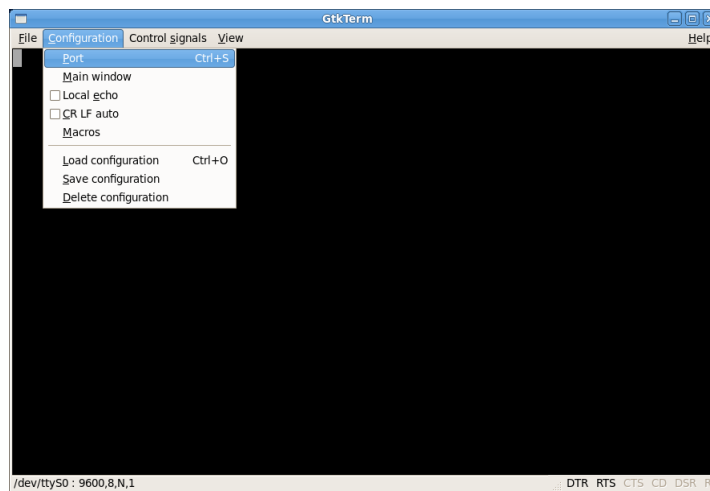


Figure 3-1 Configuration menu

14. Enter the default UART parameters, as illustrated.

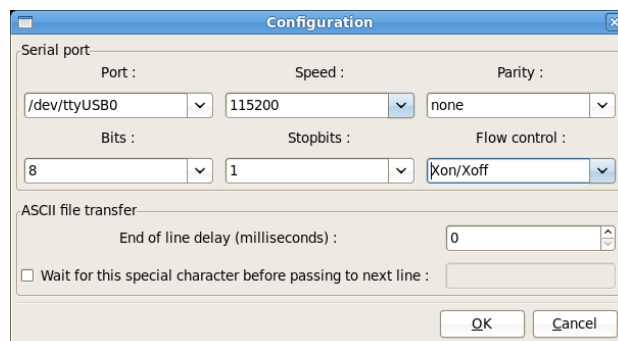


Figure 3-2 UART parameters

15. Change your port ID depending on your system.
16. Click **OK**.
17. To verify the communication link, program the Perseus FPGA with a valid bitstream designed for PetaLinux.
To program the FPGA, see FPGA development with VHDL.

3.2 Network Configurations

Because the RTDEx module included with the Perseus uses a custom Ethernet protocol, a specific network configuration is necessary on the computer that you plan to use as the host device connected to the μ TCA chassis containing the card. This section explains how to do so.

Establish a point-to-point connection through a twisted RJ45 cable between your computer Ethernet card and one of the Perseus FPGA MACs (there are two available MACs).

Note:

Your card must support Gigabit Ethernet. 10/100 Ethernet cards do not allow you to communicate with the Perseus.

3.2.1 Configuring a Windows 7 Network

Jumbo frame size and flow control

The RTDEx module has better throughput when it uses jumbo frames. For better performance of the module, it is therefore important that your Ethernet card be capable of handling jumbo frames. If your Ethernet card does not support such frames, the frame size is limited to 1500 bytes.

To enable jumbo frames and flow control:

1. Open **Control Panel**.
2. Click **Performance and Maintenance**.
3. Click **System**.
4. In the **System Properties** dialog box, on the **Hardware** tab, click **Device Manager**.
5. In the **Device Manager** window, expand **Network adapters**, and then right-click the Ethernet card connected to the Perseus.
6. On the shortcut menu that appears, click **Properties**.
7. In the **Network Connection Properties** dialog box, on the **Advanced** tab, select **Jumbo Packet** from the list of available settings, and then select the highest possible value.

Note:

If **Jumbo Packet** does not appear in the list of available settings, your Ethernet card does not support such packets and the largest frame exchanged with the RTDEx module is limited to 1500 bytes.

8. Select **Flow Control** from the list of available settings and make sure that it is enabled in TX and RX.
9. Click **OK**.

Configuring the host Ethernet card IP address

The host Ethernet card connected to the Perseus must have a static IP address to function with the RTDEx module. This IP address must be in the same subnetwork as the Perseus own IP address.

To configure the host IP address:

10. Open **Control Panel**.
11. Click **Network and Internet Connections** and then click **Network Connections**.
12. In the **Network Connections** window, double-click the local area connection that corresponds to the Ethernet card connected to the Perseus.
13. In the **Local Area Connection Status** dialog box, click **Properties**.
14. In the **Local Area Connection Properties** dialog box, on the **General** tab, clear all the check boxes in the **This connection uses the following items** list, except **Internet Protocol (TCP/IP)**.
15. Double-click **Internet Protocol (TCP/IP)**.
16. In the **Internet Protocol (TCP/IP) Properties** dialog box, on the **General** tab, select the **Use the following IP address** option.
17. In the **IP address** and **Subnet mask** text boxes, specify the IP and subnetwork addresses that you want to use.

18. Click **OK**.

MAC address

Beyond the previous configurations, you must know the MAC address of your Ethernet card to use the RTDEx module. It is used when configuring the RTDEx module.

To display the MAC address:

19. Return to go to **Local Area Connection Status** dialog box.
20. On the Support tab, click **Details** in the **Connection status** group.

The MAC address appears as property **Physical Address** in the **Network Connection Details** dialog box.

21. Make a note of the MAC address and file it for future use.

3.2.2 Configuring a Linux Network

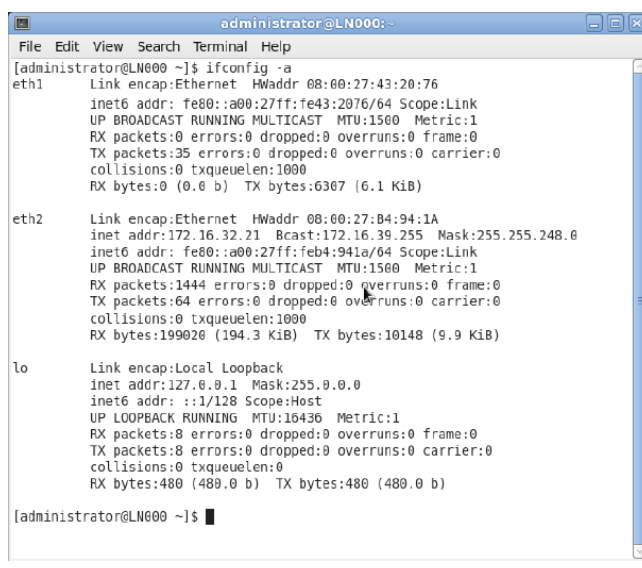
MAC address

You must first determine the MAC address of the host Ethernet card connected to the Perseus. In Linux, your Ethernet card is named `eth#` where `#` is a number.

To display the MAC address:

1. Open a terminal window.
2. At the prompt, type `ifconfig -a`.

The following information appears in the window.



```

administrator@LN000:~$ ifconfig -a
eth1: Link encap:Ethernet HWaddr 08:00:27:43:20:76
      inet6 addr: fe80::a00:27ff:fe43:2076/64 Scope:Link
      UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
      RX packets:0 errors:0 dropped:0 overruns:0 frame:0
      TX packets:35 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:0 (0.0 b)  TX bytes:6307 (6.1 KiB)

eth2: Link encap:Ethernet HWaddr 08:00:27:04:94:1A
      inet addr:172.16.32.21  Bcast:172.16.39.255  Mask:255.255.248.0
      inet6 addr: fe80::a00:27ff:feb4:941a/64 Scope:Link
      UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
      RX packets:1444 errors:0 dropped:0 overruns:0 frame:0
      TX packets:64 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:199020 (194.3 KiB)  TX bytes:10148 (9.9 KiB)

lo:    Link encap:Local Loopback
      inet addr:127.0.0.1  Mask:255.0.0.0
      inet6 addr: ::1/128 Scope:Host
      UP LOOPBACK RUNNING  MTU:16436  Metric:1
      RX packets:8 errors:0 dropped:0 overruns:0 frame:0
      TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:0
      RX bytes:480 (480.0 b)  TX bytes:480 (480.0 b)

administrator@LN000 ~$

```

Figure 3-3 Determining the MAC address

3. In the figure above, the host Ethernet card MAC address is 08:00:27:43:20:76 and its interface name is `eth1`.

Configuring the IP address

The host Ethernet card connected to the Perseus must have a static IP address to function with the RTDEx module. This IP address must be in the same subnetwork as the Perseus' own IP address.

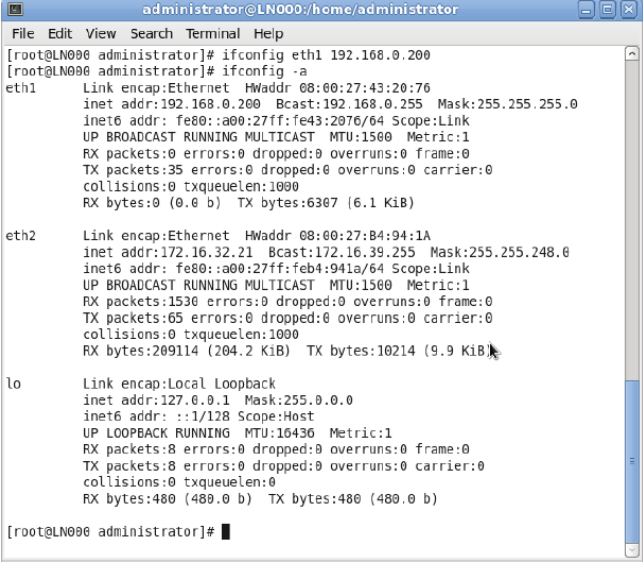
Note:

Make sure that you have administrator access privilege before proceeding. If you do not, consult your network administrator.

To configure the IP address:

- Log on as the administrator (superuser) in the root.
- Type `ifconfig eth# xxx.xxx.xxx.xxx` where # corresponds to your interface number and xxx.xxx.xxx.xxx is the IP address that you want to assign, and then press Enter.
- Type `ifconfig -a` and then press Enter to verify that your IP address is assigned.

The following information appears:



```
administrator@LN000:/home/administrator
File Edit View Search Terminal Help
[root@LN000 administrator]# ifconfig eth1 192.168.0.200
[root@LN000 administrator]# ifconfig -a
eth1      Link encap:Ethernet  HWaddr 08:00:27:43:20:76
          inet addr:192.168.0.200  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe43:2076/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:35 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 b)  TX bytes:6307 (6.1 KiB)

eth2      Link encap:Ethernet  HWaddr 08:00:27:B4:94:1A
          inet addr:172.16.32.21  Bcast:172.16.39.255  Mask:255.255.248.0
          inet6 addr: fe80::a00:27ff:feb4:941a/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1530 errors:0 dropped:0 overruns:0 frame:0
          TX packets:65 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:209114 (204.2 KiB)  TX bytes:10214 (9.9 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:8 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:480 (480.0 b)  TX bytes:480 (480.0 b)

[root@LN000 administrator]#
```

Figure 3-4 IP address of the host Ethernet card

In the figure above, the IP address for *eth1* is 198.168.0.200.

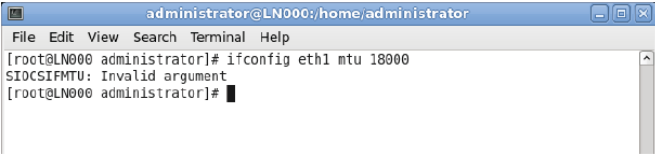
Jumbo frames

The RTDEx module has better throughput when it uses jumbo frames. For better performance of the module, it is therefore important that your Ethernet card be capable of handling jumbo frames. If your Ethernet card does not support such frames, the frame size is limited to 1500 bytes.

To enable jumbo frames:

- Log on as a user with administrator access privileges.
- Type `ifconfig eth# mtu 9000` where # is your interface number and 9000 is the size of jumbo frames, in bytes.

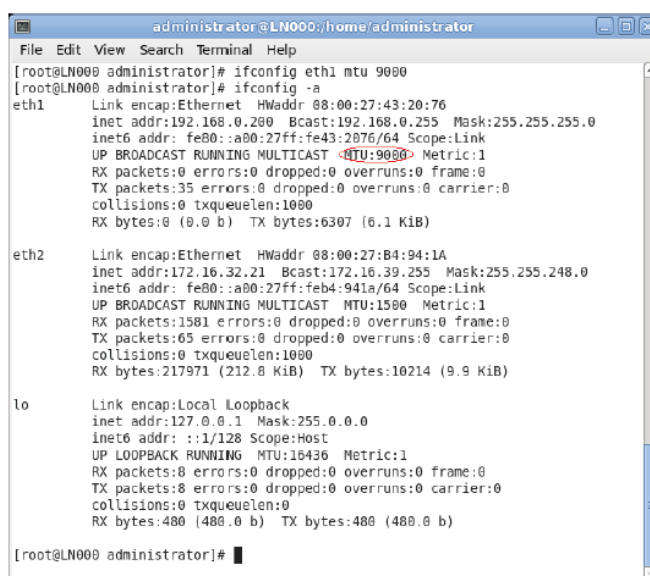
If your card does not support the 9000-byte frames, the “SIOCSIFMTU: Invalid argument” message appears, in which case you should use a lower frame value until you find one that does not produce an error message.



```
administrator@LN000:/home/administrator
File Edit View Search Terminal Help
[root@LN000 administrator]# ifconfig eth1 mtu 18000
SIOCSIFMTU: Invalid argument
[root@LN000 administrator]#
```

Figure 3-5 Invalid jumbo frame size

- Once you specify a valid jumbo frame size, type `ifconfig -a` and then press Enter.
- Verify the MTU value in the information that appears.



```

administrator@LN000:/home/administrator
File Edit View Search Terminal Help
[root@LN000 administrator]# ifconfig eth1 mtu 9000
[root@LN000 administrator]# ifconfig -a
eth1      Link encap:Ethernet  HWaddr 08:00:27:43:20:76
          inet addr:192.168.0.200  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe43:2076/64  Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:9000  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:35 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 b)  TX bytes:6307 (6.1 KiB)

eth2      Link encap:Ethernet  HWaddr 08:00:27:B4:94:1A
          inet addr:172.16.32.21  Bcast:172.16.39.255  Mask:255.255.248.0
          inet6 addr: fe80::a00:27ff:feb4:941a/64  Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1581 errors:0 dropped:0 overruns:0 frame:0
          TX packets:65 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:217971 (212.8 KiB)  TX bytes:10214 (9.9 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128  Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:8 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:480 (480.0 b)  TX bytes:480 (480.0 b)

[root@LN000 administrator]#

```

Figure 3-6 Valid jumbo frame size

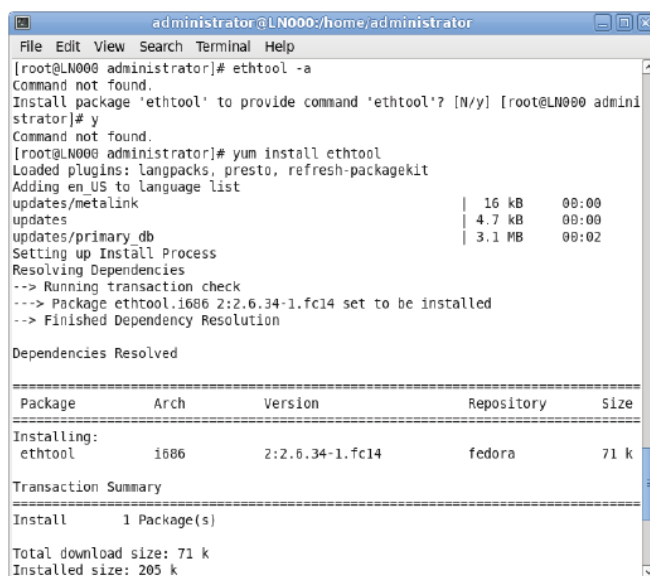
Flow control

You must now verify whether flow control is enabled for your host Ethernet card. In all the procedures below, # is your Ethernet card interface number

To enable flow control:

11. In a terminal window, type **ethtool -a eth#** and then press Enter.

If ethtool is not installed on your computer, use **yum install ethtool** to do so. You must be logged on as a user with administrator access privileges to perform this operation.



```

administrator@LN000:/home/administrator
File Edit View Search Terminal Help
[root@LN000 administrator]# ethtool -a
Command not found.
Install package 'ethtool' to provide command 'ethtool'? [N/y] [root@LN000 admini
strator]# y
Command not found.
[root@LN000 administrator]# yum install ethtool
Loaded plugins: langpacks, presto, refresh-packagekit
Adding en_US to language list
updates/metalink                                | 16 kB    00:00
updates                                          | 4.7 kB    00:00
updates/primary.db                              | 3.1 MB    00:02
Setting up Install Process
Resolving Dependencies
--> Running transaction check
--> Package ethtool.i686 2:2.6.34-1.fc14 set to be installed
--> Finished Dependency Resolution

Dependencies Resolved

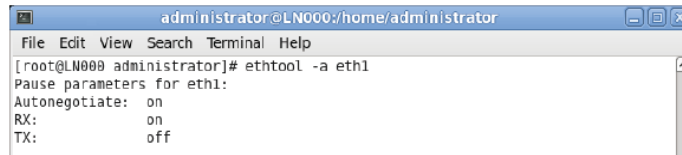
=====
Package             Arch      Version           Repository      Size
=====
Installing:
ethtool             i686      2:2.6.34-1.fc14   fedora          71 k
Transaction Summary
-----
Install      1 Package(s)

Total download size: 71 k
Installed size: 205 k

```

Figure 3-7 Installing ethtool

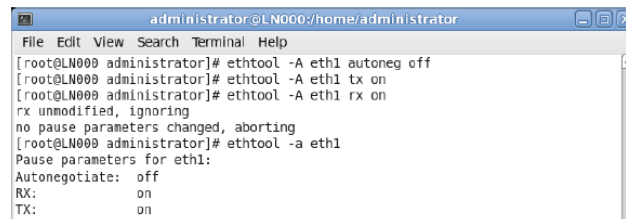
12. Type **ethtool -a eth#** and then press Enter.



```
administrator@LN000:/home/administrator
File Edit View Search Terminal Help
[root@LN000 administrator]# ethtool -a eth1
Pause parameters for eth1:
Autonegotiate: on
RX: on
TX: off
```

Figure 3-8 ethtool

13. Type **ethtool -A eth# autoneg off** and then press Enter.
14. Type **ethtool -A eth# tx on** and then press Enter.
15. Type **ethtool -A eth# rx on** and then press Enter.



```
administrator@LN000:/home/administrator
File Edit View Search Terminal Help
[root@LN000 administrator]# ethtool -A eth1 autoneg off
[root@LN000 administrator]# ethtool -A eth1 tx on
[root@LN000 administrator]# ethtool -A eth1 rx on
rx unmodified, ignoring
no pause parameters changed, aborting
[root@LN000 administrator]# ethtool -a eth1
Pause parameters for eth1:
Autonegotiate: off
RX: on
TX: on
```

Figure 3-9 Flow control

3.3 Programming an FPGA Bitstream into Flash Memory

It is possible to program an FPGA bitstream into the Perseus onboard Flash memory for default FPGA boot up configuration. Two different bitstreams can be located in the Flash memory. It is possible to select which FPGA bitstream will be used for the FPGA configuration at the next boot up.

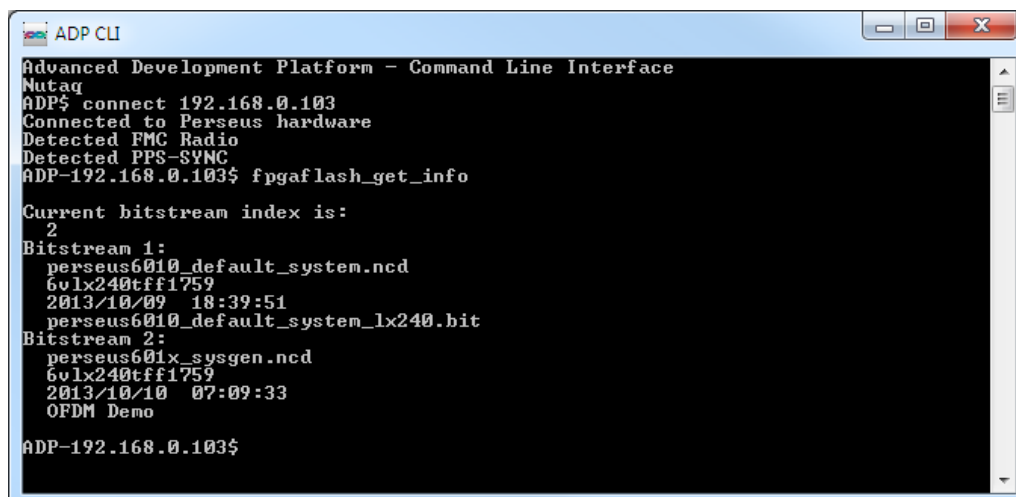
A C programming language (API) is available but the easiest way to program the flash memory is to use BAS Command Line Interface (CLI)

The *fpgaflash* CLI command can be used to write into the flash a bitstream at a specific index partition. The available partition indexes are 1 and 2.

The following commands are valid CLI instruction.

```
fpgaflash 1 "C:\download1.bit"
fpgaflash 2 "C:\download2.bit" "This is my comment"
```

The first argument is the partition index targeted, the second is the bitstream path and the third one is optional allow the user to write a comment to described the bitstream. If no comment is provided, the bitstream file name will be used instead.



```
ADP CLI
Advanced Development Platform - Command Line Interface
Nutaq
ADP$ connect 192.168.0.103
Connected to Perseus hardware
Detected FMC Radio
Detected PPS-SYNC
ADP-192.168.0.103$ fpgaflash_get_info
Current bitstream index is:
2
Bitstream 1:
perseus6010_default_system.ncd
6vlx240tff1759
2013/10/09 18:39:51
perseus6010_default_system_lx240.bit
Bitstream 2:
perseus601x_sysgen.ncd
6vlx240tff1759
2013/10/10 07:09:33
OFDM Demo
ADP-192.168.0.103$
```

Figure 3-10 FPGA Flash Information

The *fpgaflash_get_info* CLI command will show the information of the programmed bitstream in the Flash memory. The selected bitstream index for the FPGA configuration at the next boot up will be displayed. For each bitstream into the Flash memory, the following information will be displayed:

- The project name
- The FPGA type
- The date and hour of creation
- Custom field that contains either the user's comment or the bitstream file name

To select the bitstream partition index for the FPGA configuration at the next boot up, the *fpgaflash_set_index* CLI command can be used. The desired index (1 or 2) must be provided as an argument of the command.

```
fpgaflash_set_index 2
```

4 FPGA Development

4.1 Reference Design Block Diagram

The reference designs supplied with the Perseus is built with Platform Studio, using its design flow. These reference designs are inside the “examples” folder. The designs contain all the IP cores necessary to support Linux and the onboard peripherals of the Perseus. Use the design as a starting point for your projects.

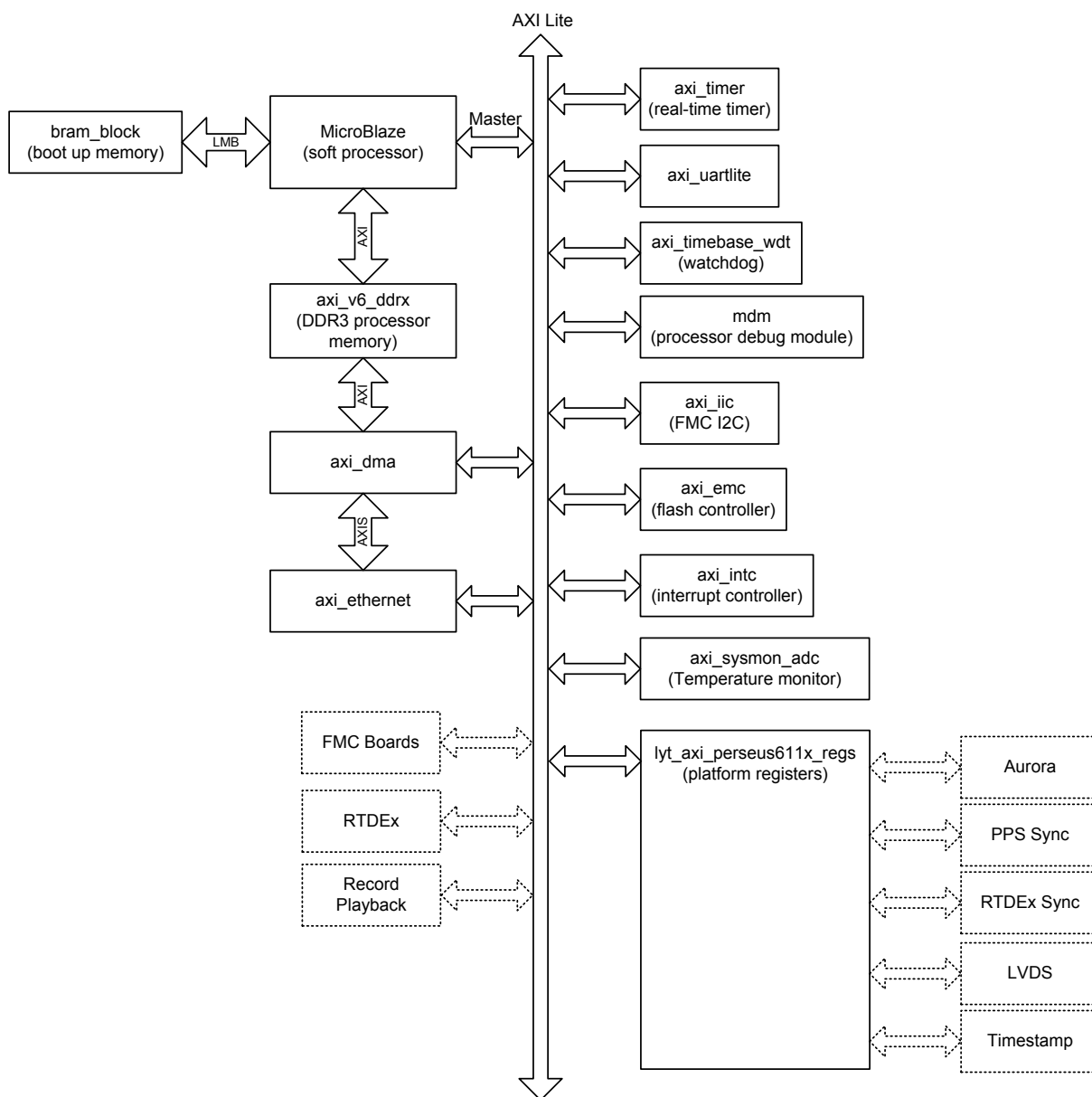


Figure 4-1 Perseus FPGA reference design block diagram

Figure 4-1 shows the system architecture and how the different cores are connected to the processor. Since no hardware processor is available in Virtex-6 FPGAs, a Xilinx MicroBlaze soft processor has to be instantiated and used as the central point of the Platform Studio projects of the Perseus.

All the solid line blocks shown in Figure 4-1 must always be present in a Perseus FPGA project. The dotted line blocks indicate Nutaq cores that can be added to the base design to take advantage of the already developed functionalities. Some blocks have their own AXI Lite decoder (FMC, RTDEx, ...) while other blocks (Aurora, PPS Sync, ...) are directly mapped to the platform registers allowing saving the logic resources involved when the core instantiates its own AXI decoder.

4.2 Perseus FPGA Address Map

Figure 4-2 shows the address map of the various IP cores connected to the MicroBlaze processor as shown in Figure 4-1. In order for the all Nutaq BAS API to work correctly, the core addresses should match exactly this address map.

Memory Mapping		User register space		System Peripheral	
0x0000 0000	Boot memory	0x7000 0000	Platform register	0x8000 0000	Processor debugging module
0x0000 8000	Reserved	0x7100 0000	Ethernet RTDEx	0x8160 0000	FMC I2C
0x6000 0000	User register space	0x7200 0000	Reserved	0x8180 0000	Interrupt controller
0x8000 0000	System peripherals	0x7300 0000	Recording & playback	0x81C0 0000	Ethernet MAC
0x9000 0000	DDR3 processor memory	0x7400 0000	DDR3 SODIMM I2C	0x8360 0000	Watchdog
0x9800 0000	Reserved	0x7500 0000	System Monitor	0x83C0 0000	Timer
0xA000 0000	Flash memory			0x8400 0000	UART
0xA400 0000	Reserved			0x8460 0000	Ethernet DMA
				0x8500 0000	FMC board #0 on FMC connector #1
				0x8501 0000	FMC board #1 on top of board #0 if in a double-stack configuration
				0x8502 0000	FMC board #2 on FMC connector #2
				0x8503 0000	FMC board #3 on top of board #2 if in a double-stack configuration

Figure 4-2 Perseus MicroBlaze address map

4.2.1 User Register Space

The user register space is a memory address space that ranges from address 0x60000000 to 0x7FFFFFFF. This memory space can be used to connect custom user cores since it accepts direct read and write accesses from the API.

For example, if a user AXI core is mapped at the base address 0x60000000, “memory_read_send” and “memory_write_send” EAPI functions can be used to access and modify the AXI register values.

4.2.2 System Peripheral Space

The system peripheral space is a memory address space that ranges from address 0x80000000 to 0x8FFFFFFF. This memory space is used by platform peripheral cores. These cores are controlled by their own drivers and do not accept direct read and write accesses from the API.

The FMC memory space that ranges from address 0x85000000 to 0x85FFFFFF, unlike the rest of the system peripheral space, can be directly accessed by reads and writes originating from the API. Even if the FMC modules have their own driver, their internal registers can be accessed for debugging purpose, thus giving more control to the user.

4.3 Module List

4.3.1 MicroBlaze

The Perseus FPGA base design uses a MicroBlaze soft processor to run Linux or a stand-alone application. For simple stand-alone applications, the program can be entirely stored into the block RAM. In the case of larger applications, the program can be stored in the flash memory or loaded in the DDR3 SDRAM or internal block RAM (or any combination thereof).

When using Nutaq's BSDK or MBDK design flow, the MicroBlaze boots Linux from the Flash memory and launches a TCP application to listen to the commands sent from the host computer.

For details, refer to the documentation accompanying Platform Studio.

4.3.2 Processor Boot Memory

This boot memory is implemented using FPGA BRAM block. The processor boot memory section ranges from address 0x00000000 to 0x00007FFF and contains the processor boot code. At startup, the MicroBlaze runs the code in this section. By default, a first-stage boot loader occupies this section of memory and allows the MicroBlaze to jump to the program stored at the beginning of the flash memory, which is by default, a second-stage boot loader called U-Boot. U-Boot loads the Linux operating system into the MicroBlaze DDR3 memory.

Instead of the first-stage boot loader, you can also build your own application and store it within this section.

If you want to develop your own application and it does not fit entirely in the BRAM memory. You can store your application into the Flash and use a boot loader to load it on system startup.

4.3.3 DDR3 Processor Memory

The DDR3 processor memory section ranges from address 0x90000000 to 0x97FFFFFF and it maps the content of an external DDR3 memory chip dedicated to MicroBlaze code. The DDR3 memory is made available to the MicroBlaze through Xilinx multiport DDR3 memory controller called `axi_v6_ddrx`.

If the default first-stage boot loader (*perseus_default_linux.elf*) is located inside the FPGA boot-up memory, Linux code will be loaded from the flash memory to this DDR3 memory during the startup of the system.

For details on Xilinx memory controller core, refer to the documentation accompanying Platform Studio.

For details about the DDR3 SDRAM chip, see your platform User Guide.

4.3.4 AXI Lite Bus

The Perseus reference design uses an FPGA version of AXI from Xilinx. The AXI-Lite is fully synchronous and allows several masters and slaves to be interconnected through it. The reference design includes one master (MicroBlaze), while the other peripherals are slaves (platform registers, flash memory, UART, ...).

4.3.5 Real-Time Timer

The real-time timer section ranges from address 0x83C00000 to 0x83C000FF and is reserved for the real-time timer. It is needed by the operating system scheduler to perform process switching.

For details, refer to the Xilinx documentation accompanying Platform Studio.

4.3.6 UART Controller

The UART controller section ranges from address 0x84000000 to 0x8400FFFF and is reserved for the UART controller. The reference design uses a UART controller to transfer commands to and from the MicroBlaze or to handle the Linux console interface in the BSDK development flow.

On the Perseus601X platform, the UART can be accessed through the AMC backplane or by using a Mestor board. On the Perseus611X platform, the UART is also accessible through the RTM backplane.

The Pico AMC backplane, the RTM backplane and the Mestor boards from Nutaq all provide a mini-USB interface to the UART bus.

If connected to a PC, PuTTY application (or any serial application) can be used to access the MicroBlaze through the UART. In the case of a console application, the stdin and stdout file descriptors can be redirected from and to the UART. For details, refer to the documentation accompanying Platform Studio.

4.3.7 Watchdog Timer

The watchdog timer section ranges from address 0x83600000 to 0x836000FF and is reserved for the watchdog timer. It allows the system to be reset after failures.

For details, refer to the Xilinx documentation accompanying Platform Studio.

4.3.8 Processor Debugging Module (mdm)

The processor debugging module section ranges from address 0x80000000 to 0x8000FFFF and is reserved for the processor debugging module. It allows the user to debug his application code using a JTAG module. The module can also send commands to the processor such as start, stop, reset or read/write registers. It is also capable of loading a binary image of Linux.

For details, refer to the Xilinx documentation accompanying Platform Studio.

4.3.9 FMC I2C

The FMC I2C section ranges from address 0x81600000 to 0x8160FFFF and is reserved for the FPGA core communication with FMCs through I2C.

The I2C communication is handled by the Xilinx “axi_iic” core. For details, refer to the Xilinx documentation accompanying Platform Studio.

4.3.10 Flash Memory Controller

The flash memory controller section ranges from address 0xA0000000 to 0xA3FFFFFF and is reserved for the flash memory. The flash memory is used to store Linux system code and FPGA bitstreams.

The base address of the flash memory controller is 0xA0000000. Using this base address, the absolute addresses of the flash memory sections described below can be calculated by adding this base address to the address of the appropriate sub-section memory space. For example, the DTB section absolute address range is 0xA0060000 to 0xA007FFFF.

The following is a list of all the default Flash memory spaces of the Perseus.

0x0000 0000 – 0x0003 FFFF — U-Boot Image

This section contains the binary code of the U-Boot application (second-state boot loader) that loads Linux to the DDR3 SDRAM.

0x0004 0000 – 0x0005 FFFF — U-Boot Environment

This section contains the environment variables (data section) of the U-Boot application.

0x0006 0000 – 0x0007 FFFF — DTB

0x0008 0000 – 0x00AF FFFF — Linux image

This section contains the Linux image (code) that is loaded to the MicroBlaze DDR3 SDRAM.

0x00B0 0000 – 0x00B1 FFFF — Configuration

0x00B2 0000 – 0x017F FFFF — JFFS2

0x0180 0000 – 0x02BF FFFF — FPGA2: Second bitstream in BPI down configuration

0x02C0 0000 – 0x03FF FFFF — FPGA1: First bitstream in BPI down configuration

Each **FPGA** section contains an FPGA bitstream stored in BPI down mode. The bitstream located in these sections can be directly loaded to the FPGA at the system startup. The **FPGA** partition size is 20 MB so it is able hold the bitstream of the biggest supported FPGA (XC6VSX475T).

4.3.11 Interrupt Controller

The interrupt controller section ranges from address 0x81800000 to 0x8180FFFF and is reserved for the FPGA interrupt controller. It handles interrupts from the FMC I2C, the DDR3 memory processor's SDMA, the Ethernet AMC 0 TMAC, the watchdog timer, the front panel UART and the PCIe.

For details, refer to the Xilinx documentation accompanying Platform Studio.

4.3.12 System Monitor

In the reference design, the axi_sysmon_adc core is provided to monitor the FPGA temperature and its different voltages.

For more details, look for “sysmon” in API documentations and refer to the documentation accompanying Platform Studio.

4.3.13 Platform Registers

The platform registers core is a specific component with registers that handle the configuration of the entire platform. These registers control the clock switches, the FMC power supplies, the user LEDs and the reset mechanism of the platform.

The platform registers core also provides the user with custom registers he can use for his design implementation as well as specific registers from FPGA cores that have their own AXI decoder.

For more detail of the platform register core of the Perseus601X refer to chapter 5.

For more detail of the platform register core of the Perseus611X refer to chapter 6.

4.3.14 Ethernet Controller and DMA

The reference design of the Perseus uses modified versions of the Xilinx tri-mode Ethernet media access controller (TEMAC) core and Xilinx DMA core to supply Ethernet connectivity to the platform.

The Ethernet controller and DMA section ranges from address 0x81C00000 to 0x81C7FFFF and is reserved for the Ethernet port 0 of the AMC backplane connector. The Ethernet port handles communications between a host application and the MicroBlaze through UDP and TCP.

The DMA engine section ranges from address 0x84600000 to 0x8460FFFF and is reserved for the DMA engine of the Ethernet module.

When used with Nutaq's Gigabit Ethernet RTDEx core, to support RTDEx transfer protocol through the same Ethernet link, the Ethernet controller routes the data to the RTDEx core instead of directly to the DMA engine.

For details on the Ethernet controller, refer to the Xilinx documentation accompanying Platform Studio.

4.3.15 FMC Board

The addresses ranging from 0x85000000 to 0x85FFFFFF are reserved to access all the FMC boards connected to the Perseus. Each FMC address space size is 0x10000 bytes wide. The Perseus601X only has one FMC connector

while the Perseus611X has two FMC connectors. Since with Nutaq's double-stack configuration, two FMC boards can be connected to the same FMC connector, the Perseus601X can support up to 2 FMC boards and the Perseus611X up to four FMC boards.

The shows the address map of the FMC cores:

Address ranges	FMC boards
0x85000000–0x8500FFFF	FMC Board #0 Address range for board at position #0 Connected directly to the FMC connector #1.
0x85010000–0x8501FFFF	FMC Board #1 Address range for board at position #1 Connected on top of the FMC board #0.
0x85020000–0x8502FFFF	FMC Board #2 Address range for board at position #2 Connected directly to the FMC connector #2.
0x85030000–0x8503FFFF	FMC Board #3 Address range for board at position #3 Connected on top of the FMC board #2.

Table 2 FMC address map

Nutaq supports various FMC cards for the Perseus carrier boards. For each supported card, an AXI IP core provides an interface between the FMC hardware and the MicroBlaze processor. All the available FMC examples are located in the “examples” directory of the software.

4.3.16 RTDEx Cores

The Perseus FPGA is equipped with interfaces allowing exchange of data in real time between the FPGA and a host device.

To learn how an RTDEx interface can be integrated into a FPGA design, refer to the RTDEx examples (GigE or PCIe) available in the software release.

For complete information on the RTDEx cores, please refer to the *RTDEx Programmer's Reference Guide.pdf* document.

4.3.17 Record Playback and the DDR3 SODIMM

You can access this DDR3 SODIMM through a wrapped MIG interface. Nutaq provides a custom IP core (lyt_axi_record_playback_v1_00_a) record and playback data to/from this memory.

The FPGA SDRAM recording module allows for the storage of bursts of data on the onboard SDRAM. The data can then be transferred at a slower rate through the RTDEx module to a remote host device for analysis. The FPGA SDRAM playback module allows the transmission of large portions of data from the FPGA SDRAM after this data is written to the SDRAM by the host device.

To learn how to use this IP core, please refer to the record/playback examples available in the software release.

For complete information on the record/playback core, please refer to the *Programmer's Reference Guide RecordPlayback.pdf* document.

4.3.18 Other Nutaq Cores

Various Nutaq IP cores (Aurora, PPS Sync, RTDEx Sync, LVDS, Timestamp) do not have their own AXI interface. This choice was made to save logic resources in the FPGA. To still be controllable from the MicroBlaze processor, these cores directly use specific registers instantiated by the platform register core (lyt_axi_perseus6010_regs or lyt_axi_perseus611x_regs).

5 Perseus601X Platform Register Core Description

5.1 Core Level Features

The platform register core has several registers. These registers control different hardware aspects of the Perseus601x such as:

- Front panel user LEDs state
- Configuration of the AMC backplane TCLK and FCLKA switch
- Control over the FMC V_{adj} power supply voltage
- Enabling of onboard oscillators
- Reset of the FPGA

The registers also provide statuses for the boards connected to the FMC connector.

Furthermore, several custom registers are made available to the user granting access to configurable registers for his custom logic without the need of instantiating his own AXI decoder.

Finally, a few Nutaq IP cores do not have their own AXI interface and use the platform specific registers instead.

5.2 Core Ports

This section presents a list of the FPGA core ports.

Port	Range	Direction	Description
Reset and miscellaneous control/status			
o_nFpgaProg_p	–	Out	Reprograms the FPGA with the bitstream stored in the flash memory
i_daughter_absent_p	–	In	Indicate if the Mestor daughter board is physically absent or not.
i_fmc1_absent_p	–	In	Indicate if there is an FMC board connected to the FMC connector.
i_fmc1_stack_absent_p	–	In	Indicate if there are FMC boards in double-stack configuration connected to the FMC connector. This status requires a valid V_{ADJ} power supply voltage. Therefore, it cannot be used during the FMC identification process prior to the FMC connector power up.
i_pcie_present_p	–	In	Indicates if the PCIe core is present in the FPGA design. This signal must be connected manually when creating a new project
i_pcie_clock_lock_p	–	In	Indicates the PCIe clock lock state. This signal must be connected to the “mmcm_lock” of the “axi_pcie” component when the PCIe is used.
o_pcie_msi_p	-	Out	Message signaled interrupt from the MicroBlaze processor.

Port	Range	Direction	Description
ov2_uart_mode_p	[1:0]	Out	Select the active UART port.
i_ddr3_init_done_p	–	In	Initialization status of the DDR3 memory used by the MicroBlaze processor.
Clock control			
o_CtrlAmctclka2Fmcclk2En_p	–	Out	Enables or disables the AMC TCLKA clock path to FMC clock 2.
o_CtrlAmctclkc2Fmcclk3En_p	–	Out	Enables or disables the AMC TCLKC clock path to FMC clock 3.
o_CtrlFmcclk02AmctclkbEn_p	–	Out	Enables or disables the FMC clock 0 clock path to AMC TCLK B.
o_CtrlFmcclk12AmctclkdEn_p	–	Out	Enables or disables the FMC clock 1 clock path to AMC TCLK D.
o_CtrlTclkdTxEn_p	–	Out	Enables the TCLKD clock in the TX direction.
o_CtrlTclkdRxDis_p	–	Out	Disables the TCLKD clock in the RX direction.
o_CtrlTclkcTxEn_p	–	Out	Enables the TCLKC clock in the TX direction.
o_CtrlTclkcRxDis_p	–	Out	Disables the TCLKC clock in the RX direction.
o_CtrlTclkbTxEn_p	–	Out	Enables the TCLKB clock in the TX direction.
o_CtrlTclkbRxDis_p	–	Out	Disables the TCLKB clock in the RX direction.
o_CtrlTclkaTxEn_p	–	Out	Enables the TCLKA clock in the TX direction.
o_CtrlTclkaRxDis_p	–	Out	Disables the TCLKA clock in the RX direction.
o_Ctrl100mhzOutEn_p	–	Out	Enables or disables the 100 MHz clock.
o_CtrlFclkaHighz_p	–	Out	Modifies the state of FCLKA.
i_fmc1_clkdir_p	–	In	Indicate the FMC bidirectional clocks 2 and 3 direction for FMC connector.
FMC V_{ADJ} power supply control			
o_CtrlVadjSel1_p	–	Out	Sets the V _{ADJ} power supply value.
o_CtrlVadjSel0_p	–	Out	
o_CtrlVadjEn_p	–	Out	Enables or disable the V _{ADJ} power supply.
User LED control			
o_CtrlLedBufOd_p	–	Out	Connected to the FPGA o_CtrlLedBufOd_p pin.
ov8_nCtrlLedGrn_p	[7:0]	Out	Connected to the FPGA o_nCtrlLed1Grn_p to o_nCtrlLed8Grn_p pin.
ov8_nCtrlLedRed_p	[7:0]	Out	Connected to the FPGA o_nCtrlLed1Red_p to o_nCtrlLed8Red_p pins.

Port	Range	Direction	Description
I²C bus access			
o_Mmcl2cReleaseReq_p	–	Out	This bit enables a request to the MMC bus. It must be set to 1 to request I ² C bus ownership.
i_Mmcl2cReleaseAck_p	–	In	When a request is sent to the MMC I ² C bus, the MMC responds by setting this bit high.
Custom Registers			
ov32_CustomReg#_p	[31:0]	Out	Available for the user logic for control or data input. # range is from 0 to 31.
iv32_CustomReg#_p	[31:0]	In	Available for the user logic for status or data output. # range is from 0 to 31.

Table 3 Perseus 601X Platform Register core port list

The specific registers for Nutaq IP cores that do not have their own AXI interface is not described in this document. For more details, refer to the Programmer's Reference Guide of the desired module.

5.3 Configuration Registers

5.3.1 Registers Map

To easily control the Perseus platform, Nutaq supplies an AXI core that must be instantiated in the processor design. To do so, it must be used with the specific Perseus API and with the generic carrier API.

Register name	Offset from core base address	Direction	Description
INFO	0x00	R	Core ID and version identifier
MODULE_CTRL_STATUS	0x04	R/W	Control and status of the modules that can be connected to the carrier
CLK_CTRL	0x08	R/W	Clock switch and enable controls
VADJ_AND_FPGA_PROG_CTRL	0x0C	R/W	V _{ADJ} power supply control and FPGA reset
LED_CTRL	0x10	R/W	User LED control
MMC_CTRL	0x14	R/W	MMC control register
CUSTOM_REGISTER	0x18 – 0x94	R/W	Custom registers available for the user logic
AURORA	0x98 – 0xAC	R/W	Aurora specific register space. Can support up to 3 Aurora cores.
PPS_SYNC	0xB0 – 0xC4	R/W	PPS Sync specific register space
LVDS	0xC8 – 0x104	R/W	LVDS specific register space Can support up to 4 LVDS cores.
RTDEX_SYNC	0x108 – 0x114	R/W	RTDEx Sync specific register space
TIMESTAMP	0x118 – 0x124	R/W	Timestamp specific register space

Table 4 Perseus 601X Registers map

5.3.2 Registers Description

Offset 0x00 — INFO

This register shows the Perseus 601X IP core identification and version number.

Bit	31 to 16	15 to 0
Name	core_id	core_version
Direction	R	R
Reset value	0xC601	0x0200

Table 5 Perseus 601X INFO register

Name	Description	Configuration
core_id	This is the Perseus 601X IP core unique ID number.	0xC601
core_version	This is the Perseus 601X IP core version number.	0x0200 = Version 2.0

Table 6 Perseus 601X INFO descriptions

Offset 0x04 — MODULE_CTRL_STATUS

Control and status of the modules that can be connected to the carrier. It indicates the presence of FMC and Mestor boards.

Bit	31 to 13	10	9	8
Name	RESERVED	pcie_msi	pcie_clock_lock	pcie_present
Direction	R	R/W	R	R
Reset value	0	0	0	0

Bit	7	6 to 4	3	2	1	0
Name	ddr3_init_done	RESERVED	fmc1_clkdir	fmc1_stack_absent	fmc1_absent	daughter_absent
Direction	R	R	R	R	R	R
Reset value	0	0	0	0	0	0

Table 7 Perseus 601X MODULE_CTRL_STATUS register

Name	Description	Configuration
daughter_absent	Indicates whether a Mestor or Mestor expander is present at the connector. Careful, some Mestor hardware version does not identify themselves correctly.	0x0: Mestor present. 0x1: Mestor absent.
fmc1_absent	Indicates whether an FMC is present at the FMC connector.	0x0: FMC present. 0x1: FMC absent.
fmc1_stack_absent	Indicates whether a second FMC is present at the FMC connector, creating an FMC stack. For this status to work correctly, the V_{ADJ} power supply of the FMC connector must be active.	0x0: FMC stack present. 0x1: FMC stack absent.
fmc1_clkdir	Indicates the direction of the FMC bidirectional clocks (CLK2_BIDIR and CLK3_BIDIR) on the FMC connector.	0x0: CLK2 and CLK 3 are driven from the carrier to the FMC. 0x1: The clocks are driven from the FMC to the carrier.
ddr3_init_done	Initialisation status of the DDR3 memory used by the MicroBlaze processor.	0x0: Initialization failed 0x1: Initialization done without error
pcie_present	Indicates if the PCIe core is present in the FPGA design. This signal must be connected manually when creating a new project.	0x0: PCIe core is not present 0x1: PCIe core is present
pcie_clock_lock	Indicates the PCIe clock lock state. This signal must be connected to the “mmcm_lock” of the “axi_pcie” component when the PCIe is used.	0x0: PCIe clock is not locked 0x1: PCIe clock is locked
pcie_msi	Message signaled interrupt from the microblaze processor.	0x0: Clear PCIe interrupt 0x1: Rise PCIe interrupt

uart_mode	Select the active UART port.	0x0: All UART ports 0x1: AMC backplane UART 0x2: Mestor UART
-----------	------------------------------	--

Table 8 Perseus 601X MODULE_CTRL_STATUS descriptions

Offset 0x08 — CLK_CTRL

Control the platform clocks.

Bit	31 to 15		14		13		12	
Name	RESERVED		CtrlGatedClkEn		Ctrl100mhzOutEn		CtrlFclkaHighz	
Direction	R		R/W		R/W		R/W	
Reset value	0		1		0		0	
Bit	11		10		9		8	
Name	CtrlAmctclka2Fmcclk2En		CtrlAmctclk2Fmcclk3En		CtrlFmcclk02AmctclkbEn		CtrlFmcclk12AmctclkdEn	
Direction	R/W		R/W		R/W		R/W	
Reset value	0		0		0		0	
Bit	7	6	5	4	3	2	1	0
Name	CtrlTclkaRxDis	CtrlTclkaTxEn	CtrlTclkbRxDis	CtrlTclkbTxEn	CtrlTclkcRxDis	CtrlTclkcTxEn	CtrlTclkdRxDis	CtrlTclkdTxEn
Direction	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	1	1	1	0	1	1	1	0

Table 9 Perseus 601X CLK_CTRL register

Bit	Description	Configuration
CtrlTclk[X]TxEn	Enables or disables the TX clock. [X] range from 'a' to 'd'	0x0: Disables the TX clock. 0x1: Enables the TX clock.
CtrlTclk[X]RxDis	Enables or disables the RX clock. [X] range from 'a' to 'd'	0x0: Enables the RX clock. 0x1: Disables the RX clock.
CtrlFmccl12AmctclkdEn	Enables or disables the FMC clock 1 clock path to AMC TCLK D.	0x0: Enables the path. 0x1: Disables the path.
CtrlFmccl02AmctclkbEn	Enables or disables the FMC clock 0 clock path to AMC TCLK B.	0x0: Enables the path. 0x1: Disables the path.
CtrlAmctclk2Fmccl3En	Enables or disables the AMC TCLK C clock path to FMC clock 3.	0x0: Enables the path. 0x1: Disables the path.
CtrlAmctclka2Fmccl2En	Enables or disables the AMC TCLK A clock path to FMC clock 2.	0x0: Enables the path. 0x1: Disables the path.
CtrlFclkaHighz	Modifies the state of FCLKA.	0x0: Sets FCLKA to low impedance. 0x1: Sets FCLKA to high impedance.
Ctrl100mhzOutEn	Enables or disables the 100 MHz clock.	0x0: Disables the 100 MHz clock. 0x1: Enables the 100 MHz clock.

CtrlGatedClkEn	Enables or disables a global clock. This control is currently unused.	0x0: Disables the clock. 0x1: Enables the clock. (default)
----------------	---	---

Table 10 Perseus 601X CLK_CTRL descriptions

Offset 0x0C — VADJ_RESET_CTRL

Configure the value of the V_{ADJ} power supply that power up the FMC connector. To prevent damaging your FMC, make sure that the voltage configuration matches your FMC card requirement before connecting it to the Perseus and enabling V_{ADJ} .

Bit	31 to 4	3	2	1 downto 0
Name	RESERVED	nFpgaProg	CtrlVadjEn	CtrlVadjSel
Direction	R	R/W	R/W	R/W
Reset value	0	1	0	“00”

Table 11 Perseus 601X VADJ_RESET_CTRL registers

Name	Description	Configuration
CtrlVadjSel	Sets the V_{ADJ} power supply voltage value for FMC connector when V_{ADJ} is power supply is active.	0x0: Set the V_{ADJ} to 1.2 V. 0x1: Set the V_{ADJ} to 1.5 V. 0x2: Set the V_{ADJ} to 1.8 V. 0x3: Set the V_{ADJ} to 2.5 V.
CtrlVadjEn	Enables or disable the V_{ADJ} power supply for the FMC connector.	0x0: Disables V_{ADJ} . 0x1: Enables V_{ADJ} .
nFpgaProg	Reprograms the FPGA with the bitstream stored in the flash memory.	0x0: Reprograms the FPGA. 0x1: Does not reprogram the FPGA.

Table 12 Perseus 601X VADJ_RESET_CTRL descriptions

Offset 0x10 — LED_CTRL

Allows you to configure the value of the front panel LEDs.

Bit	31 to 17	16	15 to 8	7 to 0
Name	RESERVED	CtrlLedBufOd	nCtrlLedGrn	nCtrlLedRed
Direction	R	R/W	R/W	R/W
Reset value	0	1	0x00	0x00

Table 13 Perseus 601X LED_CTRL register

Bit	Description	Configuration
nCtrlLedRed	Configures the state of each front panel red LED.	When the bits are 0 the LEDs light.
nCtrlLedGrn	Configures the state of each front panel green LED.	When the bits are 0 the LEDs light.
CtrlLedBufOd	Enables or disables the front panel LED buffers. When disabled, the LEDs do no light.	0x0: Enables the LED buffers. 0x1: Disables the LED buffers.

Table 14 Perseus 601X LED_CTRL descriptions

Offset 0×14 — MMC_CTRL

This register allows accessing the MMC I²C bus.

Bit	31 to 2	1	0
Name	RESERVED	Mmcl2cReleaseAck	Mmcl2cReleaseReq
Direction	R	R	R/W
Reset value	0	0	0

Table 15 Perseus 601X MMC_CTRL register

Bit	Description	Configuration
Mmcl2cReleaseAck	When a request is sent to the MMC I ² C bus, the MMC responds by setting this bit.	When this bit is <i>1</i> , the MMC acknowledges the request and grants access to the MMC I ² C bus. When this bit value is <i>0</i> after the request, the MMC denies access to the MMC I ² C bus. Once the request is sent, the bit reverts to <i>0</i> , acknowledging the end of request.
Mmcl2cReleaseReq	This bit enables a request to the MMC bus. It must be set to 1 to request I ² C bus ownership	When the bit is <i>1</i> , a bus request is sent to the MMC. When the bit value changes from <i>1</i> to <i>0</i> , an end of request is sent to the MMC.

Table 16 Perseus 601X MMC_CTRL descriptions

Offsets 0×18 to 0×94 — CUSTOM_REGISTER

You can use these 32 registers as general-purpose user-defined registers (control or data). Each custom register occupy 4 bytes (32 bits) and the read and write port can be used independently.

Bit	31 to 0
Name	CustomReg[N] <i>N from 0 to 31</i>
Direction	R/W
Reset value	0

Table 17 Perseus 601X CUSTOM_REGISTER register

6 Perseus 611X Platform Register Core Description

6.1 Core Level Features

The platform register core has several registers. These registers control different hardware aspects of the Perseus611x such as:

- Front panel user LEDs state
- Configuration of the AMC backplane TCLK and FCLKA switch
- Control over the FMC V_{adj} power supply voltage
- Enabling of onboard oscillators
- Reset of the FPGA

The registers also provide statuses for the boards connected to the FMC connector.

Furthermore, several custom registers are made available to the user granting access to configurable registers for his custom logic without the need of instantiating his own AXI decoder.

Finally, a few Nutaq IP cores do not have their own AXI interface and use the platform specific registers instead.

6.2 Core Ports

This section presents a list of the FPGA core ports.

Port	Range	Direction	Description
Reset and status			
o_fpga_reset_n_p	–	Out	Reprograms the FPGA with the bitstream stored in the flash memory
i_daughter_absent_p	–	In	Indicate if the Mestor daughter board is physically absent or not.
i_fmc1_absent_p	–	In	Indicate if there is an FMC board connected to the FMC connector #1.
i_fmc1_stack_absent_p	–	In	Indicate if there are FMC boards in double-stack configuration connected to the FMC connector #1. This status requires a valid V_{Adj} power supply voltage. Therefore, it cannot be used during the FMC identification process prior to the FMC connector #1 power up.
i_fmc1_clkdir_p	-	In	Indicate the FMC bidirectional clocks 2 and 3 direction for FMC connector #1.
i_fmc2_absent_p	–	In	Indicate if there is an FMC board connected to the FMC connector #2.
i_fmc2_stack_absent_p	–	In	Indicate if there are FMC boards in double-stack configuration connected to the FMC connector #2. This status requires a valid V_{Adj} power supply voltage. Therefore, it cannot be used during the FMC identification process prior to the FMC connector #2 power up.

Port	Range	Direction	Description
i_fmc2_clkdir_p	-	In	Indicate the FMC bidirectional clocks 2 and 3 direction for FMC connector #2.
i_pcie_present_p	-	In	Indicates if the PCIe core is present in the FPGA design. This signal must be connected manually when creating a new project
i_pcie_clock_lock_p	-	In	Indicates the PCIe clock lock state. This signal must be connected to the "mmcm_lock" of the "axi_pcie" component when the PCIe is used.
o_pcie_msi_p	-	Out	Message signaled interrupt from the MicroBlaze processor.
ov2_uart_mode_p	[1:0]	Out	Select the active UART port.
i_ddr3_init_done_p	-	In	Initialization status of the DDR3 memory used by the MicroBlaze processor.
Clock control			
o_ctrl_amctclka_to_fmc1clk2_en_p	-	Out	Enables or disables the AMC TCLKA clock path to FMC #1 clock 2.
o_ctrl_amctclka_to_fmc2clk2_en_p	-	Out	Enables or disables the AMC TCLKA clock path to FMC #2 clock 2.
o_ctrl_amctclkc_to_fmc1clk3_en_p	-	Out	Enables or disables the AMC TCLKC clock path to FMC #1 clock 3.
o_ctrl_amctclkc_to_fmc2clk3_en_p	-	Out	Enables or disables the AMC TCLKC clock path to FMC #2 clock 3.
o_mgtrefclk_125m_en_p	-	Out	Enables or disables the 125 MHz MGT clock.
o_mgtrefclk_156m_en_p	-	Out	Enables or disables the 156.25 MHz MGT clock.
o_Ctrl100mhzOutEn_p	-	Out	Enables or disables the 100 MHz MGT clock.
o_CtrlFclkaHighz_p	-	Out	Modifies the state of FCLKA.
i_mgtpll_lol_p	-	In	PLL Loss of Lock indicator of the PLL that generates MGTREFCLK1 of MGT group 111 and MGTREFCLK1 of MGT group 118.
i_mgtpll_clk1_los_p	-	In	Loss of signal indicator for the clock input 1 of the PLL that generates MGTREFCLK1 of MGT group 111 and MGTREFCLK1 of MGT group 118.
i_mgtpll_clk2_los_p	-	In	Loss of signal indicator for the clock input 2 of the PLL that generates MGTREFCLK1 of MGT group 111 and MGTREFCLK1 of MGT group 118.
FMC V_{ADJ} power supply control			
ov2_vadj1_sel_p	-	Out	Sets the V _{ADJ} power supply value for FMC connector #1.
o_vadj1_en_p	-	Out	Enables or disable the V _{ADJ} power supply for FMC connector #1.
ov2_vadj2_sel_p	-	Out	Sets the V _{ADJ} power supply value for FMC connector #2.
o_vadj2_en_p	-	Out	Enables or disable the V _{ADJ} power supply for FMC connector #2.

Port	Range	Direction	Description
User LED control			
o_led_spi_clk_p	-	Out	Clock lane of the SPI communication with the Perseus 611X CPLD to change the front panel LED states.
o_led_spi_data_p	-	Out	Data lane of the SPI communication with the Perseus 611X CPLD to change the front panel LED states.
o_led_spi_cs_n_p	-	Out	Chip select of the SPI communication with the Perseus 611X CPLD to change the front panel LED states.
I²C bus access			
o_i2c_request_p	–	Out	This bit enables a request to the I2C bus. It must be set to 1 to request I2C bus ownership. After a I2C transaction, the bit must be set back to 0 to release the bus.
i_i2c_granted_p	–	In	This bit indicates if the I2C bus access has been granted to the FPGA or not.
ov2_i2c_bus_sel_p	[1:0]	Out	Select to which I2C bus the FPGA want to communicate.
Custom Registers			
ov32_CustomReg#_p	[31:0]	Out	Available for the user logic for control or data input. # range is from 0 to 99.
iv32_CustomReg#_p	[31:0]	In	Available for the user logic for status or data output. # range is from 0 to 99.

Table 18 Perseus 611X Platform Register core port list

The specific registers for Nutaq IP cores that do not have their own AXI interface is not described in this document. For more details, refer to the Programmer's Reference Guide of the desired module.

6.3 Configuration Registers

6.3.1 Registers Map

To easily control the Perseus platform, Nutaq supplies an AXI core that must be instantiated in the processor design. To do so, it must be used with the specific Perseus API and with the generic carrier API.

Register name	Offset from core base address	Direction	Description
INFO	0x00	R	Core ID and version identifier
MODULE_CTRL_STATUS	0x04	R/W	Control and status of the modules that can be connected to the carrier
CLK_CTRL	0x08	R/W	Clock switch and enable controls
VADJ_AND_FPGA_PROG_CTRL	0x0C	R/W	V _{ADJ} power supply control and FPGA reset
LED_CTRL	0x10	R/W	User LED control
I2C_CTRL	0x14	R/W	I2C control register
CUSTOM_REGISTER	0x18 – 0x1A8	R/W	Custom registers available for the user logic
AURORA	0x1AC – 0x1E4	R/W	Aurora specific register space. Can support up to 8 Aurora cores.
PPS_SYNC	0x1E8 – 0x1FC	R/W	PPS Sync specific register space
LVDS	0x200 – 0x23C	R/W	LVDS specific register space. Can support up to 4 LVDS cores.
RTDEX_SYNC	0x240 – 0x24C	R/W	RTDEx Sync specific register space
TIMESTAMP	0x250 – 0x25C	R/W	Timestamp specific register space

Table 19 Perseus 611X Registers map

6.3.2 Registers Description

Offset 0x00 — INFO

This register shows the Perseus 611X IP core identification and version number.

Bit	31 to 16	15 to 0
Name	core_id	core_version
Direction	R	R
Reset value	0xC611	0x0200

Table 20 Perseus 611X INFO register

Name	Description	Configuration
core_id	This is the Perseus 611X IP core unique ID number.	0xC611
core_version	This is the Perseus 611X IP core version number.	0x0200 = Version 2.0

Table 21 Perseus 611X INFO descriptions

Offset 0x04 — MODULE_CTRL_STATUS

Control and status of the modules that can be connected to the carrier. It indicates the presence of FMC and Mestor boards.

Bit	31 to 13	10	9	8
Name	RESERVED	pcie_msi	pcie_clock_lock	pcie_present
Direction	R	R/W	R	R
Reset value	0	0	0	0
Bit	7	6	5	4

Name	ddr3_init_done	fmc2_clkdir	fmc2_stack_absent	fmc2_absent
Direction	R	R	R	R
Reset value	0	0	0	0
Bit	3	2	1	0
Name	fmc1_clkdir	fmc1_stack_absent	fmc1_absent	daughter_absent
Direction	R	R	R	R
Reset value	0	0	0	0

Table 22 Perseus 611X MODULE_CTRL_STATUS register

Name	Description	Configuration
daughter_absent	Indicates whether a Mestor or Mestor expander is present at the connector. Careful, some Mestor hardware version does not identify themselves correctly.	0x0: Mestor present. 0x1: Mestor absent.
fmc[N]_absent	Indicates whether an FMC is present at the connector <i>N</i> .	0x0: FMC present. 0x1: FMC absent.
fmc[N]_stack_absent	Indicates whether a second FMC is present at the connector <i>N</i> , creating an FMC stack. For this status to work correctly, the V_{ADJ} power supply of the FMC connector must be active.	0x0: FMC stack present. 0x1: FMC stack absent.
fmc[N]_clkdir	Indicates the direction of the FMC bidirectional clocks (CLK2_BIDIR and CLK3_BIDIR) on the FMC connector <i>N</i> .	0x0: CLK2 and CLK 3 are driven from the carrier to the FMC. 0x1: The clocks are driven from the FMC to the carrier.
ddr3_init_done	Initialisation status of the DDR3 memory used by the MicroBlaze processor.	0x0: Initialization failed 0x1: Initialization done without error
pcie_present	Indicates if the PCIe core is present in the FPGA design. This signal must be connected manually when creating a new project.	0x0: PCIe core is not present 0x1: PCIe core is present
pcie_clock_lock	Indicates the PCIe clock lock state. This signal must be connected to the “mmcm_lock” of the “axi_pcie” component when the PCIe is used.	0x0: PCIe clock is not locked 0x1: PCIe clock is locked
pcie_msi	Message signaled interrupt from the microblaze processor.	0x0: Clear PCIe interrupt 0x1: Rise PCIe interrupt

Table 23 Perseus 611X MODULE_CTRL_STATUS descriptions

Offset 0x08 — CLK_CTRL

Control the platform clocks.

Bit	31 to 11	10	9	8
Name	RESERVED	mgtpll_lol	mgtpll_clk2_los	mgtpll_clk1_los
Direction	R	R	R	R
Reset value	0	0	0	0
Bit	7	6	5	4
Name	ctrl_amctclkc_tofmc2_clk3_en	ctrl_amctclkc_to_fmc1clk3_en	ctrl_amctclka_to_fmc2clk2_en	ctrl_amctclka_to_fmc1clk2_en
Direction	R/W	R/W	R/W	R/W
Reset value	0	0	0	0

Bit	3	2	1	0
Name	ctrl_fclka_highz	mgtrfclk_100m_en	mgtrfclk_156m_en	mgtrfclk_125m_en
Direction	R/W	R/W	R/W	R/W
Reset value	0	0	1	1

Table 24 Perseus 611X CLK_CTRL register

Name	Description	Configuration
mgtrfclk_125m_en	Enable the 125 MHz local oscillator on the Perseus board that is connected to MGTREFCLK1 of MGT groups 110, 112, 113, 115, 116 and 117.	0x0: Disables clock. 0x1: Enables clock.
mgtrfclk_156m_en	Enable the 156.25 MHz local oscillator on the Perseus board that is connected to MGTREFCLK0 of MGT groups 111, 112, 113, 115, 116 and 117.	0x0: Disables clock. 0x1: Enables clock.
mgtrfclk_100m_en	Enable the 100 MHz local oscillator on the Perseus board that is connected to MGTREFCLK1 of the MGT group 114. When the 100 Mhz is enable and when ctrl_fclka_highz == '0', the 100 MHz clock drive the AMC FCLKA port.	0x0: Disables clock. 0x1: Enables clock.
ctrl_fclka_highz	Modifies the state of FCLKA. When mgtrfclk_100m_en == '0' and ctrl_fclka_highz == '0', the AMC FCLKA is connected to MGTREFCLK0 of MGT group 114. When mgtrfclk_100m_en == '1' and ctrl_fclka_highz == '0', the AMC FCLKA is driven by the 100 MHz local oscillator of the Perseus board. When ctrl_fclka_highz == '1', the AMC FCLKA is in high impedance.	0x0: Allows connection with FCLKA. 0x1: Sets fclka to high impedance.
ctrl_amctclka_to_fmc1clk2_en	Controls the switch to toggle from a connection between bidirectional clock 2 of FMC 1 and the FPGA to a connection between bidirectional clock 2 of FMC 1 and the AMC TCLKA. The connection between bidirectional clock 2 of FMC 1 and the FPGA can be bidirectional depending of the FMC module and the FPGA bitstream. The connection between bidirectional clock 2 of FMC 1 and the AMC TCLKA can only be from the TCLKA to the FMC clock 2 since the AMC TCLKA can only be an input on the Perseus611X board.	0x0: Connection between bidirectional clock 2 of FMC 1 and the FPGA. 0x1: Connection between bidirectional clock 2 of FMC 1 and the AMC TCLKA.
ctrl_amctclka_to_fmc2clk2_en	Same than ctrl_amctclka_to_fmc1clk2_en but for bidirectional clock 2 of FMC 2	0x0: Connection between bidirectional clock 2 of FMC 2 and the FPGA. 0x1: Connection between bidirectional clock 2 of FMC 2 and the AMC TCLKA.
ctrl_amctclkc_to_fmc1clk3_en	Same than ctrl_amctclka_to_fmc1clk2_en but between bidirectional clock 3 of FMC 1 and AMC TCLKC.	0x0: Connection between bidirectional clock 3 of FMC 1 and the FPGA. 0x1: Connection between bidirectional clock 3 of FMC 1 and the AMC TCLKC.
ctrl_amctclkc_to_fmc2clk3_en	Same than ctrl_amctclkc_to_fmc1clk3_en but for bidirectional clock 2 of FMC 2	0x0: Connection between bidirectional clock 3 of FMC 2 and the FPGA. 0x1: Connection between bidirectional clock 3 of FMC 2 and the AMC TCLKC.
mgtpll_clk1_los	Loss of signal indicator for the clock input 1 of the PLL that generates MGTREFCLK1 of MGT group 111 and MGTREFCLK1 of MGT group 118. See Si5324 datasheet.	0x0: CKIN1 present. 0x1: LOS on CKIN1.

mgtpll_clk2_los	Loss of signal indicator for the clock input 2 of the PLL that generates MGTREFCLK1 of MGT group 111 and MGTREFCLK1 of MGT group 118. See Si5324 datasheet.	0x0: CKIN2 present. 0x1: LOS on CKIN2.
mgtpll_lol	PLL Loss of Lock indicator of the PLL that generates MGTREFCLK1 of MGT group 111 and MGTREFCLK1 of MGT group 118. See Si5324 datasheet.	0x0: PLL locked. 0x1: PLL unlocked.

Table 25 Perseus 611X CLK_CTRL descriptions

Offset 0x0C — VADJ_RESET_CTRL

Allows you to configure the value of the V_{ADJ} power supplies that power up the FMC connectors. To prevent damaging your FMC, make sure that the voltage configuration match your FMC card requirement before connecting it to the Perseus and enabling V_{ADJ} .

Bit	31 to 7	6	5 downto 4	3	2	1 downto 0
Name	RESERVED	vadj2_en	vadj2_sel	fpga_reset_n	vadj1_en	vadj1_sel
Direction	R	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	“00”	1	0	“00”

Table 26 Perseus 611X VADJ_RESET_CTRL registers

Name	Description	Configuration
vadj[N]_sel	Sets the V_{ADJ} power supply voltage value for FMC connector N when V_{ADJ} is power supply is active.	0x0: Set the V_{ADJ} to 1.2 V. 0x1: Set the V_{ADJ} to 1.5 V. 0x2: Set the V_{ADJ} to 1.8 V. 0x3: Set the V_{ADJ} to 2.5 V.
vadj[N]_en	Enables or disable the V_{ADJ} power supply for the FMC connector N .	0x0: Disables V_{ADJ} . 0x1: Enables V_{ADJ} .
fpga_reset_n	Reprograms the FPGA with the bistream stored in the flash memory.	0x0: Reprograms the FPGA. 0x1: Does not reprogram the FPGA.

Table 27 Perseus 611X VADJ_RESET_CTRL descriptions

Offset 0x10 — LED_CTRL

Allows you to configure the value of the front panel LEDs.

Bit	31 to 16	15 to 8	7 to 0
Name	RESERVED	led_green_en	led_red_en
Direction	R	R/W	R/W
Reset value	0	0x00	0x00

Table 28 Perseus 611X LED_CTRL register

Name	Description	Configuration
led_red_en	Configures the state of each front panel red LEDs.	When the bits are 1 the LEDs light.
led_green_en	Configures the state of each front panel green LEDs.	When the bits are 1 the LEDs light.

Table 29 Perseus 611X LED_CTRL descriptions

Offset 0×14 — I2C_CTRL

This register allows accessing the shared I²C bus.

Bit	31 to 4	3 to 2	1	0
Name	RESERVED	i2c_bus_sel	i2c_granted	i2c_request
Direction	R	R/W	R	R/W
Reset value	0	00	0	0

Table 30 Perseus 611X I2C_CTRL register

Name	Description	Configuration
i2c_request	This bit enables a request to the I2C bus. It must be set to 1 to request I2C bus ownership. After a I2C transaction, the bit must be set back to 0 to release the bus.	0x0: I2C bus release 0x1: I2C bus request
i2c_granted	This bit indicates if the I2C bus access has been granted to the FPGA or not.	0x0: The FPGA must not used the I2C bus 0x1: I2C bus has been granted to the FPGA
i2c_bus_sel	Select to which I2C bus the FPGA want to communicate.	0x0: Carrier 0x1: FMC 1 0x2: FMC 2 0x3: RTM

Table 31 Perseus 611X I2C_CTRL descriptions**Offsets 0×18 to 0x1A4 — CUSTOM_REGISTER**

You can use these 100 registers as general-purpose user-defined registers (control or data). Each custom register occupy 4 bytes (32 bits) and the read and write port of a given custom register can be used independently.

Bit	31 to 0
Name	CustomReg[N] <i>N from 0 to 99</i>
Direction	R/W
Reset value	0

Table 32 Perseus 611X CUSTOM_REGISTER register

7 Low-Level and Host Programming

This chapter presents the functions available to control your platform.

The low-level driver functions are running on the MicroBlaze and are only used by when developing directly on the MicroBlaze processor. Each carrier board has its own library: the Perseus601x has the `perseus601x_lib` and the Perseus611x the `perseus611x_lib`. On top of these libraries, a generic library (`carrier_lib`) controls the carrier board regardless of its specific type.

The generic carrier library is the only one accessible through host programming. Nevertheless, it provides to the user all the available functionalities of the carrier he uses.

7.1 Perseus601X Driver

Driver Function	Use
Perseus601X_Open	Open an instance of the Perseus601X carrier object. Must be on a Perseus601X carrier board in order for this function to make sense.
Perseus601X_Close	De-allocate the Perseus601X carrier object.
Perseus601X_FmcAssignSlot	This function assigns to each connector of the Perseus601x carrier the detected FMC modules.
Perseus601X_FmcSetVadj	This function sets the required vadj voltage for the carrier board/connector.
Perseus601X_FmcGetVadj	This function verifies if the required Vadj voltage is compatible with what could be provided for a specific board slot. It returns the Vadj level setting needed for the board.
Perseus601X_FmcGetGA	Retrieve the Geographical Address (GA) of the specified FMC connector.
Perseus601X_GetSignalsInfo	This function gets the data pairs (LA, HA, HB) length (delay in ps) for the carrier. The delays are estimated value for material DK 4.5 (180 ps/po).
Perseus601X_GetModuleBaseAddr	Retrieve the base address corresponding to the specified module.
Perseus601X_GetModuleSize	Retrieve the memory space size of the specified module type.
Perseus601X_GetModuleMaxNumber	Retrieve the maximal number of instance of the specified module type the current carrier can handle. The retrieved instance number minus 1, is the maximal index value that can be used when calling the <code>Carrier_GetModuleBaseAddr</code> function.
Perseus601X_GetMemSpaceBaseAddr	Retrieve the base address corresponding to the specified memory space. A memory space is a range of memory address that is available to the user to connect its custom logic to the processor. Once the custom logic is mapped inside a memory space, its registers are reachable using the memory read and write functions.
Perseus601X_GetMemSpaceSize	Retrieve the memory space size of the specified memory space type. A memory space is a range of memory address that is available to the user to connect its custom logic to the processor. Once the custom logic is mapped inside a memory space, its registers are reachable using the memory read and write functions.
Perseus601X_CustomRegisterWrite	Update the value of a custom register in the platform register FPGA core.
Perseus601X_CustomRegisterRead	Retrieve the value of a custom register in the platform register FPGA core.
Perseus601X_LedSetValue	Set the state of the carrier user LEDs.
Perseus601X_LedGetValue	Get the state of the carrier user LEDs.
Perseus601X_Reboot	Reboot the carrier platform. This will reset the FPGA and program it from the Flash memory content.

Driver Function	Use
Perseus601X_I2cWrite	Perform an I2C Write transaction.
Perseus601X_I2cRead	Perform an I2C Read transaction.
Perseus601X_I2cReadNoWrData	Read data from an I2C device. This function does not send the register start address before reading from the I2C interface. It only read the incoming data. It will start reading from current internal device address and so no assumption could be made on the content read from what device address.
Perseus601X_I2cCheckRaw	Check if an i2c device is detected on the i2c bus
Perseus601X_TClkSetState	Set the state (enable or disable) of the specified TClk in the specified direction.
Perseus601X_TClkGetState	Get the state (enable, disable, or unknown) of the specified TClk in the specified direction.
Perseus601X_TClkSetTxSource	Set the driver of the specified TClk in TX direction (from carrier board to MicroTCA backplane).
Perseus601X_TClkGetTxSource	Get the driver of the specified TClk in TX direction (from carrier board to MicroTCA backplane).
Perseus601X_FmcBidirClkSetConnection	Set the connection of the specified FMC bidirectional clock on the carrier board.
Perseus601X_FmcBidirClkGetConnection	Get the connection of the specified FMC bidirectional clock on the carrier board.
Perseus601X_SetOscState	Enable or disable the specified clock oscillator located on the carrier board.
Perseus601X_GetOscState	Get the state of the specified clock oscillator located on the carrier board.
Perseus601X_FclkaSetConnection	Set the connection of the FCLKA clock on the carrier board.
Perseus601X_FclkaGetConnection	Get the connection of the FCLKA clock on the carrier board.

Table 33 Perseus 601X driver functions

7.2 Perseus611X Driver

Driver Function	Use
Perseus611X_Open	Open an instance of the Perseus611X carrier object. Must be on a Perseus611X carrier board in order for this function to make sense.
Perseus611X_Close	De-allocate the Perseus611X carrier object.
Perseus611X_FmcAssignSlot	This function assigns to each connector of the Perseus611x carrier, the detected FMC modules.
Perseus611X_FmcSetVadj	This function sets the required Vadj voltage for the carrier board/connector.
Perseus611X_FmcGetVadj	This function verifies if the required Vadj voltage is compatible with what could be provided for a specific board slot. It returns the Vadj level setting needed for the board.
Perseus611X_FmcGetGA	Retrieve the Geographical Address (GA) of the specified FMC connector.
Perseus611X_GetSignalsInfo	This function gets the data pairs (LA, HA, HB) length (delay in ps) for the carrier. The delays are estimated value for material DK 4.5 (180 ps/po).
Perseus611X_GetModuleBaseAddr	Retrieve the base address corresponding to the specified module.
Perseus611X_GetModuleSize	Retrieve the memory space size of the specified module type.
Perseus611X_GetModuleMaxNumber	Retrieve the maximal number of instance of the specified module type the current carrier can handle. The retrieved instance number minus 1, is the maximal index value that can be used when calling the Carrier_GetModuleBaseAddr function.
Perseus611X_GetMemSpaceBaseAddr	Retrieve the base address corresponding to the specified memory space. A memory space is a range of memory address that is available to the user to connect its custom logic to the processor. Once the custom logic is mapped inside a memory space, its registers are reachable using the memory read and write functions.

Driver Function	Use
Perseus611X_GetMemSpaceSize	Retrieve the memory space size of the specified memory space type. A memory space is a range of memory address that is available to the user to connect its custom logic to the processor. Once the custom logic is mapped inside a memory space, its registers are reachable using the memory read and write functions.
Perseus611X_CustomRegisterWrite	Update the value of a custom register in the platform register FPGA core.
Perseus611X_CustomRegisterRead	Retrieve the value of a custom register in the platform register FPGA core.
Perseus611X_LedSetValue	Set the state of the carrier user LEDs.
Perseus611X_LedGetValue	Get the state of the carrier user LEDs.
Perseus611X_Reboot	Reboot the carrier platform. This will reset the FPGA and program it from the Flash memory content.
Perseus611X_I2cWrite	Perform an I2C Write transaction.
Perseus611X_I2cRead	Perform an I2C Read transaction.
Perseus611X_I2cReadNoWrData	Read data from a I2C device. This function does not send the register start address before reading from the I2C interface. It only read the incoming data. It will start reading from current internal device address and so no assumption could be made on the content read from what device address.
Perseus611X_I2cCheckRaw	Check if an i2c device is detected on the i2c bus
Perseus611X_TClkSetState	Set the state (enable or disable) of the specified TClk in the specified direction.
Perseus611X_TClkGetState	Get the state (enable, disable, or unknown) of the specified TClk in the specified direction.
Perseus611X_TClkSetTxSource	Set the driver of the specified TClk in TX direction (from carrier board to MicroTCA backplane).
Perseus611X_TClkGetTxSource	Get the driver of the specified TClk in TX direction (from carrier board to MicroTCA backplane).
Perseus611X_FmcBidirClkSetConnection	Set the connection of the specified FMC bidirectional clock on the carrier board.
Perseus611X_FmcBidirClkGetConnection	Get the connection of the specified FMC bidirectional clock on the carrier board.
Perseus611X_SetOscState	Enable or disable the specified clock oscillator located on the carrier board.
Perseus611X_GetOscState	Get the state of the specified clock oscillator located on the carrier board.
Perseus611X_FClkASetConnection	Set the connection of the FCLKA clock on the carrier board.
Perseus611X_FClkAGetConnection	Get the connection of the FCLKA clock on the carrier board.

Table 34 Perseus 611X driver functions

7.3 Carrier Driver

Driver Function	Use
Carrier_FmcDiscover	Scan the system and assign detected EEPROM modules.
Carrier_FmcGetInfo	Get information about a specific board in a connector. The type of information will provide a specific answer for that type.
Carrier_FmcValidatePresence	Allow to validate if a module type is detected in the specified connector/board. If a module is identified correctly, the detection operation will be rejected if it is called for a different type of modules than what is in the connector/board. If the module is not identified (or an error is found) in the connector/board and the specified modlst has module configured for compatibility mode, it will allow the request for the operation if the FPGA core is matching.

Driver Function	Use
Carrier_FmcPowerUp	Allow to configure V_{ADJ} voltage according to what is needed for a module. According to module type operating mode, the V_{ADJ} Value provided is used or ignored, but only possibly used for module operating in compatibility mode. If a module is identified correctly and operating in safe mode, the power operation will be rejected if it is called for a different type of modules than what is in the connector/board. If the module is not identified in the connector/board and the specified modlst has module configured for compatibility mode, it will allow the request with the specified V_{ADJ} Value. If the module in the connector/board is identified but running in compatibility mode, the V_{ADJ} function will be allowed with the specified Value if the specified modlst has some modules defined in compatibility mode.
Carrier_GetSignalsInfo	This will calculates the data lanes calibration information in delays (ps) for the specified module.
Carrier_Close	Deallocate the carrier object.
Carrier_FmcGetGA	Retrieve the Geographical Address (GA) of the specified FMC connector.
Carrier_GetModuleBaseAddr	Retrieve the base address corresponding to the specified module.
Carrier_GetModuleSize	Retrieve the memory space size of the specified module type.
Carrier_GetModuleMaxNumber	Retrieve the maximal number of instance of the specified module type the current carrier can handle. The retrieved instance number minus 1, is the maximal index value that can be used when calling the Carrier_GetModuleBaseAddr function.
Carrier_GetMemSpaceBaseAddr	Retrieve the base address corresponding to the specified memory space. A memory space is a range of memory address that is available to the user to connect its custom logic to the processor. Once the custom logic is mapped inside a memory space, its registers are reachable using the memory read and write functions.
Carrier_GetMemSpaceSize	Retrieve the memory space size of the specified memory space type. A memory space is a range of memory address that is available to the user to connect its custom logic to the processor. Once the custom logic is mapped inside a memory space, its registers are reachable using the memory read and write functions.
Carrier_CustomRegisterWrite	Update the value of a custom register in the platform register FPGA core.
Carrier_CustomRegisterRead	Retrieve the value of a custom register in the platform register FPGA core.
Carrier_LedSetValue	Set the state of the carrier user LEDs.
Carrier_LedGetValue	Get the state of the carrier user LEDs.
Carrier_Reboot	Reboot the carrier platform. This will reset the FPGA and program it from the Flash memory content.
Carrier_I2cWrite	Perform an I2C Write transaction.
Carrier_I2cRead	Perform an I2C Read transaction.
Carrier_I2cReadNoWrData	Read data from a I2C device. This function does not send the register start address before reading from the I2C interface. It only read the incoming data. It will start reading from current internal device address and so no assumption could be made on the content read from what device address.
Carrier_I2cCheckRaw	Check if an i2c device is detected on the i2c bus
Carrier_TClkSetState	Set the state (enable or disable) of the specified TClk in the specified direction.
Carrier_TClkGetState	Get the state (enable, disable, or unknown) of the specified TClk in the specified direction.
Carrier_TClkSetTxSource	Set the driver of the specified TClk in TX direction (from carrier board to MicroTCA backplane).
Carrier_TClkGetTxSource	Get the driver of the specified TClk in TX direction (from carrier board to MicroTCA backplane).
Carrier_FmcBidirClkSetConnection	Set the connection of the specified FMC bidirectional clock on the carrier board.
Carrier_FmcBidirClkGetConnection	Get the connection of the specified FMC bidirectional clock on the carrier board.
Carrier_SetOscState	Enable or disable the specified clock oscillator located on the carrier board.
Carrier_GetOscState	Get the state of the specified clock oscillator located on the carrier board.

Driver Function	Use
Carrier_FClkASetConnection	Set the connection of the FCLKA clock on the carrier board.
Carrier_FClkAGetConnection	Get the connection of the FCLKA clock on the carrier board.

Table 35 Carrier driver functions

7.4 Carrier EAPI Library

EAPI Function	Use
Carrier_Reboot_send	Reboot the carrier platform.
Carrier_GetType_send	Get the type of the carrier.
Carrier_GetModuleBaseAddr_send	Retrieve the base address corresponding to the specified module.
Carrier_LedSetValue_send	Set the state of the carrier user LEDs.
Carrier_LedGetValue_send	Get the state of the carrier user LEDs.
Carrier_TClkSetState_send	Set the state (enable or disable) of the specified TClk in the specified direction.
Carrier_TClkGetState_send	Get the state (enable, disable, or unknown) of the specified TClk in the specified direction.
Carrier_TClkSetTxSource_send	Set the driver of the specified TClk in TX direction (from carrier board to MicroTCA backplane).
Carrier_TClkGetTxSource_send	Get the driver of the specified TClk in TX direction (from carrier board to MicroTCA backplane).
Carrier_FmcBidirClkSetConnection_send	Set the connection of the specified FMC bidirectional clock on the carrier board.
Carrier_FmcBidirClkGetConnection_send	Get the connection of the specified FMC bidirectional clock on the carrier board.
Carrier_SetOscState_send	Enable or disable the specified clock oscillator located on the carrier board.
Carrier_GetOscState_send	Get the state of the specified clock oscillator located on the carrier board.
Carrier_FClkASetConnection_send	Set the connection of the FCLKA clock on the carrier board.
Carrier_FClkAGetConnection_send	Get the connection of the FCLKA clock on the carrier board.
Carrier_I2cRead_send	Perform an I2C Read transaction.
Carrier_I2cWrite_send	Perform an I2C Write transaction.
Carrier_I2cReadNoWrData_send	Read data from a I2C device. This function does not send the register start address before reading from the I2C interface. It only read the incoming data. It will start reading from current internal device address and so no assumption could be made on the content read from what device address.
custom_register_read_send	Reads the value of the specified custom register.
custom_register_write_send	Write the value to the specified custom register.
i2c_scan_devices_send	Scan the specified i2c bus to detect available devices. This function is used for production purpose to check if all devices are detected correctly on the i2c bus.

Table 36 Carrier EAPI functions

7.5 Carrier Command Line Interface (CLI) Commands

CLI Function	Use
custom_register_read	Read 32 bits of data from a custom register ID.
custom_register_write	Write 32 bits of data to a custom register ID.
reboot	Rebooting Perseus. Restart CLI
fmc_info	Print information about a carrier FMC daughter card
i2c_bus_scan	Scan the specified i2c bus to detect all available devices.

Table 37 Carrier CLI function