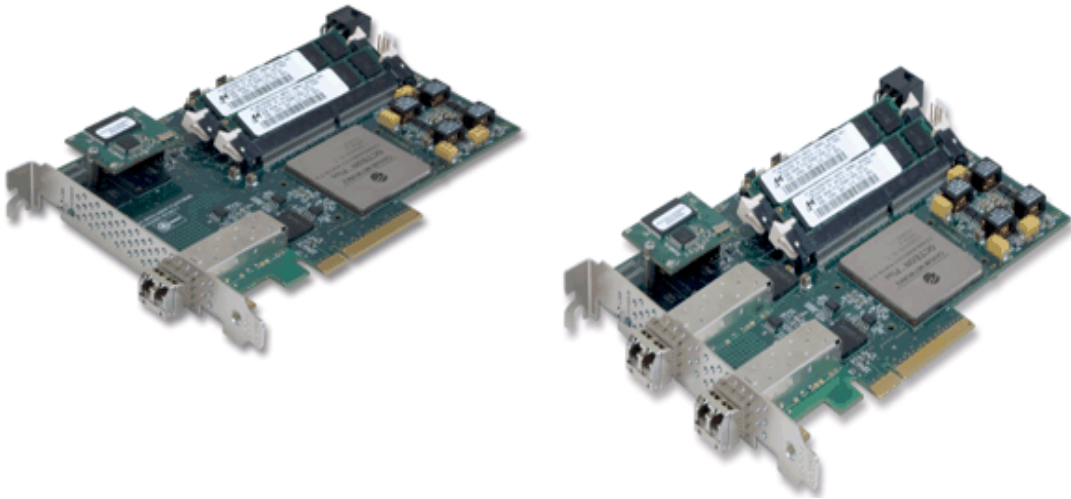


Reference Manual

WANic*-5651x Packet Processor PCI Express 10-Gigabit Ethernet Third Edition

Part No: 87002098-820



imagination at work

Table of Contents

Preface	11
1 • Overview	15
1.1 Features	15
1.2 Product Overview	16
1.3 Hardware Overview	16
1.3.1 Software Overview	17
1.3.2 Additional Software	17
1.3.3 Operating System Support	17
1.3.4 Cavium Networks Users' Organization Web Site	17
1.4 Specification	18
1.5 Emission Compliance Notice	20
1.6 Additional Important Notices	21
1.7 Warranty and Repair	22
1.7.1 Warranty	22
1.7.2 Customer Technical Support	22
2 • Hardware Description	23
2.1 Features	23
2.2 Multi-Core Processor	24
2.2.1 Reference Clock	25
2.2.2 Boot Bus	25
2.2.3 QLM Packet Interfaces	26
2.2.4 Processor Interfaces	27
2.2.5 TWSI Interface	27
2.2.6 GPIO Interface	28
2.2.7 RS-232 Serial Port Interfaces	29
2.2.8 On-Board Power Supply Interface	29
2.2.9 Clocking	30
2.3 PCI Express Connector	32
2.4 FPGA	33
2.4.1 Control and Status Registers	34
2.4.2 Persistent Memory Interface	34
2.4.3 FPGA Clocking	34
2.4.4 FPGA Interrupts	34
2.4.5 Reset Control	34
2.4.6 Power Supply Control	34
2.4.7 I2C Interface	35
2.4.8 TWSI Interface	35

2.5	Memory	36
2.5.1	Internal (Level 2) Cache Memory	36
2.5.2	DDR2 SDRAM	36
2.5.3	Boot Flash	37
2.5.4	Persistent Memory	37
2.5.5	Serial EEPROM	37
2.5.6	USB Flash Drive	38
2.6	Gigabit Ethernet Physical Device	39
2.7	Headers	40
2.7.1	J5 Header	40
2.7.2	J6 Header and JTAG Scan Chain	42
2.7.3	Enhanced JTAG Header (EJTAG)	43
2.8	Dual In-Line Package (DIP) Switch	44
2.9	Front Panel Connectors and LEDs	46
2.9.1	SFP+ Connectors	46
2.9.2	Front Panel LEDs	47
2.10	Temperature Sensor	48
2.10.1	Thermal Management	48
2.11	Power	50
3	• FPGA Registers	53
3.1	FPGA Register Descriptions	53
4	• Software Description	71
4.1	Software Overview	71
4.2	Cavium Software Development Kit	72
4.2.1	Cavium Ethernet Driver	72
4.2.2	Embedded File System	73
4.2.3	Simple Executive Library	74
4.3	User Application	74
4.4	U-Boot BootLoader	75
4.4.1	Boot Process	75
4.4.2	Building U-Boot	77
4.4.3	U-Boot Commands	78
4.4.4	Environment Variables	81
4.4.5	PCI Console Redirection – U-Boot	82
4.4.6	PCI Console Redirection – Linux	82
4.4.7	U-Boot Scripting	84
4.4.8	IP Configuration Acquisition with DHCP	84
4.4.9	Automated UA Executable Load and Boot	84
4.4.10	EEPROM	86
4.4.11	EEPROM Tuple Update and Enumeration	99

4.5 Linux Support Package (LSP)	101
4.5.1 Flash Driver	102
4.5.2 IOCTL Device Interface	106
4.5.3 IOCTL Operations for U-Boot Tuple and Environment	113
4.5.4 IOCTL Register Access Operations	114
4.5.5 IOCTL POST Error Operations	115
4.5.6 IOCTL Port Management	116
4.5.7 NPA Driver	117
4.6 Additional LSP Features	118
4.7 Examples	119
4.7.1 Linux Applications Examples	119
4.7.2 Simple Exec Application Examples	120
5 • LSP Applications	123
5.1 Diagnostic Utility	123
5.1.1 Setup	125
5.1.2 Running the Diagnostic Utility	126
5.2 In-Service Daemon	141
5.2.1 Running the In-Service Daemon	141
A • Hardware Installation and Removal Procedures	143
A.1 Precautions	143
A.2 Overview	143
A.3 Mini-RDIMMs Installation and Removal	144
A.3.1 Mini-RDIMM Installation	144
A.3.2 Mini-RDIMM Removal	145
A.4 WANic-5651x Installation and Removal	146
A.4.1 WANic-5651x Installation	146
A.4.2 WANic-5651x Removal	149
A.5 SFP+ Modules Installation and Removal	150
A.5.1 SFP+ Module Installation	151
A.5.2 SFP+ Module Removal	152
B • GNU General Public License	155
C • GE INTELLIGENT PLATFORMS EMBEDDED SYSTEMS, INC. Software License Agreement – Object Code	161
D • GE Intelligent Platforms Inc. Software License Description	167

List of Figures

Figure 2-1 WANic-56511 Block Diagram	23
Figure 2-2 WANic-56512 Block Diagram	23
Figure 2-3 QLM Grouping	26
Figure 2-4 GPIO Interface Assignment	28
Figure 2-5 WANic-56511 Clocking	30
Figure 2-6 WANic-56512 Clocking	31
Figure 2-7 FPGA Function Block Diagram	33
Figure 2-8 Mini-RDIMMs on WANic-5651x	36
Figure 2-9 USB Flash Drive Header	38
Figure 2-10 USB Flash Header Pinout	38
Figure 2-11 10G PHY Implementation	39
Figure 2-12 Header Locations	40
Figure 2-13 SDA Cabling Example	41
Figure 2-14 JTAG Scan Chain	43
Figure 2-15 S1 DIP Switch Location	44
Figure 2-16 Front Panel Connectors and LEDs	46
Figure 2-17 External Sensor	48
Figure 2-18 WANic-5651x Air Flow Path	49
Figure 2-19 WANic-5651x External Power Connector	50
Figure 3-1 FPGA Revision Register @ Base + 00h	54
Figure 3-2 Board ID and DIP Switch Register @ Base + 01h	55
Figure 3-3 LED Control Register @ Base + 02h	56
Figure 3-4 Interrupt Control Register @ Base + 03h	57
Figure 3-5 Interrupt Status Register @ Base + 04h	58
Figure 3-6 Peripheral Reset Register @ Base + 05h	59
Figure 3-7 Transmit Control Register @ Base + 06h	60
Figure 3-8 Temperature Register @ Base + 07h	61
Figure 3-9 I2C Control Register @ Base + Device Address	64
Figure 3-10 I2C Address Low Register @ Base + Low Address	65
Figure 3-11 I2C Address High Register @ Base + High Address	65
Figure 3-12 I2C Data Register @ Base + Device Address	66
Figure 3-13 Boot Flash Upper Address Control Register @ Base + 20h	67
Figure 3-14 Miscellaneous Functions Register @ Base + 21h	68
Figure 3-15 PHY Interrupt Register @ Base + 22h	69
Figure 4-1 Software Components	72
Figure 4-2 S1 DIP Switch Location	77
Figure 4-3 EEPROM Mapping	86
Figure 4-4 Flash Mapping	102
Figure 4-5 IOCTL Commands and Structures	106

Figure 4-6 IOCTL Command Types	109
Figure 4-7 Exported Kernel Symbols.	112
Figure 4-8 NPA Driver in LSP	117
Figure 5-1 Command Line Options	124
Figure 5-2 OCTEON NPA Major Number	125
Figure 5-3 Diagnostic Utility Main Menu - PCIe Host Mode	126
Figure 5-4 Diagnostic Utility Main Menu - PCIe Target Mode	126
Figure 5-5 View Test Results Display	128
Figure 5-6 EEPROM Menu	129
Figure 5-7 Run Auto Test	130
Figure 5-8 Test Configuration with Cable Attachment	131
Figure 5-9 Auto Test Configuration Menu	131
Figure 5-10 Booting U-BOOT over the PCI	133
Figure 5-11 Loading an Application over the PCI	133
Figure 5-12 Running Commands over the PCI	133
Figure 5-13 Diagnostic LED Location	134
Figure 5-14 Select Current Card	135
Figure 5-15 Flash Menu	136
Figure 5-16 Log File Location	137
Figure 5-17 Show Core Temperature	138
Figure 5-18 Memory Test Results	139
Figure 5-19 Memory Test with Command Line Parameter -a	140
Figure 5-20 Error Message Display	141
Figure A-1 Mini-RDIMM Installation	144
Figure A-2 Installation in a PCI Bus Chassis	147
Figure A-3 Connecting to the Power Supply	148
Figure A-4 SFP+ Module with Clasp	152
Figure A-5 Inserting or Removing an SFP+ Module.	153

Table 1-1 WANic-5651x Hardware Specification	18
Table 2-1 Chip Select Space	25
Table 2-2 TWSI Device Information	27
Table 2-3 GPIO Interface Bits	29
Table 2-4 PCIe Data Rates	32
Table 2-5 FPGA TWSI Devices and Addresses	35
Table 2-6 JTAG Mode S1 DIP Switch Configuration	45
Table 2-7 Boot Mode S1 DIP Switch Configuration	45
Table 2-8 Diagnostic Mode S1 DIP Switch Configuration	45
Table 2-9 Module Status LEDs	47
Table 2-10 Link Status LEDs	47
Table 2-11 WANic-5651x Cooling	49
Table 2-12 Current and Power Limits	51
Table 3-1 FPGA Memory Mapped Registers	53
Table 3-2 Build Configuration Bit Values	55
Table 3-3 Diagnostic LEDs on WANic-5651x	56
Table 4-1 POST Failures	76
Table 4-2 U-Boot Commands	78
Table 4-3 Environment Variables	81
Table 4-4 Load and boot Image Entries	85
Table 4-5 EEPROM Tuples	87
Table 4-6 Flash Driver Functions	103
Table 4-7 Flash Partitioning	104
Table 4-8 IOCTL Commands	107
Table 4-9 Tuple/U-Boot Environment IOCTL Operations	113
Table 4-10 TWSI_OP_t Device IDs:	114
Table 4-11 POST Error Operations	115
Table 4-12 ETHOOL IOCTL Operations	116
Table 4-13 Linux IOCTL Operations	116
Table 5-1 Command Line Parameters	124
Table 5-2 Diagnostic Utility Main Menu	127
Table 5-3 EEPROM Menu Options	129
Table 5-4 Test Configuration Menu Options	132
Table 5-5 Diagnostic LED Test Prompts	134
Table 5-6 Flash Menu Options	136

Preface

This document provides technical information for the WANic-56511 and WANic-56512 Packet Processors. The WANic-56511 and WANic-56512 Packet Processors (hereafter referred to as the WANic-5651x) are PCI Express, standard height, half-length packet processing cards, which comply with the PCI Express 1.1 specification. The WANic-5651x provide 10 Gigabit Ethernet I/O through front panel Small Form Pluggable Plus (SFP+) connectors with either a 4- or 8-lane (x4 or x8) PCI Express control plane connection.

Purpose

The purpose of this manual is to describe the WANic-5651x and the services the card provides. This manual includes a description of the WANic-5651x hardware and firmware, as well as information for using the accompanying Linux device drivers.

Audience

This manual is intended for the user who creates and develops applications for the WANic-5651x such as an engineer or developer.

It is assumed that the user is familiar with standard cabling, configurations of operating systems, networks, and PCI Express technology as well as C programming with the Linux Operating System.

Referenced Documentation

Documentation referenced in this manual includes the following industry and vendor documentation:

Industry Standard Specifications

- *IEEE® 802.3–Standard for Information Technology–Telecommunications and information exchange between systems–Local and metropolitan area networks–Specific requirements–Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications*, Institute of Electrical and Electronics Engineers (IEEE), www.ieee.org.
- *PCI Express® Base Specification R1.1*, PCI Special Interest Group (PCI-SIG®), www.pcisig.com.
- *PCI Express® Card ElectroMechanical Specification R1.1*, PCI-SIG, www.pcisig.com.
- *SFF-8431 Specifications for Enhanced Small Form Factor Pluggable Module (SFP+)*, Multi-Source Agreement 2008 <http://www.sffcommittee.org/ie/sffspec.html>.

Vendor Documentation

- *OCTEON™ Plus CN54/5/6/7xx Hardware Reference Manual*, 2008, Cavium® Networks, Inc.
- *QT2025™ PxKD Serial 10Gbps-to-XAUI Transceiver with Adapter EDC Data Sheet*, 2008, Applied Micro Circuits Corporation®.

Manual Organization

This manual contains the following chapters and appendices:

- ***“Chapter 1: Overview,”*** presents an overview of WANic-5651x hardware and software as well as compliance, warranty, and Customer Technical Support information.
- ***“Chapter 2: Hardware Description,”*** describes the hardware components on the WANic-5651x. This chapter also includes a description of connectors, pinouts, and Light Emitting Diodes (LEDs).
- ***“Chapter 3: FPGA Registers,”*** describes the registers to use for communications between components on the WANic-5651x.
- ***“Chapter 4: Software Description,”*** provides information on how to use software components including a bootloader, Application Programming Interface (API), and drivers.
- ***“Chapter 5: LSP Applications,”*** describes the Diagnostic Utility and the In-Service Daemon. The Diagnostic Utility verifies and configures components on the WANic-5651x. The In-Service Daemon monitors hardware continuously and reports detected faults.
- ***“Appendix A • Hardware Installation and Removal Procedures,”*** describes the procedures to install and to hot swap the WANic-5651x.
- ***“Appendix B • GNU General Public License,”*** contains a copy of the GNU General Public License.
- ***“Appendix C • GE INTELLIGENT PLATFORMS EMBEDDED SYSTEMS, INC. and its subsidiaries COMPUTER DYNAMICS OF ILLINOIS, INC. Software License Agreement – Object Code,”*** contains a copy of the Source License Agreement for the WANic-5651x object code.
- ***“Appendix D • GE Intelligent Platforms Embedded Systems, Inc. Software License Description,”*** identifies the applicable software licenses for select WANic-5651x components.

This manual also contains a glossary and an index.

Notation Conventions

This manual uses the following notation conventions:

- *Italics* emphasizes words in text, register field names, and documentation or chapter titles.
- Hexadecimal values are represented as digits followed by “h”, for example 0Ch.
- Courier text identifies text that the user must enter or text that displays on the screen. For example,
“Use the `ioctl DevShow` command to verify the driver installation.”
- **Bold Palatino Linotype text** identifies register names. For example,
“An **Interrupt Control Register** is available for each I/O bus space.”
- **Bold Courier text** identifies text in an example that the user must enter. For example,
“Load the driver manually by typing the `modprobe` command:
`modprobe -v xxxx`”
- Notes, cautions, and warnings call attention to essential information:



NOTE

A Note calls attention to important information, such as tips and advice.



CAUTION

A Caution alerts you to conditions that could damage a device, system, or data.



WARNING

A Warning calls attention to actions that can cause risk of personal injury.

- Specific term definitions, as applied in this manual include:
 - “May” means that there is flexibility that does not affect the outcome or result of the action.
 - “Should” means that the user has flexibility but it is strongly recommended to perform the specified action to achieve an optimal outcome or result.
 - “Must” means that there is no flexibility and the user is required to perform the action to achieve an optimal outcome or result.

1 • Overview

This document describes the WANic-56511 and WANic-56512 Packet Processors (hereafter referred to as the WANic-5651x.) This chapter provides an overview of the WANic-5651x hardware and software.



NOTE

Unless specified otherwise, the information in this document applies to all models of the WANic-5651x Packet Processors.

The WANic-5651x are cost-effective PCI Express cards that comply with the PCI Express 1.1 specifications. The WANic-5651x family of packet processors was designed for optimized implementation in a PCI Express-compatible chassis. PCI Express provides an ideal mix of robustness and cost effectiveness for the types of traffic found in communications applications. The tight integration with PCI Express and proprietary architecture provides Telecommunications Equipment Manufacturers (TEMs) and Original Equipment Manufacturers (OEMs) with high-performance processing power and extensive Input/Output (I/O) operations, enabling quick time-to-market application deployment.

1.1 Features

The WANic-5651x feature the following components in select configurations:

- Cavium® OCTEON™ Plus Multi-Core Packet Processor with 12 cores at speeds of 750 MegaHertz (MHz)
- 4 or 8 lanes of PCI Express to the host interface
- Up to 4 GigaByte (GB) Double Data Rate Type 2 (DDR2) in two Mini-Registered Dual In-Line Module Memory (Mini-RDIMM) packet memory devices
- 128 MegaBytes (MB) Flash for booting
- 2 GB Flash (Universal Serial Bus) disk for steady state storage
- 32 MB Persistent Memory
- Front panel Light Emitting Diodes (LEDs) for module and link status
- Debug headers via the optional Serial Debug Adapter:
 - RS-232 serial interface
 - On-board (JTAG) tool debugging
 - External (EJTAG) tool debugging
- Up to two 10 Gigabit (G) Ethernet Small Form Pluggable Plus (SFP+) transceivers on the front panel
- Linux 2.6.x Operating System support

1.2 Product Overview

The WANic-5651x are fully-integrated, packet processors that provide flexible processing capabilities for development of telecommunications and other data processing intensive applications. The WANic-5651x are PCI Express, standard height, half-length modules that offers a 10 Gigabit Ethernet external-networking front panel I/O interfaces. The WANic-5651x provide dedicated I/O by combining on-board 10 Gigabit Ethernet with PCI Express to provide high-performance network connectivity for the platform with PCI Express slots.

The WANic-5651x features a Cavium OCTEON Plus Network Multi-Core Processor with up to 12 cnMIPS™¹ cores at speeds of 750 MHz. Powerful DDR2 SDRAM with Error Correction Code (ECC) provides up to 4 GB of packet memory.

WANic-5651x is available with factory-installed front panel optical SFP+ ports of 10 Gigabits per second Ethernet (GbE) communications.

The WANic-5651x installs seamlessly under popular Linux distributions.

1.3 Hardware Overview

The WANic-5651x contain a Multi-Core Processor that runs at 750 MHz. The WANic-5651x are available with 12 cnMIPS cores.

The WANic-5651x contains up to 4 GB of DDR2 SDRAM that can provide memory data transfer speeds up to 667 MHz depending on the model. The WANic-5651x support two Very Low Profile (VLP) single- or dual-rank Mini-RDIMMs in angled-memory sockets.

Flash ROM provides 128 MB of memory for boot and application code storage. An additional non-volatile Flash disk contains 2 GB of storage and is available in configurations up to 4GB². On-board Persistent Memory provides 32 MB of storage to preserve the contents of the WANic-5651x during a reset.

The WANic-5651x is available in a variety of factory-installed configurations and offers front panel 10G Ethernet SFP+ I/O ports. The front panel GbE ports implement 10GBase-SX/LX interfaces, which are defined by the IEEE 802.3 (Gigabit Ethernet) specification suite. In addition, the SFP+ I/O ports support Direct Attach Mode (DAM) as defined by the Multi-Source Agreement Group in the SFF-8431 specification.



NOTE

Consult with your local GE Intelligent Platforms Customer Technical Support Representative for configuration information and options.

1. cnMIPS is Cavium Networks' implementation of MIPS64 with enhancements and additional built-in features. MIPS64 is a product to MIPS Technology, Inc.
2. Contact GE Intelligent Platforms Customer Technical Support for more information.

1.3.1 Software Overview

The WANic-5651x Software Developer's Kit (SDK) provides an easy-to-use development interface for the Linux Operating System. The WANic-5651x provides boot-up services, diagnostics, and device drivers, as well as a customized Application Programming Interface (API) for select on-board devices.

WANic-5651x software on the CD-ROM consists of the following components as described in "*Chapter 4: Software Description.*"

- U-Boot Bootloader – loads and boots the WANic-5651x firmware.
- Linux Support Package (LSP) – provides a suite of functions to access and to use the WANic-5651x hardware.
- Linux Operating System – contains the Debian™ distribution with OCTEON Multi-Core Processor support.
- Diagnostic Utility – configures and exercises various aspects of the WANic-5651x hardware.

Appendix B: GNU General Public License contains a copy of the *GNU General Public License*, version 2, for using and applying U-Boot, LSP, and the Cavium Software Developer's Kit.

1.3.2 Additional Software

For your convenience, the WANic-5651x CD-ROM also contains the following additional software:

- GNU Tool Chain – includes tools for building executables for the cnMIPS cores.
- GNU Debugger – is a standard debugger for the GNU software that helps diagnose a program that is executing.

Appendix B: GNU General Public License contains a copy of the *GNU General Public License*, version 2, for using and applying these tools.

1.3.3 Operating System Support

The WANic-5651x supports the following operating systems:

- Linux Kernel 2.6.x Operating System, which is included in the software distribution.
- OCTEON Simple Executive or Generic Operating System, which the user must obtain from Cavium Networks, Inc. (www.caviumnetworks.com)

1.3.4 Cavium Networks Users' Organization Web Site

The Cavium Networks Users' Organization website, www.cnusers.org, provides a platform for expanding and supporting the user community for the OCTEON Multi-Core Processor (cnMIPS) family. This website provide a means for sharing software, such as Linux-based software, on the OCTEON Multi-Core Processor. This site includes data plane functions, drivers, and Simple Executive applications.



NOTE

Building applications that may use Octeon functions require the Cavium Simple Executive (CVMX) libraries, which are provided in the Cavium Networks OCTEON SDK.

Refer to www.cnusers.org, for more information on specific applications.

1.4 Specification

Table 1-1 lists the hardware specifications for the released configurations of the WANic-5651x.



NOTE

This product is in compliance with the European Union's legislation 2002/95/EC, Restrictions of Hazardous Substances (RoHS) directive and similar regulations that may be adopted by other countries.



CAUTION

Maximum power requirements are different for each model and configuration. Consult your Customer Technical Support representative for model specific power requirements.

Table 1-1 WANic-5651x Hardware Specification

Description	WANic-56512	WANic-56511
Form Factor	PCI-Express-compliant, Standard-height half-length	
Weight	with Mini-RDIMMs and SFP+ = 0.582 lbs (0.3Kg) with Mini-RDIMMs but without any SFP+ = 0.474 lbs (0.2Kg)	
Slot Type	Requires four lane PCI Express slot	Requires eight lane PCI Express slot
Bus Type	PCI Express 1.1 compliant	
I/O Configuration	10GBase-SX/LX (Single or Multi-Mode Fiber), supports SFP+ Direct Attach Mode	
Connectors	Two SFP+, PCIe Graphics Extension Power	One SFP+, PCIe Graphics Extension Power
Processor Core Speed	750 MHz	
Core Processors	12	
Flash ROM	128 MB	
Flash Disk	2 GB	
Persistent Memory	32 MB	
DDR2 SDRAM	4 GB	
Supports two registered Mini-RDIMMs	667 MHz	
EEPROM	64 KB	
Power Requirements		
12 cnMIPS Cores	See Table 2-12 in Chapter 2.	
Additional Configurations	Contact GE Intelligent Platforms Customer Technical Support	
Mean Time Between Failure		
Main Card only	523,896 hours	581,026 hours
Compliance		
RoHS	6/6	
Emissions Class A	Australia – AS/NZS CISPR 22 Class A ITE Canada – ICES-003 Issue 4 Class A Europe – EN55022 Class A ITE Japan – VCCI Class A ITE USA – FCC 47 CFR Part 15 Class A	
Immunity	Europe – EN55024:ITE	

Description	WANic-56512	WANic-56511
Safety		Canada – CSA22.2 NO 60950-1 Europe – EN60950-1 USA – UL60950-1
Flammability		UL94V0
Environmental Requirements		
Operating Temperature		0° to +55°C (+32° to +131° F)
Storage Temperature		-40° to +85°C (-40° to +185° F)
Relative Humidity		5% to 90% (non-condensing)
Cooling		
Module with Heat sink Assembly		
[Minimum volumetric flow rate at a maximum ambient temperature (Celsius) in Linear Feet per Minute (LFM)]	175 LFM @ 25° C 285 LFM @ 40° C 495 LFM @ 55° C	205 LFM @ 25° C 305 LFM @ 40° C 505 LFM @ 55° C

1.5 Emission Compliance Notice

This equipment complies with the following international and North American emission requirements:

- **Australia and New Zealand — AS/NZS 3548/CISPR 22 Class A ITE**

- **Canada — ICES-003**

This Class A digital apparatus complies with Canadian ICES-003.
(Cet appareil numérique de la class A est conforme a la norme NMB-003 du Canada.)

- **Europe — EN55022**



WARNING

This is a Class A product. In a domestic environment, these boards may cause radio interference in which case the user may be required to take adequate measures.

- **Japan — VCCI Class A ITE**

This is a Class A product based on the standard of the Voluntary Control Council for Interference from Information Technology Equipment (VCCI). If this is used near a radio or television receiver in a domestic environment, it may cause radio interference. Install and use the equipment according to the instruction manual.

- **USA — FCC Part 15 Class A**



NOTE

The hardware has been tested and found to comply with the limits for a Class A digital device pursuant to Part 15 of the FCC rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction in this documentation, may cause harmful interference to radio communications. Operation of the equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense. Changes or modification not expressly approved by the manufacture could void the user's FCC granted authority to operate the equipment.

1.6 Additional Important Notices

- For optical/laser devices, read the following notes, cautions, and warnings:



WARNING

For this product, use only the Class 1 laser devices that have the following approvals:

- FDA 21 CFR 1040.10
- IEC 60825-1



NOTE

Protect optical SFP modules by inserting clean dust plugs into the SFP modules after the cables are extracted from them. Be sure to clean the optic surfaces of the fiber cables before plugging the dust plugs back into the optical bores of another SFP module. Avoid getting dust and other contaminants into the optical bores of your SFP modules: The optics will not work correctly when obstructed with dust.



CAUTION

It is important to disconnect or remove all cables before removing or installing an optical SFP transceiver. Failure to do so may result in damage to the cable or SFP device.



WARNING

An optical SFP transmitter is a Class 1 device. Invisible laser radiation may be emitted from disconnected fibers or connectors. Do not stare into beams or view directly with optical instruments as this may permanently damage your eyes.

Do not leave an optical SFP transceiver uncovered except when inserting or removing a cable. The safety/dust plugs keep the port clean and prevent accidental exposure to laser light.

1.7 Warranty and Repair

GE Intelligent Platforms, Inc. provides a comprehensive web site on the World Wide Web (<http://www.ge-ip.com>.) This web site contains up-to-date information including current and new products, such as the WANic family of packet processors. The web site also contains sales office locations, copyrights, trademarks, press releases, warranties, and technical support information.

1.7.1 Warranty

Warranty information is described on the GE Intelligent Platforms Embedded Systems web site at <http://www.ge-ip.com/support/embeddedsupport/warranty>. This site provides current product warranty and repair services as well as information on out-of-warranty services. For additional information on specific product warranty, contact your local support or sales representative.

1.7.2 Customer Technical Support

GE Intelligent Platforms' dedicated team of Customer Technical Support Engineers is committed to providing quality support to all their customers. Customer Technical Support Engineers are trained to assist GE Intelligent Platforms customers in the development, integration, and use of GE Intelligent Platforms products in customer applications, systems, and products to facilitate timely product development. The Customer Technical Support Service Center is staffed weekdays (except holidays) between the hours of 8:00 AM and 5:00 PM Eastern Standard Time (CST).

Use one of the following methods to contact technical support:

Email:	support.embeddedsystems.ip@ge.com
Telephone:	1-800-433-2682
Address:	GE Intelligent Platforms 12090 South Memorial Parkway Huntsville, AL 35803-3308
Hours:	8:00 AM to 5:00 PM

Before contacting technical support, make sure you have the following information available:

- Model
- Purchase receipt
- Description of problem

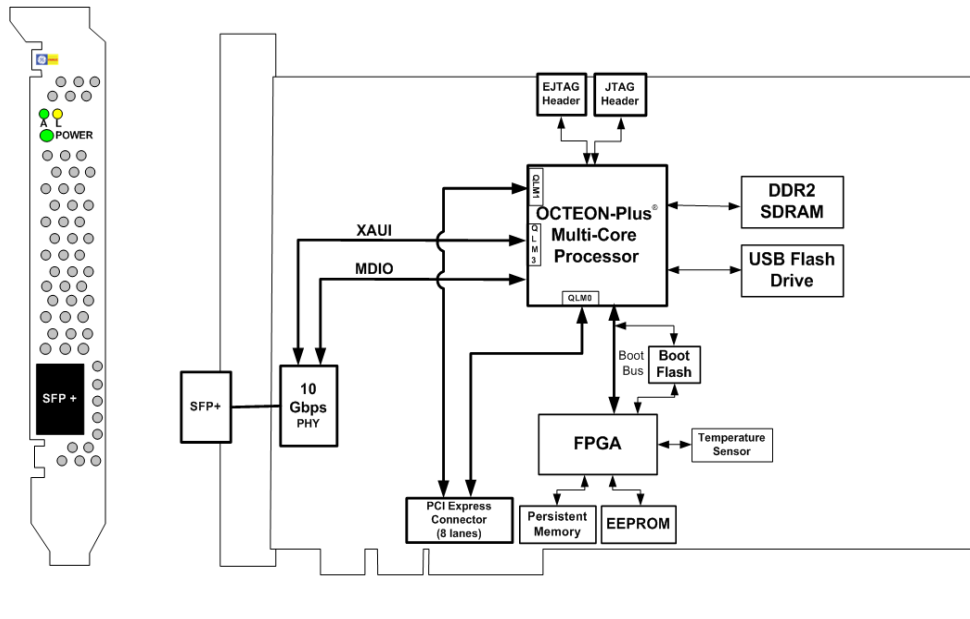
2 • Hardware Description

This chapter describes the featured hardware components on the WANic-5651x. It contains a simple block diagram of the featured components, which includes the Multi-Core Processor, memory, controllers, and external interfaces as well as connectors.

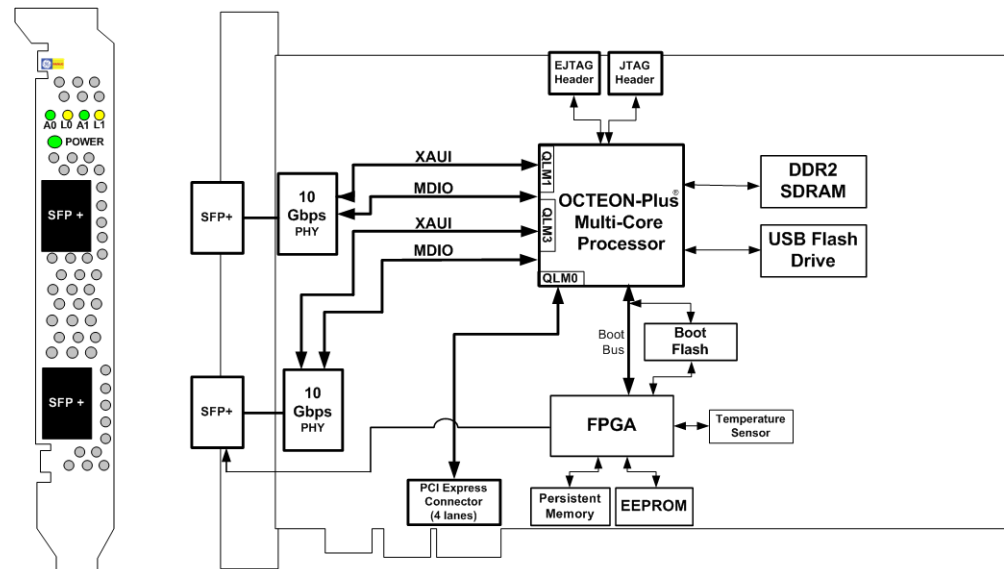
2.1 Features

The WANic-56511 features the hardware components shown in Figure 2-1. The WANic-56512 features the components shown in Figure 2-2.

WANic-56511 Figure 2-1 WANic-56511 Block Diagram



WANic-56512 Figure 2-2 WANic-56512 Block Diagram



Featured components for the WANic-5651x include the following:

- Multi-Core Processor with 12 cnMIPS cores
- x4 or x8 PCI Express
- Up to 4 GB DDR2 SDRAM via two VLP Mini-RDIMMs (two slots of 2 GB per slot)
- 128 MB Boot Flash
- 2 GB USB Flash Drive
- FPGA for access to on-board components
- 32 MB Persistent Memory
- EEPROM
- Temperature sensor
- LEDs for module status and power indications
- On-card headers provides access to:
 - OCTEON Joint Test Action Group (JTAG) debug port
 - External JTAG test/debug port
 - RS-232 console I/O port UART
- Front panel 10GbE SFP+ (SR/LR with Direct Attach Mode) XAUI to PHY

2.2 Multi-Core Processor

The WANic-5651x uses the Cavium OCTEON-Plus (CN56xx) family of packet processing Multi-Core Processors (hereafter referred to as the Processor). The Processor is a single-chip device for secure Open Systems Interconnection (OSI) Layer 2 to Layer 7 networking applications.

Each Processor option is configured at factory assembly time to provide comprehensive integrated functions.

The Processor is available with 12 cnMIPS cores with enhancements and additional built-in hardware acceleration for content and security processing. This architecture combines cnMIPS cores along with dedicated programmable coprocessor blocks capable of delivering up to 10 Gbps of application performance at 750 MHz chip clock rates on the WANic-5651x.

The Processor offers highly-flexible external networking interfaces. Interfaces on select models include a four (x4) or eight lane (x8) PCI Express (endpoint) interface and up to two 10GbE ports.

Within the WANic-5651x, the PCI Express (PCIe) interface provides connectivity to the host. Packets and control information can flow to/from the Processor using any of the XAUI (Roman numeral X, meaning ten – Attachment Unit Interface) or PCIe interfaces.

The Processor uses external power to run a dedicated on-board power supply. This permits adjustments to the core device supply voltage, as needed, without affecting other circuitry.

The Processor uses the latest technology to implement the following:

- Reference Clock
- Boot Bus
- Quad Lane Module (QLM) packet interface
- General Purpose Input/Output (GPIO) interface

2.2.1 Reference Clock

The Processor reference clock input is driven from a 50 MHz oscillator to drive the initial core clock frequency of 750 MHz. The multiplier is set by resistor straps on the board.

2.2.2 Boot Bus

The Processor Boot Bus (hereafter referred to as the Boot Bus) provides an 8-bit data path with:

- 8 Chip Selects
- Independent read and write
- 27 address lines (using big endian byte-ordering)

When power-on reset completes, Core 0 of the Processor uses the Boot Bus to obtain code from the Flash memory device that is controlled by Chip Select 0 (CS0). The boot Flash memory device connects directly to the Boot Bus using 8-bit multiplexed mode and is controlled by Chip Select 0.

FPGA registers are accessible by means of the Boot Bus using Chip Select 1 (CS1).

Chip Select 2 (CS2) provides the lower 16 MB address of Persistent Memory (PMEM).

Chip Select 3 (CS3) provides the upper 16 MB address of PMEM.

Table 2-1 identifies each Chip Select.

Table 2-1 Chip Select Space

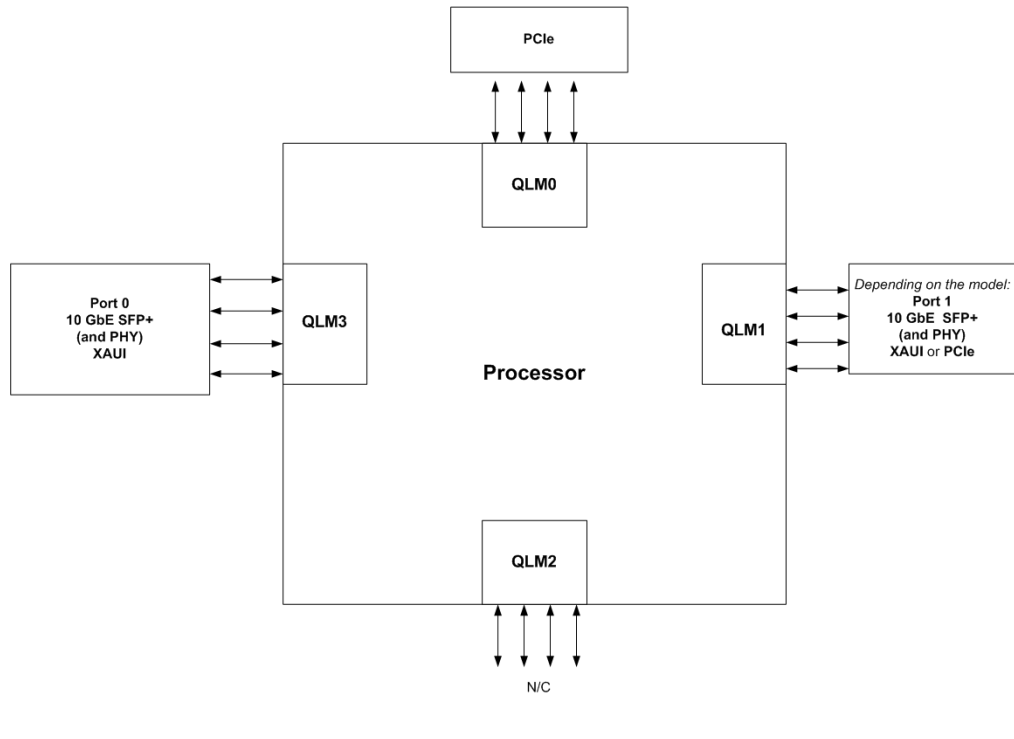
Chip Select	Address Range	Description
CS0	0x80010000_1FC00000 – 0x80010000_27BFFFFFF	Boot Flash
CS1	0x80010000_16000000 – 0x80010000_1600FFFF	FPGA
CS2	0x80010000_30000000 – 0x80010000_30FFFFFF	16 MB PMEM lower address space
CS3	0x80010000_31000000 – 0x80010000_31FFFFFF	16 MB PMEM upper address space

2.2.3 QLM Packet Interfaces

The Processor contains four Quad-Lane Modules (QLMs) for SerDes communications (for a total of 16 SerDes lanes) as shown in Figure 2-3. The WANic-5651x groups these QLMs on the Processor as follows:

- QLM0 – Four lanes of PCIe
- QLM1 – Four additional lanes of PCIe, or XAUI, dedicated to the PHY and front panel 10GbE SFP+
- QLM2 – Not connected (NC)
- QLM3 – XAUI dedicated to the PHY and front panel 10GbE SFP+

Figure 2-3 QLM Grouping



QLM0 QLM0 is a dedicated four lane PCIe endpoint interface (Ports 0–3). Four lane PCIe is a full-duplex interface that uses four self-clocked serial differential lanes in each direction to achieve up to 8 Gbps data throughput. Each lane operates at 2.5 Gbps to accommodate both data and overhead associated with 8B/10B coding.

On the WANic-56511, QLM0 provides the lower four lanes of an eight lane PCIe interface (QLM1 provides the upper four lanes).

On the WANic-56512, QLM0 provides all four PCIe lanes.

- QLM1** The WANic-56511 uses QLM1 to provide four additional lanes of PCIe. The WANic-56512 uses QLM1 to provide a XAUI link dedicated to the second front panel 10GbE SFP+ transceiver and PHY.
- QLM2** QLM2 is not used.
- QLM3** QLM3 accommodates a XAUI link dedicated to a front panel 10GbE SFP+ transceiver and PHY.

2.2.4 Processor Interfaces

The Processor contains the following additional interfaces:

- Two-Wire Serial Interface (TWSI)
- General Purpose I/O (GPIO)
- Universal Asynchronous Receiver Transmitter (UART)
- On-Board Power Supply

2.2.5 TWSI Interface

The Processor supports the TWSI interface, which can read from and write to devices on an I2C bus. The Processor has five independent TWSI ports.

1. One TWSI connects to the two Mini Registered Dual In-Line Memory Module (Mini-RDIMM) sockets (0 and 1) and is hardwired to device address **0x50** and **0x51** respectively. This TWSI interface is used to determine the characteristics of each installed Mini-RDIMM device.
2. The second TWSI is used for the PCIe Configuration EEPROM at device address 0x52.



NOTE

Depending on the configuration, the EEPROM may not be installed.

3. There is also a TWSI for the SFP+ and one for the Temperature Sensor.

The Processor supports normal and fast modes for the TWSI interface as well as both 7- and 10-bit interfaces.

When selected, the Processor and FPGA can operate as *Master* of the TWSI device to initiate a transaction.

Table 2-2 shows the high level address for the TWSI devices on the WANic-5651x and identifies their master controller.

Table 2-2 TWSI Device Information

TWSI Device	High-Level Address	Master Controller
DIMM0	0x50	Processor
DIMM1	0x51	Processor
EEPROM	0x52	FPGA
SFP +	0x53	FPGA
Temperature Sensor	0x57	FPGA

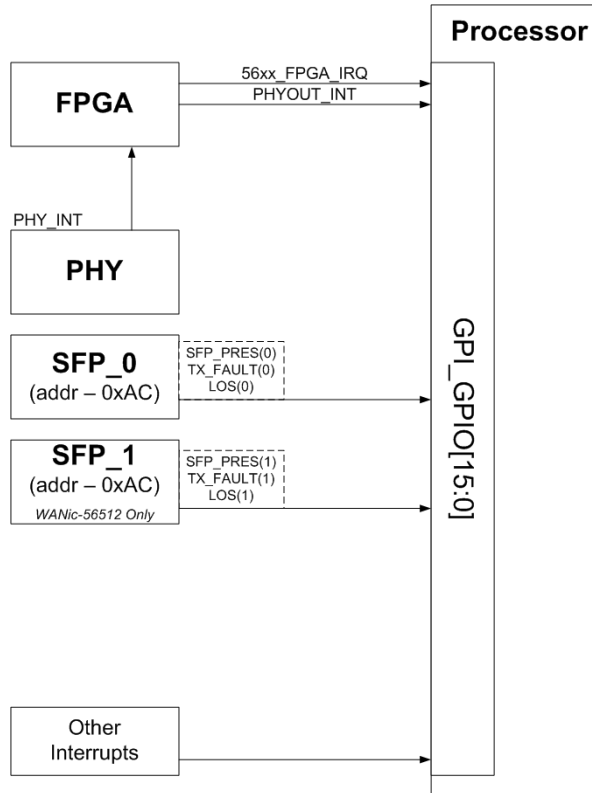
The TWSI interface communicates with any of the cnMIPS64 cores or a remote host. Software primitives provide access to the TWSI. See “[Chapter 4: Software Description](#)” for more information on TWSI software primitives.

2.2.6 GPIO Interface

The Processor provides a 16-Bit GPIO general purpose interface used as an input port to detect the following signals:

- SFP_PRES – SFP presence
- TX_FAULT – Transmit fault
- SFP LOS – SFP Loss of Signal
- PHYOUT_INT – An interrupt from the PHY interrupt (in the FPGA) to a common GPIO (GPIO11) and decoded by reading the FPGA **PHY Interrupt Register** (0x22).

Figure 2-4 GPIO Interface Assignment



Each bit within the GPIO interface can be configured independently to generate an interrupt in response to a state change for the specified devices as described in Table 2-3.

Table 2-3 GPIO Interface Bits

Bit	Field	Description	Value	Access
0	SFP_PRE0	SFP0 Presence – When set, detects SFP0 presence.	0 = Not Present 1 = Present	R/O
1	SFP_PRE1	SFP1 Presence – When set, detects SFP1 presence.	0 = Not Present 1 = Present	R/O
3–2	RSVD	Reserved		
4	TX_FAULT0	Transmit Fault 0 – When set, indicates a transmit fault on SFP0.	0 = No Fault 1 = Fault	R/O
5	TX_FAULT1	Transmit Fault 1 – When set, indicates a transmit fault on SFP1.	0 = No Fault 1 = Fault	R/O
7–6	RSVD	Reserved		
8	SFP_LOS0	SFP0 Loss of Signal – When set, indicates a loss of signal on SFP0.	0 = No LOS 1 = LOS	R/O
9	SFP_LOS1	SFP1 Loss of Signal – When set, indicates a loss of signal on SFP1.	0 = No LOS 1 = LOS	R/O
11–10	RSVD	Reserved		
12	PHYOUT_INT	PHY OUT Interrupt– When set, indicates an interrupt is present on the PHY device	0 = No interrupt 1 = Interrupt	R/O
13	RSVD	Reserved		
14	FPGA_INT	FPGA Interrupt – When set, indicates an FPGA interrupt.	0 = No interrupt 1 = Interrupt	R/O
15	RSVD	Reserved		

2.2.7 RS-232 Serial Port Interfaces

The Processor provides an interface to a general purpose asynchronous communications controllers. This industry standard 16550-style Universal Asynchronous Receiver Transmitter (UART) is capable of sending and receiving data at rates up to 10 Mega (M) baud. The Processor also provides a Baud Rate Generator, which drives the baud rate by dividing the Processor core clock frequency by a user-specified 16-bit divisor, multiplied by 16.

UART Port 0 is available for debug purposes via an optional Serial Debug Adapter (SDA) on the WANic-5651x. (See [Section 2.7.1 J5 Header](#).) The serial port is used for console access and has a baud rate programmable up to 115K.

2.2.8 On-Board Power Supply Interface

The Processor’s core and I/O are powered by dedicated on-board power supplies. The interface to the power supply permits adjustments to the core supply voltage, as needed, without affecting other circuitry.

2.2.9 Clocking

The CN56xx core reference clock input (PLL_REF_CLK) is driven by a fixed 50 MHz oscillator. A Phase Locked Loop (PLL) within the CN56xx multiplies the reference clock to derive the internal core clock frequency according to the 5-bit value strapped to the PLL_MUX_<4:0> Otheon inputs. The default for the WANic-5651x is core frequency of 750MHz

The PLL_REF_CLK must meet the following clock requirements:

- 50 MHz frequency
- 150 ± total ppm
- 45 to 60% duty cycle
- 150 ps phase jitter (Peak-to-Peak)
- 2.0 ns edge rate (measured from 10–90% single levels on single -edged clock. Differential clock is measured from –150mV to +150mV)
- 0 – 3.3 V levels

The clocking distribution scheme for the WANic-56511 is shown in Figure 2-5. Figure 2-6 illustrates the clocking for the WANic-56512.

Figure 2-5 WANic-56511 Clocking

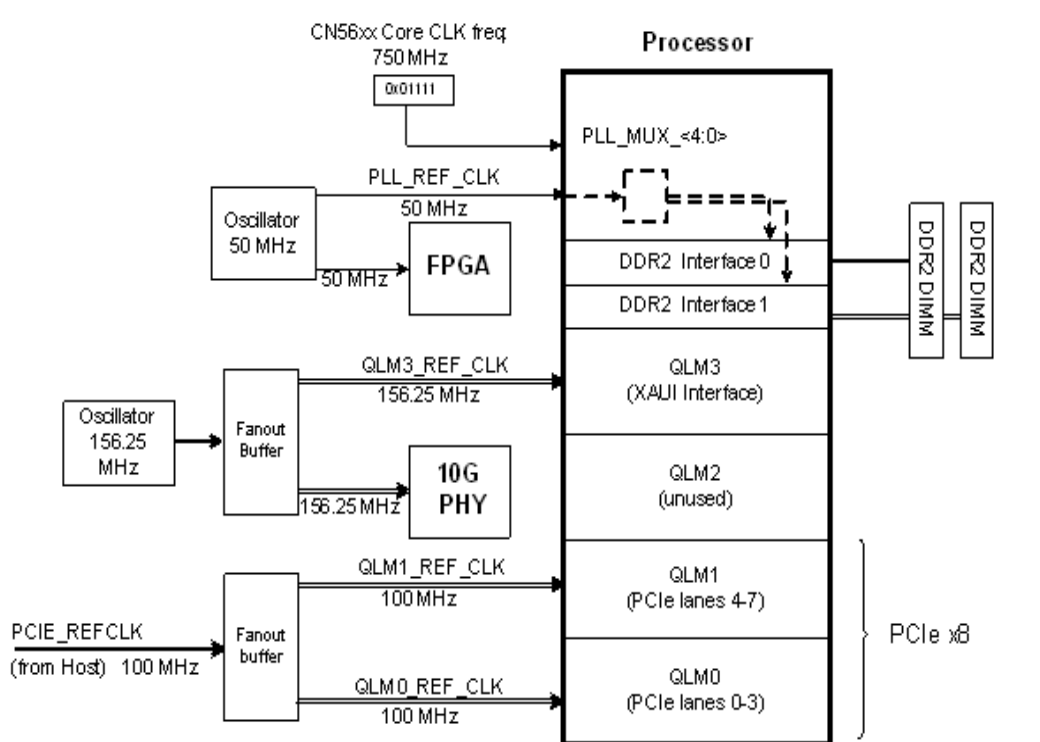
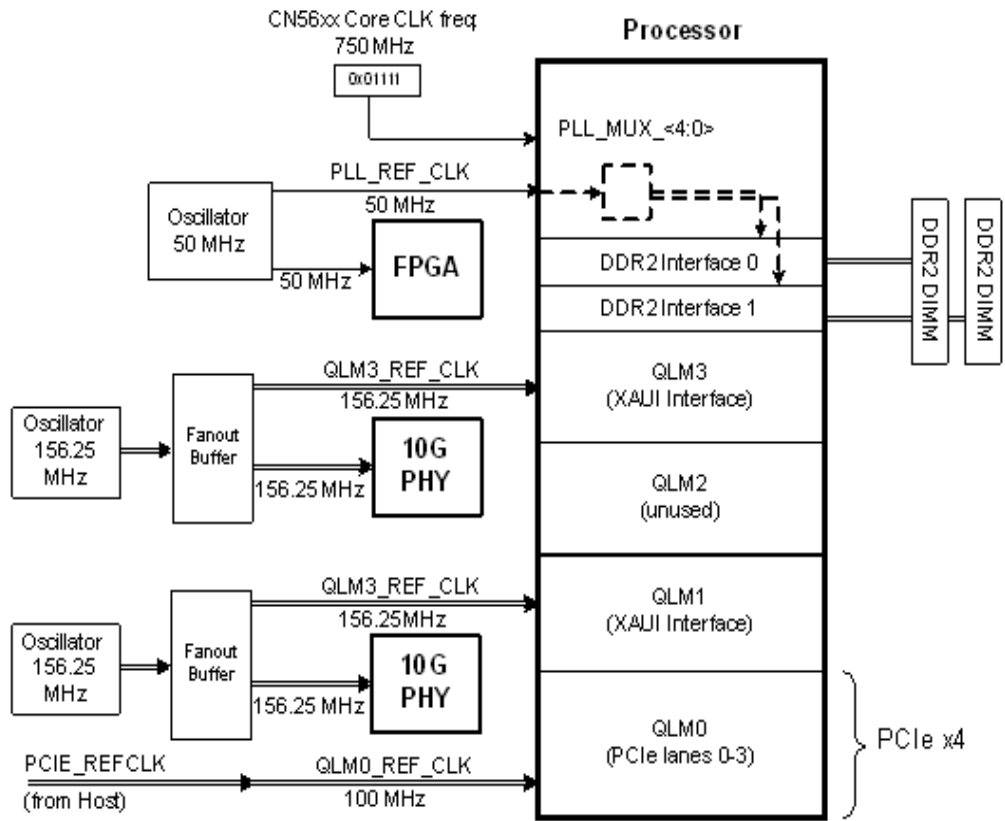


Figure 2-6 WANic-56512 Clocking



2.3 PCI Express Connector

The PCI Express Connector provides a reliable, high-speed, serial point-to-point interconnect that is downwardly compatible with the PCI bus. The PCI Express Connector achieves reliable, high-data rates by using Low-Voltage Differential Signaling (LVDS) pairs. The WANic-5651x PCIe Connector provides four (x4) or eight lane (x8) PCIe interface at data rates specified in Table 2-4.

Table 2-4 PCIe Data Rates

PCIe Implementation	Encoded Data Rate	Unencoded Data Rate
x1	5 Gbps	4 Gbps (500 MB/sec)
x4	20 Gbps	16 Gbps (2 GB/sec)
x8	40 Gbps	32 Gbps (4 GB/sec)

PCIe Edge Connector Clocking Specifications

The PCIe Edge Connector must be supplied with a reference clock from the Host that meets the following clock requirements:

- 100 MHz frequency
- 150 ± total ppm
- 45 to 55% duty cycle
- 80 ps phase jitter (Peak-to-Peak)
- 0.5 ns edge rate (measured from 10–90% single levels on single -edged clock. Differential clock is measured from –150mV to +150mV)
- 0 – 0.7 V levels
- 100 Ohm source termination

2.4 FPGA

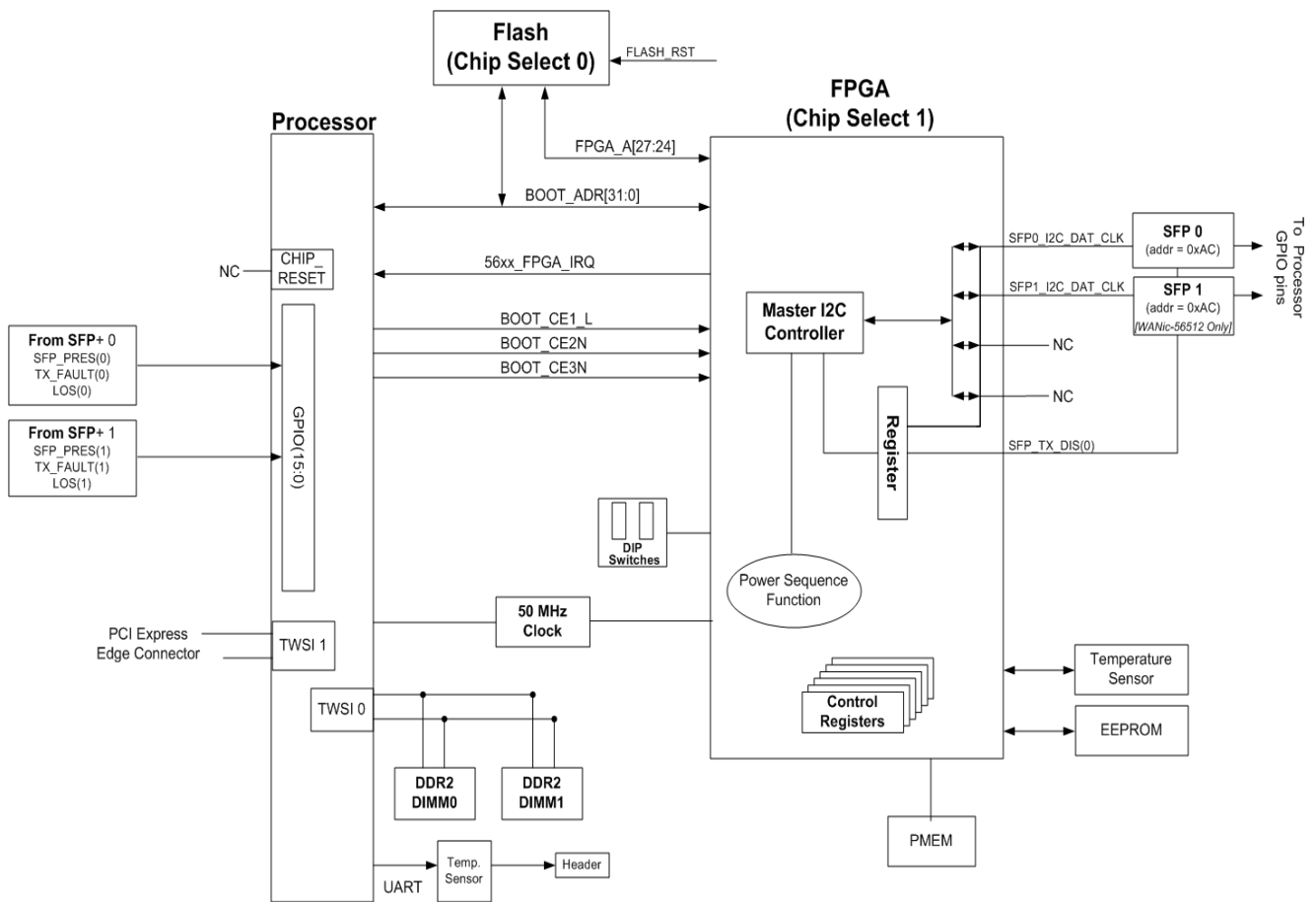
The WANic-5651x use an FPGA as the interface to various on-board hardware resources. The FPGA resides on the Boot Bus (along with the boot Flash device) and can be accessed by the cnMIPS cores or PCIe Host.

The FPGA includes independent I2C controllers to provide access to the management port of each SFP+ module, and the power sequencing function.

The FPGA provides the following functions and interfaces as shown in Figure 2-7:

- Control and Status Registers
- Persistent Memory Interface and Control
- Clock
- Interrupt Control
- Power-on Reset
- SFP+ I2C Interface
- Multi-Core Processor Interface
- TWSI Interface

Figure 2-7 FPGA Function Block Diagram



2.4.1 Control and Status Registers

The FPGA contains special registers that allow the Processor to control and to monitor transactions among interfaces and local hardware resources. Boot Chip Select 1 decodes access to various control and status registers in the FPGA. The FPGA latches the upper three address bits for the Boot Flash using the Boot Bus. A dedicated register, **Boot Flash Upper Address Control Register**, is provided for this operation.

All FPGA registers map to a contiguous block of 16 bytes within each address space. Thus, the FPGA Chip Selects can be configured to locate this bank of 32 locations to any area within each address space. In cases when the Chip Select logic configuration accesses a range of addresses larger than 16 bytes, the FPGA register image repeats.

2.4.2 Persistent Memory Interface

The FPGA has a dedicated interface for the Persistent Memory (PMEM). To access PMEM from the Boot Bus, Chip Select 2 (BOOT_CE2N) and Chip Select 3 (BOOT_CE3N) are brought to the FPGA for this purpose. Use BOOT_CE2N for the lower 16MB of PMEM, and BOOT_CE3N for the upper 16MB of PMEM.

2.4.3 FPGA Clocking

The FPGA uses a copy of the 50 MHz Processor core reference clock as its primary synchronization method.

2.4.4 FPGA Interrupts

The FPGA provides a means to generate an interrupt upon completion of a transaction on any of its I2C interface ports. Generation of interrupts can be enabled on a port-by-port basis by means of bits in the FPGA **Interrupt Control Register**. (See [Chapter 3: FPGA Registers](#) for more information on this register.)

2.4.5 Reset Control

The FPGA routes individual reset signals to the Flash memory, RDIMM memory, and 10 G PHY device. These resets signals are asserted at power-up and are de-asserted automatically when power is stable. Software can force an individual reset to these devices using the **Peripheral Reset Register**. (See [Chapter 3: FPGA Registers](#) for more information on this register.)



CAUTION

A PCI reset from the Host does **not** reset these devices.

2.4.6 Power Supply Control

The FPGA contains logic to enable various on-card power supplies to turn on in a specific sequence with controlled timing. Operation of this logic is automatic upon application of +3.3V and +12V power to the input of the card.

The FPGA is constantly monitoring the state of each supply and restricts the sequence accordingly if any faults arise.

2.4.7 I2C Interface

The FPGA provides independent I2C port interfaces for the following devices:

- Front panel SFP and module
- A 64 KB serial EEPROM
- A MAXIM® 6663 temperature sensor

The FPGA performs read and write operations on each I2C interface in response to requests from the Processor.

An independent set of registers is provided for each I2C interface. See “[Chapter 4: Software Description](#)” for a description of each I2C interface register.

2.4.8 TWSI Interface

The FPGA can be master controller for the TWSI devices listed in Table 2-5.

Table 2-5 FPGA TWSI Devices and Addresses

Device	Address
EEPROM	0x52
SFP0+ (bottom)	0x53
SFP1+ (top)	0x54
Temperature Sensor	0x57

Descriptions of each device are given in this chapter.

2.5 Memory

The packet memory on the WANic-5651x consists of DDR2 SDRAM. Flash memory provides storage for start-up boot code and Processor core images. The EEPROM provides general purpose storage of hardware information. The WANic-5651x also contains Persistent Memory and an embedded USB Flash disk drive.

2.5.1 Internal (Level 2) Cache Memory

The Processor operates primarily on internal Level 2 cache by means of a coherent memory bus. The Level 2 cache connects to two DDR2 controllers to support DDR2 SDRAM packet memory, which provides primary data storage. Internal (Level 2) cache memory is 2 MB.

2.5.2 DDR2 SDRAM

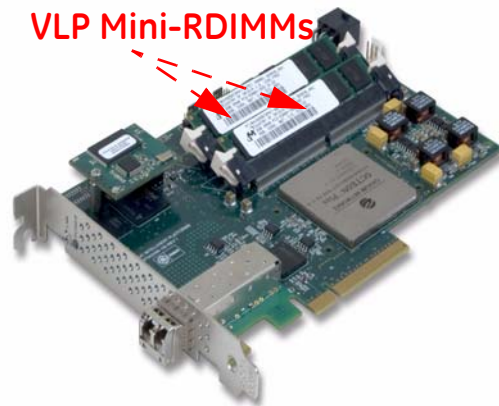
The WANic-5651x supports DDR2 memory with a 64-bit data path. An additional 8 bits is provided for Error Correction Code (ECC) support. ECC permits detection and correction of single-bit memory errors as well as two-bit error detection. Timing for the DDR2 SDRAM interface is derived from an internal PLL that locks to and multiplies from the core clock frequency.

Mini-RDIMMs

The WANic-5651x uses Very Low Profile (VLP) Mini-RDIMMs with ECC and supports both single- and dual-rank varieties. The Mini-RDIMMs connect to the board by means of two angled VLP Mini-DIMM sockets.

Figure 2-8 show the location of Mini-RDIMMs on the WANic-5651x.

Figure 2-8 Mini-RDIMMs on WANic-5651x



Serial Presence Detect

Each VLP Mini-RDIMM module features Serial Presence Detection (SPD) based on a serial EEPROM device on the Mini-RDIMM, which uses the I2C protocol to access information about each Mini-RDIMM. Mini-RDIMM sockets are connected to the Processor, TWSI, I2C interface, and hardwired to device addresses **0x50** and **0x54** respectively.

Reset Mini-RDIMM reset is asserted at power-up and de-asserted automatically once power is stable. Software can force an individual reset to the Mini-RDIMM using the **Peripheral Reset Control Register**.



CAUTION

A PCI_RESET signal from the Host does **not** reset the Mini-RDIMMs.

2.5.3 Boot Flash

An on-board Flash memory device operates as a 128M x 8 device organized as 1024 blocks of 128 KB each. Code execution is supported from this Flash during hardware initialization only. The Flash is directly connected to the Boot Bus.

Operating firmware copies the boot code from Flash into the DDR2 SDRAM packet memory for use following hardware initialization to optimize code performance. When required by operating firmware, the Flash device can be written to or erased.

The Flash is partitioned at the factory and can be reprogrammed. For information on reprogramming the Flash, as well as information on accessing and partition the Flash, see "[Section 4.5.1 Flash Driver](#)."



NOTE

The WANic-5651x also supports PCIe booting. See "[Section 2.8 Dual In-Line Package \(DIP\) Switch](#)" for more information on selecting PCIe booting.

2.5.4 Persistent Memory

Persistent Memory (PMEM) is a storage device whose contents are preserved whenever the power is on. The WANic-5651x provides 32 MB of PMEM, which retains its contents across hard and soft resets *except* for power-on resets. PMEM is available to store system events and logs messages for use after a system failure. The contents of PMEM is preserved over any system reset and available for analysis after a system failure and reboot.



NOTE

The PMEM contents is not preserved whenever power *is not* applied (for example, if there is no battery backup.)

The 32 MB of shared memory is treated as two 16 MB regions. Each region is accessed through memory Chip Select 2 (CS2) to provide the lower PMEM address and Chip Select 3 (CS3) provides the upper PMEM address.

2.5.5 Serial EEPROM

The Serial EEPROM provides 64 KB of general-purpose non-volatile data storage for on-board specific hardware configuration information, such as assembly model number, serial number, and so forth. An I2C interface connects the EEPROM to the Processor by means of the FPGA. This permits examination and modification of the EEPROM contents.

2.5.6 USB Flash Drive

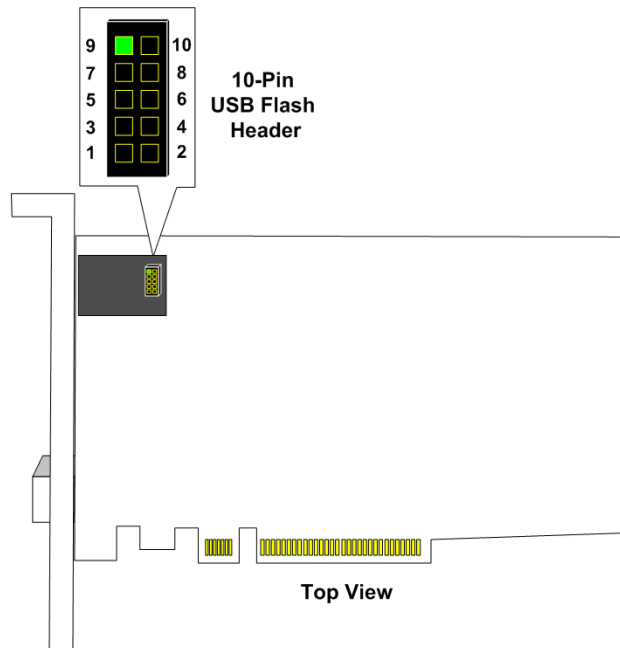
The WANic-5651x contain a 2 GB USB Flash Drive for additional storage or for application use. The USB Flash Drive contains a standard 10-pin header on the top (circuit side) of the card as shown in Figure 2-9.



NOTE

For additional USB configurations of up to 4 GB, consult a GE Intelligent Platforms Customer Support Representative.

Figure 2-9 USB Flash Drive Header



The pinout for the USB Flash drive header is described in Figure 2-10.

Figure 2-10 USB Flash Header Pinout

Pins	Type	Signal Name	Description	Illustration
1	Power	VBUS	Bus voltage supply from source(+5V)	
3	D-	i/O	Data line -	
5	D+	I/O	Data line +	
7	GND	Ground	Ground	
2, 4, 6, 8, 10	NC	Open	No connect	
9	Key	Open	Alignment	

2.6 Gigabit Ethernet Physical Device

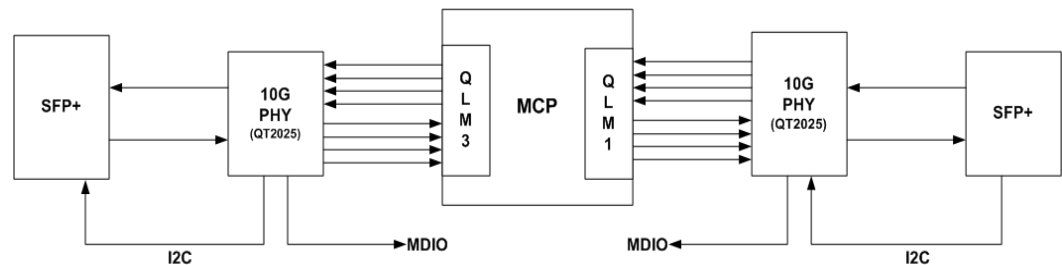
The 10G PHY device is a QT2025 from AMCC, which provides a high-performance, 10 Gbps transceiver independent port. As shown in Figure 2-11, the 10G PHY device is connected to a XAUI link on the Processor through QLM3. The 10G PHY device implements the IEEE 802.3u Physical Coding Sublayer (PCS) Clause 22 specifications for 10GBASE-X.



NOTE

The WANic-56512 implements a second 10G link using QLM1.

Figure 2-11 10G PHY Implementation



The Processor contains a dedicated system management interface (SMI) controller to communicate with this interface. The MDC is the Management Data Clock and as the frequency range of 0-8.3 MHz. The Management Data Input Output (MDIO) pin is the bidirectional management data input/output that runs synchronously to the MDC.

The 10G PHY device has an active low hardware reset pin (`/QT2025_RESET`), which must be held low for 500 μ s after the power supplies reach their nominal values. The 10G PHY device reset is asserted at power-up by the FPGA, and de-asserted automatically when power is stable. Software can also force an individual reset to the PHY device via the **Peripheral Reset Control Register**.



NOTE

A `PCI_RESET` from the host device does **not** reset this device.

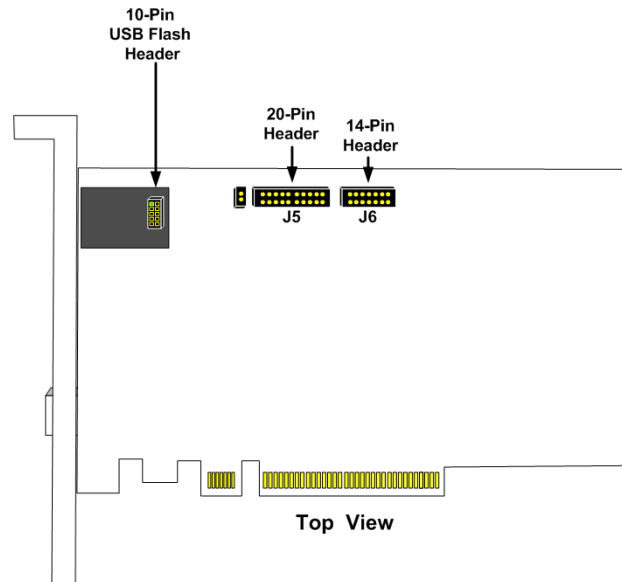
The 10G PHY requires a 156.25 MHz +/-50 PPM reference clock. As shown in Figure 2-5, the 156.25 MHz clock that feeds the PHY is a buffered copy of a Low Volt PECL oscillator. The device used to buffer the oscillator output depends on the build configuration.

2.7 Headers

The WANic-5651x contain featured headers as shown in Figure 2-12, which are located on the top (circuit side) of the board:

- J5 – is a 20-pin header for access to RS-232 and EJTAG scan chain interface
- J6 – is a 14-pin header for the JTAG interfaces

Figure 2-12 Header Locations



2.7.1 J5 Header

The EJTAG and RS-232 UART are combined on a single 20-pin header (J5). Certain signals within the JTAG port are also shared with the EJTAG scan chain. The on-board DIP Switch determines whether these signals are connected as part of the scan chain or whether they are routed out to the header.

An RS-232 header provides the signals for one UART depending on the WANic-5651x model.

Serial Debug Adapter (optional)

An optional Serial Debug Adapter (SDA) and cable assembly separates these JTAG and RS-232 signals into two independent standard boxed-headers:

- RS-232 – 10 pin connector
- EJTAG – 14-pin connector

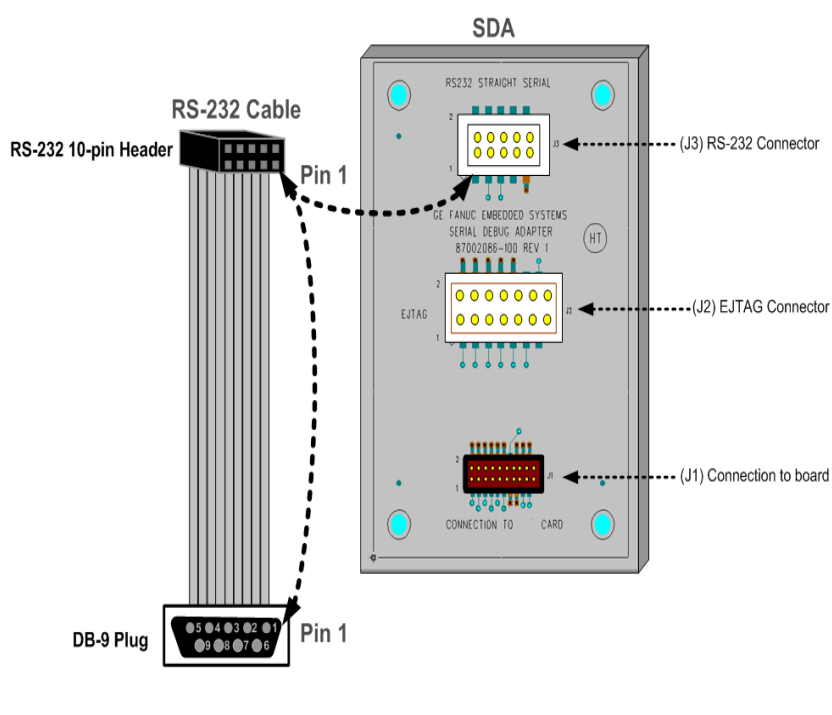
The RS-232 10-pin connector permits the attachment of a standard Insulation Displacement Connector (IDC) cable assembly consisting of a 10-pin header and a standard 9-pin RS-232 Digital Communications Equipment (DCE) interface.

The EJTAG 14-pin header supports direct attachment of EJTAG debugging tools. The header is keyed to ensure proper connector alignment when the user-supplied cable assembly is attached to the optional SDA.

Note the following:

- The pinout and gender of the DB-9 connector permits it to be directly attached to DTE devices such as a dumb terminal or terminal emulation software running on a PC. If an extension cable is required, be sure to use an *RS-232 Straight-Through Cable*. An RS-232 Straight-Through Cable has a DB-9 plug on one end and a DB-9 socket on the other. This cable does not have 'crossover' or 'null modem' functionality built into it.
- The RS-232 IDC cable assembly is *not* keyed. Make sure that the cable is oriented in such a way that Pin 1 of the DB-9 socket is on the same edge of the cable as Pin 1 of the 10-pin header on the SDA. For example, pins on the DB-9 Plug in Figure 2-13 are numbered 1–9. Following the cabling from Pin 1 on the DB-9 Plug to the socket indicates the location of the Pin 1 slot on the DB-9 socket, which can be inserted correctly onto the RS-232 Header on the SDA.

Figure 2-13 SDA Cabling Example



2.7.2 J6 Header and JTAG Scan Chain

The J6 Header provides an interface for JTAG testing. To support testing and debugging, all WANic-5651x components that provide JTAG test ports are interconnected to form a JTAG Scan Chain.



CAUTION

To ensure proper operation under normal operating conditions, the JTAG *TRST* signal must be held in its asserted state (*low*). Failure to observe this requirement can result in unpredictable behavior of the module.

Components that connect to the JTAG Scan Chain include the following:

- Processor
- One or two 10GbE PHYs
- FPGA

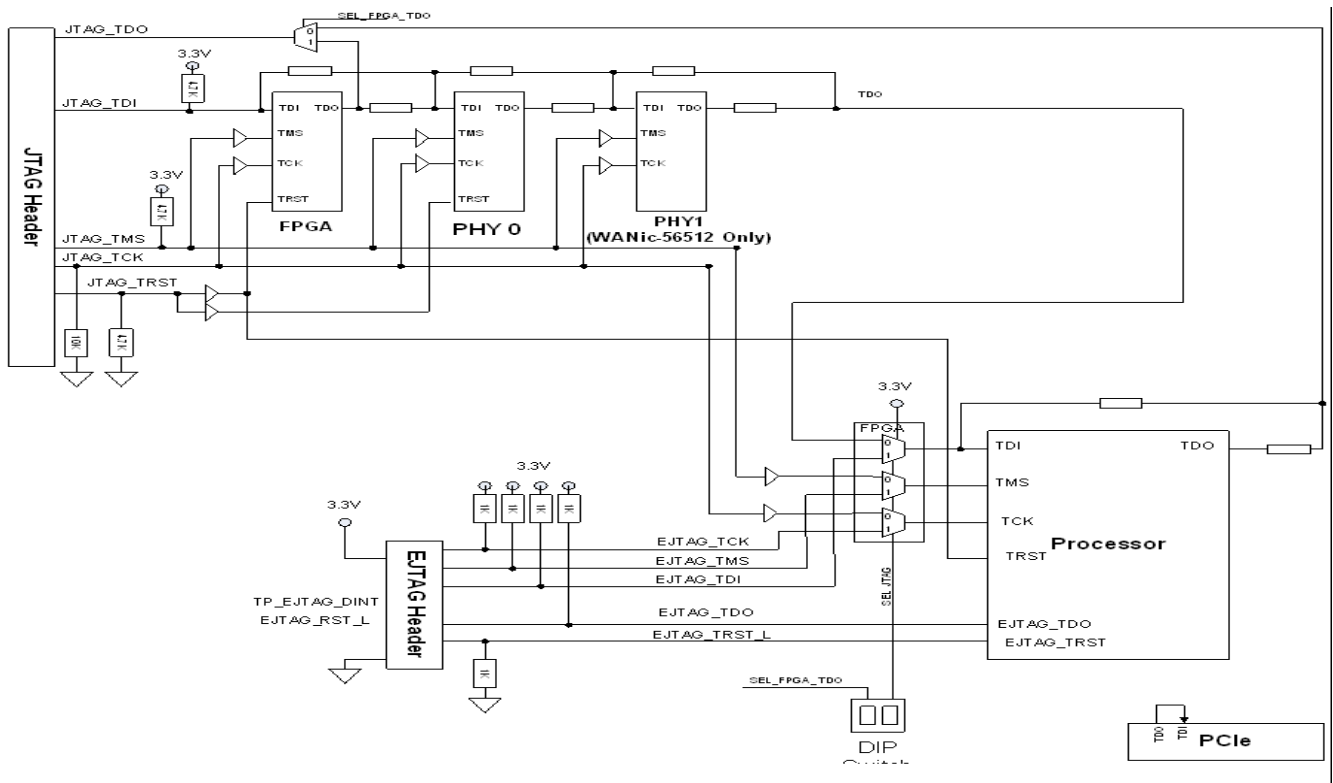
The on-board S1 DIP Switch bank permits the JTAG Scan Chain to be reconfigured to simplify programming of the FPGA, and to support the use of EJTAG debugging tools in conjunction with the Processor. See Figure 2-15 for the location of the S1 DIP Switch bank, which is comprised of four DIP Switches (1-4).

In the S1 DIP Switch bank, set Switch 1 and 2 to the 'ON' position to include all devices in the JTAG Scan Chain. This represents the default configuration for all manufacturing JTAG tests.

When Switch 1 is set to 'OFF', the FPGA Test Data Out (TDO) signal is reconnected directly to the Connector JTAG TDO signal. Since the FPGA Test Data In (TDI) input is the first in the chain, it always connects directly to the TDI input. Thus, setting Switch 1 to 'OFF' effectively isolates all other devices from the JTAG Scan Chain (from the PCIe Connector perspective). Typically, this simplifies the task of in-circuit programming the FPGA with JTAG programming tools.

Setting Switch 2 to the 'OFF' position disconnects certain Processor JTAG I/O signals from the scan chain and reconnects them to the 20-pin header. This configuration permits attachment of an Enhanced JTAG (EJTAG) debugging tool, connecting the device TDI input to the TDO output (normally unpopulated).

Figure 2-14 JTAG Scan Chain



2.7.3 Enhanced JTAG Header (EJTAG)

The Processor JTAG pins can be connected to the JTAG Scan Chain or to the Enhanced JTAG (EJTAG) header. Multiplexers, under control of the on-board S1 DIP Switches, route the JTAG pins.

The EJTAG Header is a standard 14-pin dual-row boxed header connected to the 20-pin header through the SDA card. It provides keying of a mating cable to prevent inadvertent insertion in the wrong direction. The EJTAG Header interface permits the Processor to connect to standard debugging tools during software development. It also provides a means of initially programming a boot-code loader routine into the Flash.

Since the Processor is a target device on the PCIe interface, the `PCI_RST` signal is used as the chip reset for the Processor.



NOTE

The `EJTAG_RST` signal is not used in this PCIe target application. The `EJAG_RST` is not required by the software debugger. When a reset is necessary, the Host must issue a PCI Reset (`PCI_RST`) signal.

2.8 Dual In-Line Package (DIP) Switch

The WANic-5651x contain a grouping of four DIP Switches, collectively labeled S1, as shown in Figure 2-15. Set the S1 DIP Switches to perform the following operations:

- JTAG boundary scan testing
- FPGA override for stand-alone JTAG programming of FPGA
- EJTAG configuration
- Boot mode selection
- Diagnostic utility booting

The **Board ID and DIP Switch Register** permits the Processor to determine the position of each the DIP Switch within the S1 switch bank.

Figure 2-15 shows the approximate location of the S1 DIP Switch, which is located on the bottom (soldered side) of the board. Table 2-6 through Table 2-7 summarize the DIP Switch settings and associated operations.



NOTE

When using switch 3 in the S1 DIP Switch bank, to invoke the Flash application boot process, use Environment Variables to control the process. (See [Chapter 4: Software Description](#) for more information on Environment Variables.)

Figure 2-15 S1 DIP Switch Location

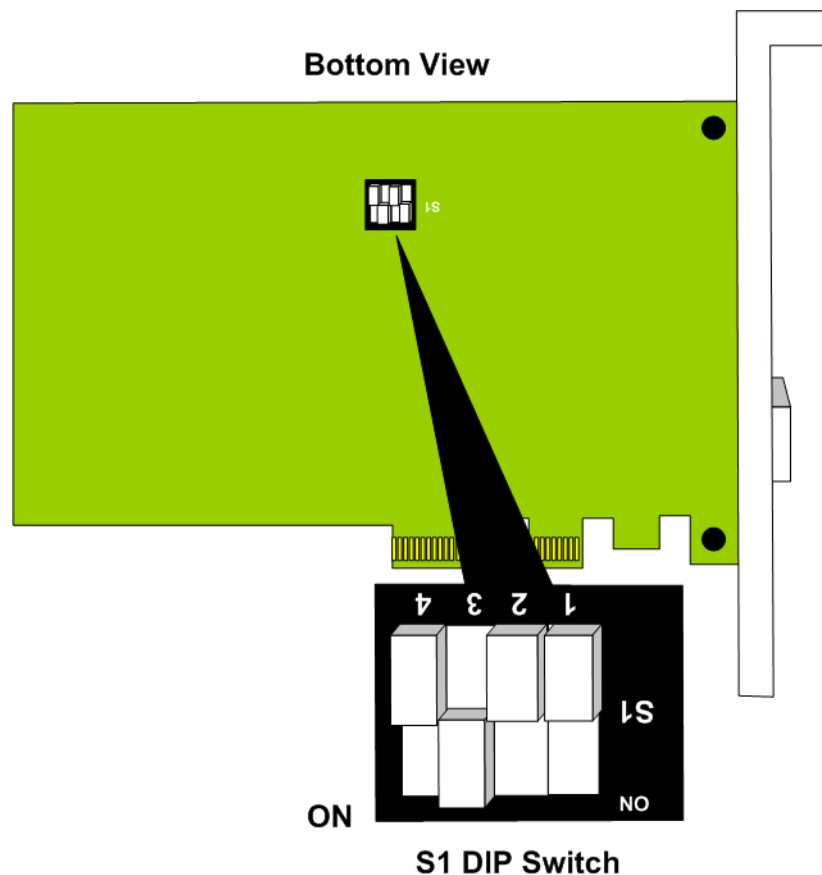


Table 2-6 JTAG Mode S1 DIP Switch Configuration

Mode	Operation	DIP SWITCH	
		1	2
JTAG	Configures the board/MPU for use with EJTAG debugger	Off	ON
	Configures the board for JTAG programming of the FPGA	Off	Off
	Not supported	ON	Off
	Configures the board for boundary scan testing on full JTAG chain	ON	ON

Table 2-7 Boot Mode S1 DIP Switch Configuration

Mode	Operation	DIP SWITCH
		3
Boot	Enables Flash Boot operations	Off
	Enables PCIe Boot operations	ON



NOTE

When using DIP Switch 3 to invoke the Flash application boot process, use Environment Variables to control the process. (See [Chapter 2: Hardware Description](#) for more information on Environment Variables.)

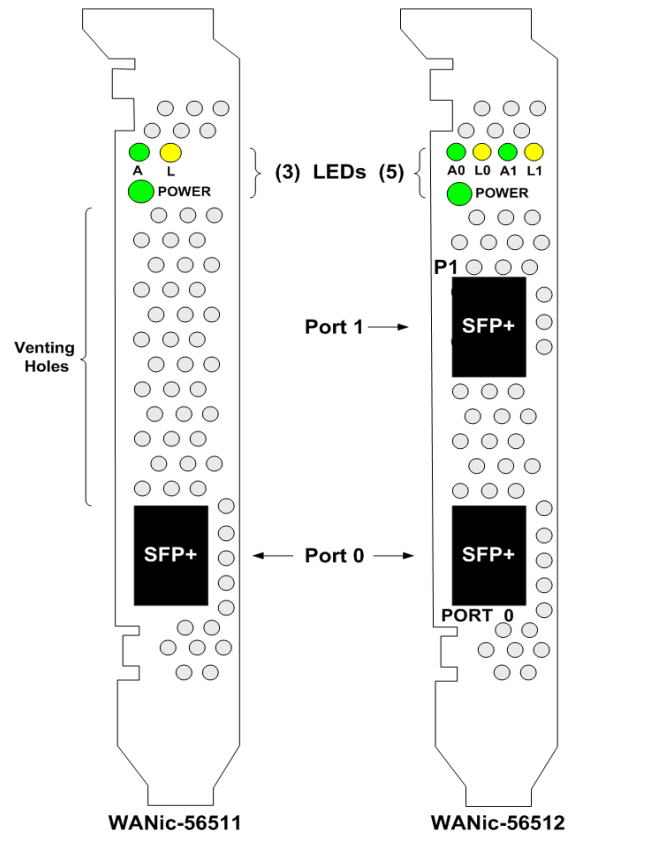
Table 2-8 Diagnostic Mode S1 DIP Switch Configuration

Mode	Operation	DIP SWITCH
		4
Diagnostic	Disables Linux auto-start	Off
	Enables auto-start Linux and Diagnostics	ON

2.9 Front Panel Connectors and LEDs

The WANic-5651x front panel connectors and LEDs are shown in Figure 2-16. (There are also numerous venting holes throughout the front panel.)

Figure 2-16 Front Panel Connectors and LEDs



2.9.1 SFP+ Connectors

SFP+ optical/fiber transceivers support 10GBase-SX/LX connectors.

Direct Attach Mode

SFP+ connectors, in conjunction with WANic-5651x software and an optional cable assembly, provide Direct Attach Mode. Direct Attach Mode, as defined by the Multi-Source Agreement Group in the SFF-8431 specification detects when a cable is attached to the SFP+ connector and sets up the interface for the SFP+ automatically. The cable assembly for Direct Attach Mode can be purchased separately from GE Intelligent Platforms (Part No. 42G7690-0003).

Refer to 8 <http://www.sffcommittee.org/ie/sffspec.html> for more information on the SFF-8431 specification.

2.9.2 Front Panel LEDs

The front panel contains Power and Link Status LEDs as shown in Figure 2-16.

Power LED

The Power status LED is described in Table 2-9

Table 2-9 Module Status LEDs

LED	Color	Indicator	State	Description
POWER	Green	Power status	Off	Fault
			On	Normal operations

Link and Activity Status LEDs

The Link and Activity status LEDs are described in Figure 2-13. The WANic-56512 contains two Link and two Activity LEDs.

Table 2-10 Link Status LEDs

LED	Color	Indicator	State	Description
A[1:0]	Green	Activity	Off	Link is down and not working properly.
			On/ Blinking	Link is working properly
L[1:0]	Yellow	Link	Off	Link is down and not working properly.
			On/ Blinking	Link is up and receiving

2.10 Temperature Sensor

All WANic-5651x models contain a Temperature Sensor to actively monitor the temperature of the Processor. The Temperature Sensor provides set points (maximums) for the temperatures. When a set point is crossed, the Temperature Sensor asserts the thermal overload signal. In addition, two shutdown outputs are triggered when the remote or local temperatures exceed the programmed set points.

The FPGA on the WANic-5651x provides two status registers to provide indications to the Host when these set points are exceeded:

- The **Interrupt Status Register** indicates to the host that there was an interrupt involving the Temperature Monitor.
- The **Temperature Status Register** provides information to the host as to the cause of the interrupt.

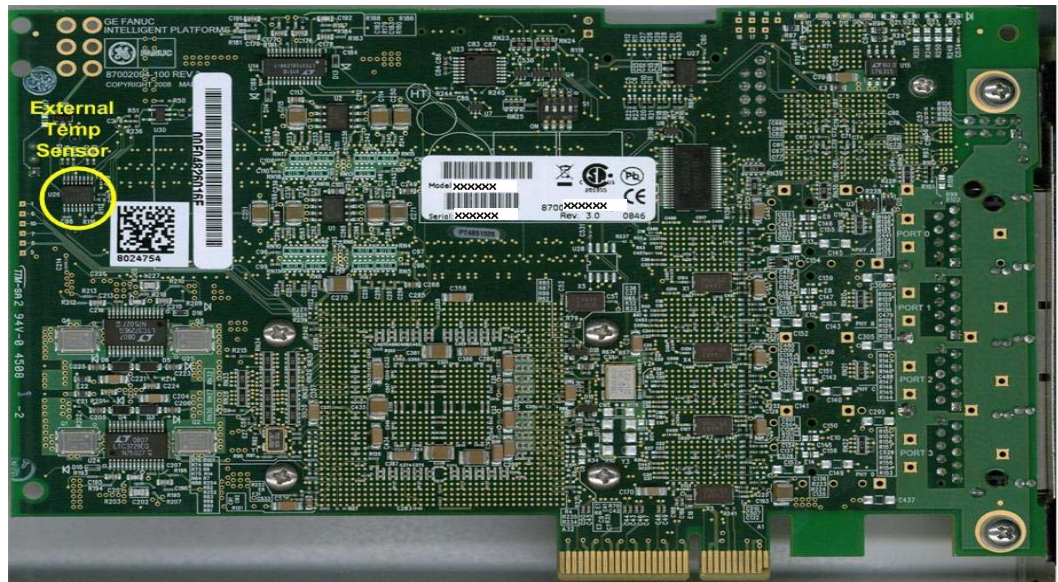
See [Chapter 3: FPGA Registers](#) for more information on these FPGA registers.

2.10.1 Thermal Management

The WANic-5651x contains two temperature sensors:

- An internal sensor within the Processor
- An external sensor (U26) on the circuit side of the board as shown in Figure 2-17

Figure 2-17 External Sensor



The temperature sensor, in conjunction with the thermal diode within the Processor, provides a direct indication of the Processor die temperature. Use the temperature sensor to determine whether there is sufficient cooling air flow to maintain a junction temperature of less than 105°C in accordance with the maximum ratings specified for the Processor.

The U26 sensor indicates the temperature of its own die and that of the Processor. The temperature accuracy is within $\pm 1^\circ\text{Celsius}$ (C) for remote temperature measurements from +60°C to +100°C.

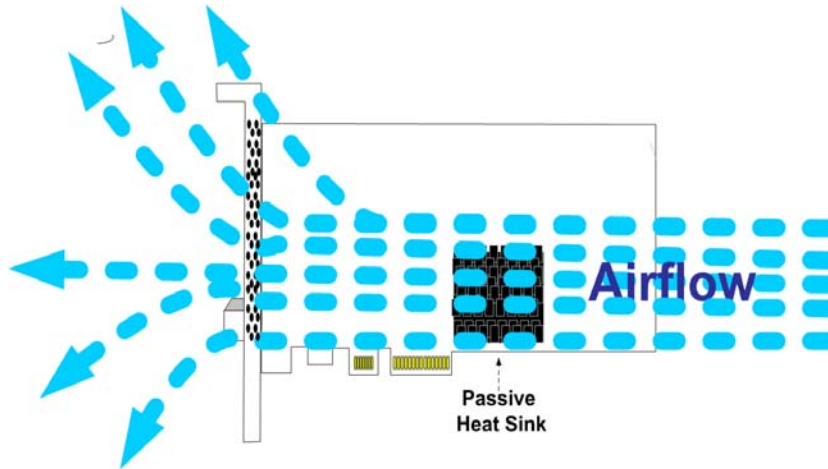
The WANic-5651x is designed for an air-cooled chassis environment. The WANic-5651x thermal requirements are satisfied by air holes in the front panel and a passive heat sink to provide sufficient air flow to maintain the appropriate temperature for the Processor. The heat sink is secured to the PCB with four screws and contacts the Processor with a thermal interface material. The minimum volumetric air flow in Linear Feet per Minute (LFM) is listed in Table 2-11

Table 2-11 WANic-5651x Cooling

Cooling	WANic-56512	WANic-56512
Board with Heat Sink Assembly		
[Minimum volumetric flow rate at a maximum ambient temperature (Celsius) in Linear Feet per Minute (LFM)]	175 LFM @ 25° C	205 LFM @ 25° C
	285 LFM @ 40° C	305 LFM @ 40° C
	495 LFM @ 55° C	505 LFM @ 55° C

The air flow path is similar to that shown in Figure 2-18 depending on the location of air vents within the chassis.

Figure 2-18 WANic-5651x Air Flow Path



2.11 Power

The WANic-5651x hardware uses multiple voltage sources. The FPGA activates the power distribution in a serial fashion.

Maximum power requirements are different for each model and configuration. Consult with Customer Technical Support representative for specific power requirements for each model.

The WANic-5651x hardware is powered through a J4 PCIe External Graphics Power Connector (+12V) located on the upper corner of the board as shown in Figure 2-19. Figure 2-19 also shows the pinout for this connector.

The J4 PCIe External Graphic Connector requires a cable and mating connector assembly (purchased separately) from an external power supply. The mating connector on the cable from the external power supply should be a Molex® 4.20mm (.165") Pitch, 6-pin, Mini-Fit Jr.™ Receptacle Housing, Dual Row connector purchased separately from Molex Incorporated, PN 45559-0002. (www.molex.com)

Instructions for connecting to the J4 PCIe Graphic Power Connector are provided in "[Appendix A • Hardware Installation and Removal Procedures.](#)"

Figure 2-19 WANic-5651x External Power Connector

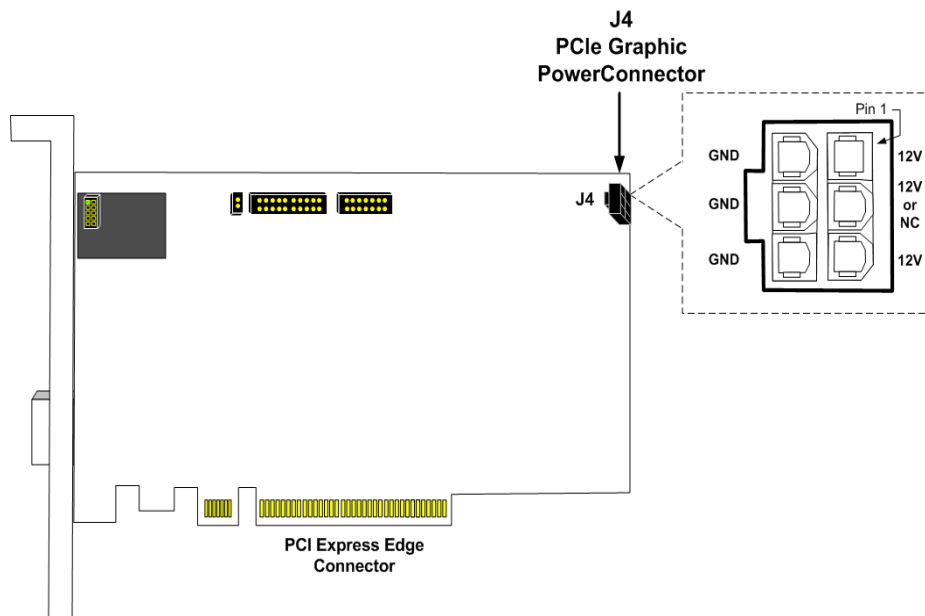


Table 2-12 identifies the current and power supply limits for both connectors. Typical user power will vary.

Table 2-12 Current and Power Limits

Model	Rail Voltage (V)	Source	Typical Maximum Watts (W)
WANic-56511	+3.3V	PCIe Edge	2.892W
	+12V	J4 External Connector	37.351W
			<i>Total: 40.242W</i>
WANic-56512	+3.3V	PCIe Edge	3.702W
	+12V	J4 External Power	39.062W
			<i>Total: 42.764W</i>

3 • FPGA Registers

This chapter describes the FPGA registers that the WANic-5651x uses to communicate with other on-board components.

3.1 FPGA Register Descriptions

The register interface provides direct access to FPGA registers. These 8-bit address registers, listed in Table 3-1, are used for the following:

- Resets
- Interrupts
- LEDs
- Temperature Sensor Control
- Status indications

The base address of the FPGA is **1600 0000** hexadecimal (h).

Access is R=Read, W=Write, O=Only.

Table 3-1 FPGA Memory Mapped Registers

Offset	Register	Description	Access
0x00	FPGA Revision	Provides the FPGA revision number.	R/O
0x01	Board ID and DIP Switch	Determines hard straps and DIP Switch settings.	R/O
0x02	LED Control	Manages the four diagnostic module status LEDs	R/W
0x03	Interrupt Control	Controls interrupts for selected devices.	R/W
0x04	Interrupt Status	Indicates interrupt status for selected devices.	R/W
0x05	Peripheral Reset	Generates a reset for selected on-card devices	R/W
0x06	Transmit Control	Manages the front panel SFP+ transmitter output.	R/W
0x07	Temperature Status	Indicates when the temperature sensor has experienced a fault.	R/W
0x08	Temperature I2C Control	Indicates an I2C transaction for the temperature monitor.	R/W
0x09	Temperature I2C Address High	Specifies the upper address for the transaction.	R/W
0x0A	Temperature I2C Address Low	Specifies the lower address for the transaction.	R/W
0x0B	Temperature I2C Data	Contains the data for the transaction.	R/W
0x0C	EEPROM I2C Control	Initiates an I2C transaction and, when necessary, reports an error for the EEPROM.	R/W
0x0D	EEPROM I2C Address High	Specifies the upper address for the transaction.	R/W
0x0E	EEPROM I2C Address Low	Specifies the lower address for the transaction.	R/W
0x0F	EEPROM I2C Data	Contains the data for the transaction.	R/W
0x10 – 0x17	Reserved		
0x18	SFP1 I2C Control	Initiates an I2C transaction and reports an error for SFP 1.	R/W
0x19	SFP1 I2C Address High	Specifies the upper address for the transaction.	R/W
0x1A	SFP1 I2C Address Low	Specifies the lower address for the transaction.	R/W
0x1B	SFP1 I2C Data	Contains the data for the transaction.	R/W
0x1C	SFP0 I2C Control	Initiates an I2C transaction and reports an error for SFP 0.	R/W
0x1D	SFP0 I2C Address High	Specifies the upper address for the transaction.	R/W
0x1E	SFP0 I2C Address Low	Specifies the lower address for the transaction.	R/W

Table 3-1 FPGA Memory Mapped Registers

Offset	Register	Description	Access
0x1F	SFP0 I2C Data	Contains the data for the transaction.	R/W
0x20	Boot Flash Upper Address Control	Breaks the 1 GB Flash into 16 MB pages.	R/W
0x21	Miscellaneous Functions	Provides several miscellaneous functions.	R/W
0x22	PHY Interrupt	Controls interrupts for on-card PHYs.	R/O

Register descriptions follow.

FPGA Revision Register

The **FPGA Revision Register** is an 8-bit read-only register that provides the FPGA major and minor hardware revision information.

The Minor Revision (`MIN_REV`) is indicated on the Diagnostic LEDs immediately at power-up. (See the **LED Control Register**).

Software clears all bits during a boot-up.

Figure 3-1 FPGA Revision Register @ Base + 00h

Bit 7	6	5	4	3	2	1	Bit 0
MAJ_REV				MIN_REV			
Bit	Field	Description			Value	Access	
3-0	MIN_REV	Minor Revision Level – Current minor hardware revision of the FPGA. This number increments for each minor hardware revision.			<minor_num>	R/O	
7-4	MAJ_REV	Major Revision Level – Current major hardware version of the FPGA. This number increments for each major hardware revision.			<major_num>	R/O	

Board ID and DIP Switch Register

The **Board ID and DIP Switch Register** is an 8-bit read-only register that indicates the configuration of the four hard straps on the WANic-5651x. This register also indicates the WANic-5651x boot mode.

Figure 3-2 Board ID and DIP Switch Register @ Base + 01h

Bit 7	6	5	4	3	2	1	Bit 0
RSVD		DIP_SW1	DIP_SW0	BOARD_ID 3	BOARD_ID 2	BOARD_ID 1	HEAT_SINK

Bit	Field	Description	Value	Access
0	HEAT_SINK	Heat Sink Presence – Indicates the type of heat sink present as passive or a fan.	0 = Fan 1 = Passive	R/O
3-1	BOARD_ID[1:0]	Board Identifier – Indicates the hard strap configuration as specified by the build configuration bits.	See Table 3-2	R/O
4	DIP_SW0	DIP Switch Indicator 0 – Indicates when the Diagnostic Utility is enabled.	0 = Enable (On) 1 = Disable (Off)	R/O
5	DIP_SW1	DIP Switch Indicator 1 – Selects the device from which the WANic-5651x is to boot.	0 = PCIe Bus 1 = Flash	R/O
7-6	RSVD	Reserved		

Table 3-2 Build Configuration Bit Values

Board_ID 3	Board_ID 2	Board_ID 1	Front End Configuration Description	Model
0	1	0	One 10G SPF+	WANic-56511
0	1	1	Two 10G SPF+	WANic-56512

LED Control Register

The **LED Control Register** is an 8-bit read/write register that controls the four diagnostic LEDs on the side of the card as shown in Figure 3-3.

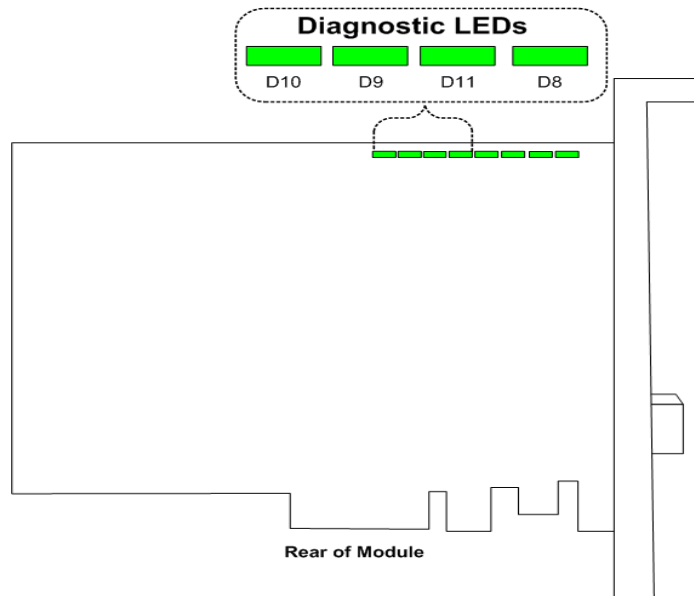
The Diagnostic LEDs power up to indicate the FPGA current minor revision (MIN_REV) immediately at power-up (before software loads the board.)

Figure 3-3 LED Control Register @ Base + 02h

Bit 7	6	5	4	3	2	1	Bit 0
RSVD				DIAG LED3	DIAG LED2	DIAG LED1	DIAG LED0

Bit	Field	Description	Value	Access
3-0	DIAG_LED[3:0]	Diagnostic LEDs - Controls the diagnostic LEDs by turning them on or off.	0 = Off 1 = On	R/W
7-4	RSVD	Reserved		

Table 3-3 Diagnostic LEDs on WANic-5651x



Interrupt Control Register

The **Interrupt Control Register** is an 8-bit read/write register that generates an interrupt upon completion of a transaction on any of the I2C interface ports.

Interrupt control bits function as masks for the corresponding bits in the **Interrupt Status Register**. When a bit in the **Interrupt Status Register** is set, the corresponding bit in the **Interrupt Control Register** is also set. Each register controls the generation of interrupts within its I/O register interface as a function of I2C transactions initiated within that register space. Neither register can control or examine the **Interrupt Control Register** mapped to the other's I/O space. All bits are set to zero by a hardware reset.

Figure 3-4 Interrupt Control Register @ Base + 03h

Bit 7	6	5	4	3	2	1	Bit 0
RSVD	TEMP_IEN	EEPROM_IEN	TEMP_I2C_IEN	RSVD		SFP1_IEN	SFP0_IEN

Bit	Field	Description	Value	Access
0	SFP0_IEN	SFP0 Interrupt Enable – When enabled, generates an interrupt to indicate a completed I2C transaction on Port 0.	0 = Disable 1 = Enable	R/W
1	SFP1_IEN	SFP1 Interrupt Enable – When enabled, generates an interrupt to indicate a completed I2C transaction on Port 1. (<i>WANic-56512 only</i>)	0 = Disable 1 = Enable	R/W
3–2	RSVD	Reserved		
4	TEMP_I2C_IEN	Temperature Sensor Interrupt Enable – When enabled, generates an interrupt to indicate a completed I2C transaction involving the temperature sensor.	0 = Disable 1 = Enable	R/W
5	EEPROM_IEN	EEPROM Interrupt Enable – When enabled, generates an interrupt to indicate a completed I2C transaction involving the serial EEPROM.	0 = Disable 1 = Enable	R/W
6	TEMP_IEN	Temperature Interrupt Enable – When enabled, generates a temperature sensor interrupt.	0 = Disable 1 = Enable	R/W
7	RSVD	Reserved		

Interrupt Status Register

The **Interrupt Status Register** is an 8-bit read/write register that indicates the status of a requested I2C transaction for selected devices. An independent **Interrupt Control Register** is available for each I/O bus space. When a bit is set, it indicates that the I2C transaction is completed with or without errors for the specified device. The host writes a 1 to each bit to clear it. A hardware reset also clears each bit.



NOTE

The **Interrupt Status Register** bits are *always* set (1) when the respective I2C transaction is completed regardless of the value of the bits in the **Interrupt Control Register**.

The **Interrupt Control Register** provides a masking function to determine whether interrupts are generated whenever the corresponding bit is set.

Figure 3-5 Interrupt Status Register @ Base + 04h

Bit 7	6	5	4	3	2	1	Bit 0
RSVD	TEMP_INT	EEPROM_INT	TEMP_I2C_INT	RSVD		SFP1_INT	SFP0_INT

Bit	Field	Description	Value	Access
0	SFP0_INT	SFP0 Interrupt – When set, indicates a completed I2C transaction involves Port 0.	0 = Normal 1 = Interrupt	R/W
1	SFP1_INT	SFP1 Interrupt – When set, indicates a completed I2C transaction involves Port 1. (WANic-56512 only)	0 = Normal 1 = Interrupt	R/W
3-2	RSVD	Reserved		
4	TEMP_I2C_INT	Temperature Sensor Interrupt – When set, indicates a completed I2C transaction involves the temperature sensor.	0 = Normal 1 = Interrupt	R/W
5	EEPROM_INT	EEPROM Interrupt Enable – When set, indicate a completed I2C transaction involves the serial EEPROM.	0 = Normal 1 = Interrupt	R/W
6	TEMP_INT	Temperature Interrupt Enable – When set, indicate a completed I2C transaction involves the temperature sensor.	0 = Normal 1 = Interrupt	R/W
7	RSVD	Reserved		

Peripheral Reset Register

The **Peripheral Reset Register** is an 8-bit read/write register that allows the Processor to force assertion of the *Reset* signal for selected on-card devices.



NOTE

PHY A maps to Port 0, and PHY B maps to Port 1.

Figure 3-6 Peripheral Reset Register @ Base + 05h

Bit 7	6	5	4	3	2	1	Bit 0
RSVD	CN56xx _RSTN	RSVD		PHY_B _RSTN	PHY_A _RSTN	DIMMs _RSTN	FLASH _RSTN

Bit	Field	Description	Value	Power Up Setting	Access
0	FLASH _RSTN	Flash Reset –When set to 0, issues a reset to the boot Flash.	0 = Reset 1 = Normal	1	R/W
1	DIMMs _RSTN	DIMMs Reset –When set to 0, issues a reset to the registered DIMM modules.	0 = Reset 1 = Normal	0	R/W
2	PHY_A _RSTN	PHY A Reset –When set to 0, issues a reset to the PHY A (Port 0) device.	0 = Reset 1 = Normal	0	R/W
3	PHY_B _RSTN	PHY B Reset –When set to 0, issues a reset to the PHY B (Port 1) device on the WANic-56512 only.	0 = Reset 1 = Normal	0	R/W
5-4	RSVD	Reserved			
6	CN56xx _RSTN	Multi-Core Processor Reset – When set, issues a reset to the Multi-Core Processor	0 = Reset 1 = Normal	0	R/W
7	RSVD	Reserved			

Transmit Control Register

The **Transmit Control Register** is an 8-bit read/write register that controls the front panel SFP+ transmitter output. A hardware reset sets all bits to zero (disable).

Figure 3-7 Transmit Control Register @ Base + 06h

Bit 7	6	5	4	3	2	1	Bit 0
RSVD						CH1 _EN	CH0 _EN

Bit	Field	Description	Value	Access
0	CH0 _EN	Channel 0 Enable - When set, enables transmitter output for SFP+ 0.	0 = Disable 1 = Enable	R/W
1	CH1 _EN	Channel 1 Enable - When set, enables transmitter output for SFP+ 1 on the WANic-56512.	0 = Disable 1 = Enable	R/W
7-2	RSVD	Reserved		

Temperature Status Register

The **Temperature Status Register** is an 8-bit read/write register that provides the status for the temperature monitor. This register provides the status for the die temperature as well as the temperature of the diode (PN) junction of the Processor.

Figure 3-8 Temperature Register @ Base + 07h

Bit 7	6	5	4	3	2	1	Bit 0
RSVD						TEMP_FAULT	RSVD

Bit	Field	Description	Value	Access
0	RSVD	Reserved		
1	TEMP_FAULT	Temperature Fault – When set, indicates that the temperature sensor has experienced a fault.	0 = Fault 1 = Normal	R/W
7–2	RSVD	Reserved		

I2C Registers

The WANic-5651x contains I2C compatible devices including:

- Front panel SFP+ I/O modules
- One EEPROM
- One temperature sensor

The FPGA provides a separate I2C port to interface with each device. Each I2C port has a separate set of four I/O registers that include the following:

- I2C Address High Register
- I2C Address Low Register
- I2C Data Register
- I2C Control Register

In response to requests from the Processor, the FPGA performs read and write transactions on each I2C port by means of these four registers.

The FPGA permits the Processor to simultaneously initiate transfers to the same device and automatically resolves simultaneous transfer requests. The FPGA carries out simultaneous transfers sequentially. An internal arbitrator performs the following:

- Determines which transfer to process first
- Coordinates the transfer of data between the respective I2C interface and the requesting device
- Notifies the requesting device when the transfer is complete

Each I2C transaction requires a specific sequence of commands for writing and reading data. The value written to the *Least Significant Bit* (LSB) of the **I2C Control Register** determines whether an I2C read or write transaction is requested.

During a transaction, the FPGA reports an I2C error in the **I2C Control Register** when the associated I2C device fails to maintain communications with the I2C interface; for example, when the host tries to access a register at a non-existent address.

Writing Data

To write data in an I2C transaction, perform the following steps:

1. Each I2C transaction requires a register address specification prior to initiating an I2C transaction. The host must specify this address by writing to the **I2C Address Register**. (**I2C High Address** and **I2C Address Low Registers**, when appropriate.)
2. Next, the host must specify the data to be transmitted by writing to the **I2C Data Register**. The **I2C Data Register** holds the data until the transaction completes.
3. After specifying the I2C address (and data), the host initiates an I2C transaction by writing to the **I2C Control Register** with the *Most Significant Bit* (MSB) set to 1. All other bits must be set to 0 (80h).

First, arbitration logic within the FPGA determines when the associated I2C port is idle. Then, the FPGA obtains the address and data and initiates the transaction.
4. To determine if the transaction is complete, the host can do the following:
 - Poll the content of the **I2C Control Register** or **Interrupt Status Register**.
 - Wait for an interrupt, if interrupts are enabled.
5. When the transaction completes, the **I2C Control Register** resets the *I2C_RUN* field (Bit 6) and the *I2C_ERR* field (Bit 7) updates.

Reading Data

To read data from an I2C transaction, perform the following steps:

1. Specify the address of the register that contains the data.
2. Initiate the I2C transaction by writing *81h* to the **I2C Control Register**.
3. Determine if the transaction is complete. The host can do the following:
 - Poll the content of the **I2C Control Register** or **Interrupt Status Register**.
 - Wait for an interrupt, if interrupts are enabled.
4. When the transaction completes, the **I2C Data Register** contains the data that was read from the specified I2C device. The host can now read the data from this register. The **I2C Control Register** resets the *I2C_RUN* field (Bit 6) and the *I2C_ERR* field (Bit 7) updates.

I2C registers descriptions are as follows.

I2C Control Register

This register is an 8-bit read/write register that contains I2C transaction status, activity, and error information. Write to the **I2C Control Register** to initiate a read or write transaction for the specified device. Read the **I2C Control Register** to determine the state of a previously initiated transfer.

The **I2C Control Register** address for specific devices is as follows:

<u>Device</u>	<u>Address</u>
Temperature Sensor	08h
EEPROM	0Dh
SFP+ 0	1Ch
SFP+ 1	18h

Figure 3-9 I2C Control Register @ Base + Device Address

Bit 7	6	5	4	3	2	1	Bit 0
I2C_ RUN	I2C_ _ERR	RSVD					I2C_ RW

Bit	Field	Description	Value	Access
0	I2C_ RW	I2C Transaction Mode – Initiates either an I2C read or write operation for the specified device. For both operations, the I2C_RUN field (Bit 7) must also be set to 'Run' (1) when writing to the I2C Control Register .	0 = Read 1 = Write	R/W
5–1	RSVD	Reserved		
6	I2C_ ERR	I2C Error – When set, indicates that the current I2C transaction completed with errors. This bit automatically reset to zero (no errors) on the next I2C transaction for the specified device.	0 = No errors 1 = Errors	R/W
7	I2C_ RUN	I2C Initiate Run – When set, initiates an I2C read or write transaction for the specified device.	0 = Inactive 1 = Run	R/W

I2C Address Low Register / I2C Address High Register

The **I2C Address Low Register** is an 8-bit read/write register that specifies an address of up to 8-bits. For addresses that exceed 8 bits, specify the upper bits in the **I2C Address High Register**.



NOTE

Values written in both registers must be right justified.

The **I2C Address Register** address for specific devices is as follows:

<u>Device</u>	<u>High Address</u>	<u>Low Address</u>
Temperature Sensor	Not applicable	0Bh
EEPROM	0Eh	0Fh
SFP+ 0	1Dh	1Eh
SFP+ 1	19h	1Ah

Figure 3-10 I2C Address Low Register @ Base + Low Address

Bit 7	6	5	4	3	2	1	Bit 0
I2C_A7	I2C_A6	I2C_A5	I2C_A4	I2C_A3	I2C_A2	I2C_A1	I2C_A0

Figure 3-11 I2C Address High Register @ Base + High Address

Bit 7	6	5	4	3	2	1	Bit 0
I2C_A15	I2C_A14	I2C_A13	I2C_A12	I2C_A11	I2C_A10	I2C_A9	I2C_A8

where **I2C_A[15:0]** is the device address.

I2C Data Register

The **I2C Data Register** contains the data associated with each I2C transaction. This register is an 8-bit read/write register. Figure 3-12 provides the format for this register.

- For write operations, specify the data by writing to this register *prior* to initiating the transaction.
- For read operations, this register contains data read from an I2C device when the transaction is completed.

The **I2C Data Register** address for specific devices is as follows:

<u>Device</u>	<u>Address</u>
Temperature Sensor	0Ch
EEPROM	10h
SFP+ 0	1Fh
SFP+ 1	1Bh

Figure 3-12 I2C Data Register @ Base + Device Address

Bit 7	6	5	4	3	2	1	Bit 0
I2C_D7	I2C_D6	I2C_D5	I2C_D4	I2C_D3	I2C_D2	I2C_D1	I2C_D0

where **I2C_D[7:0]** is data.

Boot Flash Upper Address Control Register

The **Boot Flash Upper Address Control Register** is an 8-bit read/write register that provides the upper address location for the boot Flash to break the 1 GB Flash into 16 MB pages.

Figure 3-13 Boot Flash Upper Address Control Register @ Base + 20h

Bit 7	6	5	4	3	2	1	Bit 0
RSVD					A26	A25	A24

Bit	Field	Description	Value	Access
0–2	A26–A24	Page number	000–111	R/W
7–3	RSVD	Reserved		

Miscellaneous Functions Register

The **Miscellaneous Functions Register** is an 8-bit read-only register that reports the several miscellaneous functions including the following:

- Controls the output of GPIO13.
- Sets the key memory to all zeros.
- Controls the diagnostic clock from the Processor.

At reset, each bit is set to zero.

Figure 3-14 Miscellaneous Functions Register @ Base + 21h

Bit 7	6	5	4	3	2	1	Bit 0
RSVD				QLM0 _REV_LANES	GPIO13 _OUT	ZERO _KEYS	MSC_ CLKOUT_EN

Bit	Field	Description	Value	Access
0	MSC_ CLKOUT_EN	Clock Output Enable – When set, enables the diagnostic clock out of the Processor running at core clock speed.	0 = Disable 1 = Enable	R/W
1	ZERO _KEYS	Zero Keys Enable – When enabled, sets the key memory to all zeros.	0 = Disable 1 = Enable	R/W
2	GPIO13_OUT _EN	GPIO13 Output – Controls GPIO13 data output between the Processor and FPGA.	0 = Low 1 = High	R/W
3	QLM0 _REV_LANES	QLM0 Lane Reversal – When enabled, assigns lane reversal for QLM0_REV_LANES.	0 = No reversal 1 = Lane reversal	R/W
7-4	RSVD	Reserved		

PHY Interrupt Register

The **PHY Interrupt Register** is an 8-bit read-write and read-only register that indicates the status of a PHY.

The read-only bits of this register report the current state of the interrupt bits directly from the PHY. The bits remain active until the issue is resolved at the PHY.

The interrupt generated by a PHY is a current indication of its status. As soon as an interrupt is serviced, the signal is de-asserted. Therefore, Bit 3 of this register updates the current state of the PHY interrupt every clock, or 20ns.

A hardware reset, sets all bits to 0.



NOTE

PHYA map to physical Port 0 and PHYB maps to Port 1.

Figure 3-15 PHY Interrupt Register @ Base + 22h

Bit 7	6	5	4	3	2	1	Bit 0
RSVD				NOT_ PHY_A_ INTN	NOT_ PHY_B_ INTN	RSVD	

Bit	Field	Description	Value	Reset	Access
1-0	RSVD	Reserved			
2	NOT_ PHY_B_ INTN	PHY B Interrupt Inverted – When a 1 is registered in the FPGA, this indicates an active interrupt from PHY B on the WANic-56512.	0 = De-asserted 1 = Asserted	0	R/O
3	NOT_ PHY_A_ INTN	PHY A Interrupt Inverted – When a 1 is registered in the FPGA, this indicates an active interrupt from PHY A.	0 = De-asserted 1 = Asserted	0	R/O
7-4	RSVD	Reserved			

4 • Software Description

The Processor on the WANic-5651x provides “embedded Linux-based” boot firmware and application software. The software manages the hardware, configuration, and monitoring of data processing, and also provides services to the host. The software executes on the WANic-5651x, which is accessible through a networking application programming interface (API). The API provides easy software integration for creating customized applications to configure and to monitor the WANic-5651x. The software makes it easy to configure a variety of signaling, gateway, real-time control, traffic processing, and network interface applications.

4.1 Software Overview

The WANic-5651x software executes on one or more cnMIPs64 cores, which are specified by the user. The SDK for the WANic-5651x consists of the following components, most of which are shown in Figure 4-1:

- U-Boot Bootloader – based on the Universal Bootloader (U-Boot) provides initialization and reset operations.
- Linux Support Package (LSP) – provides a suite of functions and drivers to access and to use the WANic-5651x hardware and provides a platform for the Control Plane.
- Linux Operating System – Debian™ distribution with Cavium OCTEON™ support.
- Diagnostic and Configuration Utility – is a Linux image that provides tests to configure and to verify the WANic-5651x hardware. (See [Chapter 5: LSP Applications](#) for more information.)
- GNU Tool-Chain – provides a variety of tools for building executables for the Processor.
- GNU Debugger – is a standard debugger for the GNU software that helps diagnose a program as it is executing.



NOTE

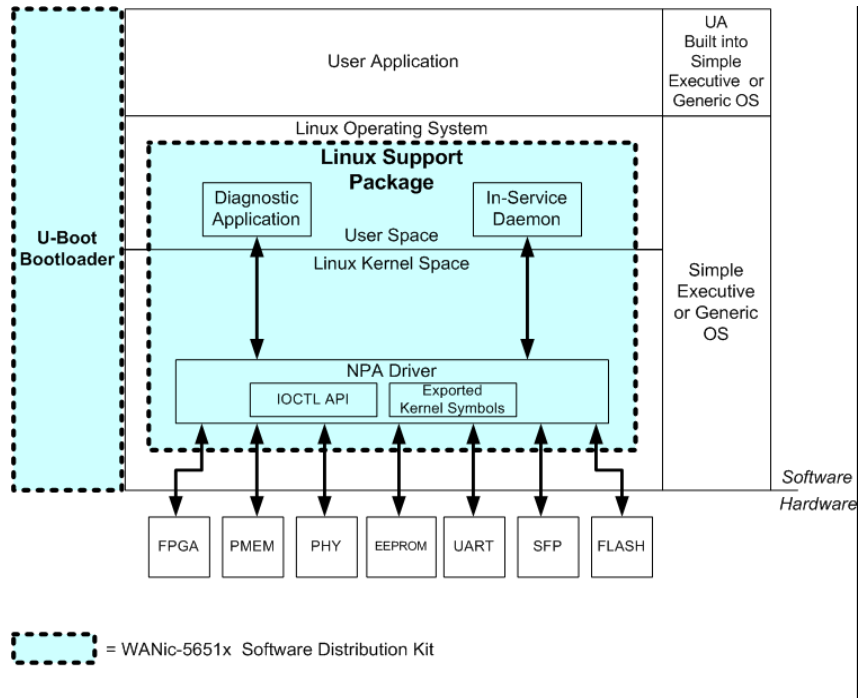
See [Appendix B: GNU General Public License](#) and [Appendix C: GE INTELLIGENT PLATFORMS EMBEDDED SYSTEMS, INC. and its subsidiaries COMPUTER DYNAMICS OF ILLINOIS, INC. Software License Agreement – Object Code](#) for software licensing information.

WANic-5651x software uses one of the following operating systems (which the user must obtain) to create the User Application:

- Linux Operating System, Version 2.6, or
- OCTEON Simple Executive, or
- Generic Operating System

See the Cavium Network Users site www.cnusers.org for more information on the OCTEON Simple Executive.

Figure 4-1 Software Components



4.2 Cavium Software Development Kit

The Cavium Software Development Kit (SDK) is included in the WANic-5651x distribution software CD-ROM for rapid development of the OCTEON Multi-Core Processor. The SDK includes drivers, libraries, files, and other tools to facilitate application development. The SDK is covered by the GPLv2 license (see [Appendix B: GNU General Public License](#) for more information.) This section briefly describes selected SDK features.

4.2.1 Cavium Ethernet Driver

The SDK provides the Cavium Ethernet Driver, which is contained in the following directory:

```
$OCTEON_ROOT/linux/kernel_2.6/linux/drivers/net/cavium-ethernet.
```

The default Cavium Linux kernel configuration builds the Cavium Ethernet Driver as a module. For instructions on building the embedded Linux kernel, see the installation instructions on the software CD-ROM.

After the Cavium Ethernet Driver is loaded, two new XAUI interfaces are available: XAUI0, XAUI1. These two ports correspond to the front panel ports in the following order:

- XAUI0 -> port 0
- XAUI1 -> port 1

Both XAUI ports function as standard Linux XAUI interfaces, and are configurable with Linux tools, such as `ifconfig` and `ethtool`.

When EtherPCI is enabled in the NPLSP configuration, the Ethernet PCI interfaces (`pci0` and `pci1`) are also available. For instructions on building and configuring Ethernet over PCI, see the installation instructions on the software CD-ROM.

4.2.2 Embedded File System

The Cavium SDK provides the Embedded File System, which is contained in the following directory on the software CD-ROM:

```
$OCTEON_ROOT/linux/embedded_rootfs
```

The Embedded File System is a RAM-based file system, which uses BusyBox (an open-source software application licensed under the *GNU General Public License*, version 2) as its base, and also supplies the following optional Linux applications and utilities:

- libpcap
- libpopt
- bridge-utils
- ethtool
- flex
- ipsec-tools
- iptables
- iproute2
- mii-tools
- mtd-tools
- net-tools
- openSSL
- pciutils
- readline
- strace
- tcpdump
- wireless-tools
- zlib
- OCTEON Utilities
- Toolchain Utilities

To configure the Embedded File System, run the following commands:

```
# cd $OCTEON_ROOT/linux/embedded_rootfs  
# make menuconfig
```

Now, configure the file system using the Configuration Menu. Once the configuration is complete, Save the changes and Exit.

For instructions on building the newly configured Embedded File System, see the installation instructions on the software CD-ROM.

4.2.3 Simple Executive Library

The Cavium SDK provides the Simple Executive Library, which is contained in the following directory on the software CD-ROM:

```
$OCTEON_ROOT/executive
```

The Simple Executive Library is a runtime library that also provides a hardware abstraction layer across all Octeon processors. For an overview of the Simple Executive Library, see the file:

```
$OCTEON_ROOT/executive/README.txt
```

Simple Executive sample applications are provided with this release, and contained in the directory:

```
$OCTEON_ROOT/cav-gefes/examples/simple_exec
```

For instructions on how to build the examples, see “[Section 4.7 Examples](#).”

4.3 User Application

The User Application (UA) is the software component that controls and manages processing. The WANic-5651x U-Boot loads and boots the UA. The UA operation is distributed across the multiple cores of the Processor.

The UA can be composed of:

- A unique set of tasks performed by each core, or
- Similar or identical tasks distributed across multiple cores in an attempt to balance core loading

The UA should always strive to prevent core spin-locks and idle time during high loading periods among the cores.

The UA can interface to the LSP when it configures or monitors the WANic-5651x hardware.

The UA can also interface to the LSP when it runs foreground or background diagnostics for WANic-5651x hardware testing. The UA can assert the LSP API for configuring, monitoring, programming, and testing operations of the WANic-5651x hardware.

4.4 U-Boot BootLoader

The WANic-5651x U-Boot Bootloader is based on the U-Boot open source multi-platform bootloader, which is used for a wide range of embedded processor architectures. The WANic-5651x U-Boot bootloader (hereafter referred to as U-Boot) supports interactive commands, environment variables, command scripting, and Flash located user application read/write. U-Boot is covered by the GPLv2 License. (See [Appendix B: GNU General Public License](#) for more information.)

4.4.1 Boot Process

The WANic-5651x contains two U-Boot images on the Flash:

- Failsafe – a small portion of this image always executes first to verify that the Normal image is present, and when present verifies the integrity of the Normal image. The Failsafe image continues to execute when the Normal image fails or is not present.
- Normal – boot image runs when verified by the Failsafe image.

During the boot process, U-Boot executes from Flash on Core 0 only. A system reset transfers execution to the Failsafe image immediately. The Failsafe image performs a checksum validation on the Normal image Flash sectors.

- If the checksum is valid, the Failsafe image transfers execution to the Normal image.
- If the checksum is not valid, the Failsafe image continues executing.

After reset, U-Boot transfers execution to RAM. In RAM, it conducts the operations dictated by the contents of EEPROM. U-Boot supports non-volatile configuration retrieval.

U-Boot software performs basic initialization of the WANic-5651x. When initialization is complete, U-Boot checks the application images configuration settings in NVRAM to determine which application images to load. The application images are decompressed from the network or Flash, and loaded into memory. When an application image is not found, U-Boot enters Command Line Mode and waits for more input.

Once the images are validated and loaded to memory, U-Boot transfers control to the UA software and also passes environment variables that are relevant to the application images. (Environment variables are described further in [Section 4.4.4 Environment Variables](#).) U-Boot provides special commands for displaying information and performing various operations. U-Boot commands are described in Table 4-2.

Tuples

U-Boot uses *tuples* to define and to store the configuration of the WANic-5651x in NVRAM. (A tuple provides a set of configuration definitions to pass from one application to another or to an operating system.) The WANic-5651x U-Boot extends the tuple configuration definitions to U-Boot commands and includes new tuples for the WANic-5651x as described in the section “[EEPROM Tuples](#).”

POST U-Boot also performs Power On Self Test (POST) operations after initialization. POST operations validate the hardware integrity.



NOTE

POST operations run only when Switch 4 on the DIP Switch S1 bank is On, or if the 'postall' environment variable is set. See [Section 2.8 Dual In-Line Package \(DIP\) Switch](#) for more information on the S1 DIP Switches.

The WANic-5651x provides the following short and long POST tests:

- Flash checksum – validates a Flash checksum on an individual sector or a collection of sectors
- RAM – validates RAM integrity
- PHY – validates PHY integrity
- PMEM – validates PMEM integrity
- DIMM0 – validates DIMM0 interface access via SPD
- DIMM1 – validates DIMM1 interface access via SPD

POST failure results in error code indications. The resulting action is determined by the associated test failure for that particular test. Failure action can include the following:

- System reset
- System halt
- System partial continuation to the U-Boot menu for manual debugging

After a power loss, POST results are stored in 16 bytes of EEPROM (as shown in Figure 4-3) to indicate the failures described in Table 4-1.

Table 4-1 POST Failures

Value	Description
FF	No error
00	Flash Checksum Failure
01	RAM Failure
02	PHY Failure
03	PMEM Failure
05	DIMM0 Failure
06	DIMM1 Failure
08	PMEM Failure

4.4.2 Building U-Boot

U-Boot can be configured for Flash-based execution or through the PCI Express interface.

Flash Boot

To build U-Boot for Flash boot, enter the following commands at the build system prompt in the U-Boot source directory :



NOTE

This booting method supports only a single executable image, which can be booted on Core 0 only.

1. Configure U-Boot to build for Flash-based execution:

Model	Image	Command
WANic-5651x	Failsafe	<code>make octeon_w56xx_failsafe_config</code>
	Normal	<code>make octeon_w56xx_config</code>

2. Build the Flash-based U-Boot image using the make command.

make

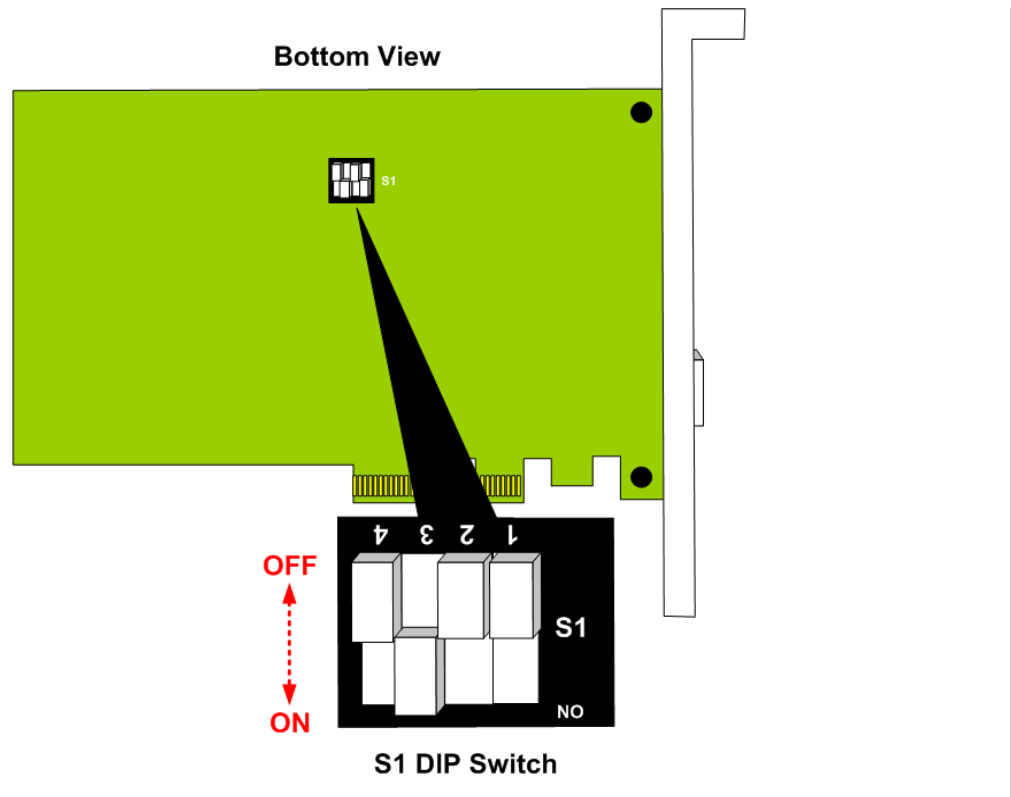
3. Using the Flash Menu in the Diagnostic Utility, copy the Flash-based boot image to the on-board Flash.

Re-Programming U-Boot

Re-program U-Boot using the S1 bank of DIP Switches as shown in Figure 4-2. To re-program U-Boot, perform the following steps:

1. Set DIP Switch 4 to the 'On' position.
2. Run the `npaDiag` application in the embedded Linux pre-loaded in the Flash.

Figure 4-2 S1 DIP Switch Location



4.4.3 U-Boot Commands

Table 4-2 lists U-Boot commands in alphabetical order. For more information on these commands, see the *DENX U-Boot and Linux Guide (DULG)* <http://www.denx.de/wiki/DULG/Manual>.

Table 4-2 U-Boot Commands

Command	Description
?	Displays online help.
askenv	Gets the environment variables from "stdin".
autoscr	Runs the Shell script under U-Boot from memory.
base	Displays or sets a base address that is used as an address offset for memory commands. The default value of the base address is 0.
bootelf	Boots from an ELF image in memory.
bootoct	Boots from the OCTEON Executive Elf image in memory.
bootoctelf	Boots a generic Elf image in memory. Note: This command does not support Simple Executive applications. Instead use the <code>bootoct</code> command.
bootoctlinux	Boots from a Linux ELF image in memory.
bootp	Boots the image over the network using the BOOTP/TFTP protocol.
cmp	Compares the contents of two memory areas.
coninfo	Displays information about the available console I/O devices. The output contains information such as device name, flags, current usage, and so forth.
cp	Copies memory areas.
crc32	Calculates a CRC-32 checksum over a range of memory.
dhcp	Invokes the DHCP client to obtain the IP/boot parameters.
defaultenv	Uses default environment variables.
echo	Displays the same argument typed on the console.
eeeprom	EEPROM sub-system.
erase	Erases the contents of one or more seconds of the Flash memory.
fatinfo	Prints information about the file system.
fatload	Loads binary file from a DOS file system.
fatloadalloca	Loads binary files from a DOS file system and allocates a named bootmem block for file data.
fatls	Lists files in a directory.
flinfo	Displays information for all available Flash memory banks.
fr	Reads the Flash.
freeprint	Prints a list of free bootmem blocks.
go	Starts standalone applications at address "addr."
gunzip	Un-compresses an in-memory gzipped file.
help	Displays online help. Without arguments, it displays a short listing of all available U-Boot commands.
ide	IDE sub-system.
itest	Returns true or false on an integer compare.
loadb	Loads a binary file over serial lines (kermit mode).

Table 4-2 (continued) U-Boot Commands

Command	Description
loop	Performs an infinite loop on an address range. To stop the loop, reset the card.
md	Displays memory contents both as hexadecimal and ASCII data.
mii	MII Utility command.
mm	Interactively modifies the memory contents.
mtest	Performs a simple Random Access Memory test. This test fails when applied to ROM or Flash.
mw	Initializes (fills) memory with some value.
namealloc	Allocates a named bootmem block.
namedfree	Frees a named bootmem block.
namedprint	Prints a list of named Boolean blocks.
netintstat	Obtains network interface status.
nm	Memory modify (constant address).
pcie <flag> <addr>	Accesses the PCI Express, where: <flag> = r(read, w(rite), or d(lump). <addr> = address For example, the command, <code>phy r 0x004</code> , reads from the PCI Express Configuration Register, PCIEEP_CFG001, located at address 0x004.
phy <flag> <phyaddr> <phyreg><value>	Accesses the PHY, where: <flag> = r(read, w(rite), or d(lump) <phyaddr> = Address of PHY <phyreg> = PHY register <value> = Write-only value For example, the command <code>phy w 0x40 17 0x1234</code> , writes to Phy Register 17 at address 0x40.
phyl <n> <location>	Toggles the Line or Mac loopback for internal Phy testing or toggles Transmit Enable for Phy testing, where: <n> = PHY number <location> = l(line) or m(ac) or t(ransmit enable)
ping	Sends ICMP ECHO_REQUEST to network host.
printenv	Displays one, several or all variables of the U-Boot environment.
protect	Enables or disables Flash protection. Sets certain parts of the Flash memory to read-only mode, or makes the Flash memory writable again.
rarpboot	Boots the image over the network using RARP/TFTP protocol.
read64	Reads a 64-bit word from 64-bit address.
read64b	Reads a 8-bit word from 64-bit address.
read64l	Reads a 32-bit word from 64-bit address
read64w	Reads a 16-bit word from 64-bit address
read_cmp	Reads and compares memory to value.
readi2c	Reads data from an I2C device.
readpci	Reads data from a PCI device.
readpmem	Reads data from Persistent Memory.
reset	Reboots the system.
run	Runs commands in an environment variable.
saveenv	Saves the environment variables to persistent storage.
sdump <reg>	Dumps status registers, where: <reg> = g(MX) or p(CX)

Table 4-2 (continued) U-Boot Commands

Command	Description
setenv	Sets the environment variables.
sfpinit	Initiates the SFPs.
sfpshow	Reads and displays current SFP installation.
sleep	Delays execution for a specified number of seconds (in decimal).
testmem	Tests memory.
testphy	Tests the PHY.
testpmem	Tests Persistent Memory.
tftpboot	Boots the image over the network using the TFTP protocol.
tlv_eeprom	EEPROM data parsing for the module.
validatenfi	Validates the Normal Flash image.
version	Displays the monitor version and build date of the U-Boot image running on your system.
write64	Writes 64-bit words to a 64-bit address.
write64b	Writes 8-bit words to a 64-bit address.
writel2c	Writes data to an I2C device.
write64l	Writes 32-bit word to a 64-bit address.
write64w	Writes 16-bit word to a 64-bit address.
writelpci	Writes data to a PCI device.
writelpmem	Writes data to Persistent Memory.

4.4.4 Environment Variables

Once the software image is validated and loaded to memory, U-Boot transfers control to the application software and also passes environment variables that are relevant to the application image. Environment variables also control the Flash-based application boot process. Environment variables can be modified from default values listed in Table 4-3, where 'h' represents a hexadecimal default value.

Table 4-3 Environment Variables

Variable	Default	Description
swfb_active	None	Activates the Flash application boot process.
swfb_cmd	bootoctlinux	Starts the application.
swfb_cmd_arg	coremask = 0FFF	Indicates a command argument list passed to the application.
swfb_flash_addr	b800 0000h	Identifies the Flash address of the application.
swfb_image_size	180 0000h	Indicates the application image size.
swfb_ram_addr	600 0000h	Provides the starting RAM address for the application to be copied and executed. When this value is not specified, no copy operation is performed and the image is executed from the Flash address.
mdio_alloc	Yes	Creates shared named alloc for mdio global lock required when using both npaDriver and Cavium-Ethernet driver together.
pci_console_active	<i>Not set by default</i>	Enables PCI console redirection. Use with oct-pci-console.
ethact	octeth0	Sets the current active Ethernet interface.
autoload	Yes	Auto loads DHCP and TFTP.
baudrate	115200	Defines the serial console baud rate.
bootloader_flash_update	protect off 0xb7e00000 0xb7efffff;erase 0xb7e00000 0xb7effrfff;cp.b \$(fileaddr) 0xb7e00000 0xf0000 validatenfi	Programs the Normal U-Boot image into Flash following TFTP transfer into memory. Use with 'run' command.



NOTE

If during the attempt to obtain Environment Variables, U-Boot encounters an invalid CRC, the environment variables are not retrieved from Flash and a default environment is used. A message similar to the following displays:

```
*** Warning - bad CRC, using default environment
```

To save the environment variables, perform the following steps:

1. Modify the environment.
2. Issue the U-Boot `saveenv` command to store the changes into non-volatile memory. The `saveenv` command burns the default RAM-based environment variables into Flash. The `saveenv` command also burns a CRC into Flash, which is used to test the integrity of the Flash-based IP Configuration Acquisition with DHCP.

4.4.5 PCI Console Redirection – U-Boot

PCI console redirection allows all serial output to be redirected to PCI host, where users can run PCI host tool to take serial input and display serial output.



NOTE

Host utils in Cavium CDK 1.8.1 are located in `host/pci/oct-pci-*` and host utils in CDK 1.9.0 are located in `host/remote-utils/pct-remote-*`.

To set U-Boot for PCI Console Redirection, perform the following steps:

1. Once U-Boot has loaded, set environment variable `pci_console_active` to 'yes'

```
# setenv pci_console_active yes
# saveenv
```

2. Reset the bootloader. U-Boot loads, and provides output to inform you: 'Using PCI console, serial port disabled.'
3. Perform this step only when using Cavium CDK 1.9.0. When using Cavium CDK 1.8.1, skip this step and continue with Step 4.

Before using any host remote tools, make sure that on the PCI host the `OCTEON_REMOTE_PROTOCOL` is set for the specified PCI target device

```
# export OCTEON_REMOTE_PROTOCOL=PCI:0
```

4. Run the remote pci console application to access U-Boot serial output:

```
# host/remote-tools/oct-remote-console --noraw 0
```

Output is now redirected to PCI host.

Use <CTRL+C> to break out (only in --noraw mode) from the PCI host.

4.4.6 PCI Console Redirection – Linux

PCI console redirection allows all serial output to be redirected to PCI host where user can run PCI host tool to take serial input and to display serial output.



NOTE

Host utils in Cavium CDK 1.8.1 are located in `host/pci/oct-pci-*` and host utils in CDK 1.9.0 are located in `host/remote-utils/pct-remote-*`.

To set Linux for PCI Console Redirection, perform the following steps:

1. Once U-boot has loaded, set environment variable `pci_console_active` to 'yes'

```
# setenv pci_console_active yes
# saveenv
```

2. Reset the bootloader. U-Boot loads, and provides output to inform you: 'Using PCI console, serial port disabled.'

3. **Perform this step only when using Cavium CDK 1.9.0. When using Cavium CDK 1.8.1, skip this step and continue with Step 4.**

Before using any host remote tools, make sure that on the PCI host the OCTEON_REMOTE_PROTOCOL is set for the specified PCI target device

```
# export OCTEON_REMOTE_PROTOCOL=PCI:0
```

4. When booting the Linux kernel image, pass in the kernel parameter console=pci to have all output redirected to PCI host:

```
# bootoctlinux 0 coremask=fff console=pci
```

5. Run the remote pci console application to access U-Boot serial output:

```
# host/remote-tools/oct-remote-console --noraw 0
```

Output is now redirected to PCI host.

Use <CTRL+C> to break out (only in --noraw mode) from the PCI host.

4.4.7 U-Boot Scripting

U-Boot allows the storage of commands or command sequences in an EEPROM tuple that executes during initialization. U-Boot provides storage for up to four scripts; however, only one script can be active at any given time.

Although scripting performs predefined operations during initialization, it does not provide error recovery procedures. Since error recovery can be critical to the successful load and boot of an application, the SIPI/PFI/SFI/LABI based application provides an alternative to scripting. To create and to update a script tuple in EEPROM, see [Section 4.4.11 EEPROM Tuple Update and Enumeration](#) in this chapter.

SIPI/PFI/SFI/LABI Application

U-Boot permits unique Source IP Information (SIPI) for each Ethernet interface. Additionally, U-Boot can apply the same source IP address information to all Ethernet interfaces. U-Boot primarily applies Primary File Information (PFI) and Secondary File Information (SFI), if necessary, to acquire the UA once the source IP address is determined. Then, U-Boot applies load and boot information (LABI) to load and boot the application on user-specified cores.

See the section in this chapter entitled “*EEPROM*” for information to create SIPI, PFI, SFI, and LABI tuples.

4.4.8 IP Configuration Acquisition with DHCP

U-Boot uses the networks Dynamic Host Control Protocol (DHCP) for IP configuration acquisition. The DHCP adds an Ethernet device to your network by automatically detecting and assigning an IP address to each connected device. However, some networks do not incorporate DHCP. These networks are referred to as ‘static’ and require you to assign an IP address to each device manually.

U-Boot asserts the DHCP protocol on a designated Ethernet port. U-Boot ceases DHCP assertion when the IP configuration is obtained. IP configuration includes the following:

- IP address
- Subnet mask
- Default gateway
- Default TFTP server

4.4.9 Automated UA Executable Load and Boot

Optionally, U-Boot supports loading the automated UA executable image into DDR2 RAM. U-Boot obtains an image from the removed Trivial File Transfer Protocol (TFTP) server of the local Flash memory. U-Boot supports the following application executable:

- OCTEON Simple Executive-Based ELF image
- Linux-Based ELF image
- Generic ELF image

U-Boot supports both:

- Unique image loading for each cnMIPS 64 core
- Single executive loading for multiple cores

The automated UA executable load is controlled by the configuration for each application executable. Up to 16 application executable entries can reside in the load table, By default, two entries are defined in Table 4-4. Within this table, primary and secondary file names are user-defined.

When the application executable is transferred completely into DDR2 RAM, U-Boot boots the specified cores.



NOTE

Core 0 is used for the execution of U-Boot only. Therefore, specify Core 0 as the last core in the load sequence. Subsequent loading is unavailable.

Table 4-4 Load and boot Image Entries

Load and Boot Image Entries	Data Plane Image	Controller Image
Primary location ID	TFTP Server	TFTP Server
Primary TFTP Ethernet Port ID	Port 0, GbE	Port 0, GbE
Primary TFTP server IP address	DHCP Server IP Address	DHCP Server IP Address
Primary TFTP retry frequency	5 seconds	5 seconds
Primary TFTP number of retries	3 times	3 times
Primary file name	<user-defined>	<user-defined>
Primary file type	Octeon Simple Executive	Linux-based
Secondary location ID	Flash	
Secondary TFTP Ethernet Port ID	Not applicable (N/A)	N/A
Secondary TFTP server IP address	N/A	N/A
Secondary TFTP retry frequency	N/A	N/A
Secondary TFTP number of retries	N/A	N/A
Secondary file name	<user-defined>	<user-defined>
Secondary file type	Octeon Simple Executive	Linux-based
DDR2 RAM Address Image Destination	0x7000000	0x7000000
DDR2 RAM Address Boot Location	0x7000000	0x7000000
Core Asset Mask	0xFFFFE – 15 cores [number limited to (max_cores -1) of part]	0x0001 – Core 0

4.4.10 EEPROM

The 64 KB serial EEPROM (hereafter referred to as EEPROM) provides non-volatile data storage for on-board specific hardware configuration information. After reset, U-Boot moves the software image to RAM and then transfers execution to RAM. From RAM, it conducts all subsequent operations. The content of the EEPROM dictates these operations. The EEPROM stores and supports the extended EEPROM tuples listed in Table 4-5.

EEPROM Mapping

The EEPROM provides three storage areas as show in Figure 4-3.

Figure 4-3 EEPROM Mapping

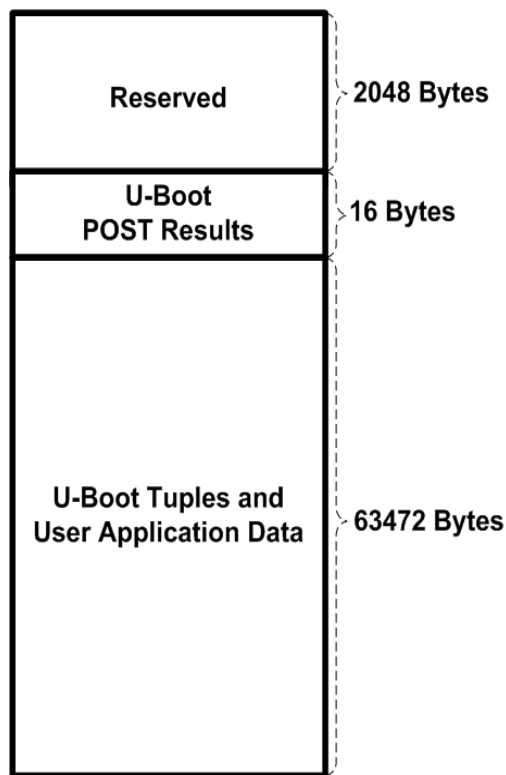


Table 4-5 summarized the new EEPROM tuples for the WANic-5651x U-Boot. These tuples supplement the tuple definitions to U-Boot commands.

Table 4-5 EEPROM Tuples

Tuple	Maximum	Description
CSUMTESTI	Up to four	Validates the Checksum test
LABI	Up to 16	Loads and boots information definitions
MEMTEST	Up to four	Creates a Memory test
PHYTESTI	One	Creates a PHY test
PFI	Up to 16	Obtains Primary File Information
SCRIPT	Up to four	Creates and updates a script
SCRIPTACTIVE	One	Indicates the active script
SFI	Up to 16	Obtains Secondary File Information
SIPI	Up to 10	Defines the Source IP address
UARTCONI	One	Changes the UART console data rate

Tuple Format

The WANic-5651x U-Boot uses the existing U-Boot storage format with a total tuple size of 128 bytes. The tuple consists of both a header and body. The header is defined as follows:

```
typedef struct {
    uint16_t type;
    uint16_t length;
    uint8_t minor_version; /*Minor version increment indicates backward compatible change*/
    uint8_t major_version; /*Major version must be changed when incompatible change made*/
    uint16_t checksum;
} octeon_eeprom_header_t;
```

The body of the tuple is available exclusively for the unique attributes of the specific tuple. For example, the LABI tuple is formed as follows:

```
/****** Load And Boot Information *****/
#define BOOT_COMMAND_MAX_LEN          32
#define IMAGE_ADDR_MAX_LEN           20
#define CORE_ASSERT_MASK_MAX_LEN      8
struct octeon_eeprom_load_and_boot_info_v1
{
    octeon_eeprom_header_t header;
    uint8_t boot_command[BOOT_COMMAND_MAX_LEN]; /* Boot Command */
    uint8_t image_addr[IMAGE_ADDR_MAX_LEN]; /* Image Address */
    uint8_t core_assert_mask[CORE_ASSERT_MASK_MAX_LEN]; /* Core Assert Mask */
};
#define OCTEON_EEPROM_LOAD_AND_BOOT_INFO_VER 1
typedef struct octeon_eeprom_load_and_boot_info_v1
    octeon_eeprom_load_and_boot_info_t
```

Displaying a Tuple Address

To display the address of a tuple, enter the command:

```
# tlv_eeprom display
```

The address of all the tuples display similar to the following:

```
=====
MAC_ADDR_TYPE (0x4) tuple found: at addr 0x810
type: 0x4, len: 0x10, csum: 0x1d7, maj_ver: 1, min_ver: 0
MAC base: 00:e0:48:26:01:6f, count: 4
=====
BOARD_DESC_TYPE (0x2) tuple found: at addr 0x820
type: 0x2, len: 0x24, csum: 0x420, maj_ver: 1, min_ver: 0
Board type: W5434 (0x4e26)
Board revision major:1, minor:0
Chip type (deprecated): Unsupported Chip (0x16f)
Chip revision (deprecated) major:4, minor:0
Board ser #: unknown
=====
UBOOT_FAILSAFE_INFO_TYPE (0x52) tuple found: at addr 0x844
type: 0x52, len: 0x50, csum: 0x89d, maj_ver: 2, min_ver: 0
Version Number: U-Boot GEF-3.0.1.v/SDK1.8.1-294
This U-Boot Version Is Inactive
FLASH Partition: UBOOT
    FLASH Offset(Bytes): 0x00000000(0KBytes)
    FLASH Length(Bytes): 0x001c0000(1792KBytes)
FLASH Partition: Environment
    FLASH Offset(Bytes): 0x001c0000(1792KBytes)
    FLASH Length(Bytes): 0x00020000(128KBytes)
FLASH Partition: NFI(Normal File Information)
    FLASH Offset(Bytes): 0x001e0000(1920KBytes)
    FLASH Length(Bytes): 0x00020000(128KBytes)
=====
UBOOT_NORMAL_INFO_TYPE (0x53) tuple found: at addr 0x894
type: 0x53, len: 0x50, csum: 0x8ff, maj_ver: 2, min_ver: 0
Version Number: U-Boot GEF-3.0.1.v/SDK1.8.1-294
This U-Boot Version Is Active
FLASH Partition: UBOOT
    FLASH Offset(Bytes): 0x00200000(2048KBytes)
    FLASH Length(Bytes): 0x001c0000(1792KBytes)
FLASH Partition: Environment
    FLASH Offset(Bytes): 0x003c0000(3840KBytes)
    FLASH Length(Bytes): 0x00020000(128KBytes)
FLASH Partition: NFI(Normal File Information)
    FLASH Offset(Bytes): 0x003e0000(3968KBytes)
    FLASH Length(Bytes): 0x00020000(128KBytes)
=====
```

Deleting a Tuple

To delete a tuple, obtain the address of the tuple then enter the following command:

```
# tlv-eeprom delete <address>
```

Descriptions of WANic-5651x EEPROM tuples follow.

Checksum Validation

Name CSUMTESTI

Prototype `tlv_eeeprom set csumtesti [id] [start] [end] [csum location] [csum algorithm][failure action]`

Parameters

<i>id</i>	Identifies which region to check, where: 1 = First 2 = Second 3 = Third 4 = Fourth
<i>start</i>	Starting address
<i>end</i>	Ending address
<i>csum location</i>	Checksum location
<i>csum algorithm</i>	Checksum algorithm, where: 0 = 8-bit checksum calculation with an 8-bit resultant 1 = 16-bit checksum calculation with a 16-bit resultant 2 = 32-bit checksum calculation with a 32-bit resultant 3 = 64-bit checksum calculation with a 64-bit resultant
<i>failure action</i>	Action that caused the failure, where: 0 = Console error message and continue as normal 1 = Console error message and system hangs

Description Validates the checksum and permits up to four regions of memory to be specified for checksum validation with each region containing a specific checksum. The [id] parameter specifies the region to check.

Example The following example illustrates a checksum validation region with:

- A starting address of 0xbfc00000.
- An ending address of 0xbfc2fff7.
- A checksum location of 0xbfc2fff8.
- A checksum algorithm of 3.
- A failure action of 0.

```
tlv_eeeprom set csumtesti 0 0xbfc00000 0xbfc2fff7 0xbfc2fff8 3 0
```

Load and Boot Information

Name	LABI								
Prototype	<code>tlv_eeeprom set labi [id] [boot command] [image address] [command arguments]</code>								
Parameters	<table><tr><td><code>id</code></td><td>Identifier</td></tr><tr><td><code>boot command</code></td><td>Boots an ELF image, where: <code>bootoctlinux</code> = Boots Linux ELF image <code>bootoct</code> = Boots the OCTEON Executive ELF image <code>bootelf</code> = Boots a generic ELF image</td></tr><tr><td><code>image address</code></td><td>Address where the image resides in RAM</td></tr><tr><td><code>command arguments</code></td><td>Input arguments</td></tr></table>	<code>id</code>	Identifier	<code>boot command</code>	Boots an ELF image, where: <code>bootoctlinux</code> = Boots Linux ELF image <code>bootoct</code> = Boots the OCTEON Executive ELF image <code>bootelf</code> = Boots a generic ELF image	<code>image address</code>	Address where the image resides in RAM	<code>command arguments</code>	Input arguments
<code>id</code>	Identifier								
<code>boot command</code>	Boots an ELF image, where: <code>bootoctlinux</code> = Boots Linux ELF image <code>bootoct</code> = Boots the OCTEON Executive ELF image <code>bootelf</code> = Boots a generic ELF image								
<code>image address</code>	Address where the image resides in RAM								
<code>command arguments</code>	Input arguments								
Description	Loads and boots information definition. The LABI tuple is stored in serial EEPROM. A LABI tuple is permitted for each core of the Processor (up to 12 cores.) The LABI is referenced when the associated PFI and SFI results in an application resident in memory, which is ready for activation on one or more Processor cores.								



NOTE

Enter a dash to indicate values for parameters that are not necessary.

Example The following example configures the LABI 0 for booting an application.
`tlv_eeeprom set labi 0 bootoctlinux 21000000 coremask=ff00`

In the above example:

- The `bootoctlinux` command boots the Linux Elf image.
- The image is located at address 21000000 (hex).
- The core mask directs U-Boot to use this image to boot cores 15–8. (Other arguments can be specified along with the `coremask`.)
- U-Boot terminates once Core 0 is loaded and booted with an executable image.

Memory Testing

Name	MEMTESTI												
Prototype	<code>tlv_eeeprom set memtesti [id] [start] [end] [csum location] [csum algorithm][failure action]</code>												
Parameters	<table><tr><td><i>id</i></td><td>Identifies which region to check, where: 0 = First 1 = Second 2 = Third 3 = Fourth</td></tr><tr><td><i>start</i></td><td>Starting address</td></tr><tr><td><i>end</i></td><td>Ending address</td></tr><tr><td><i>csum location</i></td><td>Checksum location</td></tr><tr><td><i>csum algorithm</i></td><td>Checksum algorithm, where: 0 = 8-bit checksum calculation with an 8-bit resultant 1 = 16-bit checksum calculation with a 16-bit resultant 2 = 32-bit checksum calculation with a 32-bit resultant 3 = 64-bit checksum calculation with a 64-bit resultant</td></tr><tr><td><i>failure action</i></td><td>Action that caused the failure, where: 0 = Console error message and continue as normal 1 = Console error message and system hangs</td></tr></table>	<i>id</i>	Identifies which region to check, where: 0 = First 1 = Second 2 = Third 3 = Fourth	<i>start</i>	Starting address	<i>end</i>	Ending address	<i>csum location</i>	Checksum location	<i>csum algorithm</i>	Checksum algorithm, where: 0 = 8-bit checksum calculation with an 8-bit resultant 1 = 16-bit checksum calculation with a 16-bit resultant 2 = 32-bit checksum calculation with a 32-bit resultant 3 = 64-bit checksum calculation with a 64-bit resultant	<i>failure action</i>	Action that caused the failure, where: 0 = Console error message and continue as normal 1 = Console error message and system hangs
<i>id</i>	Identifies which region to check, where: 0 = First 1 = Second 2 = Third 3 = Fourth												
<i>start</i>	Starting address												
<i>end</i>	Ending address												
<i>csum location</i>	Checksum location												
<i>csum algorithm</i>	Checksum algorithm, where: 0 = 8-bit checksum calculation with an 8-bit resultant 1 = 16-bit checksum calculation with a 16-bit resultant 2 = 32-bit checksum calculation with a 32-bit resultant 3 = 64-bit checksum calculation with a 64-bit resultant												
<i>failure action</i>	Action that caused the failure, where: 0 = Console error message and continue as normal 1 = Console error message and system hangs												
Description	Validates the checksum and permits up to four regions of memory to be specified for checksum validation with each region containing a specific checksum. The [id] parameter specifies the region to check.												
Example	<p>The following example illustrates a checksum validation region with:</p> <ul style="list-style-type: none">• A starting address of 0xBFC00000.• An ending address of 0xBFC2FFF7.• A checksum location of 0xBFC2FFF8.• A checksum algorithm of 3.• A failure action of 0. <pre>tlv_eeeprom set memtesti 0 0xbfc00000, 0xbfc2fff7, 0xbfc2fff8 3 0;</pre>												

PHY Testing

Name	PHYTESTI						
Prototype	<code>tlv_eeprom set phytesti [algorithm][device sel map][failure action]</code>						
Parameters	<table><tr><td><i>algorithm</i></td><td>Algorithm, where: the PHY presence detection test is: 0 = Enable 1 = Disable</td></tr><tr><td><i>device sel map</i></td><td>Device selection map is a bit map with the least significant bit corresponding to PHY0.</td></tr><tr><td><i>failure action</i></td><td>Action that caused the failure, where: 0 = Console error message and continue as normal 1 = Console error message and system hangs</td></tr></table>	<i>algorithm</i>	Algorithm, where: the PHY presence detection test is: 0 = Enable 1 = Disable	<i>device sel map</i>	Device selection map is a bit map with the least significant bit corresponding to PHY0.	<i>failure action</i>	Action that caused the failure, where: 0 = Console error message and continue as normal 1 = Console error message and system hangs
<i>algorithm</i>	Algorithm, where: the PHY presence detection test is: 0 = Enable 1 = Disable						
<i>device sel map</i>	Device selection map is a bit map with the least significant bit corresponding to PHY0.						
<i>failure action</i>	Action that caused the failure, where: 0 = Console error message and continue as normal 1 = Console error message and system hangs						
Description	Creates a test function for the PHY by detecting the PHY presence and returning the status of the test.						
Example	<p>The following example configures an eight PHY test that executes on Ports 0–1.</p> <pre>tlv_eeprom set phytesti 0 0x0003 0</pre> <p>where:</p> <ul style="list-style-type: none">• Algorithm 0 supports a PHY presence detection test.• The device selection map is a bit map with the least significant bit corresponding to PHY0.						

Primary File Information

Name	PFI														
Prototype	<code>tlv_eeeprom set pfi [id] [flash floc] [flash fsize] [SIPI ID] [TFTP fname] [TFTP serve IP addr] [TFTP num retries]</code>														
Parameters	<table><tr><td><i>id</i></td><td>Identifier</td></tr><tr><td><i>flash floc</i></td><td>Flash file location</td></tr><tr><td><i>flash fsize</i></td><td>Flash file size</td></tr><tr><td><i>SIPI ID</i></td><td>Source IP address</td></tr><tr><td><i>TFTP fname</i></td><td>TFTP filename</td></tr><tr><td><i>TFTP serve IP addr</i></td><td>TFTP server IP address</td></tr><tr><td><i>TFTP num retries</i></td><td>Number of retries for TFTP</td></tr></table>	<i>id</i>	Identifier	<i>flash floc</i>	Flash file location	<i>flash fsize</i>	Flash file size	<i>SIPI ID</i>	Source IP address	<i>TFTP fname</i>	TFTP filename	<i>TFTP serve IP addr</i>	TFTP server IP address	<i>TFTP num retries</i>	Number of retries for TFTP
<i>id</i>	Identifier														
<i>flash floc</i>	Flash file location														
<i>flash fsize</i>	Flash file size														
<i>SIPI ID</i>	Source IP address														
<i>TFTP fname</i>	TFTP filename														
<i>TFTP serve IP addr</i>	TFTP server IP address														
<i>TFTP num retries</i>	Number of retries for TFTP														
Description	Uses the TFTP to obtain the PFI definition from an TFTP server, or to use a Flash file. A PFI tuple is permitted for each core of the Processor (up to 12).														



NOTE

Enter a dash to indicate values that are not required.

Example The following example configures the PFI to acquire an application from a TFTP server.

```
tlv_eeeprom set pfi 0--0 vmlinux.64 10.71.1.11 4
```

In the above example:

- PFI is 0 if configured.
- A dash specifies the Flash file location and Flash File size since they are not needed.
- SIPI0 is the source IP address information.
- The TFTP server is specified as 10.71.1.11.
- The number of retries is 4.

Scripting

Name	SCRIPT				
Prototype	<code>tlv_eeeprom set script [id] [script command] { }{script command}.....</code>				
Parameters	<table><tr><td><i>id</i></td><td>Script ID</td></tr><tr><td><i>script command</i></td><td>Script command</td></tr></table>	<i>id</i>	Script ID	<i>script command</i>	Script command
<i>id</i>	Script ID				
<i>script command</i>	Script command				
Description	Creates and updates a sequence of U-Boot commands to execute during initialization. The WANic-5651x provides storage for up to four scripts; however, only one active script is permitted at any given time. See the <code>scriptactive</code> function for more information on enabling an active script.				
Example	<p>The following example creates a script tuple with two commands:</p> <pre>tlv_eeeprom set script 0 dhcp setenv serverip 10.71.1.11</pre> <p>Add subsequent commands to this script tuple as follows:</p> <pre>tlv_eeeprom set script 0 tftpboot 21000000 vmlinux.64</pre>				

Script Active

Name	SCRIPTACTIVE
Prototype	<code>tlv_eeeprom set scriptactive [id]</code>
Parameters	<i>id</i> Script ID
Description	Specifies the active script.
Example	<p>The following example creates a <code>Scriptactive</code> tuple with Script 0 as the active script:</p> <pre>tlv_eeeprom set scriptactive 0</pre> <p>The WANic-5651x executes the following commands from Script 0 during initialization:</p> <pre>dhcp setenv serverip 10.71.1.11 tftpboot 21000000 vmlinux.64</pre>



NOTE

See the `SCRIPT` tuple for more information on creating the script used in this example.

Source IP Address Information

Name	SIPI	
Prototype	<code>tlv_eeeprom set sipi [id] [ethernet portid] [DHCP enable] [DHCP num retries] [IP addr] [subnet mask] [default GW IP addr];</code>	
Parameters	<i>id</i>	Identifier
	<i>ethernet portid</i>	Ethernet Port ID
	<i>DHCP enable</i>	Enables or disables DHCP, where: 0 = Disable 1 = Enable
	<i>DHCP num retires</i>	Number of retires
	<i>IP address</i>	Internet Protocol address
	<i>subnet mask</i>	Subnet mask
	<i>default GW IP addr</i>	Default gateway internet protocol address
Description	Defines the source IP address information. A SIPI tuple is permitted for each WANic-5651x Ethernet interface.	



NOTE

The IP address, subnet mask, and default gateway are received from the DHCP server when DHCP is enabled.

Example The following example configures SIPI 0 for Ethernet 0 with DHCP enabled. The IP address, subnet mask, and default gateway are received from the DHCP server and are not entered as part of SIPI 0.

```
tlv_eeeprom set sipi 0 octeth0 1 0---
```

In the example above, the IP address, Subnet Mask, and Default Gateway entered with DHCP enabled overrides any value from the DHCP server.

The following example, configures a static IP address:

```
tlv_eeeprom set sipi 0 octeth0 0 0 10.72.1.21 255.255.255.0.0 10.72.1.1
```

In the above example:

- The ID is 0.
- The Ethernet port ID is `octeth0`.
- DHCP is enabled.
- The IP address is `10.72.1.21`.
- The subnet mask is `255.255.255.0`.
- The default gateway IP address is `10.72.1.1`.

Secondary File Information

Name	SFI												
Prototype	<code>tlv_eeeprom set sfi [<i>flash floc</i>] [<i>flash fsize</i>] [<i>SIPI ID</i>] [<i>TFTP fname</i>] [<i>TFTP serve IP addr</i>] [<i>TFTP num retries</i>]</code>												
Parameters	<table><tr><td><i>flash floc</i></td><td>Flash file location</td></tr><tr><td><i>flash fsize</i></td><td>Flash file size</td></tr><tr><td><i>SIPI ID</i></td><td>Source IP identifier</td></tr><tr><td><i>TFTP fname</i></td><td>TFTP file name</td></tr><tr><td><i>TFTP serve IP addr</i></td><td>TFTP server IP address</td></tr><tr><td><i>TFTP num retries</i></td><td>Number of retries for the TFTP</td></tr></table>	<i>flash floc</i>	Flash file location	<i>flash fsize</i>	Flash file size	<i>SIPI ID</i>	Source IP identifier	<i>TFTP fname</i>	TFTP file name	<i>TFTP serve IP addr</i>	TFTP server IP address	<i>TFTP num retries</i>	Number of retries for the TFTP
<i>flash floc</i>	Flash file location												
<i>flash fsize</i>	Flash file size												
<i>SIPI ID</i>	Source IP identifier												
<i>TFTP fname</i>	TFTP file name												
<i>TFTP serve IP addr</i>	TFTP server IP address												
<i>TFTP num retries</i>	Number of retries for the TFTP												
Description	Creates the SFI definition. A SFI tuple is permitted for each core of the Multi-Core Processor for up to 12 cores. The SFI is referenced when either the associated PFI is not defined, or the TFTP transfer associated with the PFI encounters an error condition.												
Example	<p>The following example configures SFI 0 for acquiring an allocation from Flash.</p> <pre>tlv_eeeprom set sfi bfc80000 80000 0--0</pre> <p>In the above example:</p> <ul style="list-style-type: none">• The Flash-based application file address must be specified (<i>bfc80000</i>).• The Flash-based application file size is specified (<i>80000</i>) because the application is to be transferred to RAM.• <i>SIPI 0</i> must be specified even though it is not reference for Flash-based operations.• A dash is specified for both the TFTP file name and TFTP server IP address as this information is not required.• The number of retries is <i>0</i>.												

UART Configuration

Name	UARTCONI
Prototype	<code>tlv_eeprom set uartconi [<i>baudrate</i>])</code>
Parameters	<i>baudrate</i> Baud rate
Description	Changes the console UART data rate by creating the UARTCONI tuple in non-volatile storage.
Example	The following example configures the console UART data rate for 115200 bps. <code>tlv_eeprom set uartconi 115200</code>

4.4.11 EEPROM Tuple Update and Enumeration

The WANic-5651x U-Boot updates the enumerated EEPROM tuple types as follows:

```
enum eeprom_types_enum {
    EEPROM_NULL_TYPE = 0,
    EEPROM_CLOCK_DESC_TYPE,
    EEPROM_BOARD_DESC_TYPE,
    EEPROM_CHIP_CAPABILITY_TYPE,
    EEPROM_MAC_ADDR_TYPE,
    EEPROM_VOLT_MULT_TYPE,
    EEPROM_NIC_XL_DESC_TYPE,
#ifdef CONFIG_OCTEON_wnpa38xx
/* Source IP Information */
    EEPROM_SIPI0_TYPE,
    EEPROM_SIPI1_TYPE,
    EEPROM_SIPI2_TYPE,
    EEPROM_SIPI3_TYPE,
    EEPROM_SIPI4_TYPE,
    EEPROM_SIPI5_TYPE,
    EEPROM_SIPI6_TYPE,
    EEPROM_SIPI7_TYPE,
    EEPROM_SIPI8_TYPE,
    EEPROM_SIPI9_TYPE,
/* Primary File Information */
    EEPROM_PFI0_IMAGE_TYPE,
    EEPROM_PFI1_IMAGE_TYPE,
    EEPROM_PFI2_IMAGE_TYPE,
    EEPROM_PFI3_IMAGE_TYPE,
    EEPROM_PFI4_IMAGE_TYPE,
    EEPROM_PFI5_IMAGE_TYPE,
    EEPROM_PFI6_IMAGE_TYPE,
    EEPROM_PFI7_IMAGE_TYPE,
    EEPROM_PFI8_IMAGE_TYPE,
    EEPROM_PFI9_IMAGE_TYPE,
    EEPROM_PFI10_IMAGE_TYPE,
    EEPROM_PFI11_IMAGE_TYPE,
    EEPROM_PFI12_IMAGE_TYPE,
    EEPROM_PFI13_IMAGE_TYPE,
    EEPROM_PFI14_IMAGE_TYPE,
    EEPROM_PFI15_IMAGE_TYPE,
/* Secondary File Information */
    EEPROM_SFI0_IMAGE_TYPE,
    EEPROM_SFI1_IMAGE_TYPE,
    EEPROM_SFI2_IMAGE_TYPE,
    EEPROM_SFI3_IMAGE_TYPE,
    EEPROM_SFI4_IMAGE_TYPE,
    EEPROM_SFI5_IMAGE_TYPE,
    EEPROM_SFI6_IMAGE_TYPE,
    EEPROM_SFI7_IMAGE_TYPE,
    EEPROM_SFI8_IMAGE_TYPE,
    EEPROM_SFI9_IMAGE_TYPE,
    EEPROM_SFI10_IMAGE_TYPE,
    EEPROM_SFI11_IMAGE_TYPE,
    EEPROM_SFI12_IMAGE_TYPE,
    EEPROM_SFI13_IMAGE_TYPE,
    EEPROM_SFI14_IMAGE_TYPE,
    EEPROM_SFI15_IMAGE_TYPE,
```

```

/* Load And Boot Information */
    EEPROM_LABI0_IMAGE_TYPE,
    EEPROM_LABI1_IMAGE_TYPE,
    EEPROM_LABI2_IMAGE_TYPE,
    EEPROM_LABI3_IMAGE_TYPE,
    EEPROM_LABI4_IMAGE_TYPE,
    EEPROM_LABI5_IMAGE_TYPE,
    EEPROM_LABI6_IMAGE_TYPE,
    EEPROM_LABI7_IMAGE_TYPE,
    EEPROM_LABI8_IMAGE_TYPE,
    EEPROM_LABI9_IMAGE_TYPE,
    EEPROM_LABI10_IMAGE_TYPE,
    EEPROM_LABI11_IMAGE_TYPE,
    EEPROM_LABI12_IMAGE_TYPE,
    EEPROM_LABI13_IMAGE_TYPE,
    EEPROM_LABI14_IMAGE_TYPE,
    EEPROM_LABI15_IMAGE_TYPE,
/* Script ID Active */
    EEPROM_SCRIPT_ID_ACTIVE_TYPE,
                                /* Script ID Active */
/* Script 0 */
    EEPROM_SCRIPT0_TYPE, /* Script */
/* Script 1 */
    EEPROM_SCRIPT1_TYPE, /* Script */
/* Script 2 */
    EEPROM_SCRIPT2_TYPE, /* Script */
/* Script 3 */
    EEPROM_SCRIPT3_TYPE, /* Script */
/* POST Information */
    EEPROM_CSUMTESTI0_TYPE,
    EEPROM_CSUMTESTI1_TYPE,
    EEPROM_CSUMTESTI2_TYPE,
    EEPROM_CSUMTESTI3_TYPE,
    EEPROM_MEMTESTI0_TYPE,
    EEPROM_MEMTESTI1_TYPE,
    EEPROM_MEMTESTI2_TYPE,
    EEPROM_MEMTESTI3_TYPE,
    EEPROM_PHYTESTI_TYPE,
    EEPROM_RLDRAMTI_TYPE,
    EEPROM_SERDESTESTI_TYPE,
/* UART-Console Information */
    EEPROM_UARTCONI_TYPE,
/* UART-MMC Information */
    EEPROM_UARTMMCI_TYPE,
#endif /* CONFIG_OCTEON_wnpa38xx */
    EEPROM_MAX_TYPE,
/* Start of range (inclusive) for customer use */
    EEPROM_CUSTOMER_RESERVED_START = 0xf000,
                                /* End of range (inclusive) for customer use */
    EEPROM_CUSTOMER_RESERVED_END = 0xff00,
    EEPROM_END_TYPE = 0xffff
};

```

4.5 Linux Support Package (LSP)

The LSP provides a suite of functions and tools for accessing and using the WANic-5651x hardware. Each component of the LSP is covered by the *GNU General Public License*, version 2 License. (See [Appendix B: GNU General Public License](#) for more information.)

The LSP supports the following WANic-5651x devices:

- EEPROM
- Flash
- GbE PHYs
- SFP+

The LSP contains the following functions, drivers, and APIs to assist in creating the UA:

- EEPROM Tuple Storage API
- EEPROM Error Code Storage and Retrieval API
- TWSI Primitives for I2C
- PHY/SFP Configuration and Status Functions API
- Flash Driver
- NPA Driver
 - Linux /proc/ File System
 - I/O control (IOCTL) API
- In-Service Daemon
- Diagnostic Application

The LSP provides functions to read and to write tuples into EEPROM. An EEPROM API allows the storage and retrieval of error codes during diagnostic testing.

Primitives for the industry standard Two-Wire Serial Interface (TWSI) provide I2C access on the WANic-5651x and support select devices.

The LSP provides an API for configuration and status of the PHY and SFPs.

The Flash Driver provides an API for all accessible hardware devices on the WANic-5651x.

The NPA Driver provides the API for communication with the on board WANic-5651x hardware. The NPA Driver also creates the /proc file. The /proc file provides access to hardware and firmware information. The NPA Driver also contains an IOCTL API, which allows a UA to communicate with the hardware.

The In-Service Daemon is a background process that monitors the hardware platform continuously and reports any faults detected.

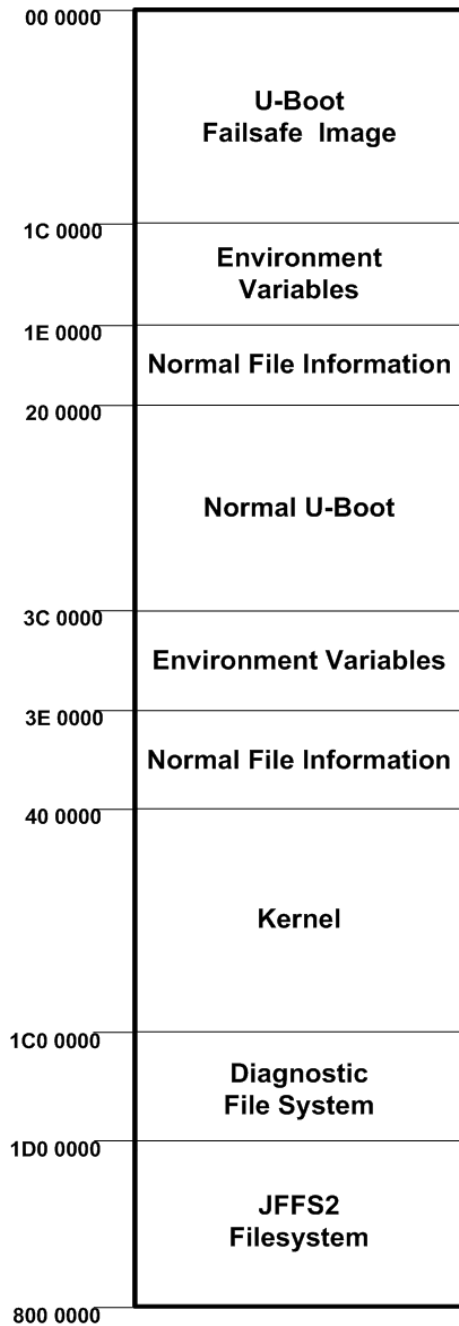
The Diagnostic Application provides in-service tests as well as U-Boot and Linux upgrade utilities.

4.5.1 Flash Driver

Flash Mapping

The WANic-5651x U-Boot maps the Flash on the Boot Bus/Chip Select 0. The Flash contains 128 MB. The base address is **17C0 0000** (B7C0 0000). Figure 4-3 shows the default factory-installed Flash mapping.

Figure 4-4 Flash Mapping



The Flash Driver provides a standard character device that allows information to be read, deleted, and retrieved from the Flash. Typically, this device is identified as `/dev/mtd0`.



NOTE

All paths are relative to the Linux Kernel source.

Flash Driver Functions

The Flash Driver functions in Table 4-6 are standard Linux system calls:

- Typically, these functions are defined in `include/sys/ioctl.h` and `include/unistd.h` in the Linux Kernel source.
- Micro `MEMGETINFO` and `MEMERASE` are found in the standard Memory Technology Device (MTD) subsystem for Linux micros and are defined in `include/mtd/mtd-abi.h`.
- Micro `SEEK_SET` is the standard file control micro. This function is defined in `include/fcntl.h`.
- Data structure `mtd_info_user` and `erase_info_user` are also defined in `include/mtd/mtd-abi.h`.

Table 4-6 Flash Driver Functions

Action	Functions
Get Flash Information	<code>int ioctl (int fd, MEMGETINFO, struct mtd_info_user * mtc);</code>
Erase	<code>int ioctl (int fd, MEMERASE, struct erase_info_user erase);</code>
Seek	<code>_off_t lseek (int fd, _toffset, SEEK_SET);</code>
Read	<code>ssize_t read(int fd, void * buf, size_t nbytes);</code>
Write	<code>ssize_t (int fd, void * buf, size_t n);</code>

Accessing the Flash

To access the Flash without creating programming code, use the `mtd_debug` function found in the MTD Utility. The MTD Utility is a collection of tools for various Flash devices found on <http://sourceware.org/jffs2/>.



NOTE

The `mtd_debug` function assumes that the input/output file is in binary format.

Use `mtd_debug` in the following format:

```
mtd_debug info <device>
mtd_debug read <device> <offset> <len> <dest-filename>
mtd_debug <device> <offset> <len> <source-filename>
mtd_debug erase <device> <offset> <len>
```

An example follows:

```
mtd-utils# mtd_debug info /dev/mtd0
mtd.type = MTD_NANDFLASH
mtd.flags = MTD_CAP_NANDFLASH
mtd.size = 8388608 (8M)
mtd.erasize = 8192 (8K) (Note: Must be a multiple of 8)
mtd.size = 512 (Note: Must be a multiple of 512)
mtd.ecctype = MTD_ECC_SW
regions = 0
```

Partitioning the Flash

By default, the Flash is partitioned at the factory as shown in Figure 4-3, and described in Table 4-7.

Table 4-7 Flash Partitioning

Size	Access	Description
1408 KB	Read-Only	U-Boot Failsafe and Normal images plus reserved
128 KB	Read-Only	U-Boot Environment Variables
40 MB	Read/Write	Compressed Kernel
Remaining MB (depending on the configuration)	Read/Write	JFFS2 File System

This partitioning layout can be customized. For example, it can be useful to create two dedicated Flash regions: one for read-only and one for write-only.

To customized Flash partitioning, use the `mtdparts` command line option. For example, the following command partitions the Flash into the factory default settings:

```
mtdparts=phys_mapped_flash:1408k(bootloader)ro,128k(bootloader_env)ro,40960k(kernel),-(jffs2)
```

To display the Flash partitioning, enter the `cat /proc/mtd` command from the Linux command line. Partitioning for the Flash displays similar to the following:

```
cat /proc/mtd
dev:   size  erasesize  name
mtd0: 00100000 00020000 "bootloader"
mtd1: 00020000 00020000 "bootloader_env"
mtd2: 02800000 00020000 "kernel"
mtd3: 016e0000 00020000 "jffs2"
```


Reprogramming the Linux Kernel in Flash

The LSP allows reprogramming of the Linux kernel in Flash:

- Automatically
- Manually
- Via the Diagnostic Utility

To reprogram the Linux Kernel in Flash, use one of the following methods:

- To manually re-program the Linux Kernel in Flash, perform the following steps:
 - a. Copy the new U-Boot image and determine the size, where size is X bytes.
 - b. Erase the Flash by entering:

```
mtd_debug erase /dev/mtd2 0 Y
```

where:
 - Y is the smallest number that is greater than X bytes, and Y is also a multiple of 131072.
 - 131072 is the Erase Size of the Flash. This means the specified Erase Size must be a multiple of 131072 when erasing *any part* of the Flash.
 - c. Write the new Linux Kernel image to Flash by entering:

```
mtd_debug write /dev/mtd2 0 <IMAGE> X
```

where:
 - $<IMAGE>$ is the new Linux Kernel image file name including full path
 - X is the size of the new Linux Kernel image
- Use the Diagnostic Utility (npaDiag) Flash Menu option. See “[Chapter 5: LSP Applications](#)” for more information.

Reprogramming U-Boot in Flash

The LSP allows you to reprogram the U-Boot Bootloader manually or via the Diagnostic Utility.

To reprogram the U-Boot Bootloader, use one of the following methods:

- To manually reprogram the U-Boot Bootloader, perform the following steps:
 - a. Copy the new U-Boot Bootloader image and determine the size, where size is X bytes.
 - b. Erase the Flash by entering:

```
mtd_debug erase /dev/mtd0 0 Y
```

where:
 - Y is the smallest number that is greater than X bytes, and Y is also a multiple of 131072.
 - 131072 is the Erase Size of the U-Boot Bootloader. This means the specified Erase Size must be a multiple of 131072 when erasing *any part* of the U-Boot Bootloader.
 - c. Write the new U-Boot Bootloader image to Flash by entering:

```
mtd_debug write /dev/mtd0 0 <IMAGE> X
```

where:
 - $<IMAGE>$ is the new Linux Kernel image file name including full path
 - X is the size of the new Linux Kernel image
- Use the Diagnostic Utility (npaDiag) Flash Menu option. See “[Chapter 5: LSP Applications](#)” for more information.

Running the Flash File System

To use the Flash file system, run:

```
mount -t jffs2 /dev/mtdblock3 MOUNT_POINT
```

where $MOUNT_POINT$ is the location in which to mount the Flash file system.

Refer to Cavium SDK document and <http://sourceware.org/jffs2/> for more information on JFFS2.

4.5.2 IOCTL Device Interface

The WANic-5651x IOCTL API supports commands that allow the UA to interface to the hardware. The IOCTL API within the NPA Driver supports the following:

- EEPROM tuples
- PHY devices
- LEDs
- TWSI (I2C access)
- PCIe
- U-Boot environment variables
- Port devices

The IOCTL code contains commands and structures to configure and to monitor all hardware features enabled on the WANic-5651x. Figure 4-5 and Table 4-8 list supported IOCTL commands and related structures, which can be sent to the NPA Driver through the IOCTL interface.

Figure 4-5 IOCTL Commands and Structures

```
enum eNpaCommands
{
_WRITE_TLV_TUPLE = 0, /**<Writes a TUPLE to the EEPROM*/
_DELETE_TUPLE,      /**<Deletes a TUPLE from the EEPROM*/
_GET_TLV_TUPLE,     /**<Retrieves TUPLE data from the EEPROM*/
_GET_NEXT_TUPLE,   /**<Retrieves the next TUPLE while enumerating TUPLE
                    values*/
_STORE_POST_ERROR, /**<Store a 8bit POST error code in the EEPROM*/
_GET_POST_ERROR,   /**<Retrieves all 16 8bit POST error codes from EEPROM*/
_CLEAR_POST_ERROR, /**<Clears all POST results from the EEPROM*/
_TWSI_WRITE,       /**<Writes a value to the specified TWSI device*/
_TWSI_READ,        /**<Reads a value from the specified TWSI device*/
_PHY_WRITE,        /**<Writes a value to the PHY registers*/
_PHY_READ,         /**<Reads a value from the PHY registers*/
_PHY_VS_WRITE,     /**<Writes a value to the SFP PHY registers*/
_PHY_VS_READ,      /**<Reads a value from the SFP PHY registers*/
_SET_LED_STATE,    /**<Sets the LED state*/
_GET_LED_STATE,    /**<Retrieves the current LED state*/
_DEVICE_COUNT,     /**<In PCI Target configuration, returns the number of
                    PCI devices found*/
_PEEK,             /**<In PCI Target configuration, peek at an address*/
_POKE,             /**<In PCI Target configuration, poke at an address*/
_GET_BOARD_INFO,   /**<Retrieves version information for the board*/
_SET_ENVIRONMENT,  /**<'setenv' for U-Boot/LSP environment variable*/
_GET_ENVIRONMENT,  /**<'getenv' for U-Boot/LSP environment variable*/
_GET_NEXT_ENVIRONMENT /**<Used to iterate through the environment list*/
_SET_PORT_ADMIN_STATE, /**<Sets a port state to 'up' or 'down'; allows up
                        on link or forces down on link*/
_GET_PORT_ADMIN_STATE, /**<Queries the current admin 'up' or 'down' state*/
_GET_PORT_STATE,      /**<Queries information about the port state*/
_SET_PORT_STATE,      /**<Queries information about the port state*/
_GET_PORT_STAT,       /**<Retrieves the statistic at index number specified*/
_GET_PORT_STAT_COUNT, /**<Retrieves a count of the number of statistics
                        that the port provides*/
_RESET_PORT_STATS,    /**<Resets the statistic counters*/
_SET_LOOPBACK,        /**<Sets the loopback state via NPA_LOOPBACK_INFO_t*/
_GET_LOOPBACK,        /**<Gets the loopback state via NPA_LOOPBACK_INFO_t*/
_NPA_DEV_EX_CMD = 0xF0 /**<Extended device specific commands*/
};
```

IOCTL Commands

Each of these enumerated commands translates to a hardware function. Table 4-8 details how each IOCTL command references its corresponding hardware functionality.

Table 4-8 IOCTL Commands

Command	Structure	Description
<code>_WRITE_TLV_TUPLE</code>	<code>NPA_TUPLE_INFO_t</code> filled structure describes the tuple to write	Creates a new tuple in EEPROM or replaces the tuple if one already exists.
<code>_DELETE_TUPLE</code>	<code>NPA_TUPLE_INFO_t</code> filled structure describes the tuple to delete	Deletes a tuple from EEPROM.
<code>_GET_TLV_TUPLE</code>	<code>NPA_TUPLE_INFO_t</code> filled structure describes the tuple to retrieve	Retrieves tuple data from the EEPROM. If a tuple is not found, this function returns with bit 31 set (negative).
<code>_GET_NEXT_TUPLE</code>	<code>NPA_TUPLE_INFO_t</code> filled structure describes the previous tuple retrieved. The tuple is filled with the next tuple data on return.	Retrieves the next tuple while enumerating tuple values. When the tuple sequence is exhausted, a zero returns.
<code>_STORE_POST_ERROR</code>	<code>POST_ERROR_e</code> set the POST code to store	Stores an 8-bit POST error code in EEPROM.
<code>_GET_POST_ERROR</code>	<code>POST_ERROR_e</code> retrieves all 16 post error codes.	Retrieves all 16 8-bit POST error codes from EEPROM.
<code>_CLEAR_POST_ERROR</code>	<code>void</code> Clears all 16 POST error codes	Clears or deletes all POST results from EEPROM.
<code>_TWSI_WRITE</code>	<code>TWSI_OP_t</code> filled structure describes the following: <i>dev_addr</i> —I2C device address <i>addr</i> —Memory location <i>data</i> —Write data	Write a value to the specified TWSI device.
<code>_TWSI_READ</code>	<code>TWSI_OP_t</code> filled structure describes: <i>dev_addr</i> —I2C device address <i>addr</i> —Memory location	Reads a value from the specified TWSI device.
<code>_PHY_WRITE</code>	<code>PHY_OP_t</code> filled structure describes: <i>interface</i> —Interface to assert [0-7] <i>reg</i> —Register number <i>mask</i> —Bit mask to assert during write operation <i>newVal</i> —New value to be written <i>wait</i> —Completion indicator	Writes a value to the SFP PHY registers.
<code>_PHY_READ</code>	<code>PHY_OP_t</code> filled structure describes: <i>interface</i> —interface to assert [0-7] <i>reg</i> —Register number <i>val</i> —Pointer to a register value	Reads a value from the SFP PHY registers.
<code>_PHY_VS_WRITE</code>	<code>PHY_OP_t</code> filled structure describes: <i>interface</i> —Interface to assert [0-7] <i>reg</i> —Register number <i>mask</i> —Bit mask to assert during write operation <i>newVal</i> —New value to be written <i>wait</i> —Completion indicator	Writes a value to the SFP PHY registers.
<code>_PHY_VS_READ</code>	<code>PHY_OP_t</code> filled structure describes: <i>interface</i> —Interface to assert [0-7] <i>reg</i> —Register number <i>val</i> —Pointer to a register value	Reads a value from the SFP PHY registers.
<code>_SET_LED_STATE</code>	Integer <code>LED_TYPE_e</code> flags indicate LED state to set	Sets the LED state.
<code>_GET_LED_STATE</code>	Filled with <code>LED_TYPE_e</code> Flags indicating snapshot state on return	Retrieves the current LED state.
<code>_DEVICE_COUNT</code>	Returns the number of boards in the system.	In PCI Target configuration, returns the number of PCI devices found.

Table 4-8 (continued) IOCTL Commands

<code>_PEEK</code>	<code>NPA_REG_RW_OP_t</code> <i>DevID</i> — Device IDs	In PCI Target configuration, peeks at an address.
<code>_POKE</code>	<code>NPA_REG_RW_OP_t</code> <i>DevIDx</i> —Device IDs	In PCI Target configuration, pokes at an address.
<code>_GET_BOARD_INFO</code>	<code>NPA_BOARD_INFO_t</code> structure Returns information about the LSP, FPGA, IPMC and MMC.	Retrieves version information for the board.
<code>_SET_ENVIRONMENT</code>	<code>NPA_ENV_ENTRY_t</code> structure Returns information about the environment variable specified in <i>envName</i> .	Initiates a 'setenv' for U-Boot/LSP environment variable.
<code>_GET_ENVIRONMENT</code>	<code>NPA_ENV_ENTRY_t</code> structure Sets the environment variable specified in <i>envName</i> to <i>envValue</i>	Initiates a 'getenv' for U-Boot /LSP environment variable.
<code>_GET_NEXT_ENVIRONMENT</code>	<code>NPA_ENV_ENTRY_t</code> structure Returns environment information for the next environment variable after the one specified in <i>envName</i> .	Iterates through the environment list.
<code>_SET_PORT_ADMIN_STATE</code>	<code>NPA_PORT_INFO_t</code> structure Sets the port specified by the port on the device specified by <i>devId</i> to the mode specified by <i>eAdminState</i> .	Sets a port state to 'up' or 'down'; Allows 'up' on link or forces 'down' on link.
<code>_GET_PORT_ADMIN_STATE</code>	<code>NPA_PORT_INFO_t</code> structure Returns the administrative state on the port specified by the port on the device specified by <i>devId</i> .	Queries the current admin 'up' or 'down' state.
<code>_GET_PORT_STAT</code>	<code>NPA_PORT_INFO_t</code> structure Return the statistic specified by <i>statIndex</i> . The name of the statistic at <i>statIndex</i> returns in <i>statName</i> , and the counter value returns in <i>statHigh</i> and <i>statLow</i> .	Queries and retrieves information about the port state.
<code>_SET_PORT_STATE</code>	<code>NPA_PORT_INFO_t</code> structure Sets the port specified by the port on the device specified by <i>devId</i> to the operational mode specified by <i>eOperState</i> .	Queries and sets information pertaining to the port state.
<code>_GET_PORT_STATE</code>	<code>NPA_PORT_INFO_t</code> structure Returns the operational mode of the port specified by the port on the device specified by <i>devId</i> .	Retrieves the statistic at the specified index number.
<code>_GET_PORT_STAT_COUNT</code>	<code>NPA_PORT_INFO_t</code> structure Returns the number of statistics available to the specified device.	Retrieves a count of the number of statistics that the port provides.
<code>_RESET_PORT_STATS</code>	<code>NPA_PORT_INFO_t</code> structure Resets the statistical counters for the device specified by <i>devId</i> .	Resets the statistic counters.
<code>_SET_LOOPBACK</code>	<code>NPA_LOOPBACK_INFO_t</code> structure Sets the port specified by the port on the device specified by <i>devId</i> to the loopback mode specified by <i>type</i> .	Sets the loopback state using <code>NPA_LOOPBACK_INFO_t</code> .
<code>_GET_LOOPBACK</code>	<code>NPA_LOOPBACK_INFO_t</code> structure Returns the loopback mode of the port specified by port on the device specified by <i>devId</i> .	Get the loopback state via <code>NPA_LOOPBACK_INFO_t</code> .
<code>_NPA_DEV_EX_CMD=0xF0</code>	<code>NPA_DEV_EX_CMD_t</code> structure Sends an uninterpreted IOCTL command to the device specified in <i>devId</i> .	Extended device specific commands = 0xF0.

IOCTL Command types are shown in Figure 4-6.

Figure 4-6 IOCTL Command Types

```

typedef struct _NPA_TUPLE_INFO {
    octeon_eeprom_header_t hdr;
    uint8_t data[ OCTEON_EEPROM_MAX_TUPLE_LENGTH ];
} NPA_TUPLE_INFO_t;

typedef struct _TWSI_OP {
    uint8_t device_id;/**< TWSI device ID */
    uint16_t addr;/**< address to perform operation on */
    uint16_t data;/**< data to write or data that was read */
    uint16_t mask;/**< mask to apply while writing data to address */
} TWSI_OP_t;

typedef struct _PHY_OP {
    uint8_t tidNum;/**< PHY device ID */
    uint16_t treg;/**< address to perform operation on */
    uint16_t tmask;/**< mask to apply while writing data to address */
    uint16_t tval;/**< data to write or data that was read */
} PHY_OP_t;

typedef struct
{
    uint8_t running_rev_major; /**< The IPMC currently running revision */
    uint8_t running_rev_minor; /**< The IPMC currently running revision */
    uint8_t active_rev_major; /**< The active IPMC image revision */
    uint8_t active_rev_minor; /**< The active IPMC image revision */
    uint8_t backup_rev_major; /**< The backup IPMC image revision */
    uint8_t backup_rev_minor; /**< The backup IPMC image revision */
    uint8_t failsafe_rev_major; /**< The failsafe IPMC image revision */
    uint8_t failsafe_rev_minor; /**< The failsafe IPMC image revision */
    uint8_t soak_counter; /**< The soak test counter number */
    uint8_t active_image_number; /**< The active image number */
    uint8_t hardware_rev; /**< The IPMC hardware revision number */
    uint8_t test_mode; /**< The IPMC test mode */
} NPA_IPMC_INFO_t;

typedef NPA_IPMC_INFO_t NPA_MMC_INFO_t;
typedef struct _NPA_BOARD_INFO {
    uint32_t driver_rev; /**< The version number for the active BSP
        version */
    uint32_t boot_normal_rev; /**< The version number for the UBOOT Normal
        Partition if available */
    int32_t boot_failsafe_rev; /**< The version number for the UBOOT Failsafe
        Partition if available */
    uint32_t fpga_rev; /**< The version number for the FPGA */
    uint8_t bPciHostMode; /**< Set to TRUE(1) if the BSP is running in
        PCI HOST Mode, FALSE(0) in PCI Target Mode*/
    uint16_t board_type; /**< Set to the value of the Board Module
        Type */
    uint8_t boot_normal_active; /**< Indicates if the Normal or Failsafe
        Partition is active */
    uint8_t boot_activate_flash;/**< Indicates if the Primary or Backup
        Flash is active */
    uint8_t npu_id; /**<The letter ID of the current Octeon NPU*/
    uint16_t mcr; /**< The master control register value */
    NPA_IPMC_INFO_t ipmc_info; /**< Contains all of the IPMC revision
        information and more */
    NPA_MMC_INFO_t mmc_info; /**< Contains all of the MMC revision
        information and more */
} NPA_BOARD_INFO_t;

typedef struct _NPA_REG_RW_OP {
    uint32_t devId; /**< specifies the device to perform the operation on */
    uint64_t addr; /**< specifies the address of the register */
    uint64_t data; /**< specifies the data qword to write or the data qword
        read */

```

```

uint64_t  mask;  /**<specifies the data mask used during register write */
} NPA_REG_RW_OP_t;

/**< The loopback IOCTL structure */
typedef struct _NPA_LOOPBACK_INFO_t {
eNpaFunction devId;      /**<specifies the device id to use for operation */
uint16_t  port;         /**<if a port is necessary, specifies port number*/
eNpaLoopbackTypes type;  /**< specifies the type of loopback */
} NPA_LOOPBACK_INFO_t;

typedef struct _NPA_PORT_INFO_t {
eNpaFunction devId;      /**<specifies the device id to use for operation */
uint16_t  port;         /**<if a port is necessary, specifies port number*/
eNpaOperStateeeOperState;  /**< Sets or gets the port operational state */
eNpaAdminStateeeAdminState;  /**< Sets or gets the port admin state */

uint32_t  statIndex;    /**< The count of statistics, or the statistic
                        to retrieve */
char      statName[128];  /**< The statistic friendly name */
int32_t  statHigh;      /**< The high 32 bits of the statistic */
uint32_t  statLow;     /**< The low 32 bits of the statistic */
} NPA_PORT_INFO_t;

typedef struct _NPA_ENV_ENTRY_t {
char envName[NPA_MAX_ENVNAME_SIZE];
char envValue[NPA_MAX_ENVVALUE_SIZE];
} NPA_ENV_ENTRY_t;

typedef struct _NPA_DEV_EX_CMD_HEADER_t {
NpaFunctiondevId;      /**< specifies the device id to use for the
                        operation\
                        */
int      cmd;
int      cmdLen;
} NPA_DEV_EX_CMD_HEADER_t;
#define NPA_DEV_EX_CMD_HEADER_SIZE sizeof(NPA_DEV_EX_CMD_HEADER_t)

typedef struct _NPA_DEV_EX_CMD_t {
eNpaFunction devId;      /**< specifies the device id to use for the
                        operation */
int      cmd;
int      cmdLen;
uint8_t  cmdBuf[(500 - sizeof(unsigned int)) - NPA_DEV_EX_CMD_HEADER_SIZE];
} NPA_DEV_EX_CMD_t;
#define NPA_DEV_EX_CMD_SIZE      sizeof(NPA_DEV_EX_CMD_t)

typedef struct _NPA_IOCTL_CMD {
unsigned int devNumber;
union {
NPA_TUPLE_INFO_t      TupleInfo;
TWSI_OP_t             TwsInfo;
PHY_OP_t              PhyOpInfo;
NPA_BOARD_INFO_t     BoardInfo;
NPA_LOOPBACK_INFO_t  LoopbackInfo;
NPA_PORT_INFO_t      PortInfo;
NPA_ENV_ENTRY_t       EnvInfo;
NPA_REG_RW_OP_t      RegInfo;
NPA_DEV_EX_CMD_t      DeviceExtendedInfo;
unsigned char         error_code[ 16 ];
unsigned char         ledState;
unsigned char         buffer[500 - sizeof(unsigned int)]; /* IOCTL structure is
always 512 bytes */
};
} NPA_IOCTL_CMD_t;

```

Proc File System

The `/proc/` file system provides access to information about the state of the WANic-5651x hardware and software.

Edit the `/proc/driver/` and `/proc/net/` directories to include information about the following:

- Basic XAUI PIP/PKO statistics
- Temperature sensor readings
- U-Boot environment variables
- Named allocated memory space

The `/proc/driver/` directory contains information for specific drivers in use by the kernel. The `/proc/net/` directory contains information about Linux networking.

Enter the following commands:

```
cat /proc/driver/gefes<x>/ethernetstats
cat /proc/driver/gefes<x>/temperature
where x =device number
```

The `/environment` `proc` entry is a writable file when setting environment variables from embedded Linux.

Values are added or change when echoed to the file:

```
echo var=val > /proc/drivers/gefes<x>/environment
where x = device number
```

When the LSP driver is configured for `PCI_HOST_MODE`, the named memory allocation is listed in the `proc` entry, `named_alloc_list`. Enter the following command:

```
cat /proc/driver/gefes<x>/named_alloc_list
where x = device number
```

Proc File Updates

The NPA Driver creates and updates the `/proc/` file system for temperature sensor readings, basic XAUI PIP/PKO statistics, U-Boot environment variable `named_alloc_list`, and Processor performance monitoring registers.

- Proc Ethernet, temperature and U-Boot environments configured for `PCI_HOST_MODE`:

```
/proc/drivers/gefes<x>/ethernetstats
/proc/drivers/gefes<x>/temperature
/proc/drivers/gefes<x>/environment
/proc/drivers/gefes<x>/named_alloc_list
where x = device number
```

- Proc Ethernet, temperature and U-Boot environment configured for `PCI_TARGET_MODE`:

```
/proc/drivers/gefes<x>/ethernetstats
/proc/drivers/gefes<x>/temperature
/proc/drivers/gefes<x>/environment
where x = device number
```

Exported Kernel Symbols

The NPA Driver also contains Exported Kernel Symbols, which are global kernel functions. Figure 4-7 identifies the kernel functions that are exported from the NPA Driver.

Figure 4-7 Exported Kernel Symbols.

```
int npaHwWdog_SetPreTimeout( int pretimeo );
int npaHwWdog_GetPreTimeout( void );
int npaHwWdog_SetTimeout( int timeo );
int npaHwWdog_GetTimeout( void );
int npaHwWdog_Pet( void );
int npaHwWdog_EnableWdog( int pretimeo, int timeo );
int npaWdog_SetNotifier( void (*wdog_event_handler)( void *, int cause ),
void * param );

#if defined(NPA_CONFIG_FAULT)
    int npa_bind_fault_reporter ( void (*fault_reporter) ( int size, void *
data_ptr ));
    int npa_unbind_fault_reporter ( void );
    int npa_set_fault_processing( eNpaFunction eFunctionID, int bEnable );
    int npa_bind_event_reporter ( void (*event_reporter) ( int size, void *
data_ptr ));

    int npa_unbind_event_reporter ( void );
#endif

struct npaObject_t; /* unused when called externally in PCI_HOST_MODE */
u64 npaGetBootCfgAddr( struct npaObject_t * devObj, unsigned int chipssel
);
```


4.5.3 IOCTL Operations for U-Boot Tuple and Environment

The UA interacts with U-Boot tuples and the U-Boot environment through the IOCTL operations. Each tuple operation listed in Table 4-9 uses the structure, `NPA_TUPLE_INFO_t`, to pass information from the UA to the driver. Each environment IOCTL uses the `NPA_ENV_ENTRY_t` structure to describe the environment values to read or write.

Table 4-9 Tuple/U-Boot Environment IOCTL Operations

Tuple	Operation
<code>_WRITE_TLV_TUPLE</code>	Adds the tuple information contained within the <code>NPA_TUPLE_INFO_t</code> structure.
<code>_DELETE_TUPLE</code>	Removes the information described by the tuple in <code>NPA_TUPLE_INFO_t</code> .
<code>_GET_TLV_TUPLE</code>	Retrieves information about the tuple type described by <code>NPA_TUPLE_INFO_t</code> .
<code>_GET_NEXT_TUPLE</code>	Iterates through all tuples. This operation retrieves information about the tuple that is stored after the one that is described by <code>NPA_TUPLE_INFO_t</code> and stores the new tuple's information in <code>NPA_TUPLE_INFO_t</code> .
<code>_SET_ENVIRONMENT</code>	Equivalent to the U-Boot <code>setenv</code> command line operation. This IOCTL adds the environment variable described by the <code>NPA_ENV_ENTRY_t</code> structure into the U-Boot environment. Setting an environment variable to an empty string removes it from the U-Boot environment.
<code>_GET_ENVIRONMENT</code>	Equivalent to the U-Boot <code>getenv</code> command line operation. This IOCTL retrieves the value associated with the environment variable described by <code>NPA_ENV_ENTRY_t</code> .
<code>_GET_NEXT_ENVIRONMENT</code>	Iterates through all environment variables. This operation retrieves information about the environment variable that is stored after the one that is described by <code>NPA_ENV_ENTRY_t</code> . To begin the iteration, set the variable name to the empty string.

4.5.4 IOCTL Register Access Operations

The UA can peek and poke low level register access to each of the devices that the LSP supports. Each IOCTL operation uses its relevant structure in the `NPA_IOCTL_CMD_t` structure.

- `_TWSI_WRITE`
- `_TWSI_READ`

Use the `TWSI_OP_t` structure to read from and to write to all devices connected to the FPGA I2C interface. These devices include those listed in Table 4-10.

Table 4-10 `TWSI_OP_t` Device IDs:

Name	Value
<code>BOARD_EEPROM_TWSI_ADDR</code>	0x52
<code>BOARD_SFP0_TWSI_ADDR</code>	0x53
<code>BOARD_SFP1_TWSI_ADDR</code>	0x54
<code>BOARD_SFP2_TWSI_ADDR</code>	0x55
<code>BOARD_SFP3_TWSI_ADDR</code>	0x56
<code>BOARD_TEMP_SENSOR_TWSI_ADDR</code>	0x57
<code>BOARD_RTM_SFP0_TWSI_ADDR</code>	0x70
<code>BOARD_RTM_SFP1_TWSI_ADDR</code>	0x71
<code>BOARD_RTM_SFP2_TWSI_ADDR</code>	0x72
<code>BOARD_RTM_SFP3_TWSI_ADDR</code>	0x73
<code>BOARD_RTM_SFP4_TWSI_ADDR</code>	0x74
<code>BOARD_RTM_SFP5_TWSI_ADDR</code>	0x75
<code>BOARD_RTM_SFP6_TWSI_ADDR</code>	0x76
<code>BOARD_RTM_SFP7_TWSI_ADDR</code>	0x77
<code>BOARD_RTM_SFP8_TWSI_ADDR</code>	0x78
<code>BOARD_RTM_SFP9_TWSI_ADDR</code>	0x79
<code>BOARD_RTM_SFP10_TWSI_ADDR</code>	0x7A
<code>BOARD_RTM_SFP11_TWSI_ADDR</code>	0x7B
<code>BOARD_BASE0_ADDR</code>	0x80
<code>BOARD_BASE1_ADDR</code>	0x81
<code>BOARD_10G_RTM_10G_DATA_SFP0_ADDR</code>	0x70
<code>BOARD_10G_RTM_10G_DATA_SFP1_ADDR</code>	0x71
<code>BOARD_10G_RTM_1G_DATA_SFP0_ADDR</code>	0x72
<code>BOARD_10G_RTM_1G_DATA_SFP1_ADDR</code>	0x73
<code>BOARD_10G_RTM_1G_CONTROL_SFP0_ADDR</code>	0x7A
<code>BOARD_10G_RTM_1G_CONTROL_SFP1_ADDR</code>	0x7B

4.5.5 IOCTL POST Error Operations

The UA can interact with stored POST information through the IOCTL operations listed in Table 4-11. Each operation returns or sets information using the `error_code[16]` parameter.

Table 4-11 POST Error Operations

Tuple	Operation
<code>_STORE_POST_ERROR</code>	Adds the post error code described by <code>error_code[0]</code> to the POST error store.
<code>_GET_POST_ERROR</code>	Retrieves all 16 available POST error history values and stores them in <code>error_code[]</code> .
<code>_CLEAR_POST_ERROR</code>	Clears the history of all POST error codes.

4.5.6 IOCTL Port Management

Management of the port devices on the WANic-5651x occurs through two IOCTL avenues:

- Standard Linux network IOCTL interface exposed with ETHTOOL IOCTLs (This is the primary control interface to the device.)
- Extended LSP port IOCTLs

Devices on the WANic-5651x supports the following standard IOCTL interface operations:

- ETHTOOL IOCTL operations supported include those listed in Table 4-12:

Table 4-12 ETHTOOL IOCTL Operations

Operation	Description
self_test_count	Retrieves the number of self tests that the driver supports.
self_test	Runs the self test specified by the IOCTL.
get_drvinfo	Retrieves information about the driver and the supported hardware.
get_strings	Retrieves user-readable strings for the statistic counters.
get_stats_count	Retrieves the number of statistic counters supported by the driver.
get_ethtool_stats	Retrieves the statistic counter referenced by the IOCTL.
get_link	Retrieves the current link state.

- Linux IOCTL operations support include those listed in Table 4-13:

Table 4-13 Linux IOCTL Operations

Operation	Description
change_mtu	Alters the MTU size accepted by the device.

IOCTL Extended Port Control

In addition to the ETHTOOL and Linux IOCTL interfaces, the LSP also provides the following additional IOCTLs for extended port control.

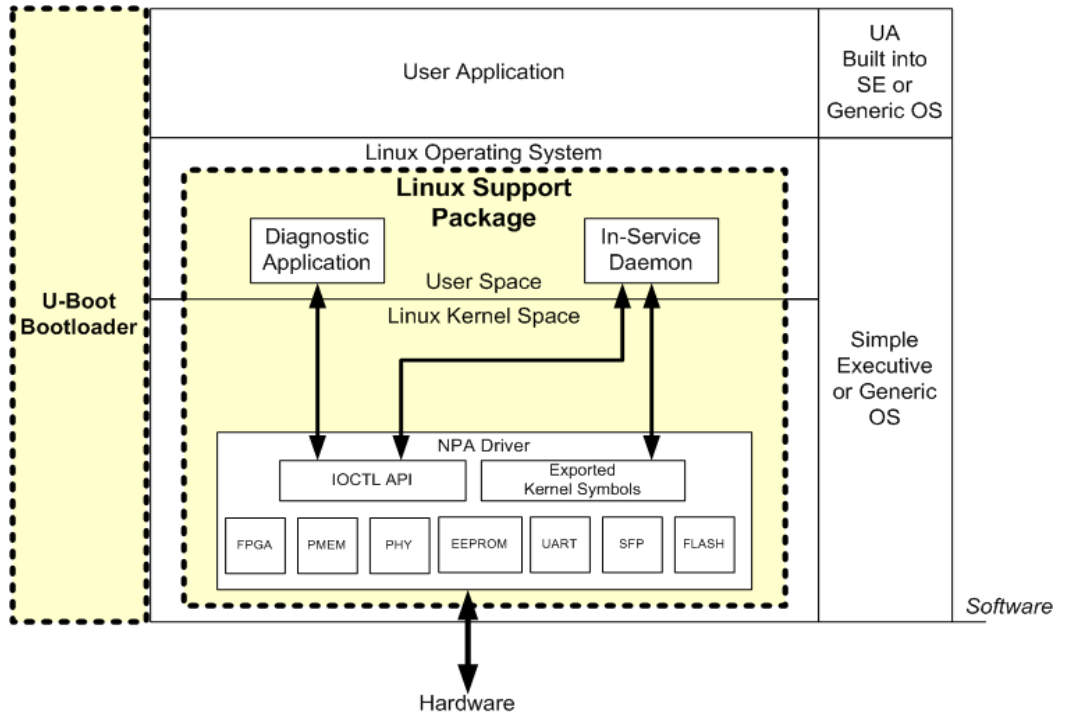
- `_SET_PORT_ADMIN_STATE`, `_GET_PORT_ADMIN_STATE` using the `NPA_PORT_INFO_t` structure – Allows the user to administratively force an interface link to be up or down. Forcing a link down causes it to *not drive* a carrier signal onto the network. The eDevMarvell supports this operation. Any action performed on these devices reflects automatically in any connected SFP or base interface. For example, administratively forcing eDevMarvell Port 0 down causes the front panel SFP 0 to be administratively forced down.
- `_RESET_PORT_STATS` – Allows the user to reset the statistics that are reported through the ETHTOOL IOCTL commands.

4.5.7 NPA Driver

The NPA Driver is a Linux kernel module that provides the API for communications with the WANic-5651x on-board hardware, and also creates and updates the `/proc` file directory. The `/proc` directory contains a hierarchy of special files, which represent the current state of the kernel and allows applications and users to per into the kernel's view of the system.

The NPA Driver contains the IOCTL API and Exported Kernel Symbols as shown in Figure 4-8.

Figure 4-8 NPA Driver in LSP



4.6 Additional LSP Features

The LSP provides the following additional features:

- Interrupt generation
- Exception processing
- In-Service testing
- Memory mapping
- Testing POST results

Interrupt Generation

The LSP provides hardware configuration for interrupt generation caused by hardware failures that do not result in errors through exception processing. The LSP introduces all interrupt request (IRQ) definitions and Linux IRQ handling infrastructures changes required to permit the binding of Linux IRQ handlers into the kernel through the standard `request_irq` interface.

Exception Passing

The LSP provides processor exception handling to the point of full trace back dumps to the standard Linux log.

In-Service Testing

The LSP provides in-service tests for each major device. It provides a test plan, which lists executed test cases with test results to verify complete functionality and known errata list. It also has an automated test suite that verifies functionality.

Memory Mapping

The LSP provides a memory map, which identifies all reserved areas that are not available to Linux. It identifies the differences between physical memory and that which is reported as memory.

POST Results

The LSP also supports retrieval of POST results from non-volatile memory.

Direct Attach Mode

LSP software provides Direct Attach Mode for 10GbE SFP+ front panel connectors. Direct Attach Mode, detects when a cable is attached to tan SFP+ connector and sets up the interface for the SFP+ automatically.

4.7 Examples

For your convenience, this section describes Linux and Simple Executive application examples, which are also available on the software CD-ROM in the following directories:

- `$OCTEON_ROOT/cav-gefes/examples/linux`
- `$OCTEON_ROOT/cav-gefes/examples/simple_exec`

4.7.1 Linux Applications Examples

The `$OCTEON_ROOT/cav-gefes/examples/linux` directories contains folders for the following two Linux application examples:

- Hello
- Board Information (boardinfo)

Example 1: Hello

This is a basic 'Hello World' Linux application. To build this example, enable the 'Examples' option in the NPLSP Configuration Menu, or run the 'make all' command from the '`$OCTEON_ROOT/cav-gefes/examples/linux/hello`' directory.

When successfully compiled, the binary 'Hello' is built in the current directory, as well as copied into the 'Extra-files' directory of the embedded file system.

After rebuilding the embedded kernel, run the command '`/bin/hello`' command. Output displays as follows:

```
~ # /bin/hello
Hello world
```

Example 2: Board Information

This example performs simple open, ioctl, and close calls to the NPA driver, and displays current information about the board.

To build this example, enable the 'Examples' option in the NPLSP Configuration Menu, or run the 'make all' command from the '`$OCTEON_ROOT/cav-gefes/examples/linux/boardinfo`' directory.

When successfully compiled, the binary 'boardinfo' is built in the current directory, as well as copied into the 'Extra-files' directory of the embedded file system.

After rebuilding the embedded kernel, run the command, '`/bin/boardinfo.`' Output displays similar to the following:

```
~ # /bin/boardinfo
Running LSP Version      : 1.22.0
Running UBoot Failsafe  : 0.0.0
Running UBoot Normal    : 3.1.0
Running FPGA Version    : 2.2
Running in PCI mode     : host
Running on NPU          : A
```

4.7.2 Simple Exec Application Examples

The '\$OCTEON_ROOT/cav-gefes/examples/simple_exec' directories contains folders for two Simple Executive application examples:

- Hello
- Intercept

Example 3: Hello

This example application performs a simple loop on all active cores displaying Hello World with an incrementing counter.

To build this example, enable the 'Examples' option in the NPLSP Configuration Menu, or run the 'make all' command from the '\$OCTEON_ROOT/cav-gefes/examples/simple_exec/hello' directory.

When successfully compiled, the binary 'hello_exe' is built in the current directory. To run Simple Executive applications, perform the following steps:

1. Copy the 'hello_exe' Simple Executive application to a TFTP server
2. Set a static IP address, or obtain an IP address through DHCP from U-Boot:

```
# dhcp
```

or

```
# set ipaddr <i.p. address>
```
3. Set the TFTP server IP address in U-Boot, for example:

```
# set serverip 192.168.1.1
```
4. From U-Boot, transfer the simple exec binary to the target system:

```
# tftpboot 20000000 hello_exe
```
5. Boot the simple exec application:

```
# bootoct 20000000 coremask=fff
```

Output similar to the following displays approximately once every second:

```
PP0:~CONSOLE-> Core[0]: Hello world - 0
PP2:~CONSOLE-> Core[2]: Hello world - 0
PP4:~CONSOLE-> Core[4]: Hello world - 0
PP7:~CONSOLE-> Core[7]: Hello world - 0
PP9:~CONSOLE-> Core[9]: Hello world - 0
PP10:~CONSOLE-> Core[10]: Hello world - 0 PP6:~CONSOLE->
Core[6]: Hello world - 0 PP5:~CONSOLE-> Core[5]: Hello world
- 0 PP8:~CONSOLE-> Core[8]: Hello world - 0 PP11:~CONSOLE->
Core[11]: Hello world - 0 PP1:~CONSOLE-> Core[1]: Hello world
- 0 PP3:~CONSOLE-> Core[3]: Hello world - 0
```


Example 5: Intercept

This application example intercepts incoming packets to any of the Cavium Ethernet ports, and drops the packets if the size exceeds a certain limit.

The application starts up and waits for Linux to load the Cavium Ethernet driver. This application runs in conjunction with a booted Linux kernel and the Cavium Ethernet driver, and does not work properly unless the Cavium Ethernet driver is loaded.

When the driver is loaded, all the Cavium Ethernet ports are configured so that all incoming packets are received by the application instead of Linux.

Then, all packets display on the screen, and any packets exceeding the maximum size limit are dropped. Packets within the maximum size limit are forwarded on to Linux.

To build this example, enable the 'Examples' option in the NPLSP Configuration Menu, or run the 'make all' command from the '\$OCTEON_ROOT/cav-gefes/examples/simple_exec/intercept' directory.

When successfully compiled, the binary 'intercept_exe' is built in the current directory. To run the Simple Executive applications, perform the following steps:

1. Copy the 'intercept_exe' Simple Executive application to a TFTP server.
2. Copy the embedded kernel image to a TFTP server.
3. Set a static IP address or obtain an IP address through DHCP from U-Boot:

```
# dhcp
```

4. Set the TFTP server ip address in U-Boot, for example:

```
# set serverip 192.168.1.1
```

5. From U-Boot, transfer the simple exec binary to the target system:

```
# tftpboot 30000000 intercept_exe
```



NOTE

Ignore messages that indicate that data is being loaded outside of the reserved load area.

6. From U-Boot, transfer the Linux kernel to the target system:

```
# tftpboot 20000000 vmlinux-1.22
```

7. Boot the Linux kernel on the upper cores:

```
# bootoctlinux 20000000 coremask=ff0
```

8. Boot the Simple Executive application on the lower cores:

```
# bootoct 30000000 coremask=00f
```

Once booted, the serial console displays both Linux and Simple Executive output and also accepts Linux input. The default IP addresses for the four Cavium Ethernet ports are:

```
eth0 - 192.168.0.100
```

```
eth1 - 192.168.1.100
```

```
eth2 - 192.168.2.100
```

```
eth3 - 192.168.3.100
```

9. Change at least one of the IP addresses to a valid address on your subnet by running the following Linux command:

```
# ifconfig <interface> <i.p. address>
```

For example, # ifconfig eth0 192.168.0.100

10. To test the application, ping one of the Ethernet interfaces from another system on your subnet:

```
# ping <i.p. address>
```

When the ping is successful, Simple Executive output displays similar to the following:



NOTE

The Simple Executive application handles the packet, and then forwards it to Linux.

```
PP0:~CONSOLE-> Packet work group: 0
PP0:~CONSOLE-> Packet Length: 60
PP0:~CONSOLE-> Input Port: 16
PP0:~CONSOLE-> QoS: 0
PP0:~CONSOLE-> Buffers: 1
PP0:~CONSOLE-> Buffer Start:41d6cc080
PP0:~CONSOLE-> Buffer I : 0
PP0:~CONSOLE-> Buffer Back:1
PP0:~CONSOLE-> Buffer Pool: 0
PP0:~CONSOLE-> Buffer Data: 41d6cc140
PP0:~CONSOLE-> Buffer Size: 1856
PP0:~CONSOLE-> 0180c20000000002
PP0:~CONSOLE-> 7ef02e4c00264242
PP0:~CONSOLE-> 0300000000008000
PP0:~CONSOLE-> 000142efafca0000
PP0:~CONSOLE-> 073f800000d001cd
PP0:~CONSOLE-> 206380ed02001400
PP0:~CONSOLE-> 02000f0000000000
PP0:~CONSOLE-> 00000000
```

11. Next, ping one of the Ethernet interfaces while specifying a size bigger than the maximum packet size limit. The maximum packet size is 1024, which can be changed in the example by changing the size of the defined 'MAX_PACKET_SIZE' inside of 'intercept.c' and then rebuilding.

```
# ping -s 1024 <i.p. address>
```

A message similar to the following displays to indicate that packets are too large and are being dropped:

```
PP0:~CONSOLE-> Rcvd 1066 byte pkt that exceeds max size of
1024, dropping being displayed on the simple exec console.
The packet is being dropped without ever being forwarded to
Linux.
```

5 • LSP Applications

This chapter describes the WANic-5651x Diagnostic Utility and the In-Service Daemon. It describes how to run tests to verify and to configure the functionality of components on the WANic-5651x, as well as a process to continuously monitor the hardware.

5.1 Diagnostic Utility

The Diagnostic Utility allows the user to perform data loop tests, which can exercise all or selected Ethernet ports with configurable options. The Diagnostic Utility gives EEPROM access for setting U-Boot parameters, such as LABI, SIPI, PFI, SFI, MAC, and board information EEPROM configurations. All POST and data loopback test results can also be retrieved from the EEPROM through the Diagnostic Utility. The Diagnostic Utility also provides an upgrade utility for U-Boot and Flash application images.

The WANic-5651x Diagnostic Utility is a Linux image that provides the following specific diagnostic tests:

- Transmit/Receive – verify the WANic-5651x Ethernet interfaces.
- Configuration Options – configure the EEPROM to specify boot parameters.
- PCIe utilities – provides services for:
 - Booting U-Boot over the PCIe interface
 - Loading applications over the PCIe to specified target memory locations
 - Sending U-Boot commands to the target over PCIe
- Flash programming menu
- POST and Test Results – verify the most recent POST diagnostics and automatic transmit/receive results.

The Diagnostic Utility also provides a log file and a status file. The log file contains various diagnostic utility information. The status file stores data-loop test results.



NOTE

Features and screens captures displayed in this manual may vary from those displayed by your software. Please consult with a GEIP Customer Technical Support engineer to make sure you install the latest software update.

General diagnostic tests include the following:

- Memory DDR
- PMEM
- USB storage

Command Line Options

In addition, the Diagnostic Utility provides Command Line Parameters to run tests, and to set or display select diagnostic parameters. Table 5-1 lists the Command Line Parameters. When you enter the **-h**(elp) parameter, a screen similar to Figure 5-1 displays all the Command Line Parameters and settings (where applicable.)

Table 5-1 Command Line Parameters

Test Parameters and Settings	Description
--resultshow=[all/post/dataloop/memtests]	Default is to show all
--display=yes	Displays the results and exits
--showtp=yes	Display throughput during data-loop test
--skipmemtest=yes	Skips auto-run memory tests, use with -a option
--skipdatatest=yes	Skips auto-run data-loop test, use with -a option
--skipusbtest=yes	Skips USB storage test, use with -a option
-a	Automatically runs the data loop and memory tests from the command line.
-f	Enables FCC mode.
-h	Displays all the command line parameters.
-l	Sets the name for the diagnostic Log file.
-m	Sets the path to diagnostics Log and Status files.
-o	Puts the interface ports in loop mode.
-r	Sets the name for the diagnostic Status file.
-s	Sets the packet frame size.
-t	Set the data loop test time duration.

Figure 5-1 Command Line Options

```
# npaDiag -h

--resultshow=[all/post/dataloop/memtests]  default is show all
--display=yes                               displays the results and exits
--showtp=yes                                display throughput during data-loop test
--skipmemtest=yes                           skips auto-run memory tests, use with -a option
--skipdatatest=yes                           skips auto-run data-loop test, use with -a option
--skipusbtest=yes                           skips USB storage test, use with -a option

-l <logfile>: set the logfile location
-r <status>: set the status file location
-s <size>: set the frame size
-t <time>: set the test time in seconds
-m <log dir>: set the log file directory location
-f run FCC dataloop test mode
-a auto run dataloop test
-o put interface ports in loopmode
```

5.1.1 Setup

The Diagnostic Utility can run in either PCI Host Mode or PCI Target Mode.

PCI Host Mode

To run the Diagnostic Utility in PCI Host Mode, you must have the following:

- WANic-5651x module installed according to the directions provided by in *“Appendix A • Hardware Installation and Removal Procedures.”*
- Host serial or network terminal.
- Ethernet connection from host terminal to the WANic-5651x, or serial connection to the UART.

Connect to the WANic-5651x, use either Telnet or a serial connection to the host terminal with a serial cable.

- When using Telnet with the factory-installed Linux application, log into a Telnet session to connect to the Linux target running on the WANic-5651x. Enter the following Telnet command:

```
telnet 192.168.x.100
      where x is the Ethernet Interface Number 0–3
```

- When using a serial connection, connect one end of a serial cable to the 5-pin connector on the WANic-5651x SDA and the other end to the host terminal. Use a terminal setting of 115200n1.

PCI Target Mode

In PCI Target Mode, the Processor is a PCI target controlled from the PCI Host System. Once all mode and application binaries are compiled, the user installs and runs them as they would in PCI Host Mode but from the PCI system.

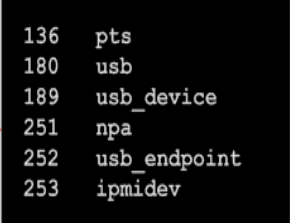
NPA Character Device

Create the NPA character device, which allows the user application to pass commands to the NPA Driver. To create the NPA character device, perform the following steps after the npaDriver.ko module is installed.

1. Get the OCTEON device major number by entering the following command:
cat /proc/devices

A listing similar to that shown in Figure 5-2 displays. The major number is the number associated with the npa.

Figure 5-2 OCTEON NPA Major Number



```
136 pts
180 usb
189 usb_device
251 npa
252 usb_endpoint
253 ipmidev
```

Major number →

2. Create the device by entering the following command and inserting the major number.

```
mknod /dev/npa c<major-number>0
```

5.1.2 Running the Diagnostic Utility

To run the Diagnostic Utility, perform the following:

1. Start the Diagnostic Utility.

Enter the following command to start the Diagnostic Utility:

```
# npaDiag <option>
```

When the Main Menu displays, it prompts you to enter a Menu Choice.

The Main Menu displays in either PCIe Host Mode or PCIe Target Mode, depending on the operating mode. Figure 5-3 display the PCIe Host Mode screen and Figure 5-4 displays the PCIe Target Mode screen. Table 5-2 describes both PCIe Host Mode and PCIe Target Mode options on the Diagnostic Utility Main Menu.



NOTE

The vmlinux image kernel/file system included in the LSP package supports the Diagnostic Utility. When the vmlinux image is executed, the Linux application is configured to conduct an automated loop test following the Linux boot process and exercises all Ethernet ports.

Figure 5-3 Diagnostic Utility Main Menu - PCIe Host Mode

```
1. View Test Results          10. Flash Menu
2. EEPROM Menu              11. Log File Locations
3. PCI Configuration Menu   12. Reserved
4. Peek/Poke Menu          13. MEM Test
5. Run Auto Test            14. Reserved
6. Auto Test Configuration  15. USB Storage Test
8. Diagnostic LED test
0. Exit
Menu Choice (0) : █
```

Figure 5-4 Diagnostic Utility Main Menu - PCIe Target Mode

```
1. View Test Results          10. Flash Menu
2. EEPROM Menu              11. Log File Locations
3. PCI Configuration Menu   12. Show core temperature
4. Peek/Poke Menu          13. Reserved
7. PCI Host Tools Menu
8. Diagnostic LED test
9. Select Current Card
0. Exit
Menu Choice (3) : 0 █
```

Table 5-2 Diagnostic Utility Main Menu

Menu Item	Description
1. View Test Results	Displays the results from all the diagnostic tests performed automatically during the most recent boot of the board.
2. EEPROM Menu	Displays the EEPROM Menu.
3. PCI Configuration Menu	Displays the PCI Configuration registers.
4. Peek/Poke Menu	Allows the peeking and poking of board components.
5. Run Auto Test	Executes the transmit/receive test using the settings in Table 5-4.
6. Auto Test Configuration Menu	Displays the Auto Test Configuration Menu.
7. PCI Host Tools Menu	Allows the PCI boot, PCI load, PCI command, and PCI reset functionality.
8. Diagnostic LED Test	Toggles the LED on and off.
9. Select Current Card	Selects the current card in a multi-card system.
10. Flash Menu	Displays options to view and update Flash partitioning.
11. Log File Location	Display the path to the log files.
12. Reserved	
13. Mem Test	Conducts memory read/write tests for PMEM and/or SDRAM. Only SDRAM test is allowed to run when <code>npa_mem_test</code> named <code>allocated space</code> is set in U-Boot prior to booting Linux. (For example, named <code>alloc npa_mem_test <size> <addr> .</code>)
14. Reserved	
15 USB Storage Test	Conducts USB storage device read/write test if device is available and mounted.
0. Exit	Leaves the Diagnostic Utility.



NOTE

Where appropriate, default values are shown in angle brackets <>.

A description of each Main Menu option follows.

View Test Results

Select **1. View Test Results** to display the most recent POST test results and the most recent Auto Test results. (See **5. Run Auto Test** option.) The Diagnostic Utility displays PASSED or FAILED results from the WANic-5651x most recent boot, similar to that shown in Figure 5-5.

Figure 5-5 View Test Results Display

```
POST PASSED
Diagnostic status
Retrieving data-loop results file...
  xau10,10Gbps Full: PASSED
  xau11,10Gbps Full: PASSED
Retrieving Persistent memory test results file...
  cpu0: All Memory Tests: PASSED
  cpu1: All Memory Tests: PASSED
  cpu2: All Memory Tests: PASSED
  cpu3: All Memory Tests: PASSED
  cpu4: All Memory Tests: PASSED
  cpu5: All Memory Tests: PASSED
  cpu6: All Memory Tests: PASSED
  cpu7: All Memory Tests: PASSED
  cpu8: All Memory Tests: PASSED
  cpu9: All Memory Tests: PASSED
  cpu10: All Memory Tests: PASSED
  cpu11: All Memory Tests: PASSED
Retrieving USB Storage test results...
  USB Storage R/W test: PASSED
Retrieving LED test results...
  LED test: Incomplete
Retrieving Octeon Model Check results...
  Octeon Model Check: PASSED
```


EEPROM Menu

Select **2. EEPROM Menu** to display the EEPROM Menu options as shown in Figure 5-6. Use these options to modify the EEPROM contents on the WANic-5651x.

Figure 5-6 EEPROM Menu

```
Menu Choice (1) : 2
1. Write MAC address
2. Write Board Description
3. Set Source IP information
4. Set Primary File Information
5. Set Secondary File Information
6. Set Load and Boot Information
7. Display SIPI/PFI/SFI/LABI Information
8. Clear POST codes
0. Exit
Menu Choice (0) :
```

Table 5-3 EEPROM Menu Options

Option	Description
1. Write MAC Address	Displays the MAC address for the WANic-5651x and also allows the programming of a new MAC address.
2. Write Board Description	Sets the revision and serial numbers for the module.
3. Set Source IP Information	Sets the source IP address information (SIPI) for each Ethernet interface.
4. Set Primary File Information	Defines the Primary File Information (PFI) definitions.
5. Set Secondary File Information	Defines the Secondary File Information (SFI) definitions.
6. Set Load and Boot Information	Defines the Load and Boot Information (LABI) definitions.
7. Display SIPI/PFI/SFI/LABI Information	Shows all the SIPI, PFI, SFI and LABI information.
8. Clear POST codes	Erases all POST codes.
0. Exit	Leaves the EEPROM Menu, initializes the EEPROM, and returns to the Main Menu.

Run Auto Test

Select **5. Run Auto Test** to run a transmit/receive test using the parameters specified in option **6. Auto Test Configuration Menu**. When the Auto Test completes, the latest test result is updated.



NOTE

If a Telnet connection is established to the WANic-5651x, the connection may be lost when the Auto Test executes.

Figure 5-7 Run Auto Test

```
test_loop(): xau11 link is 10Gbps Full

test_loop(): starting data-loop test...
INFO: Using peer xau10 for xau10
Starting data loop thread for xau10
INFO: [xau10] sched_setaffinity TxRxthread on PID 944 cpu_set 1 ret: 0
INFO: [xau10] sched_getaffinity TxRxthread cur_mask = 1 ret: 0
ERROR: TxRxthreadProc: [xau10] using broadcast for peer
INFO: Using peer xau11 for xau11
Starting data loop thread for xau11
INFO: [xau11] sched_setaffinity TxRxthread on PID 944 cpu_set 2 ret: 0
INFO: [xau11] sched_getaffinity TxRxthread cur_mask = 2 ret: 0
ERROR: TxRxthreadProc: [xau11] using broadcast for peer
Iter:0
xau10: tx:00000259 rx:00000259
xau11: tx:00000253 rx:00000253

Iter:1
xau10: tx:00000510 rx:00000510
xau11: tx:00000504 rx:00000504
```

Auto Test Configuration Menu

Select **6. Auto Test Configuration Menu** to display test configuration options as shown in Figure 5-9. Use these menu options to modify the configuration before selecting **5 Run Auto Test**.



CAUTION

A power cycle of the WANic-5651x causes the configuration to be lost.

To test the WANic-5651x, attach a Loopback Plug to the SFP+.

Figure 5-8 Test Configuration with Cable Attachment

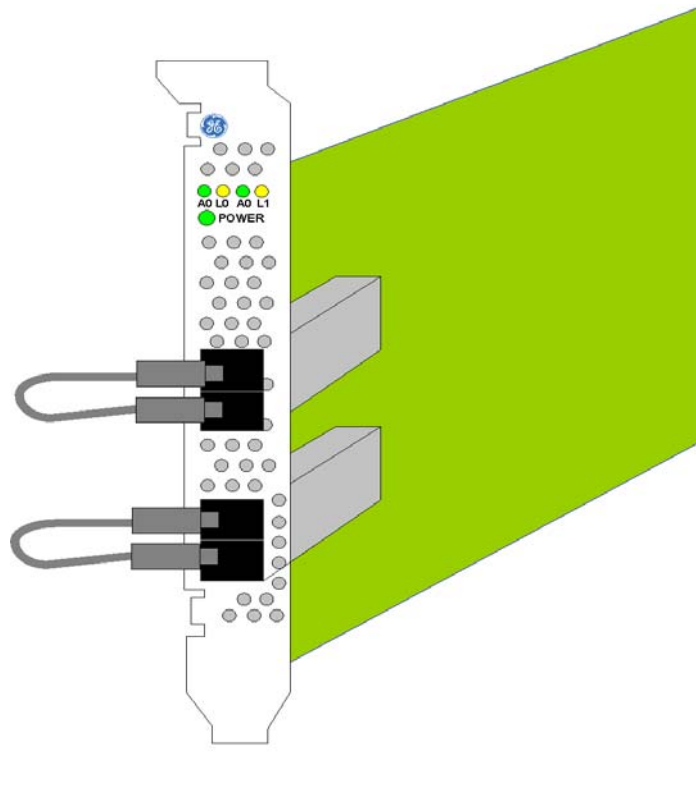


Figure 5-9 Auto Test Configuration Menu

```
Menu Choice (0) : 6
1. Frame Size
2. Enable/Disable Port to be tested
3. Number of seconds to run test
4. Reserved
5. Put Interfaces into Line Loop Mode [DISABLED]
6. Stop loop test on failure [DISABLED]
8. Display data-loop throughput [DISABLED]
9. Put Interfaces into Self Peer Mode [DISABLED]
0. Exit
Menu Choice (0) :
```

Table 5-4 Test Configuration Menu Options

Option	Description	Default
1. Frame Size	Specifies the frame size of the transmitted packets [96 – 1500].	1500
2. Enable/Disable Port to be Tested	Toggles to enable/disable testing on the specified port.	Enable All
3. Number of Seconds to run test	Specifies the number of seconds to run the test. [1 – 86401] (24 hours+1 second). -1 = continuous	30 seconds
4. Reserved		
5. Put interfaces into Line Loop Mode [DISABLED]	Loops incoming data back out the same interface.	Disabled
6. Stop loop test on failure [DISABLED]	When data-loop test exceeds a specified error threshold limit, the data loop test stops.	Disabled
7. Enter EPS value	Sets stop on error threshold value. NOTE: This option is set only when Option 6 is enabled.	
8. Display data-loop throughput [DISABLED]	Shows transmit and receive data throughput.	Disabled
9. Put Interfaces into Self-Peer Mode [DISABLED]	Interface sends data out and expects to receive data on the same interface. The Port is paired with itself.	Disabled
0. Exit	Leaves the Test Configuration Menu and returns to the Main Menu.	None

PCI Host Tools Menu *(PCI Target Mode Only)*

Select **7. PCI Host Tools Menu** to display PCI options as shown in Figure 5-10. PCI host tool options include the following functions:

- Booting U-BOOT over the PCI
- Loading an application over the PCI
- Running command over the PCI
- Performing a soft PCI reset



NOTE

The PCI Host Tools Menu options require hexadecimal values only.

Figure 5-10 through Figure 5-12 show examples of several of these options.

Figure 5-10 Booting U-BOOT over the PCI

```
Menu Choice <0> : 7
1.  Boot U-Boot over PCI
2.  Load App over PCI
3.  Run command over PCI
4.  Soft PCI reset
0.  Exit
Menu choice <0> : 1
Filename: ../../bootloader-3.0.4.c/bin/u-boot-octeon_Wnpaxxxx_ram_debug.bin
INFO: Initializing DDR interface 0
INFO: DRAM configured: 1024 Mbytes
INFO: Copied ../../bootloader-3.0.4.c/bin//u-boot-octeon_Wnpaxxxx_ram_debug.bin
Into memory total bytes copied 334656
█
```

Figure 5-11 Loading an Application over the PCI

```
Menu Choice <1> : 2
Filename: /mnt/tftpboot/npa-rootfs/npa-1.17.7/vmlinux-1.17.7
address:0
INFO: Address was 0, loading to 0x20000000
INFO: DRAM configured: 1024 Mbytes
INFO: Copied /mnt/tftpboot/npa-rootfs/npa-1.17.7/vmlinux-1.17.7 into memory
Menu Choice <0> : 2
1 bytes copied 13680968
1.  Boot U-boot over PCI
2.  Load App over PCI
3.  Run command over PCI
4.  Soft PCI reset
0.  Exit
Menu Choice (2) : █
```

Figure 5-12 Running Commands over the PCI

```
Menu Choice <2> : 3
Command: bootoctlinux 0 coremask=fff █
```

Diagnostic LED Test

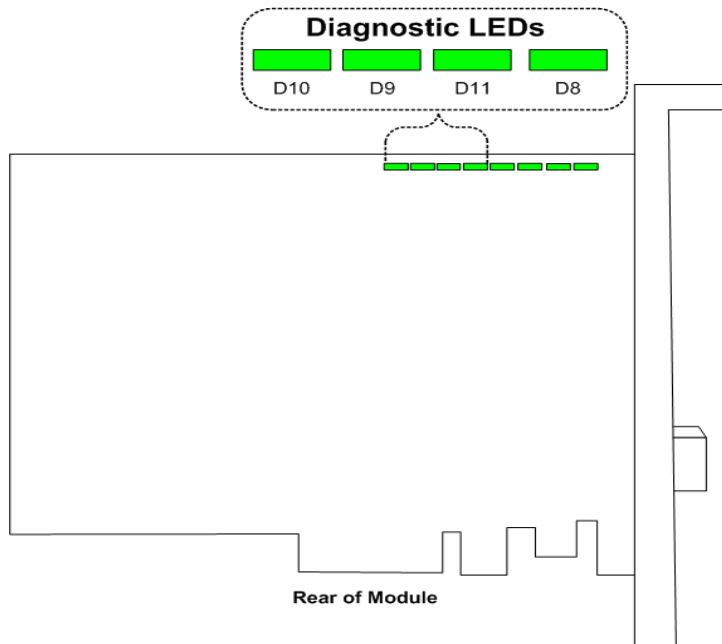
Select **8. Diagnostic LED Test** to verify that the test is toggling the LEDs on and off. This option prompts you with a message as shown in Table 5-5. View the Diagnostic LEDs, located approximately as shown in Figure 5-13, to verify that the LEDs are turning on and off.

To stop the test, press the **Enter** key.

Table 5-5 Diagnostic LED Test Prompts

```
Menu Choice <0> : 8
All leds turn ON? <N>: y
All leds turn OFF? <y>: █
```

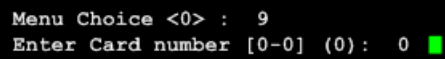
Figure 5-13 Diagnostic LED Location



Select Current Card *(PCI Target Mode Only)*

Select **9. Select Current Card** to choose the card to use in a multi-card system. The Diagnostic Utility prompts you to enter the card number similar to that shown in Figure 5-14.

Figure 5-14 Select Current Card



```
Menu Choice <0> : 9
Enter Card number [0-0] (0): 0 ■
```

Flash Menu

Select **10. Flash Menu** to display and access selected operations to a specified partition in Flash memory. Figure 5-14 shows an example of this menu.



NOTE

The Flash Menu options require decimal values only.

Figure 5-15 Flash Menu

```
Menu Choice <0> : 10
1. Show Flash Info
2. Erase Flash Partition
3. Read Flash Partition
4. Write Flash Partition
5. Update Bootloader Partition
6. Update LINUX Partition
0. Exit
Menu Choice (0) : 5
filename : u-boot-octeon_wnpa3850.distro
INFO: Attempting to update the bootloader...
INFO: Flashing u-boot-octeon_wnpa3850.distro to /dev/mtd0
Jan 1 01:22:01 (none) user.info npaDiag: Attempting to update the bootloader..
INFO: file u-boot-octeon_wnpa3850.distro has been validated
Jan 1 01:22:01 (none) user.info npaDiag: Flashing u-boot-octeon_wnpa3850.dis
Jan 1 01:22:01 (none) user.info npaDiag: file u-boot-octeon_wnpa3850.distro
INFO: Erased /dev/mtd0
Jan 1 01:22:06 (none) user.info npaDiag: Erased /dev/mtd0
INFO: Copied 349920 bytes from u-boot-octeon_wnpa3850.distro to address 0x000h
Jan 1 01:22:08 (none) user.info npaDiag: Copied 349920 bytes from u-boot-oct
INFO: Wrote u-boot-octeon_wnpa3850.distro to /dev/mtd0
INFO: bootloader upgrade was successful
```

Table 5-6 Flash Menu Options

Option	Description
1 Show Flash Info	Displays selected information about the Flash.
2 Erase Flash Partition	Erases a whole Flash partition.
3 Read Flash Partition	Initiates a read to a specified Flash partition.
4 Write Flash Partition	Initiates a write to a specified Flash partition.
5 Update Bootloader Partition	Downloads a new copy of U-Boot bootloader to the specified partition.
6 Update Linux Partition	Updates the Linux partition by downloading a new copy of Linux to the specified partition.
0 Exit	Exits from the Flash Menu and returns to the Main Menu.

Log File Location

Select **11. Log File Location** to display the path to the status and log files as shown in Figure 5-16.

Figure 5-16 Log File Location

```
Menu Choice (11) : 11
Current dependent file location:
  status: /home/root/log/npadiag_8011410_A.log
  logfile: /home/root/log/npastatus_8011410_A
```

Show Core Temperature

Select **12. Show Core Temperature** to display the local and remote temperature settings (in Celsius) as shown in Figure 5-17.

The `local` temperature is the temperature reading of the Local Temperature Sensor. The `remote` temperature is the temperature reading of the Oction internal sensor as read by the Local Temperature Sensor.

Figure 5-17 Show Core Temperature

```
Menu Choice (12) : 12
temperature: local temp 28C, remote 45C
```

Memory Test

Select **13. Mem Test** to run either a DDR2 or PMEM memory test, and to display results as shown in Figure 5-18. The Diagnostic Utility prompts you to select either DDR2 or PMEM:

- When you select DDR2 (0), `npa_mem_test` named `alloc` must be created *before* booting Linux.
- When you select PMEM (1), no preconditions are required.

After prompting you to select a memory test, the Diagnostic Utility also prompts you to Start (1) or Stop (0) the test.

To automatically run the memory test from the command line, use the `-a` Command Line Parameter, which displays similar to that shown in Figure 5-19.



NOTE

Before booting Linux, create `npa_mem_test` named `alloc` reserved memory space from within U-Boot. For example, `namedalloc npo_mem_test <size> <addr>`. Failure to create the memory space will not allow the DDR2 test to run.

Figure 5-18 Memory Test Results

```
Menu Choice <0> : 13
Select memory test <DDR2> Mem = 0, PMEM = 1>: 0
Start or Stop? <start = 1, stop = 0>: 1
Starting memory test thread for cpu0
Starting memory test thread for cpu1
Starting memory test thread for cpu2
Starting memory test thread for cpu3
Starting memory test thread for cpu4
Starting memory test thread for cpu5
Starting memory test thread for cpu6
Starting memory test thread for cpu7
Starting memory test thread for cpu8
Starting memory test thread for cpu9
Starting memory test thread for cpu10
Starting memory test thread for cpu11
```

Figure 5-19 Memory Test with Command Line Parameter -a

```
Running DDR2 Memory tests...
Starting memory test thread for cpu0
Starting memory test thread for cpu1
Starting memory test thread for cpu2
Starting memory test thread for cpu3
Starting memory test thread for cpu4
Starting memory test thread for cpu5
Starting memory test thread for cpu6
Starting memory test thread for cpu7
Starting memory test thread for cpu8
Starting memory test thread for cpu9
Starting memory test thread for cpu10
Starting memory test thread for cpu11
.....
Stopping memory test thread for cpu0
Stopping memory test thread for cpu1
Stopping memory test thread for cpu2
Stopping memory test thread for cpu3
Stopping memory test thread for cpu4
Stopping memory test thread for cpu5
Stopping memory test thread for cpu6
Stopping memory test thread for cpu7
Stopping memory test thread for cpu8
Stopping memory test thread for cpu9
Stopping memory test thread for cpu10
Stopping memory test thread for cpu11
```

5.2 In-Service Daemon

The In-Service Daemon provides a runtime background process that monitors hardware continuously and reports detected faults. The In-Service Daemon is dependent upon the NPA Driver. The In-Service Daemon logs all errors to the Linux syslog and `/var/log/messages`.

5.2.1 Running the In-Service Daemon

To run the In-Service Daemon, enter the following command:

```
# npaDaemon <option>
```

where options include:

-h	Displays all command line parameters
-f	Runs in non-daemon mode; runs in foreground.
-i <n seconds greater than 10>	Number of seconds to wait before the next service check

When you enter the `'cat /var/log/messages'` command, a screen similar to Figure 5-20 displays.

Figure 5-20 Error Message Display

```
~ # cat /var/log/messages
Jan 1 00:00:10 (none) daemon.info init: Starting pid 897, console /dev/ttyS0: '
Jan 1 00:01:34 (none) user.info npaDaemon: Found PHY [0] OUI: 0x141
Jan 1 00:01:34 (none) user.info npaDaemon: Found PHY [1] OUI: 0x141
Jan 1 00:01:34 (none) user.info npaDaemon: Found PHY [2] OUI: 0x141
Jan 1 00:01:34 (none) user.info npaDaemon: Found PHY [3] OUI: 0x141
Jan 1 00:01:34 (none) user.info npaDaemon: Flash Primary device size 128MB fou
Jan 1 00:01:34 (none) user.info npaDaemon: FPGA version is 0.12
Jan 1 00:01:34 (none) user.info npaDaemon: npaDaemon v1.04 running
Jan 1 00:01:34 (none) user.info npaDaemon: Found PHY [0] OUI: 0x141
Jan 1 00:01:34 (none) user.info npaDaemon: Found PHY [1] OUI: 0x141
Jan 1 00:01:34 (none) user.info npaDaemon: Found PHY [2] OUI: 0x141
Jan 1 00:01:34 (none) user.info npaDaemon: Found PHY [3] OUI: 0x141
Jan 1 00:01:34 (none) user.info npaDaemon: Flash Primary device size 128MB fou
Jan 1 00:01:34 (none) user.info npaDaemon: WNP15450 FPGA check
Jan 1 00:01:34 (none) user.info npaDaemon: WNP15450 Local PHY Status
Jan 1 00:01:34 (none) user.info npaDaemon: WNP15450 Flash Device Status
Jan 1 00:01:34 (none) user.info npaDaemon: Test sequence count is [1]
~ # █
```


A • Hardware Installation and Removal Procedures

This chapter describes how to install or remove the WANic-5651x from a PCI Express chassis. It also includes directions for insertion and removal of Mini-RDIMMs and an SFP+.

A.1 Precautions

Before working with any GE Intelligent Platforms component, take the necessary precautions to prevent electrostatic discharge (ESD), which can damage the module. Use the following precautions to prevent ESD when removing the card from the antistatic packaging:



1. Always ground yourself *before* touching the module. You can ground yourself by either touching a grounded unpainted metal surface or by using an ESD-protective wrist strap from your wrist to a bare, unpainted metal section of the chassis. This prevents electrostatic discharge from damaging the module.
2. Always keep the card in its static-protective envelope or packaging if it is not installed in the system.
3. Always hold the card by its faceplate or edges. Avoid touching the components or connections on the module.

A.2 Overview

Installing a WANic-5651x involves the following operations:

- Install Mini-RDIMMs.
- Install SFP+ modules, if appropriate
- Install the WANic-5651x
- Connect to the J4 PCIe Graphics Power Connector

A.3 Mini-RDIMMs Installation and Removal

The DDR SDRAM is implemented as two socketed Mini-RDIMMs, which are factory installed. If you need to install or remove the Mini-RDIMMs, follow the directions in this section.



NOTE

Install the Mini-RDIMMs onto the WANic-5651x *before* installing the board into a chassis.



CAUTION

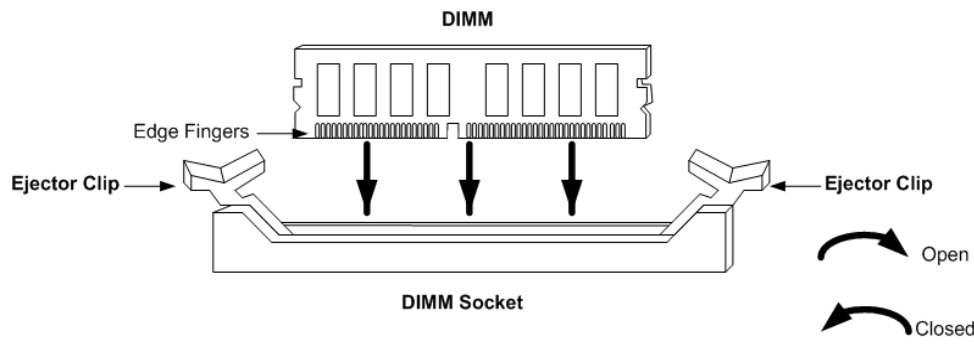
Observe all ESD safety precautions *before* starting this procedure. Failure to follow ESD safety precautions may result in damage to components.

A.3.1 Mini-RDIMM Installation

To install Mini-DIMMs on the WANic-5651x, perform the following steps:

1. Locate the two empty Mini-RDIMM slots.
2. Make sure the socket ejector clips are in the open position as shown in Figure A-1.
3. Remove the Mini-RDIMM memory module from the packaging. Be sure to hold the Mini-RDIMM modules by the edges. Do not touch the components nor the contacts on the module.
4. Align the Mini-RDIMM with the empty socket on the card as shown in Figure A-1.

Figure A-1 Mini-RDIMM Installation



5. Gently, but firmly, press down on the top edge of the Mini-RDIMM to glide the edge fingers into the socket. Make sure the Mini-RDIMM is seated properly in the socket. When seated properly, the ejector clips rise to the vertical position to snap and close to secure the Mini-RDIMM in the socket.



NOTE

Use extreme care when installing an Mini-RDIMM. Applying excessive pressure, can damage the socket or the edge fingers on the Mini-RDIMM.

A.3.2 Mini-RDIMM Removal

To remove the Mini-RDIMM, perform the following steps:

1. Safely remove the WANic-5651x from the chassis and place the card on an ESD-safe surface.
2. Select the Mini-RDIMM to remove.
3. Press down on the Mini-RDIMM socket ejectors clips simultaneously on both end of the Mini-RDIMM slot connector until the Mini-RDIMM ejects from the socket.
4. Lift the Mini-RDIMM from the socket. Pull the Mini-RDIMM up and out of the socket. Place the Mini-RDIMM on an ESD-safe surface or store it in ESD packaging.

A.4 WANic-5651x Installation and Removal

The WANic-5651x insert into a PCIe slot with a minimum of four lanes.



CAUTION

Although it is acceptable to install the WANic-5651x in a slot with more than four lanes, **do not** install the WANic-5651x in a slot with fewer than four lanes.

WANic-56512 models contain the XAUI board-to-board connector. To use this feature, an optionally purchased Flex Circuit is required to connect the boards seated in two adjacent and vertical PCIe slots.

WANic-5651x contains a J4 Connector, which must be attached to an external power supply.



NOTE

Always be sure to follow the safety precautions.

A.4.1 WANic-5651x Installation

To install the WANic-5651x into a chassis, perform the following steps:

1. Turn *off* power to the computer.
2. Disconnect the cables from the back of the computer.
3. Remove the cover from the computer chassis.
4. Ground yourself before handling the card. (*See* previous “CAUTION.”)
5. Locate the available slot in the chassis.



CAUTION

To use the XAUI feature on the WANic-56512, make sure the WANic-56xx cards are placed in adjacent slots within the chassis.

6. If appropriate, remove the screw that secures the filler panels to the card cage frame for these slots. Then, remove the filler panels.
7. Remove the card from its static-protective packaging. Do not touch the components on the module.

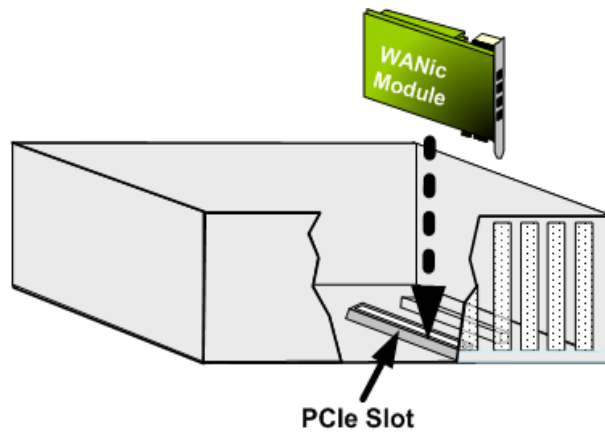


CAUTION

Ground yourself *before* handling the module. While the card is still in the static-protective envelope, hold the card by the edges with one hand while you slide the static-protective envelope off the card with the other hand. Save the package in case you need it for future use.

- Align the card with the appropriate slot and insert the card into the slot. Slide the card into the slot gently, but firmly.

Figure A-2 Installation in a PCI Bus Chassis



- Attach the screw that secures the card to the card cage frame.
- Attach the SFP+ to the card as described in "[Section A.5.1 SFP+ Module Installation](#)."

11. Attach a cable and connector assembly (purchased separately) from an external power supply to the J4 Power Connector on the WANic-56512 as shown in Figure A-3.

The cable assembly is an optional component, which can be purchased from GE Intelligent Platforms (Part No. CBL-PWR) or another vendor. This 10 inch cable splits one (LP6) power connector into two LP4 power connectors.

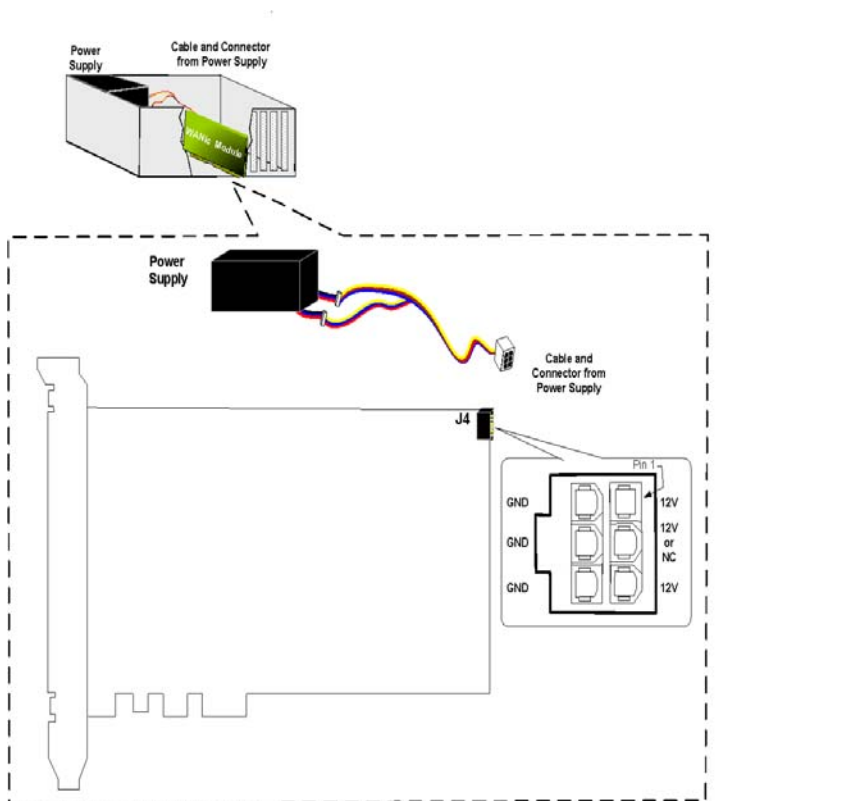
The connector on the cable assembly to the external power supply must be a Molex® 4.20mm (.165") Pitch, 6-pin, Mini-Fit Jr.™ Receptacle Housing, Dual Row connector purchased separately from Molex Incorporated, PN 45559-0002.



NOTE

Access the Molex web site at www.molex.com for a listing of connectors. Contact GE Intelligent Platforms' Customer Technical Support for additional information on cables and connectors.

Figure A-3 Connecting to the Power Supply



12. Replace the computer chassis cover.
13. Replace the cables.
14. Power on the computer.
15. Run the diagnostic utility as described in "[Chapter 5: LSP Applications.](#)"

A.4.2 WANic-5651x Removal

To remove the WANic-5651x, perform the following steps:

1. Ground yourself before handling the module.



CAUTION

Always ground yourself before touching the module. You can ground yourself by touching a grounded unpainted metal surface, such as a computer chassis. This prevents electrostatic discharge from damaging the module.

2. Turn off the power to the computer.
3. Disconnect the cables from the computer chassis.
4. Remove the cover from the computer chassis.
5. Identify the slot from which you intend to remove the module. Remove the screw that secures the card to the frame.
6. On WANic-56512 only, disconnect the XAUI Flex Circuit from each card.
7. On WANic-5651x, disconnect the connector from the power supply that is attached to the card.
8. Grasp the sides of the card. (Remember not to touch the components on the module.) Slide the card out of the slot.



NOTE

Due to the heat generated when the system is running, the card may feel warm when removing it. The card will cool if you wait several minutes before removing it.

9. Store the card in a static-protective envelope. If you are replacing the card with a new one, see "[Section A.4.1 WANic-5651x Installation](#)." If you are not installing a new module, continue with Step 10.
10. Install the filler cards into the empty slots. Attach the screws that secures the filler cards to the card cage frame.
11. Replace the computer chassis cover.
12. Plug in the appropriate cables.

A.5 SFP+ Modules Installation and Removal

SFP+ modules can be inserted and removed in the front panel of a WANic-5651x after the WANic-5651x is inserted in a chassis.

Hot swapping an SFP+ on a WANic-5651x is supported only when using the GE Intelligent Platforms BSP (npaDriver) running in Linux.



CAUTION

U-Boot **does not** support hot swapping of an SFP+. If an SFP+ is hot swapped while in U-Boot, you must run the `sfpininit` U-Boot command from the U-Boot prompt to re-initialize all SFP ports after U-Boot is installed. See '[Section 4.4.3 U-Boot Commands](#)' for a listing of U-Boot commands.

For fiber optical modules, please observe the following warnings, cautions, and notes:



WARNING

For this product, use only the Class 1 laser device that have the following approval:

- FDA21 CFR 1040.10
- IEC 60825-1

Invisible laser radiation may be emitted from disconnected fibres or connectors. Do not Stare into beams or view directly with optical instruments as this may permanently damage your eyes.



CAUTION

1. It is important to disconnect or remove all cables before removing or installing an optical SFP+ transceiver. Failure to do so may result in damage to the cable or SFP+ device.

2. Do not leave an optical SFP+ transceiver uncovered except when inserting or removing a cable. The safety/dust plugs keep the port clean and prevent accidental exposure to laser light.



NOTE

Protect optical SFP+ modules by inserting clean dust plugs into the SFP+ modules after the cables are extracted from them. Be sure to clean the optic surfaces of the fiber cables before plugging the dust plugs back into the optical bores of another SFP+ module. Avoid getting dust and other contaminants into the optical bores of your SFP+ modules: The optics will not work correctly when obstructed with dust.

A.5.1 SFP+ Module Installation

To insert an SFP+ module into a socket connector on the WANic-5651x module, perform the following steps:

1. Ground yourself or attach an ESD-preventive wrist strap from your wrist to a bare metal surface of the chassis.
2. Remove the SFP+ from the anti-static packaging.



NOTE

SFP+ modules have different socket configuration. Make sure you have the proper orientation for the SFP+.

3. Align the SFP+ module to the front of the socket opening.



NOTE

Since SFP+ modules have various latch designs, follow the instructions provided with the SFP+ for inserting the transceiver properly.

4. Glide the SFP+ module into the slot as shown in Figure A-5. Be sure to feel or hear the module snap into place.
5. If the module contains a clasp, close the clasp by moving the clasp up and then pressing it firmly into the locked position.
6. Remove dust plugs and save them for future use.
7. Insert the cable connector into the SFP+ module.

A.5.2 SFP+ Module Removal

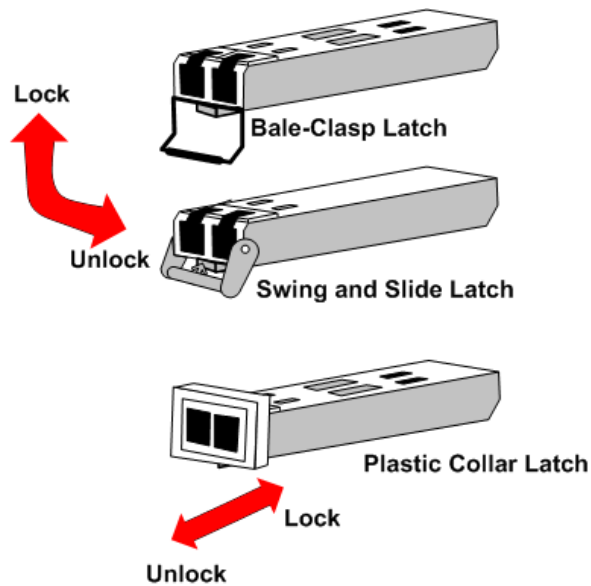
To remove an SFP+ module from a WANic-56512 module, perform the following steps:

1. Ground your self or attach an ESD-preventive wrist strap from your wrist to a bare metal surface of the chassis.
2. Disconnect the cable from the SFP+ module.

For optical SFP+ transceivers, insert the dust plugs into the SFP+ transceiver immediately.

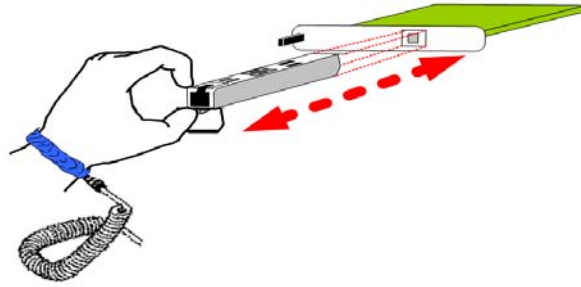
3. Unlock and remove the SFP+ module. Since SFP+ modules have various latch designs, follow the instructions provided with the SFP+ for removing the transceiver. For example, if the module contains a bale-latch, such as those shown in Figure A-4, open the latch on the SFP+ module by pressing it in the appropriate direction with your index finger or a long narrow tool such as a flat-blade screw driver.

Figure A-4 SFP+ Module with Clasp



4. Grasp the SFP+ module between your thumb and index finger. Then, carefully slide the SFP+ out of the socket connector on the module as shown in Figure A-5.

Figure A-5 Inserting or Removing an SFP+ Module.



5. Store the SFP+ transceiver in static-protective packaging.

B • GNU General Public License

The following information is exactly as provided by the Free Software Foundation, Inc.

B.1 Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA.

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

B.2 Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps:

1. copyright the software, and
2. offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

B.3 GNU General Public License

B.3.1 Terms and Conditions For Copying, Distribution and Modification

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:
 - a. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
 - b. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.
 - c. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if

the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:
 - a. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
 - b. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
 - c. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.
5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.
6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.
7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and “any later version”, you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

B.3.2 NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

C • GE INTELLIGENT PLATFORMS EMBEDDED SYSTEMS, INC. and its subsidiaries COMPUTER DYNAMICS OF ILLINOIS, INC. Software License Agreement – Object Code

CAREFULLY READ THE FOLLOWING TERMS AND CONDITIONS BEFORE OPENING THIS PACKAGE OR SIGNIFYING YOUR ACCEPTANCE BY CLICKING THE APPROPRIATE DIALOG BOX. OPENING THIS PACKAGE, CLICKING THE APPROPRIATE DIALOG BOX OR USING ANY PART OF THE SOFTWARE SIGNIFIES YOUR ACCEPTANCE OF THESE TERMS AND CONDITIONS. IF YOU DO NOT AGREE WITH THEM, PROMPTLY RETURN THE PACKAGE UNOPENED AND UNUSED ALONG WITH ANY OTHER ITEM THAT WAS INCLUDED IN THE SAME CATALOG NUMBER FOR FULL CREDIT.

You, as the Customer, agree as follows:

1. DEFINITIONS

“GEIP” shall mean the GE Intelligent Platforms Embedded Systems business providing Licensed Software to Customer pursuant to this Agreement, whether GE Intelligent Platforms Embedded Systems, Inc., its subsidiaries, or Computer Dynamics of Illinois, Inc.

“GEIP Software” shall mean those portions of the Licensed Software owned by GEIP or its affiliate company.

“Licensed Software” shall mean the software, in object code form only, supplied by GEIP pursuant to this Agreement.

“Licensed Product” shall mean the Licensed Software and/or its accompanying documentation.

“Third Party Software” shall mean those portions of the Licensed Software owned or licensed by a third party, including but not limited to operating system code, that is embedded within the Licensed Software.

2. LICENSE

2.1 Except as provided in section 2.2 below, you are granted only a personal, nontransferable, nonexclusive license to use the Licensed Software only as embedded in or to be used on a single GEIP hardware product. You may copy the Licensed Product, for backup purposes only, in support of your use of the Licensed Software, limited to one copy. No other copies shall be made unless authorized in writing by GEIP. You must reproduce and include all applicable copyright notices on any copy. You may not reverse compile or otherwise reverse engineer, or modify the Licensed Software. The Licensed Software, comprising proprietary trade secret information of GEIP and/or its licensors, shall be held in confidence by Customer and protected from disclosure to third parties. No title to the intellectual property is transferred. Licensed Software shall not be copied, reproduced, or used for any other purpose outside of operation of the GEIP hardware, and shall not be used on any other piece of hardware other than the GEIP hardware with which it was provided.

2.2 If you transfer the GEIP hardware product on which the Licensed Software is used, you may transfer the Licensed Software to the end user of the hardware product provided that the end user agrees to be bound by terms no less restrictive than the provisions of this Agreement, and provided that all proprietary

markings are maintained. Any other transfer is void and automatically terminates this license. You shall use your best efforts to enforce such agreement and shall promptly report any violation or suspected violation to GEIP. In the event you do not enforce such agreement after a breach, you shall, to the extent permissible by applicable law, grant GEIP the right to enforce such agreement.

2.3 The Licensed Software may include Third Party Software licensed to GEIP. The owner of the Third Party Software (the "Third Party") and its licensors are intended third party beneficiaries of this Agreement, and the provisions of this Agreement relating to the Licensed Software, as the same incorporates Third Party Software, are made expressly for the benefit of, and are enforceable by, the Third Party and its licensors. The Third Party and its licensors retain ownership of all copies of the Third Party Software. Unless a pass-through warranty covering the Third Party Software is extended directly to you by the Third Party, all Third Party Software is provided "AS IS" without warranty of any kind, and each of Third Party and its licensors disclaim all warranties, either express or implied, including but not limited to the implied warranties of merchantability, title, non-infringement or fitness for a particular purpose with regard to the Third Party Software. The Third Party shall not have any liability for special, indirect, punitive, incidental or consequential damages. Unless otherwise expressly stated by GEIP, you must make your own provision for any required operating system software licenses even if the Licensed Software contains some operating system code.

2.4 In addition to the Licensed Software, GEIP may provide certain files embedded in or to be used on the GEIP hardware product which may be subject to the terms of the GNU General Public License (GPL) or the GNU Lesser General Public License (LGPL), a current link to which may be found at: <http://www.gnu.org>. The Licensed Product is not subject to the GPL or LGPL, and Customer has no license to take any action, and shall take no action, which would have the effect of subjecting the Licensed Software or any portion of the Licensed Software to the terms of the GPL or LGPL. Customer may consult the user documentation for identifications and further information.

2.5 If the Licensed Software or associated documentation is provided to any U.S. Government entity, unit, or agency, the restrictions set forth at section 52.227-19(c) ("Commercial computer software - restricted rights") of the Federal Acquisition Regulations (FARs) shall apply. If the Licensed Software or associated documentation is provided to the U.S. Government, Department of Defense (DOD), or any entity, unit, or agency thereof, the restrictions set forth at section 252.227-7015 ("Technical Data - Commercial Items") of the DOD FAR Supplement (DFARS) shall also apply.

2.6 For the rights granted in this Agreement, Customer shall pay to GEIP the then-current catalog price (license fee) for each copy of the Licensed Software provided by GEIP to Customer, or the price for the GEIP hardware product in which the Licensed Software is embedded, whichever is applicable.

2.7 Customer shall pay all import duties and registration fees and all sales, use and excise taxes (and any other assessments in the nature of taxes however designated) on the Licensed Product or its license to use the Licensed Product, or resulting from this Agreement, exclusive of taxes based on GEIP' net income.

3. WARRANTY

3.1 GEIP warrants that the GEIP Software will be in substantial conformance with GEIP's standard published user documentation pertaining thereto as of the date of shipment by GEIP. If, within ninety (90) days of date of shipment, it is shown that the GEIP Software does not meet this warranty, and such Licensed Software is returned to GEIP with a copy of your purchase confirmation, GEIP will, at its option, either correct the defect or error in the GEIP Software, free of charge, or make available to Customer satisfactory substitute software, or return to Customer all payments made as license fees (or fees paid for the GEIP hardware product in which the Licensed Software is embedded which are allocable to the Licensed Software, whichever is applicable) and terminate the license with respect to the GEIP Software affected. GEIP does not warrant that operation of the GEIP Software will be uninterrupted or error free or that it will meet Customer's needs. All other portions of the Licensed Software are provided "as is" without warranty of any kind.

3.2 THE FOREGOING WARRANTIES ARE EXCLUSIVE AND ARE IN LIEU OF ALL OTHER WARRANTIES WITH RESPECT TO THE LICENSED PRODUCT WHETHER WRITTEN, ORAL, IMPLIED OR STATUTORY. NO IMPLIED OR STATUTORY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE SHALL APPLY. NO WARRANTY ARISING FROM COURSE OF PERFORMANCE, COURSE OF DEALING, OR USAGE OF TRADE SHALL APPLY.

4. LIMITATION OF LIABILITY

4.1 GEIP'S LIABILITY FOR ALL CLAIMS OF ANY KIND, WHETHER BASED ON CONTRACT, INDEMNITY, WARRANTY, TORT (INCLUDING NEGLIGENCE), STRICT LIABILITY, FAILURE OF A REMEDY TO ACCOMPLISH ITS ESSENTIAL PURPOSE, OR OTHERWISE, FOR ALL LOSSES OR DAMAGES ARISING OUT OF, CONNECTED WITH, OR RESULTING FROM THIS AGREEMENT, OR THESE TERMS AND CONDITIONS, OR FROM THE PERFORMANCE OR BREACH THEREOF, OR FROM THE LICENSED PRODUCT OR ANY PART THEREOF, OR FROM ANY SERVICE FURNISHED HEREUNDER (INCLUDING REMEDIAL WARRANTY EFFORTS), SHALL, IN THE AGGREGATE, IN NO CASE EXCEED THE LICENSE FEES FOR THE LICENSED PRODUCT OR THE PRICE OF THE GEIP HARDWARE PRODUCT IN WHICH IT IS EMBEDDED, WHICHEVER IS APPLICABLE, GIVING RISE TO THE CLAIM. ALL SUCH LIABILITY SHALL TERMINATE UPON THE EXPIRATION OF THE WARRANTY PERIOD AS SET FORTH IN SECTION 3.

4.2 IN NO EVENT, WHETHER BASED ON CONTRACT, INDEMNITY, WARRANTY, TORT (INCLUDING NEGLIGENCE), STRICT LIABILITY, FAILURE OF A REMEDY TO ACCOMPLISH ITS ESSENTIAL PURPOSE, OR OTHERWISE, SHALL GEIP, ITS EMPLOYEES OR SUPPLIERS BE LIABLE FOR INDIRECT, SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES, INCLUDING, BUT NOT LIMITED TO LOSS OF PROFITS OR REVENUE, LOSS OF USE OF ANY PROPERTY, COST OF CAPITAL, COST OF PURCHASED POWER, COST OF SUBSTITUTE EQUIPMENT, FACILITIES, OR SERVICES, DOWNTIME COSTS, OR CLAIMS OF CUSTOMERS AND TRANSFEREES OF THE CUSTOMER FOR SUCH DAMAGES EVEN IF GEIP HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, AND THE CUSTOMER WILL INDEMNIFY GEIP, ITS EMPLOYEES AND SUPPLIERS AGAINST ANY SUCH CLAIMS FROM THE CUSTOMER'S CUSTOMERS. IF THE LICENSED PRODUCT WILL BE

FURNISHED BY THE CUSTOMER TO A THIRD PARTY BY CONTRACT OR RELATE TO A CONTRACT BETWEEN THE CUSTOMER AND A THIRD PARTY, THE CUSTOMER SHALL OBTAIN FROM SUCH THIRD PARTY A PROVISION AFFORDING GEIP AND ITS SUPPLIERS THE PROTECTION OF THIS SUBSECTION AND THE PRECEDING SUBSECTION.

4.3 The Licensed Product is not intended for use in any nuclear facility or application, or any life-support equipment or other application where failure of the products could lead directly to death, personal injury or severe physical or environmental damage. If so used, GEIP disclaims all liability for any damages arising as a result of the hazardous nature of the application in question, including but not limited to nuclear or environmental damage, injury or contamination, and Customer shall indemnify, hold harmless and defend GEIP, its officers, directors, employees and agents against all such liability, whether based on contract, warranty, tort (including negligence), strict liability, or any other legal theory, regardless of whether GEIP had knowledge of the possibility of such damages.

4.4 If GEIP furnishes Customer with advice or other assistance concerning any products or systems which is not required pursuant to this agreement, the furnishing of such advice or assistance will not subject GEIP to any liability, whether in contract, indemnity, warranty, tort, (including negligence), strict liability or otherwise.

5. INDEMNITY

5.1 GEIP warrants that the GEIP Software shall be delivered free of any rightful claim of any third party for infringement of any United States patent or copyright. If promptly notified in writing and given full authority, information and assistance, GEIP shall defend, or may settle, at its expense, any suit or proceeding against Customer so far as based on a claimed infringement which would result in a breach of this warranty, and GEIP shall pay all damages and costs finally awarded therein against Customer due to such breach, other than damages and costs arising from any willful infringement by Customer after receipt of notice of the claimed infringement. GEIP shall not be responsible for any compromise or concession made by Customer without the prior written consent of GEIP. In case the GEIP Software is in such suit held to constitute such an infringement and its use for the purpose intended for such software is enjoined, GEIP shall, at its expense and option, either procure for Customer the right to continue using said software, or replace same with noninfringing software, or modify same so it becomes noninfringing, or remove the software and refund the license fees pertaining thereto or the fees paid for the GEIP hardware product in which the GEIP Software is embedded which are allocable to the GEIP Software (less reasonable depreciation for any period of use) and any transportation costs separately paid by Customer. The foregoing states the entire liability of GEIP for patent or copyright infringement by the Licensed Product or any part thereof.

5.2 The indemnity under the preceding paragraph shall not apply if the infringement or claim is based in whole or in part upon any use of GEIP Software in conjunction with any other product in a combination not furnished by GEIP as a part of this transaction. As to any such use in such combination, or any improper or unauthorized use, modification, installation, or operation of the GEIP Software, GEIP assumes no liability whatsoever for patent or copyright infringement and Customer will hold GEIP harmless against any infringement claims arising therefrom.

6. TERM AND TERMINATION

6.1 You may terminate the license granted hereunder at any time by destroying the Licensed Product together with all copies thereof and notifying GEIP in writing that all use of the Licensed Product has ceased and that same has been destroyed.

6.2 GEIP may terminate this Agreement or any license hereunder upon notice to Customer if Customer breaches any of the terms and conditions of this Agreement or if Customer attempts to assign this Agreement or any license hereunder without the prior written consent of GEIP. Within twenty (20) days after any termination of this Agreement, Customer shall certify in writing to GEIP that all use of the Licensed Product has ceased, and that the same has been destroyed.

6.3 All provisions of this Agreement related to disclaimers of warranty, limitation of liability, GEIP's intellectual property rights, or export shall survive any expiration or termination and remain in effect. Termination of this Agreement or any license hereunder shall not relieve Customer of its obligation to pay any and all outstanding charges hereunder nor entitle Customer to any refund of such charges previously paid.

7. EXPORT

7.1 If you intend to export (or reexport), directly or indirectly, the Licensed Product or technical data relating thereto or any portion thereof, it is your responsibility to assure compliance with U.S. and other applicable export control laws and to obtain any required licenses or approvals in your own name. You are also responsible for the accuracy and completeness of any information or certification you provide for purposes of export control compliance.

8. GENERAL

8.1 This Agreement shall be governed by the laws of the State of New York, without regard to its conflict of law provisions. The provisions of the United Nations Convention on the International Sale of Goods shall not apply to this Agreement.

8.2 YOU ACKNOWLEDGE THAT YOU HAVE READ THIS AGREEMENT, UNDERSTAND IT AND AGREE TO BE BOUND BY ITS TERMS AND CONDITIONS. YOU FURTHER AGREE THAT IT IS THE COMPLETE AND EXCLUSIVE STATEMENT OF THE AGREEMENT BETWEEN US AND SUPERSEDES ANY PROPOSAL OR PRIOR AGREEMENT, ORAL OR WRITTEN, AND ANY OTHER COMMUNICATIONS BETWEEN US RELATING TO THE SUBJECT MATTER OF THIS AGREEMENT. FURTHER, NO CHANGE OR AMENDMENT TO THIS AGREEMENT SHALL BE EFFECTIVE UNLESS AGREED TO BY WRITTEN INSTRUMENT SIGNED BY A DULY AUTHORIZED REPRESENTATIVE OF GEIP.

1/13/10

GFJ-353

D • GE Intelligent Platforms Embedded Systems, Inc.

Software License Description

for the WANic-5651x Packet Processors

Software License Descriptions

This document describes the licensing of the individual software components included on this product. The following components are included as firmware provided in Flash memory.

U-Boot

The U-Boot bootloader firmware is covered by the GNU General Public License (GPL), version 2. Please see the GNU GPL v2, Part Number: 21-LIC-GPLV2-0 for more information regarding the terms of this license.

Linux Image

The Linux image (kernel and embedded file system) is covered by the GNU GPL v2. Please see the GNU GPL v2, Part Number: 21-LIC-GPLV2-01 for more information regarding the terms of this license.

This glossary contains a listing of terms, acronyms, and definitions.

10GBASE, 1000BASE-BX, 1000BASE-T, 1000BASE-X, or 1000BASE-SX	<p>A standard for Gigabit Ethernet connectivity, which provides transmission speeds up to 1000 megabits per second (or one million bits per second.)</p> <ul style="list-style-type: none"> • 10GBASE applies to 10 Gigabit Ethernet. • 1000BASE-BX applies to Gigabit Ethernet over backplanes and printed wiring boards. • 1000BASE-X is used for Gigabit Ethernet over optical cables. • 1000BASE-T is used for Gigabit Ethernet over twisted-pair copper cable. • 1000BASE-SX is used for short wavelength laser over multimode fiber, and has a maximum distance of 550 meters.
Control Plane	The software that manages the establishment and maintenance of connections in the network, including protocols and mechanisms to disseminate this information, and algorithms for engineering an optimal path between end points.
DDR2	Double Data Rate Type 2. A type of synchronous DRAM Technology that permits the transfer of two words of data per clock period.
DIMM	Dual in-line Memory Module.
DRAM	Dynamic Random Access Memory.
EEPROM	Electrically Erasable Programmable Read Only Memory. A type of non-volatile memory that permits its content to be selectively modified on a location-by-location basis. Flash memory is a special category of EEPROM, which can be reprogrammed on a sector-by-sector basis or in its entirety only after the prior content of those sectors is erased.
ECC	Error Correction Code.
ESD	ElectroStatic Discharge.
Ethernet	Ethernet is a local area network technology specified in the IEEE 802.3 standard.
Flash Memory	Sometimes called Flash RAM, Flash memory is a type of constantly-powered nonvolatile memory that can be erased and reprogrammed in units of memory called blocks.
GB	GigaByte.
Gigabits per second (Gbps)	Gigabits per second is one billion bits per second.
Gigabit Ethernet (GbE)	Gigabit Ethernet provides higher level of backbone support at 1000 megabits per second (1 gigabits or 1 billion bits per second).
GND	Ground provides connections to circuit ground at both ends. Ground ensures that the transmit signal levels stay within the common mode input range of receivers.
GMII	Gigabit Media Independent Interface provides a simple interconnection between MAC and PHY device that is independent of the physical media types.
GPIO	General Purpose Input and Output.
IC	Integrated Circuit.
I2C	Inter-Integrated Circuit (I2C). Two-wire interface commonly used to connect low-speed peripherals in an embedded system.
IEEE	Institute of Electrical and Electronic Engineers, Inc standards organization.

IEEE 802.3	A local area network protocol suite commonly known as Ethernet.
I/O	Input/Output.
IP	Internet Protocol.
KB	KiloByte.
LED	Light Emitting Diode.
LOS	Loss of Signal.
MAC	Media Access Control.
MB	MegaByte.
Mbps	Megabits per second.
MDI	Medium Dependent Interface.
MDIO	Management Data Input/Output.
MPX	Multi-Processor Extension.
MTBF	Mean Time Between Failure.
Multi-Core Processor	A Multi-Core Processor is an integrated circuit to which two or more processors have been attached for enhanced performance, reduced power consumption, and more efficient simultaneous processing of multiple tasks.
NMI	Non-Maskable Interrupt.
NVRAM	Non-Volatile Random Access Memory.
OEM	Original Equipment Manufacturers.
PCI	Peripheral Component Interconnect.
PCIe	PCI Express.
PHY	A generic electronics term referring to a special electronic integrated circuit or functional block of a circuit that provides PHYSical access to a digital connection cable. Also know as a PHYSical layer device.
PICMG	PCI Industrial Computer Manufacturers Group.
Ping	A basic Internet program that allows a user to verify that a particular IP address exists and can accept requests.
QLM	Quad-Lane Modules.
RoHS	The European Restriction of Hazardous Substance in electrical and electronic equipment. sold or used in the European Union after July 1, 2006. These substances are lead, mercury, cadmium, hexavalent chromium, polybrominated biphenyls, and polybrominated diphenyl ethers.
ROM	Read Only Memory.
SDRAM	Static Dynamic Random Access Memory.
SFP	Small Form Pluggable.
SMI	System Management Interrupt.
TCK	Test Clock.
TDI	Test Data In.

TDO	Test Data Out.
TEM	Telecommunications Equipment Manufacturers.
TMS	Test Mode Select.
Tuple	An ordered set of values.
XAU	Ten (Roman Numeral X) Attachment Unit Interface.

Symbols

/proc file111, 117

Numerics

20-Pin Header
 10-pin breakout 40
 14-pin breakout 40

A

Active script tuple95
 Auto test..... 131, 133

B

Board ID and DIP Switch Register55
 Boot
 bus 25
 firmware 71
 Flash77
 BusyBox73

C

Cable
 IDE41
 serial debug adapter40
 Cavium Ethernet Driver72
 Checksum tuple89
 Command Line Parameters124
 Connector
 PCI Express 32
 SFP+46
 CSUMTESTI tuple87

D

DB-9 connector41
 DDR2 SDRAM36
 Diagnostic LED, test134
 Diagnostic Utility123
 Auto Test Configuration Menu131
 Command Line Parameters124
 Diagnostic LED Test134
 EEPROM menu129
 Flash Menu..... 136
 Log File Location..... 137
 Main Menu127
 memory test139
 PCI Host mode125
 PCI Host Tools Menu133
 PCI Target mode 125
 run auto test130

running126
 Select Current Card 135
 Show Core Temperature 138
 view test results 128

DIMMs

installation 144
 Serial Presence Detect 36

DIP Switch 44
 illustration..... 44
 register 44

Direct Attach Mode 46, 118

Dust plugs 21, 150

E

EEPROM 37, 86
 enumeration 99
 mapping 86
 programming 129
 tuple API101
 tuples 87

EJTAG Header 43

Electrostatic discharge (*See ESD*)

Embedded File System 73

configuration 73

Emission compliance 20

Environmental

requirements 19

variables 81

ESD 143

Examples

intercepts121

Linux applicaton.....119

LSP119

Simple Exec Applicatio120

External JTAG (EJTAG) 43

F

Firmware 71

Flash 37

accessing104

base address 102

device identification 102

displaying partitions 104

functions103

mapping102

partitioning104

programming136

running the file system 105

FPGA 33

base address 53

register mapping..... 34

registers	53
Free Software Foundation	155
Front panel	
LEDs.....	47
SFP connectors.....	46

G

Generation of interrupts	57
Gigabit Ethernet PHY tuple	92
GNU General Public License	155
GPIO interface	29

H

Hardware	
components	23
initialization.....	37
Header	
EJTAG	43
J5.....	40
J6.....	42
Heat sink	49

I

I2C Address High Register	65
I2C Address Low Register	65
I2C Control Register	64
I2C Data Register	66
I2C registers	62
IDE cable	41
Inserting an SFP module	151
In-Service Daemon	141
Installation	
Mini-RDIMMs	144
precautions.....	143
WANic card	146
Interrupt Control Register	57, 58
IOCTL	
API.....	106
command types.....	109
commands	107
examples.....	106
Ioctl	
environment information.....	113
Ethtool	116
extended device control	116
register access	114
standard Linux	116
storing POST information.....	115
tuple operation	113

J

J6 Header	42
JTAG scan chain	42

L

LABI tuple.....	87, 90
LED Control Register	56
LEDs	47
description	47
Level 2 cache	36
Linux Support Package (<i>See LSP</i>)	
LSP	101
additional features 1.....	18
description	101
Diagnostic Utility.....	123
examples	119
functions.....	101

M

Memory	36
DDR2 SDRAM	36
EEPROM	37
Flash.....	37
MEMTEST tuple	87
Module	
installation	146
removal	149
mtd_debug.....	105
Multi-Core Processor	24
configuration	24
fan controller	48
GPIO interface.....	28
Level 2 cache	36
power supply	24
TWSI interface	27
UARTs.....	29

N

Notation convention	13
NPA Driver	117

O

Operating system support.....	71
Optical/fiber transceiver	46

P

Payload Reset Register	59
PCI Express	
connector	32

interface	26
PCI Host Mode	125
PCI Target Mode,	125
PCIe Edge Connector	32
Persistent memory	37
PFI tuple	87, 93
PHYTESTI tuple	87
POST error codes	76
Power supplies	50
Primary File Information (PFI)	84
Primitives, TWSI	101

R

Reading data	63
Registers	
I2C Address High	65
I2C Address Low	65
I2C Control.....	64
I2C Data.....	66
Interrupt Control.....	58
Interrupt Status	58
Payload Reset	59
Transmit Control.....	60
Related specifications and documentation.....	11
Removing	
dust plugs	151
SFP module	152
WANic card	149
Reprogramming	
boot loader	77
Linux Kernel in Flash	105
U-Boot.....	105

S

S1 DIP Switch	42, 44
SCRIPT tuple	87
SCRIPTACTIVE tuple	87
Secondary File Information (SFI)	84
Serial Debug Adapter cable	40
Serial Presence Detect	36
SFI tuple	87, 97
SFP module	150
holding.....	153
inserting	151
removing	152
storing	153
unlocking.....	152
Simple Exec Library	74
SIPI tuple	87
Software components	72
Source IP Information (SIPI).....	84
Static-protective packaging	143

T

Telnet	125
Temperature Monitor.....	48
Transmit Control Register	60
Tuple	
CSUMTESTI	89
EEPROM	87
format	87
TWSI	
interface	27
primitives.....	101
TWSI_OP_t structure	114

U

UA	
composition	74
loading.....	74
UART	29
UARTCONI tuple	87
U-Boot	75
building.....	77
commands.....	78
POST error codes	76
reprogramming	105
versions	
normal.....	75
Unlocking the SFP module	152
User Application (<i>See UA</i>)	
Voltage	51

W

Writing data	63
--------------------	----

© 2009-2010 GE Intelligent Platforms Embedded Systems, Inc. All rights reserved.

An asterisk (*) indicates a trademark of GE Intelligent Platforms, Inc. and/or its affiliates. All other trademarks are the property of their respective owners.

Confidential Information - This document contains Confidential/Proprietary Information of GE Intelligent Platforms, Inc. and/or its suppliers or vendors. Distribution or reproduction prohibited without permission.

THIS DOCUMENT AND ITS CONTENTS ARE PROVIDED "AS IS", WITH NO REPRESENTATIONS OR WARRANTIES OF ANY KIND, WHETHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO WARRANTIES OF DESIGN, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. ALL OTHER LIABILITY ARISING FROM RELIANCE UPON ANY INFORMATION CONTAINED HEREIN IS EXPRESSLY DISCLAIMED.

GE Intelligent Platforms Information Centers

Americas:

1 800 322 3616 or 1 256 880 0444

Asia Pacific:

86 10 6561 1561

Europe, Middle East and Africa:

Germany +49 821 5034-0

UK +44 1327 359444

Additional Resources

For more information, please visit the GE Intelligent Platforms Embedded Systems web site at:

www.ge-ip.com

