

GE
Intelligent Platforms

Reference Manual

WANic-58xx Packet Processor
Sixth Edition

Part No: 87002020-820



Table of Contents

Preface	11
1 • Overview	15
1.1 Features.	15
1.2 Product Overview.	15
1.3 Hardware Overview.	16
1.4 Software Overview.	17
1.4.1 Software Components	17
1.4.2 Additional Software	17
1.4.3 Operating System Support	17
1.4.4 Cavium Networks Users' Organization Web Site	18
1.5 Specification.	19
1.6 Emission Compliance Notice.	20
1.7 RoHS and WEEE Compliance.	20
1.8 Warranty and Repair.	21
1.8.1 Warranty	21
1.8.2 Customer Technical Support	21
1.9 Handling Precautions.	22
1.9.1 ESD Precautions	22
1.9.2 Optical/Laser Safety Notices	22
2 • Hardware Description	25
2.1 Hardware Components.	25
2.2 Multi-Core Processor.	26
2.2.1 Multi-Core Processor Interfaces	27
2.2.2 Multi-Core Processor Boot Bus	29
2.3 PCI/PCI-X Interface.	29
2.4 FPGA.	30
2.5 Memory.	31
2.5.1 DDR2 SDRAM	31
2.5.2 Optional RLDRAM II	31
2.5.3 Flash	32
2.5.4 Serial EEPROM	32
2.6 Gigabit Ethernet PHY Devices.	33
2.7 SFP Front Panel Connectors.	33
2.8 Dual In-Line Package (DIP) Switches.	34
2.8.1 S1 DIP Switch	34
2.8.2 S2 DIP Switch	35
2.9 Reference Clock Circuitry.	37
2.10 Front Panel.	38
2.10.1 LEDs	38

2.10.2 SFP Modules	39
2.11 Diagnostic LEDs	40
2.12 Headers	40
2.12.1 JTAG Scan Chain	40
2.12.2 Enhanced JTAG	41
2.12.3 RS-232 Serial Port	42
2.13 Temperature Monitor	44
2.14 Power and Cooling	45
2.14.1 Power Supplies	45
2.14.2 Thermal Management	45
3 • FPGA Registers	47
3.1 FPGA General Registers	48
3.2 I2C Registers	57
3.2.1 Writing Data	57
3.2.2 Reading Data	58
4 • Software Features	63
4.1 Overview	63
4.2 Cavium Software Development Kit	64
4.2.1 Cavium Ethernet Driver	64
4.2.2 Embedded File System	65
4.2.3 Simple Exec Library	66
4.3 User Application	66
4.4 U-Boot Bootloader	67
4.4.1 UART Command Interface	67
4.4.2 Building U-Boot	68
4.4.3 Boot Progress Status Messages	70
4.4.4 IP Configuration Acquisition with DHCP	71
4.4.5 U-Boot Scripting	72
4.4.6 Environment Variables	73
4.4.7 Automated UA Executable Load and Boot	74
4.4.8 U-Boot Error Handling	75
4.4.9 U-Boot Commands	76
4.5 EEPROM	78
4.5.1 EEPROM Mapping	78
4.5.2 EEPROM Tuples	79
4.5.3 EEPROM Tuple Update and Enumeration	93
4.5.4 Tuple Format	95
4.6 Linux Support Package (LSP)	96
4.6.1 Flash Driver	97
4.7 NPA Driver	101
4.7.1 Linux /proc/ File System	101
4.7.2 IOCTL Device interface	102
4.7.3 IOCTL Commands	103

4.7.4 Proc File Updates	107
4.7.5 Exported Kernel Symbols	107
4.8 Examples	108
4.8.1 Linux Applications Examples	108
4.8.2 Simple Exec Application Examples	109
5 • WANic-58xx Diagnostic Utility	113
5.1 Overview	113
5.1.1 Setup	115
5.1.2 Running the Diagnostic Utility	116
A • Hardware Installation and Removal Procedures	129
A.1 Installation Precautions	129
A.2 Installation Overview	129
A.3 Mini-RDIMM Installation and Removal	130
A.3.1 Mini-RDIMM Installation	130
A.3.2 Mini-RDIMM Removal	130
A.4 WANic-58xx Installation and Removal	131
A.4.1 WANic-58xx Installation	131
A.4.2 WANic-58xx Removal	133
A.5 SFP Module installation and Removal	134
A.5.1 SFP Module Installation	135
A.5.2 SFP Module Removal	136
B • GNU General Public License	137
C • GE INTELLIGENT PLATFORMS, INC. Software License Agreement – Object Code	143
D • GE INTELLIGENT PLATFORMS, INC. Software License Description	149
Glossary	151
Index	155

Figure 2-1 Block Diagram	26
Figure 2-2 GPIO Interface to SFPs	28
Figure 2-3 Flash Connection to Boot Bus	32
Figure 2-4 DIP Switches	35
Figure 2-5 Reference Clock Circuitry	37
Figure 2-6 Front Panel LEDs	38
Figure 2-7 RJ-45 Connector	39
Figure 2-8 Diagnostic LEDs	40
Figure 2-9 JTAG Scan Chain and EJTAG Interface	41
Figure 2-10 Serial Cable Pinout	43
Figure 2-11 Serial Cable Connection	43
Figure 2-12 Air Flow on the WANic-58xx	46
Figure 3-1 FPGA Revision Register @ Base + 00h	49
Figure 3-2 Board ID and DIP Switch Register @ Base + 01h	50
Figure 3-3 LED Control Register @ Base + 02h	51
Figure 3-4 Diagnostic LEDs	51
Figure 3-5 Interrupt Control Register @ Base + 03h	52
Figure 3-6 Interrupt Status Register @ Base + 04h	53
Figure 3-7 Peripheral Reset Register @ Base + 05h	54
Figure 3-8 Transmit Control Register @ Base + 06h	55
Figure 3-9 Fan and Temperature Register @ Base + 07h	56
Figure 3-10 I2C Control Register @ Base + Address	59
Figure 3-11 I2C Address Low Register @ Base + Low Address	60
Figure 3-12 I2C Address High Register @ Base + High Address	60
Figure 3-13 I2C Data Register @ Base + Address	61
Figure 4-1 Software Block Diagram	64
Figure 4-2 S2 DIP Switch Bank	69
Figure 4-3 EEPROM Mapping	78
Figure 4-4 Flash Mapping	97
Figure 4-5 IOCTL Commands and Structures	102
Figure 4-6 IOCTL Command Types	105
Figure 4-7 Exported Kernel Symbols	107
Figure 5-1 Command Line Options	114
Figure 5-2 Diagnostic Utility Main Menu	116
Figure 5-3 View Test Results Display	118
Figure 5-4 EEPROM Menu	119
Figure 5-5 Run Auto Test	120
Figure 5-6 Test Configuration with Cable Attachment	121
Figure 5-7 Auto Test Configuration Menu	121
Figure 5-8 Diagnostic LED Location	123

Figure 5-9 Flash Menu	124
Figure 5-10 Log File Location	125
Figure 5-11 Show Core Temperature	126
Figure 5-12 Memory Test Results	127
Figure 5-13 Memory Test with Command Line Parameter -a	127
Figure A-1 DIMM Installation	130
Figure A-2 Installation in a PCI Bus Chassis	131
Figure A-3 Connecting to the Power Supply	132
Figure A-4 SFP Module with Clasp	136
Figure A-5 Inserting or Removing an SFP Module.	136

Table 1-1 Featured Components	15
Table 1-2 WANic-58xx Hardware Specification	19
Table 2-1 GPIO Signal Description	28
Table 2-2 JTAG Mode S1 DIP Switch Configuration	35
Table 2-3 Boot Mode S1 DIP Switch Configuration	35
Table 2-4 Diagnostic Mode S1 DIP Switch Configuration	35
Table 2-5 S2 DIP Switch	36
Table 2-6 Board Status LEDs	38
Table 2-7 RJ-45 Pin Assignment	39
Table 2-8 UART0 Signals	42
Table 2-9 L4 Power Supply Pinout	45
Table 2-10 Current and Power Limits	45
Table 2-11 Thermal/Power Budget	45
Table 3-1 FPGA Memory Mapped Registers	47
Table 4-1 Boot Progress Status Messages	70
Table 4-2 Environment Variables	73
Table 4-3 Load and Boot Image Default Entries	74
Table 4-4 POST Failures	75
Table 4-1 U-Boot Commands	76
Table 4-2 EEPROM Tuples	79
Table 4-3 Flash Driver Functions	98
Table 4-4 Flash Partitioning	99
Table 4-5 IOCTL Commands	103
Table 5-1 Command Line Parameters	114
Table 5-2 Diagnostic Utility Main Menu	116
Table 5-3 EEPROM Menu Options	119
Table 5-4 Test Configuration Menu Options	122
Table 5-5 Diagnostic LED Test Prompts	123
Table 5-6 Flash Menu Options	124

Preface

This document provides technical information for the WANic-58xx Packet Processor. The WANic-58xx Packet Processors (hereafter referred to as the WANic-58xx), provide 10/100/1000 Megabits per second (Mbps) Ethernet speeds and feature a Peripheral Component Interconnect Extended (PCI-X[®]) 64-bit, 133 MegaHertz (MHz) control and data plane host interface.

The WANic-58xx provide four Gigabit Ethernet (GbE) ports in a flexible configuration of either copper or fiber front access through Small Form Factor Pluggable (SFP) modules. The WANic-58xx are available in a variety of configurations. (Unless specified otherwise, WANic-58xx refers to all models.)

The WANic-58xx complies with the Restriction of Hazardous Substances (RoHS) directive.



NOTE

This manual only describes the WANic-58xx as the WANic-38xx is no longer sold by GE Intelligent Platforms. If you have questions about this product or any other GE Intelligent Platforms product, please contact GE Intelligent Platforms Customer Technical Support.

Audience

This manual is intended for the engineers or developers who use the WANic-58xx to develop telecommunication and data processing applications.

It is assumed that the user is familiar with standard cabling, configurations of operating systems, networks, and PCI-X 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*, American National Standards Institute (ANSI) 2005, www.ieee.org.
- *PCI-X Electrical and Mechanical Addendum to the PCI Local Bus Specification Revision 2.0a*, PCI-SIG, 2003, www.pcisig.com.
- *PCI Local Bus Specification Revision 3.0* PCI-SIG, 2002, www.pcisig.com.

Vendor Documentation

- *Cavium Networks OCTEON CN38xx/58xx Hardware Reference Manual*, Cavium Networks, www.caviumnetworks.com.

Manual Organization

This manual contains the following chapters and appendices:

- *“Chapter 1 • Overview,”* presents an overview of WANic-58xx hardware and firmware as well as compliance information and warranty.
- *“Chapter 2 • Hardware Description,”* describes the hardware components on the WANic-58xx. It also includes a description of connector pinouts and Light Emitting Diodes (LEDs).
- *“Chapter 3 • FPGA Registers,”* describes the register for communications between components on the WANic-58xx.
- *“Chapter 4 • Software Features,”* provides information on how to use software components to including a boot loader and Application Programming Interfaces (APIs).
- *“Chapter 5 • WANic-58xx Diagnostic Utility,”* describes the diagnostic utility, which verifies and configures components on the WANic-58xx.
- *“Appendix A • Hardware Installation and Removal Procedures,”* describes the procedures to install or hot swap the WANic-58xx.
- *“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: GE Intelligent Platforms Embedded Systems Software License Agreement – Object Code,”* contains a copy of the terms and agreements for the GE Intelligent Platforms software license agreement object code.
- *Appendix D • GE INTELLIGENT PLATFORMS EMBEDDED SYSTEMS, INC. Software License Description for the WANic-58xx,”* contains a description of the software license.

This manual also contains a glossary and an index.

Notation Conventions

This manual uses the following notation conventions:

- *Italics* emphasizes words in text 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 independent **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 using the `modprob` command:

```
#modprob -v xxxx
```
- Notes, cautions, and warnings call attention to essential information:



NOTES

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 chapter provides an overview of the WANic-58xx hardware and software. The high-bandwidth and high-speed PCI-X interface on the WANic-58xx is designed for optimized implementation in a wide range of compatible platforms. The WANic-58xx complies with the PCI Local Bus Specification Revision 3.0 as a dual-slot, short card.

1.1 Features

The WANic-58xx features the components in Table 1-1 in several configurations:

Table 1-1 Featured Components

Featured Component	WANic-58xx
Cavium™ OCTEON™ Multi-Core Processor	600 MHz
Service Communications Protocol (SCP) with cnMIPS64 ¹ cores	None
Network Service Processor (NSP) or SCP with cnMIPS64™ cores	cn5860-NSP 16
Double Data Rate Type 2 (DDR2) packet memory	2 GB
Flash Read Only Memory (ROM)	128 MB
Optional, Reduced Latency DRAM II (RLDRAM II)	256 MB
Front Panel I/O Gigabit Ethernet (GbE) ports	4
Small Form Pluggable (SFP) Transceivers	0 to 3
Linux 2.6.x Operating System	Supported

1. cnMIPS64 is the Cavium implementation of MIPS64 by MIPS Technology Inc.

Additional featured components include:

- Light Emitting diodes (LEDs) for link status and activity indication
- Field Programmable Gate Array (FPGA) as an interface to various on-board components
- PCI/PCI-X interface for Control and Data Plane operations

1.2 Product Overview

The WANic-58xx is a fully-integrated packet processor that provide flexible processing capabilities for development of telecommunications and other packet processing intensive applications. The WANic-58xx is a network interface board with GbE serial links that provide the main Data Plane traffic connection. The WANic-58xx integrate a PCI-X interface to provide access to the 64-bit,133-MHz local bus for the main Control and Data Plane connection.

The WANic-58xx is available with four factory-installed front panel GbE I/O ports. Optionally, these front panel I/O ports can be configured at the factory for 10/100/1000Base-T or 1000Base-SX transmissions through the use of appropriate copper or fiber SFP modules.

1.3 Hardware Overview

The WANic-58xx uses the OCTEON Multi-Core Processor technology for extensive logic integration and flexibility. The OCTEON Multi-Core Processor scales up to 600 MHz on select models, and provides an interface to powerful DDR2 SDRAM with Error Correction Code (ECC). On select models, DDR2 SDRAM provides up to 2 GB of main memory and is available with up to 16 cnMIPS cores.

The WANic-58xx is available with four front panel Ethernet I/O ports configured at the factory. The four front panel GbE ports implement the 10/100/1000Base-T and 1000Base-SX interfaces, which are defined by the IEEE 802.3 (Gigabit Ethernet) specification.

RLDRAM memory (for the Multi-Core Processor) provides fast random access with extremely high bandwidth and high density for latency-sensitive tasks. RLDRAM is optional on all WANic-58xx models.

Flash ROM provides 128 MB of memory for boot and application code storage.

The 64 KiloByte (KB) serial EEPROM on the WANic-58xx contains board-specific non-volatile data information for storage and retrieval.

FPGA registers provide the Multi-Core Processor and PCI-X host with real-time access to on-board devices. Status and control of some devices are indirectly accessed through Inter-Integrated Circuit (I2C) interfaces, which the FPGA provides.

1.4 Software Overview

The WANic-58xx Software Developer's Kit (SDK) provides an easy-to-use development interface for the Linux Operating System. The WANic-58xx provides boot-up services, diagnostics, and device drivers, as well as a customized Application Programming Interface (API) for select on-board devices.

1.4.1 Software Components

Software for the WANic-58xx consists of the following components as described in "*Chapter 4 • Software Features.*"

- U-Boot Bootloader – loads and boots the WANic-58xx firmware.
- Linux Support Package (LSP) – provides a suite of functions to access and to use the WANic-58xx hardware.
- Linux Operating System – contains the Debian™ distribution with OCTEON Multi-Core Processor support.



NOTES

Contact your GE Intelligent Platforms sales representative for additional operating system support.

- Diagnostic Utility – configures and exercises various aspects of the 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 Cavium Software Developer's Kit.

1.4.2 Additional Software

For your convenience, the WANic-58xx 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.

1.4.3 Operating System Support

The WANic-58xx supports the following operating systems:

- Linux Kernel Operating System, Version 2.6.x¹, which is included in the software distribution to support both the Multi-Core Processor and the host
- OCTEON Simple Executive or Generic Operating System, which the user must obtain from Cavium Networks. (www.caviumnetworks.com.)

¹Where x is an update or patch to Version 2.6.

1.4.4 Cavium Networks Users' Organization Web Site

The Cavium Networks Users' Organization web site, www.cnusers.org, provides a platform for expanding and supporting the user community for the OCTEON Multi-Core Processor (MIPS64) family. This web site provide a means for sharing software such as Linux-based software on the OCTEON Multi-Core Processor. This site includes data plane functions, RGMII drivers, and Simple Executive applications.



NOTE

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

Refer to the www.cnuser.org web site for more information on implementing specific applications.

1.5 Specification

Table 1-2 lists the hardware specifications for WANic-58xx models.

Table 1-2 WANic-58xx Hardware Specification

Description	WANic Model	
	5862N-2RT	5864N-2R
Form Factor	PCI R3.0 dual-slot, short card	
Bus Type	PCI-X/PCI interface (33 or 66 MHz / 66 or 133 MHz)	
Processor	CN5860 NSP @ 600 MHz	
cnMIPS64 cores	16	16
Flash	128 MB	
DDR2 SDRAM	2 GB	2 GB
Optional RLDRAM	256 MB	
EEPROM	64 KB	
SFPs	Copper	Fiber
Cooling Device	Passive Heat Sink	
Power Requirements		
Maximum	Less than 42W	
Compliance		
CE Mark	Yes	
Emissions Class A (See also the section entitled “Emission Compliance Notice”)	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	
Safety	Canada – CSA22.2 NO 60950-1 Europe – EN60950-1 USA – UL60950-1	
Flammability	UL94V0	
Environmental Requirements		
NEBS	Designed for compliance to NEBS requirements, based upon design practices as well as internal and external testing.	
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)	
Operating Altitude	3.9624 Kilometers (13,000 feet) maximum	
Mean Time Between Failure (MTBF)		
Passive Heat Sink Only/ without Fan	852,639 hours	
Warranty		
Module	Two years	

1.6 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 — EN5022**



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**



NOTES

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.

1.7 RoHS and WEEE Compliance

GE Intelligent Platforms product initiative encourages the design of electronic products with European Union Directive 2002/95/EC – Reduction of Hazardous Substances (RoHS) environmentally-safe recycling and recovery in mind, as well as the Waste Electrical and Electronic Equipment (WEEE) reduction in the amount of hazardous chemicals used in electronic manufactures. For information on the GE Intelligent Platforms RoHS and WEEE implementation, please refer to the www.ge-ip.com website.

1.8 Warranty and Repair

GE Intelligent Platforms, Inc. provides a comprehensive 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 Telum family of AMC modules or ATCA Carrier Blades. The web site also contains sales office locations, copyrights, trademarks, press releases, warranties, and technical support information.

1.8.1 Warranty

Warranty information is described on the GE Intelligent Platforms' 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 specific product warranty.

1.8.2 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 Central Time (CT).

Use one of the following methods to contact technical support:

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

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

- Model number
- Serial number
- Purchase receipt
- Description of problem
- System BIOS
- Driver Version

1.9 Handling Precautions

Read the following ESD and laser/optical precautions before handling any GE Intelligent Platforms component.



NOTE

GE Intelligent Platforms, Inc. assumes no liability for the user's failure to comply with these required precautions.

1.9.1 ESD Precautions

Always take the necessary precautions to prevent ElectroStatic Discharge (ESD). ESD can damage the WANic-58xx, which has ESD sensitive components.

All products should be in an anti-static plastic bag or conductive foam for storage or shipment. Work at an approved anti-static workstation when unpacking the WANic-58xx. Also, **always** use anti-static grounding straps to prevent damage due to ESD.

Use the following precautions to prevent ESD when removing the card from the antistatic packaging.



CAUTION

Always ground yourself *before* touching the board. You can ground yourself by either touching a grounded unpainted metal surface for at least two seconds, or by using an ESD-protective wrist strap from your wrist to a bare, unpainted metal section of the grounded chassis or rack. This prevents electrostatic discharge from damaging the board.

When working with a chassis, if the chassis cannot be placed upon a grounded antistatic mat, connect a grounding strap between the electrical input ground and the facility electrical service ground.

Always keep the board in its static-protective packaging when it is not installed in the system.

Always hold the board by its handles or edges. Avoid touching the components or connections on the board.



WARNING

Depending on the chassis, open equipment enclosures and chassis can expose hazardous voltage which may cause electric shock to the installer. Make sure line power to the equipment is disconnected before servicing the chassis and components.

1.9.2 Optical/Laser Safety Notices

- For optical/laser SFP/SFP+ devices, 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

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.



NOTES

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.

2 • Hardware Description

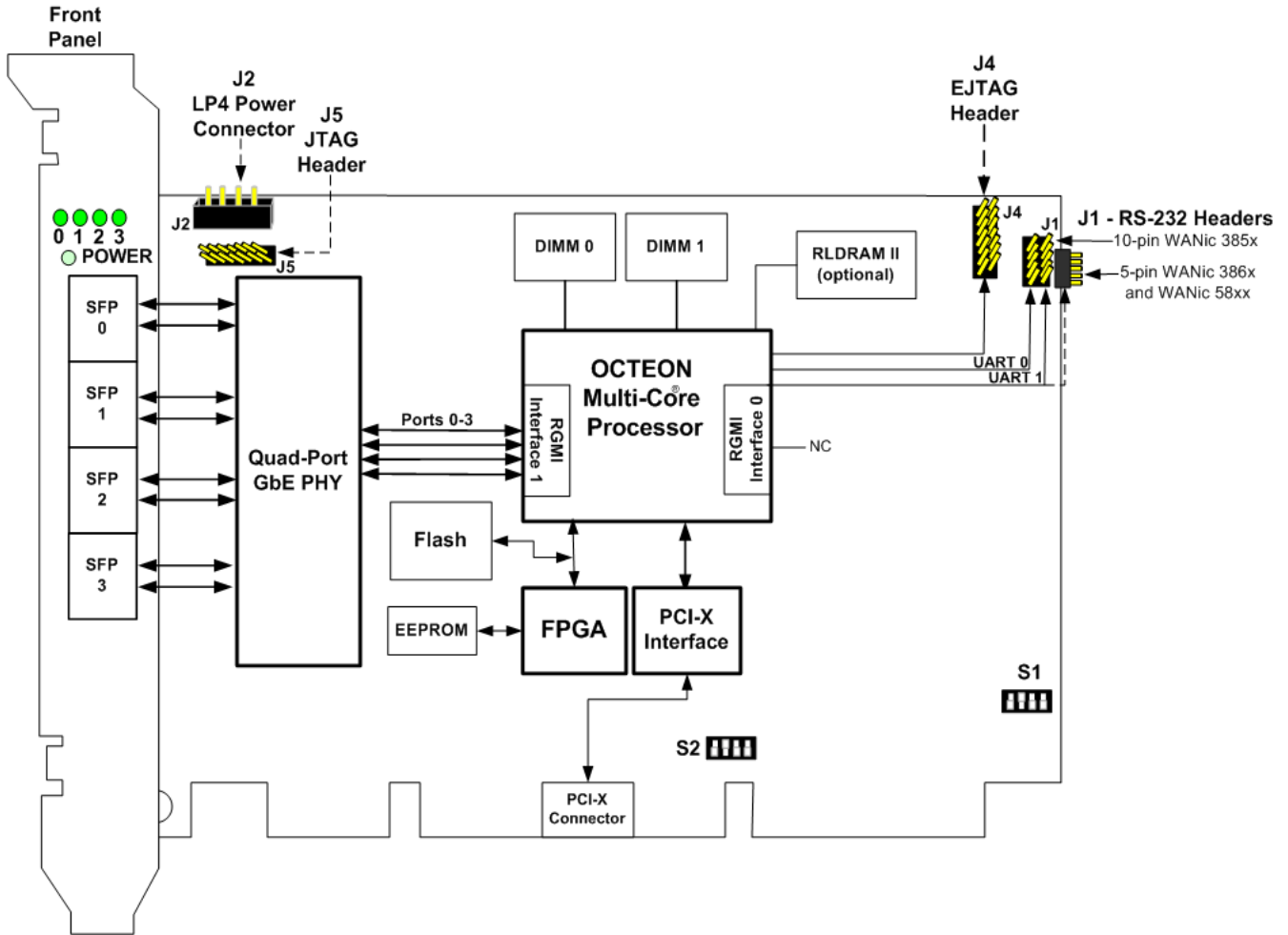
This chapter describes the hardware components on the WANic-58xx. This chapter contains a simple block diagram of the components, which include the Multi-Core Processor, memory, controllers, and external interfaces as well as connectors.

2.1 Hardware Components

The WANic-58xx features the following hardware components as shown in Figure 2-1:

- Cavium OCTEON Multi-Core IP Packet Processor with up to 16 cnMIPS64 cores running at rates up to 600 MHz
- PCI-X Interface, 64-bit, 133 MHz
- Field Programmable Gate Array (FPGA) for access to on-board components
- DDR2 SDRAM with 128-bit bus width plus ECC implemented in registered mini-Dual In-line Memory Modules (mini-RDIMMs)
- Up to 256 MB of optional RLDRAM II in supported configurations
- Flash Read Only Memory (ROM)
- Four front panel Ethernet I/O ports
- SFP copper or fiber-optic transceivers for front panel I/O configuration
- On-card headers provide access to the following:
 - OCTEON Joint Test Action Group (JTAG) debug port
 - OCTEON Enhanced JTAG debug port
 - RS-232 console I/O port
 - LP4 power connector
- Dual In-line Package (DIP) switches:
 - S1 for board configuration
 - S2 for PCI/PCI-X bus type and speed selection
- Front panel link status and activity LEDs for SFPs

Figure 2-1 Block Diagram



2.2 Multi-Core Processor

All models of the WANic-58xx use the Cavium OCTEON family of multi-core packet processors. The OCTEON Multi-Core Processor (hereafter referred to as the Multi-Core Processor) is a single-chip packet processor for Open Systems Interconnection (OSI) Layer 2 to Layer 7 networking applications. The Multi-Core Processor provides enhanced performance, reduced power consumption, and efficient simultaneous processing of multiple tasks.

Each Multi-Core Processor option is configured at assembly time to provide comprehensive integrated functions. User choices include the following factory configured selections:

- Number of embedded cnMIPS cores
- Core clock frequencies

On select models, the Multi-Core Processor is available with up to 16 cnMIPS cores with enhancements and additional built-in hardware acceleration for content and security processing. This scalable architecture combines the cnMIPS cores with dedicated programmable coprocessor blocks to deliver up to 4 Gbps of application performance at conservative 600 MHz chip clock rates on select models.

The Multi-Core Processors offer highly-flexible external networking interfaces with four integrated Reduced Gigabit Media Independent Interface (RGMII) Ethernet ports. This provides connectivity to GbE for data plane I/O.

The Multi-Core Processor interfaces with a PCI-X host interface that can be used as a control and data interface. The PCI-X interface supports 66 MHz or 133 MHz for the PCI-X, and 33 MHz or 66 MHz for the PCI interface.

The Multi-Core Processor is powered by a dedicated on-board power supply. This permits adjustments to the core supply voltage, as needed, without affecting other circuitry.

2.2.1 Multi-Core Processor Interfaces

The Multi-Core Processor contains the following interfaces:

- RGMII 1
- Two-Wire Serial Interface (TWSI)
- General Purpose I/O (GPIO)

RGMII Interface 1

The Multi-Core Processor RGMII Interface 1 functional block provides data transfer to and from the GbE PHY device using the four Reduced Gigabit Media Independent Interface (RGMII) interface ports. The GbE PHY performs serialization and de-serialization (SerDes) data transfers to and from each SFP on the WANic-58xx front panel.



NOTE

RGMII Interface 0 is not implemented in this version of the WANic-58xx.

TWSI Interface

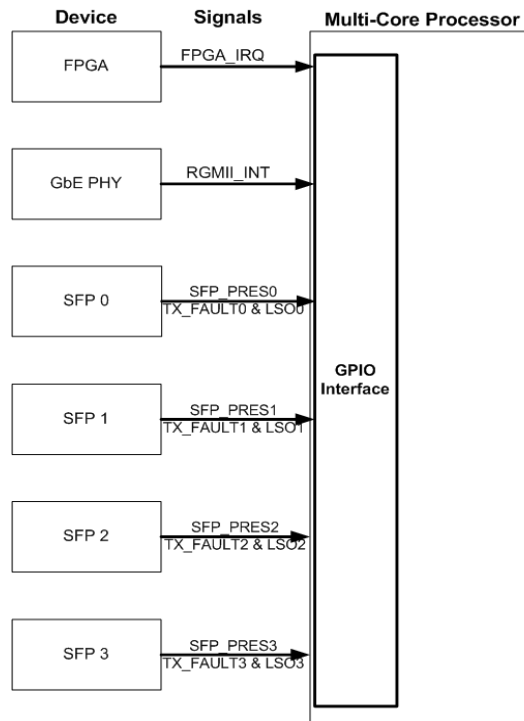
The Multi-Core Processor uses a Two-Wire Serial Interface (TWSI), which is similar to an I2C interface. For the TWSI, the Multi-Core Processor supports normal and fast modes as well as 7-bit and 10-bit interfaces. The Multi-Core Processor can operate as a master TWSI device. Any of the cores or a remote PCI host can communicate with the TWSI device. The Multi-Core Processor uses the TWSI to determine the mini-RDIMM module type. Software primitives provide access to the industry standard TWSI on the WANic-58xx. (See “[Chapter 4: Software Features](#)” for a description of these software primitives.)

GPIO Interface

The Multi-Core Processor provides a General Purpose I/O (GPIO) interface that provides an input port to detect the following signals for each SFP as shown in Figure 2-2:

- SFP presence (SFP_PRES)
- Transmit fault (TX_FAULT)
- SFP Loss of signal (SFP_LOS)

Figure 2-2 GPIO Interface to SFPs



The on-card signals in Table 2-1 are allocated to the Multi-Core Processor GPIO pins.

Table 2-1 GPIO Signal Description

GPIO PINS	Signal	Description
0–3	SFP_PRES	SFP presence detect
4–7	TX_FAULT	Transmit fault detected
8–11	SFP_LOS	SFP loss of signal
12	RGMII_INT	RGMII interrupt
13	RSVD	Reserved
14	FPGA_INT	FPGA interrupt
15	RSVD	Reserved

2.2.2 Multi-Core Processor Boot Bus

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

- 28 address lines (using big endian byte-ordering)
- Eight Chip Selects (control pins) that provide access to devices connected to the Boot Bus
- Independent read and write

The Boot Bus connects directly to the Flash, which is controlled by Chip Select 0. FPGA registers are also accessible through the Boot Bus by using Chips Select 1.

When power-on reset occurs, Core 0 of the Multi-Core Processor uses the Boot Bus to obtain code for the core that controls Chip Select 0 unless the WANic-58xx is configured to boot from the PCI-X (using the S1 DIP Switch).

2.3 PCI/PCI-X Interface

The WANic-58xx is a target (add-in) card on the PCI/PCI-X Interface. The Multi-Core Processor interfaces directly to the PCI/PCI-X interface and receives the PCI/PCI-X clock from the host. The S2 DIP Switch provides the means to configure the capabilities of the PCI/PCI-X Bus for the WANic-58xx, which the host determines at reset. The following capabilities can be selected:

- For PCI-X — 66 MHz or 133 MHz
- For PCI — 33 MHz or 66 MHz

A PCI/PCI-X Bus Controller uses the power-on reset state of the PCI bus to generate the input clock for all devices on the bus.

- For PCI-X — The bus controller drives the associated initialization pattern to all devices on the bus to configure them according to the sampled state of the bus signals at reset. The Multi-Core Processor configures the internal PCI-X bus logic automatically based on the initialization pattern initiated by the PCI/PCI-X Bus Controller.
- For PCI — Logic is configured according to the timing of the input clock signal.

See “[Section 2.8 “Dual In-Line Package \(DIP\) Switches”](#)” for more information on configuring the PCI/PCI-X Bus.

2.4 FPGA

The FPGA on the WANic-58xx provides a software interface to various on-board hardware resources. The FPGA contains registers that allow the Multi-Core Processor and PCI/PCI-X host to control and to monitor transactions among interfaces and local hardware resources.

The FPGA routes individual reset signals to the Flash memory, mini-RDIMM memory, and GbE transceiver devices. These resets are asserted at power-up, and are de-asserted automatically once power is stable. Software can also force an individual reset to these devices using the **Peripheral Reset Register**. (See “[Chapter 3.1: FPGA General Registers](#)” for more information on this and other FPGA registers.)



NOTE

A PCI_RESET from the host *does not* directly reset these devices.

The FPGA registers interface to the Multi-Core Processor by means of the Boot Bus and Chip Select logic. The Multi-Core Processor must be configured to assert this Chip Select to enable access to FPGA registers in the desired range of the address space according to software needs.

All FPGA registers are mapped to a contiguous block of 32 bytes within the address space. In cases when the Chip Select logic configuration accesses a range of addresses larger than 32 bytes, the FPGA register image repeats.

I2C Interface

The FPGA also provides independent I2C ports to six interfaces with the following devices:

- Four front panel SFPs
- A 64 KB serial EEPROM
- A temperature sensor

The FPGA performs read and write operations on each I2C interface in response to requests from the Multi-Core Processor through specific FPGA registers. See “[Chapter 3.2: I2C Registers](#)” for a description of each FPGA register.

2.5 Memory

The main memory on the WANic-58xx consists of DDR2 SDRAM and, optionally, RLD RAM memory. Flash memory provides storage for initial boot code.

2.5.1 DDR2 SDRAM

The WANic-58xx supports up to 2 GB¹ of DDR2 SDRAM memory operating at speeds up to 667 MHz and a full 128-bit data path. An additional 16 bits provide ECC support. ECC permits 1-bit error correction, and 2- and 3-bit error detection. Timing for the DDR2 SDRAM interface is derived from an independent reference clock and is not dependent on the processor core frequency. The Multi-Core Processor directly controls the DDR2 SDRAM.

Mini-RDIMMs

The DDR2 SDRAM is implemented as two socketed miniature registered Dual In line Memory Modules (Mini-RDIMMs) with 512 MB or 1 GB each depending on the model. The WANic-58xx use registered mini-RDIMMs with ECC support. Each vertical Mini-RDIMM socket provides a 64-bit data bus and ECC. To ensure the clock signal integrity, Mini-RDIMMs are built with address line registers and clock Phase Lock Loop (PLL). The WANic-58xx supports both single- and dual-rank Mini-RDIMMs.



NOTES

Contact Customer Technical Support for a listing of supported memory.

Serial Presence Detect

Each Mini-RDIMM module features a Serial Presence Detect (SPD) based on a serial EEPROM device, which uses the I2C protocol to access information about each Mini-RDIMM. Mini-DIMM sockets (0 and 1) are connected to the Multi-Core Processor, TWSI I2C interface, and hardwired to device addresses **0x50** and **0x54** respectively.

Reference Clock

Timing for the DDR2 SDRAM is derived from an independent reference clock input. The reference clock frequency is chosen to match the speed of the DDR2 SDRAM devices connected to the interface.

The DDR2 SDRAM reference clock is derived from a crystal and PLL frequency multiplier. The reference clock frequency can be programmed through hard strap selection at the factory to permit DDR2 data transfer rates of up to 667 MHz.

2.5.2 Optional RLD RAM II

The RLD RAM II (hereafter referred to as RLD RAM) is optional on all WANic-58xx models. Consult with your GE Intelligent Platforms sales representative for detailed information.

The RLD RAM interface supports up to 256 MB of externally connected RLD RAM over two independent 18-bit data paths. The effective data transfer rate is further enhanced by interleaving memory accesses between the two banks. The WANic-58xx supports two optional RLD RAM devices per 18-bit data path, for a total of four RLD RAMs.

The Multi-Core Processor generates the RLD RAM clocks, which run at half the core clock frequency. The data transfer rate matches the core clock frequency.

1. Additional configuration are available. Please contact GE intelligent Platforms Customer Technical Support or Sales for information.

2.5.3 Flash

The WANic-58xx contains a 128 MB Flash EEPROM device (hereafter referred to as the Flash). Code execution is supported from this Flash during hardware initialization only. The Flash is connected directly to the Multi-Core Processor Boot Bus as shown in Figure 2-3.

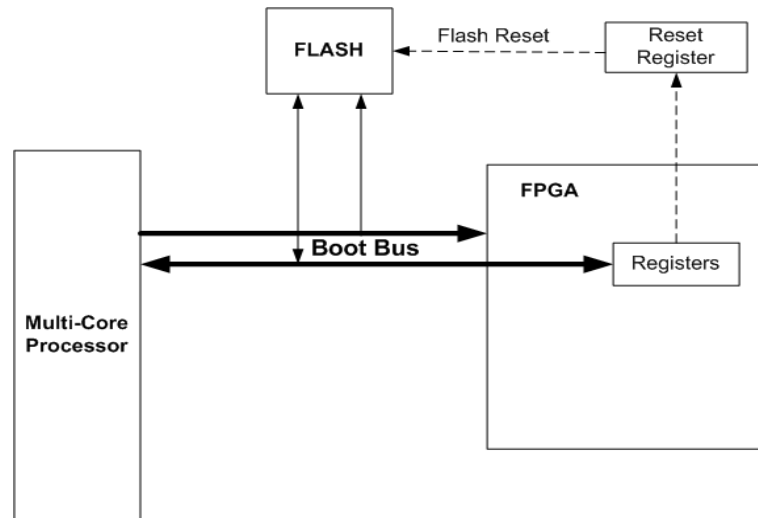


NOTE

Flash boot occurs when DIP Switch 3 is set to the Off position.

To optimize code performance, the operating firmware copies the boot code into the DDR2 SDRAM for use following hardware initialization. The Flash device has the ability to be written to, or erased, when required by operating firmware.

Figure 2-3 Flash Connection to Boot Bus



The Flash has an active low hardware reset pin ($\overline{\text{FLASH_RESET}}$). This hardware reset pin terminates all operations in progress and resets the Flash, after which the Flash is ready for a new operation. The Flash reset is asserted at power-up and is automatically de-asserted once power is stable. Software forces a reset of the Flash by setting the appropriate bit in the **Peripheral Reset Register**. (See “[Chapter 3: FPGA Registers](#)” for more information on this and other FPGA registers.)



NOTE

A `PCI_RESET` from the host does not directly reset the Flash.

2.5.4 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 Multi-Core Processor by means of the FPGA. This permits examination and modification of the EEPROM contents.

2.6 Gigabit Ethernet PHY Devices

The WANic-58xx contain a GbE physical (PHY) device, which is a Marvell® 88E1143 quad-port PHY. This GbE PHY is a high-performance quad GbE device that provides four independent ports and interconnects serial data on the Multi-Core Processor by means of four RGMII interface ports. The GbE PHY implements the IEEE 802.3z Physical Coding Sublayer (PCS).

This GbE PHY is factory-configured to support either copper or optical SFPs in Serial Gigabit Media Independent Interface (SGMII) mode. Copper SFPs are desirable when using SGMII mode because copper provides the ability to operate the link at 10, 100, and 1000BASE-T data rates.

In SGMII mode, four 1000Base-T Ethernet ports are provided through four SFP receptacles on the front panel. These four ports implement all standard PHY interface functions defined in the IEEE 802.3-2000 specification and supports standard 802.3 Ethernet frames plus 802.1p/q VLAN functions. When the GbE PHY is in use, the Multi-Core Processor automatically determines the type of SFP and controls them directly using an I2C interface for each SFP.

The GbE PHY contains several registers to manage the device. These registers are interconnected with the Multi-Core Processor by means of a standard two-wire Management Data I/O (MDIO) interface. The Multi-Core Processor contains a dedicated System Management Interface (SMI) Controller for communicating with this interface.

The GbE PHY has four identical ports, each with associated management registers. The MDIO address is comprised of a 5-Bit PHY address field and a 5-Bit register address field. When accessing the GbE PHY management registers within the WANic-58xx, note that PHY addresses 0-3 correspond to I/O ports 0-3.

The GbE PHY reset is asserted at power-up, and de-asserted automatically once power is stable. Software can also force an individual reset to the PHY device using the **Peripheral Reset Register**.



NOTE

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

2.7 SFP Front Panel Connectors

The WANic-58xx provides a flexible configuration of copper SFPs or single-mode or multi-mode optical SFP modules. SFP modules insert into front panel slots in the WANic-58xx to provide network connections to external line interfaces. SFP copper ports use a standard RJ-45 connector. SFP fiber ports use a standard LC connector.

The FPGA provides an I2C interface for each SFP. The I2C interface can be used to determine the type of SFP installed, and to configure each SFP accordingly.

The GbE PHY also provides an LED interface to indicate link status and activity for each SFP. See [“Section 2.10.1 “LEDs”](#) for more information.



NOTES

Contact Customer Technical Support for a list of certified SFPs.

2.8 Dual In-Line Package (DIP) Switches

As shown in Figure 2-4, the WANic-58xx contains two sets of DIP Switches, which toggle on and off:

- S1
- S2

2.8.1 S1 DIP Switch

The S1 DIP Switch has a bank of four dual-position DIP Switches, which are located towards the edge of the board and collectively labeled S1 as shown in Figure 2-4. Use the S1 DIP Switch to perform the following:

- FPGA programming
- EJTAG access
- Selects the Boot mode
- Boot the diagnostic utility

The **Board ID and DIP Switch Register** indicates the boot selection as either booting from the Flash or the PCI-X host. The following considerations should be noted:

- For JTAG Boundary Scan Tests, DIP Switch 1 and 2 must be in the On position.
- For JTAG testing of the FPGA *only*, DIP Switch 1 must be in the Off position to access the FPGA.
- When using EJTAG with a debugging probe for the Multi-Core Processor, DIP Switch 2 must be in the Off position.
- Use DIP Switch 3 to select either Flash or PCI-X boot mode.
- Turn DIP Switch 4 to the On position to start the on-board diagnostics.

Figure 2-4 DIP Switches

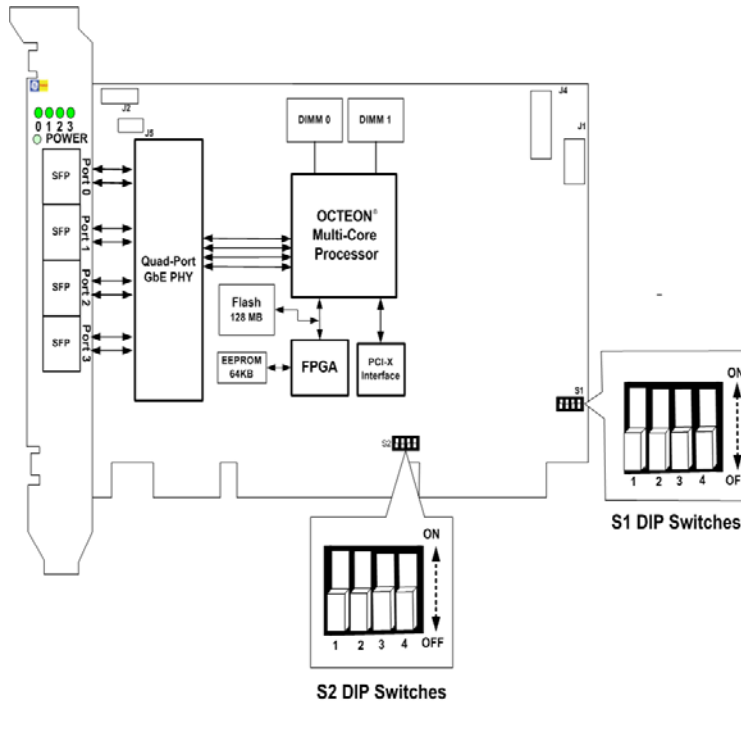


Table 2-2 through Table 2-3 summarize the DIP Switch settings for JTAG, Boot, and Diagnostic modes.

Table 2-2 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 2-3 Boot Mode S1 DIP Switch Configuration

Mode	Operation	DIP SWITCH
		3
Boot	Enables Flash Boot operations	Off
	Enables PCI-X boot operations	ON

Table 2-4 Diagnostic Mode S1 DIP Switch Configuration

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

2.8.2 S2 DIP Switch

The S2 DIP Switch contains a bank of four two-position DIP Switches located towards the bottom center of the board and labeled S2 as shown in Figure 2-4. Use the S2 DIP Switches to configure the PCI/PCI-X bus type and speed.

In Table 2-5 a dash (—) indicates that the switch is not used for that bus type. The default setting for each switch is Off.

Table 2-5 S2 DIP Switch

SW1	SW2	SW3	SW4	Bus Type	Speed
Off	Off	—	—	PCI-X (<i>default</i>)	133 MHz
Off	ON	—	—	PCI-X	66 MHz
ON	—	Off	—	PCI	66 MHz
ON	—	ON	—	PCI	33 MHz

2.9 Reference Clock Circuitry

Figure 2-5 illustrates the Multi-Core Processor reference clock circuitry. The Multi-Core Processor core reference clock input is driven by a fixed 50 MHz oscillator. A PLL within the Multi-Core Processor derives the internal core clock frequency according to certain values on the Multi-Core Processor. The 50 MHz oscillator permits the core clock frequencies of 600 MHz on the WANic 58xx models to be driven from the common reference frequency.

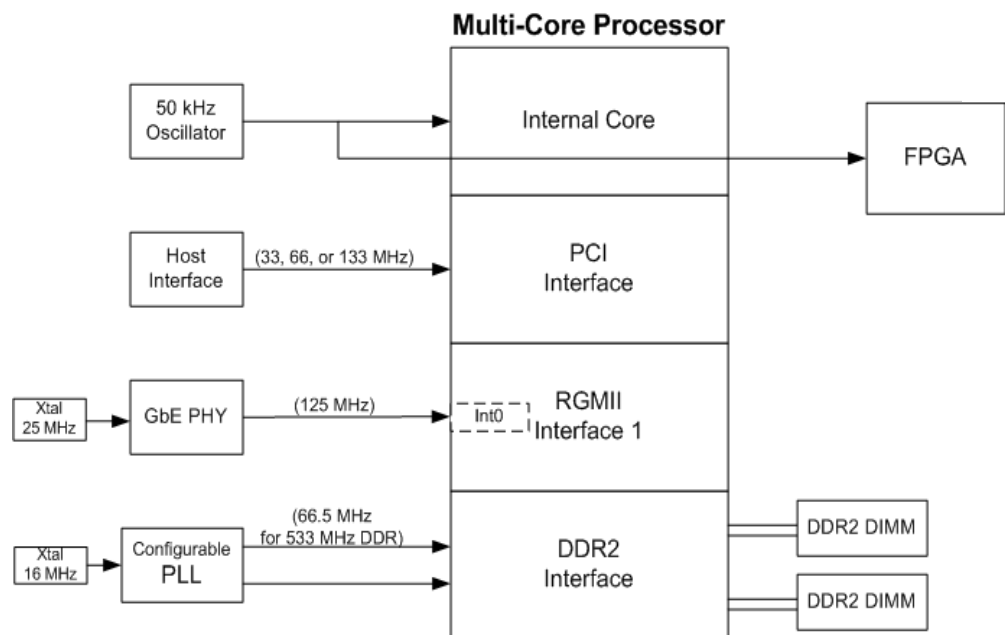
The 50 MHz signal is also routed to the FPGA clock input.

The Multi-Core Processor receives the PCI-X/PCI clock from the host PCI-X interface.

The GbE PHY reference clock is derived from the 25 MHz crystal (Xtal). The 125 MHz GbE PHY clock is generated by an integrated PLL and routed to the Multi-Core Processor.

The DDR2 reference clock is derived from a 16 MHz crystal and configurable clock synthesizer PLL. Within the Multi-Core Processor, a PLL uses the reference clock frequency to create the DDR2 clocks.

Figure 2-5 Reference Clock Circuitry



2.10 Front Panel

The WANic-58xx contains SFP front panel modules as well as power and link status activity LEDs. SFP modules provide optical or copper interfaces.

2.10.1 LEDs

The WANic-58xx contain five front panel LEDs as shown in Figure 2-6. Status signals include a power LED and four link status and activity LEDs, one for each SFP.

Figure 2-6 Front Panel LEDs

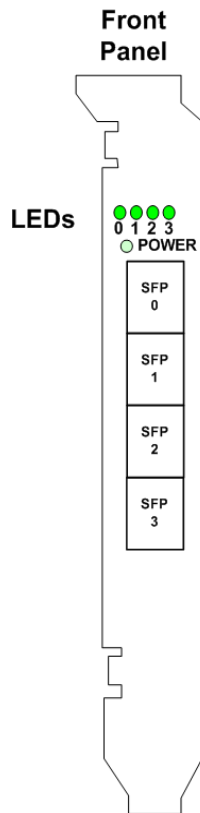


Table 2-6 Board Status LEDs

LED	Color	Indicator	State	Description
SFP [3-0]	Green	Link/Status activity	Off	Link is down.
			Blinking	Link is in use transmitting or receiving data.
			On steadily	Link is up.
POWER	Green	Power Status	On	Normal operation.
			Off	Disabled, or below the minimum operating threshold.

2.10.2 SFP Modules

SFP modules provide flexible connectivity to the type of media used for I/O. The WANic-58xx uses either copper or fiber SFP transceivers.

- Fiber transceivers support 1000Base-SX connectors.
- 1000Base-T transceivers allow the use of copper twisted-pair media with standard RJ-45 connectors. The 1000Base-T Ethernet port on the front panel of the WANic-58xx is a medium dependent interface (MDI) connection. This port uses a standard RJ-45 connector. Table 2-7 shows the pin assignment for the RJ-45.

Figure 2-7 RJ-45 Connector

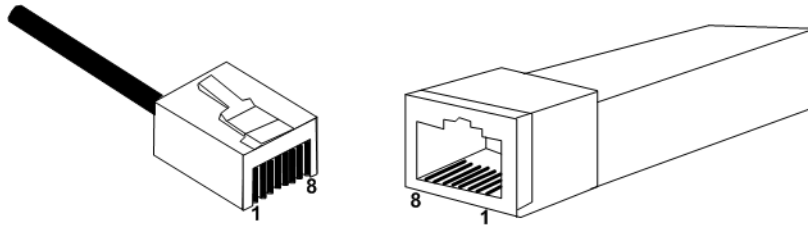


Table 2-7 RJ-45 Pin Assignment

Pin Number	Signal	Description	Pin Number	Signal	Description
1	MDI_DA+	Data Access	5	—	
2	MDI_DA-	Data Access	6	MDI_DB-	Data Access
3	MDI_DB+	Data Access	7	—	
4	—		8	—	



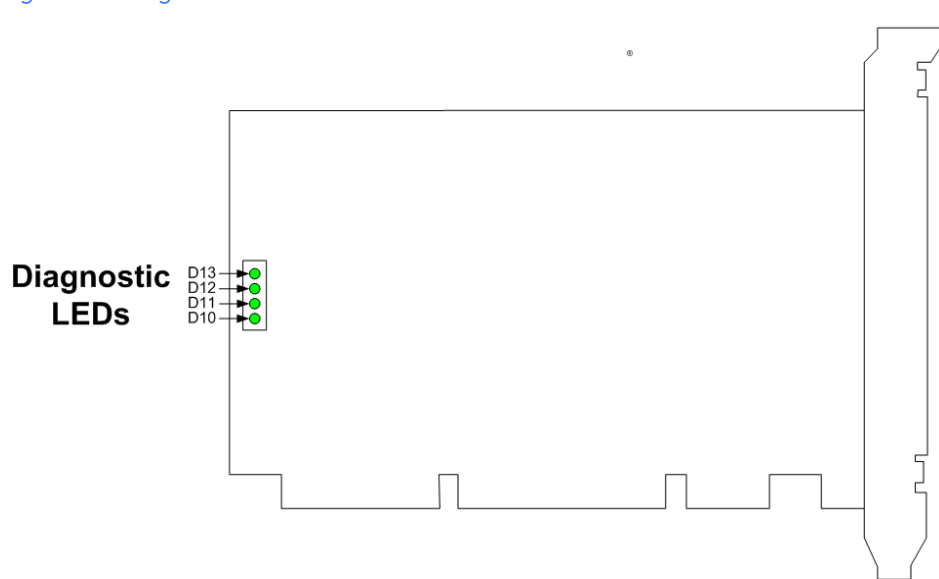
NOTE

Contact your GE intelligent Platforms Customer Support Representative for a listing of approved SFPs.

2.11 Diagnostic LEDs

The WANic-58xx contains four diagnostic LEDs located on the side of the card as shown in Figure 2-8. Use the **LED Control Register** to control these diagnostic LEDs.

Figure 2-8 Diagnostic LEDs



2.12 Headers

The WANic-58xx contains three headers:

- J5 — is a 14-pin header for the JTAG interface
- J4 — is a 14-pin header for the EJTAG interface
- J1 — is either a 5-pin or 10-pin header for the RS-232 interface

Certain signals within the JTAG port are also shared with the EJTAG scan chain. Use the on-board DIP Switch to select 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 or two Universal Asynchronous Receiver Transmitters (UARTs) depending on the model.

See [Section 2.12.3 "RS-232 Serial Port"](#) for more information on these UARTs.

2.12.1 JTAG Scan Chain

To support testing and debugging, the WANic-58xx has connected all components with JTAG test ports to a JTAG scan chain. (See Figure 2-9.) The JTAG header is a standard 14-pin 0.100" dual-row boxed header located towards the back of the board, as shown in Figure 2-1, which provides access to the JTAG scan chain. The JTAG scan chain connects to the following components:

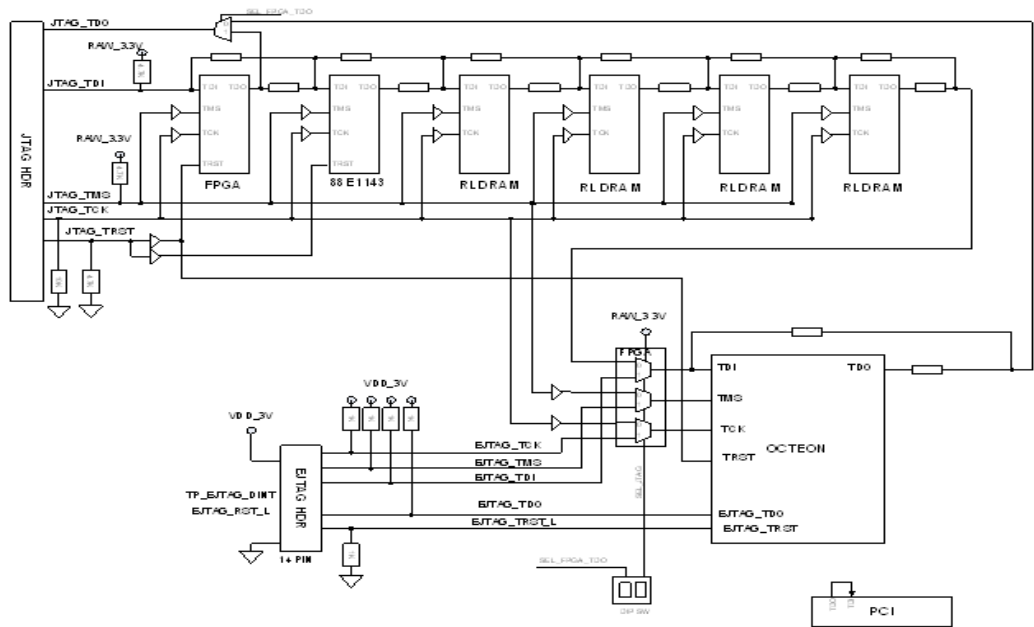
- Multi-Core Processor
- RLDRAM (when installed)
- FPGA
- GbE PHY

2.12.2 Enhanced JTAG

The Multi-Core Processor JTAG pins can be connected to the JTAG scan chain or routed out to an Enhanced JTAG (EJTAG) header. Multiplexers, under control of the on-board DIP Switch, route the JTAG pins.

The EJTAG header is a standard 14-pin, dual-row boxed header with 0.1 pin spacing. It provides keying of a mating cable to prevent inadvertent insertion in the wrong direction. The EJTAG header interface permits the Multi-Core 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 device.

Figure 2-9 JTAG Scan Chain and EJTAG Interface



1. RLD RAM is not included in the JTAG Scan Chain when RLD RAM is not installed.

2.12.3 RS-232 Serial Port

The Multi-Core Processor provides a general purpose serial communications controller. This industry standard 16550-style Universal Asynchronous Receiver Transmitter (UART) is capable of sending and receiving data at rates up to 115K bits per second (bps). The Multi-Core Processor also provides a baud rate generator, which drives the baud rate by dividing the Multi-Core Processor core clock frequency by a user-specified 16-bit divisor, multiplied by 16.

Use the RS-232 serial port(s) for console access to the Multi-Core Processor. The RS-232 serial port is programmable up to the default baud rate of 115Kbps. For RS-232 serial port connections, the WANic-58xx use a 5-pin header. These UART ports are accessible for debug purposes.

On the WANic-58xx, UART Port 0 uses the 5-pin header (J1). The 5-pin header signals for UART 0 are described in Table 2-8.

Table 2-8 UART0 Signals

Pin	PCB Pin	Signal	Description
1	1	CTS	Clear To Send
2	3	RX	Receive
3	5	TX	Transmit
4	7	RTS	Request To Send
5	9	GND	Ground

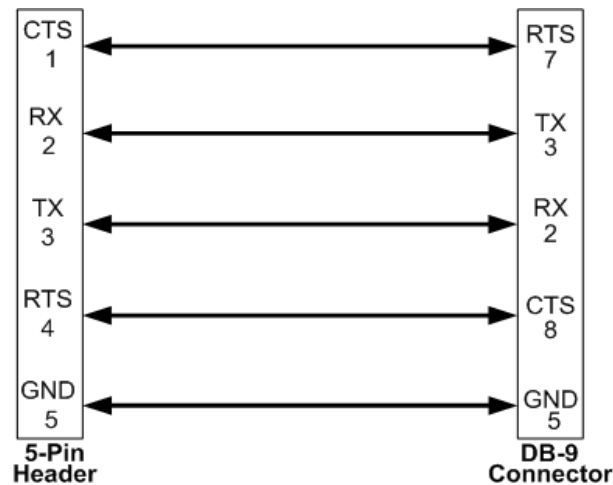
An optional Serial Cable is available to allow the UART device to terminate to a DB-9 female connector. The following section provides information on the Serial Cable. Contact a GE Intelligent Platforms sales representative for additional information on this optional Serial Cable.

Serial Cable (Optional)

On the WANic-58xx, the optional Serial Cable connects the UART on the 1x5 0.1" pitch right angle J1 connector to a standard DB-9 Connector for terminal input.

The 5-pin RS-232 connector permits the attachment of a standard Insulation Displacement Connector (IDC) cable consisting of a 5-pin header and a standard 9-pin RS-232 DCE interface. The pinout for this cable is shown in Figure 2-10.

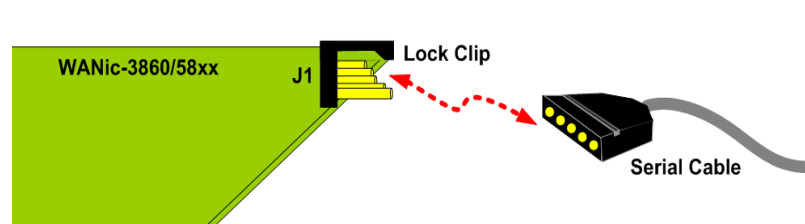
Figure 2-10 Serial Cable Pinout



NOTES

1. When attaching the Serial Cable Connector to the J1 5-Pin Connector, make sure the groove on the top of the Serial Cable Connector aligns with the Lock Clip (or notch) on the J1 5-pin Connector. (See Figure 2-11, which illustrates the Lock Clip.) The Serial Cable connector snaps into place when the Lock Clip attaches properly.
2. The pinout and gender of the DB-9 Connector permits it to be attached directly to Data Terminal Equipment (DTE) devices, such as a dumb terminal or terminal emulation software running on a personal computer. If an extension cable is required, 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. The cable **does not** have built-in crossover or null modem functionality.
3. The RS-232 IDC cable assembly is not keyed.

Figure 2-11 Serial Cable Connection



2.13 Temperature Monitor

The Multi-Core Processor requires a cooling mechanism, which is a passive heat sink.

All models contain a Temperature Monitor to actively monitor the temperature of the Multi-Core Processor.

The Temperature Monitor provides set points for the temperatures. When a set point is crossed, the Temperature Monitor 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-58xx 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 **Fan and 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.

2.14 Power and Cooling

The WANic-58xx hardware uses multiple voltage sources and is designed for an air-cooled chassis environment.

2.14.1 Power Supplies

The WANic-58xx hardware is powered through the on-card PCI-X interface (+3.3V) and also through a dedicated LP4 power connector (+12V) located on the upper left corner of the board as shown in Figure 2-1. The pinout for this L4 Power Supply is shown in Table 2-9.

Table 2-9 L4 Power Supply Pinout

Pin	Signal
1	12V
2	GND
3	GND
4	No connect

The remaining device voltages (1.2V, 1.8V, and 2.5V) are derived from the LP4 +12V power through the on-card DC/DC converters. Table 2-10 identifies the current and power limits.

Table 2-10 Current and Power Limits

Power Supply	Voltage	Maximum Operating Current	Maximum Power
LP4 Connector	+12V	3.25A	39W
PCI-X Connector	3.3V	1.8A	6W

2.14.2 Thermal Management

The WANic-58xx meet the thermal/power budget as described in Table 2-11.

Table 2-11 Thermal/Power Budget

Model	Configuration	Power
WANic-5860	16 cnMIPS cores @ 600 MHz	42W



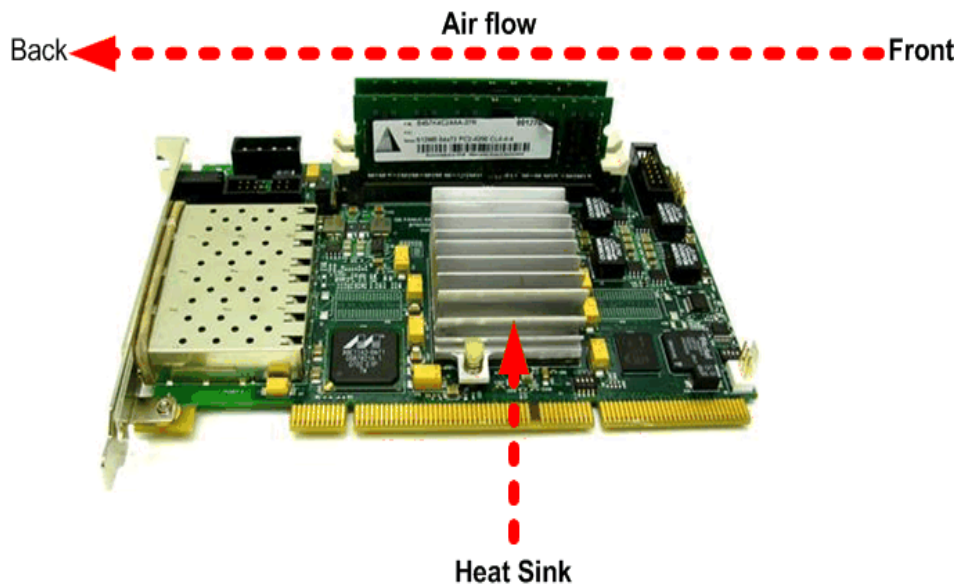
CAUTION

Maximum power requirements are different for each WANic-58xx model and configuration. Consult with Customer Technical Support for specific power requirements for each model.

Depending on the model configuration, thermal requirements are satisfied by either a direct cooling mechanism (fan-heat sink) or a passive heat sink requiring sufficient air flow. The heat sink is attached to the Multi-Core Processor and other high thermal-load components. The heat sink with fin orientation, as shown in Figure 2-12, is in the same direction as the standard front-to-back air flow path provided by the chassis.

An LM83 temperature monitor, in conjunction with a thermal diode within the Multi-Core Processor, provides a direct indication of the processor die temperature. Use the LM83 temperature monitor to determine whether there is sufficient cooling air flow to maintain a junction temperature of less than 110°C in accordance with the maximum ratings specified for the Multi-Core Processor.

Figure 2-12 Air Flow on the WANic-58xx



NOTES

Contact GE intelligent Platforms Customer Technical Support for detailed air flow requirements for various technical environments.

3 • FPGA Registers

This chapter describes the registers that the Multi-Core Processor uses to communicate with other on-board components.



NOTE

While the Multi-Core Processor can access FPGA registers directly, FPGA registers are also available to the external host indirectly through the PCI-X bus.

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

- Reset
- Interrupts
- LEDs
- Temperature Sensor and Fan Controller
- Status indications

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

Access in Table 3-1 is as follows:

R=Read

W=Write

O=Only

I/O Mapping space is in the Multi-Core Processor.

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 front panel module status LEDs.	R/W
0x03	Interrupt Control	Manages interrupts for selected devices.	R/W
0x04	Interrupt Status	Indicates interrupt status for selected devices.	R/O
0x05	Peripheral Reset	Manages the reset logic for the on-board peripherals.	R/W
0x06	Transmit Control	Manages SFP transmitter output.	R/W
0x07	Fan and Temperature Status	Indicates die temperature and fan speed status.	R/W
0x08	Temperature I2C Control	Manages the temperature sensor transactions.	R/W
0x09	Reserved		
0x0A	Temp. I2C Address	Specifies the address for the transaction.	R/W
0x0B	Temp. I2C Data	Contains the data for the transaction.	R/W
0x0C	EEPROM I2C Control	Initiates an I2C transaction and 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

Table 3-1 FPGA Memory Mapped Registers

Offset	Register	Description	Access
0x10	SFP3 I2C Control	Initiates a read or write transaction for SFP3.	R/W
0x11	SFP3 I2C Address High	Specifies the upper bits of an address for SFP3.	R/W
0x12	SFP3 I2C Address Low	Specifies the lower bits of an address for SFP3.	R/W
0x13	SFP3 I2C Data	Specifies data associated with a read or write transaction on SFP3.	R/W
0x14	SFP2 I2C Control	Initiates a read or write transaction for SFP2.	R/W
0x15	SFP2 I2C Address High	Specifies the upper bits of an address for SFP2.	R/W
0x16	SFP2 I2C Address Low	Specifies the lower bits of an address for SFP2.	R/W
0x17	SFP2 I2C Data	Specifies data associated with a read or write transaction on SFP2.	R/W
0x18	SFP1 I2C Control	Initiates a read or write transaction for SFP1.	R/W
0x19	SFP1 I2C Address High	Specifies the upper bits of an address for SFP1.	R/W
0x1A	SFP1 I2C Address Low	Specifies the lower bits of an address for SFP1.	R/W
0x1B	SFP1 I2C Data	Specifies data associated with a read or write transaction on SFP1.	R/W
0x1C	SFP0 I2C Control	Initiates a read or write transaction for SFP0.	R/W
0x1D	SFP0 I2C Address High	Specifies the upper bits of an address for SFP0.	R/W
0x1E	SFP0 I2C Address Low	Specifies the lower bits of an address for SFP0.	R/W
0x1F	SFP0 I2C Data	Specifies data associated with a read or write transaction on SFP0.	R/W

FPGA registers include general registers and I2C registers.

3.1 FPGA General Registers

General register descriptions are as follows.

FPGA Revision Register

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

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

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 - This lower-bit number increments for each minor hardware revision of the module.	0x00	R/O
7-4	MAJ_REV	Major Revision Level - This upper-bit number increments for each major hardware revision of the module.	0x02	R/O

Board ID and DIP Switch Register

The **Board ID and DIP Switch Register** is an 8-bit read-only. In addition to configuration indicators for the heat sink and RLDRAM, this register also indicates the position of the user-controlled S1 DIP Switches. User-controlled DIP Switches can be in the on (closed) or off (open) position.

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

Bit 7	6	5	4	3	2	1	Bit 0
RSVD		S1_DIP_SW1	S1_DIP_SW0	RSVD		BOARD_ID1	BOARD_ID0

Bit	Field	Description	Value	Access
0	BOARD_ID0	Heat Sink Identifier – Indicates the type of heat sink present as either active or passive.	0 = Active 1 = Passive	R/O
1	BOARD_ID1	RLDRAM – Indicates whether RLDRAM is configured on the board.	0 = No RLDRAM 1 = RLDRAM	R/O
3–2	RSVD	Reserved		
5–4	S1_DIP_SW[1:0]	DIP Switch Indicator – Indicates the position of DIP Switch 3 and 4.	0 = On (Closed) 1 = Off (Open)	R/O
7–6	RSVD	Reserved		

LED Control Register

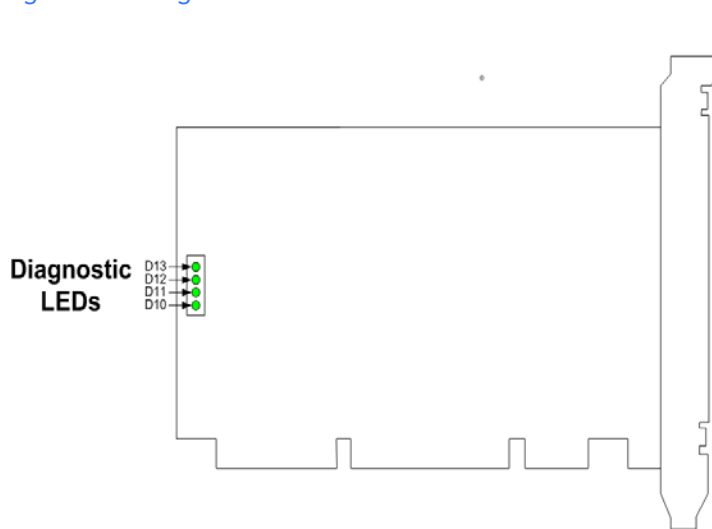
The **LED Control Register** is an 8-bit read/write register that controls the four green Diagnostic LEDs on the side of the board as shown in Figure 3-4.

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 LED - Illuminates the specified diagnostic LED when set to 1.	0 = Off 1 = On	R/W
7-4	RSVD	Reserved		

Figure 3-4 Diagnostic LEDs



Interrupt Control Register

The **Interrupt Control Register** is an 8-bit read/write register that generates an interrupt from a specific device upon completion of a transaction on an I2C port or temperature alarm. Generation of interrupts is enabled on a port-by-port basis using specific bits in this register. A hardware reset sets all bits to 0.

Bits within this register function as masks for the corresponding bits in the **Interrupt Status Register**. When a bit in the **Interrupt Status Register** is set and the corresponding bit in the **Interrupt Control Register** is also set, an interrupt generates on the corresponding I/O register space.

The FPGA also generates an interrupt to the host when the temperature threshold is exceeded or a fan fault occurs. To the host, this interrupt appears to be from the Multi-Core Processor but it is really from the FPGA.

See Figure 2-6 for the location of SFP ports.

Figure 3-5 Interrupt Control Register @ Base + 03h

Bit 7	6	5	4	3	2	1	Bit 0
FAN_FAULT	TEMP_FS_IEN	EEPROM_IEN	TEMP_I2C_IEN	SFP3_IEN	SFP2_IEN	SFP1_IEN	SFP0_IEN

Bit	Field	Description	Value	Access
0	SFP0_IEN	SFP0 Interrupt Enable – When enabled, generates an interrupt to indicate that there is 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.	0 = Disable 1 = Enable	R/W
2	SFP2_IEN	SFP2 Interrupt Enable – When enabled, generates an interrupt to indicate a completed I2C transaction on Port 2.	0 = Disable 1 = Enable	R/W
3	SFP3_IEN	SFP3 Interrupt Enable – When enabled, generates an interrupt to indicate a completed I2C transaction on Port 3.	0 = Disable 1 = Enable	R/W
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 64KB serial EEPROM.	0 = Disable 1 = Enable	R/W
6	TEMP_FS_IEN	Temperature/Fan Speed Interrupt Enable – When enabled, generates an interrupt to indicate a temperature/fan-speed condition that may exceed the threshold.	0 = Disable 1 = Enable	R/W
7	FAN_FAULT	Fan Fault – When enabled, generates an interrupt to indicate a fan failure.	0 = Disable 1 = Enable	R/W



This register controls interrupts generated on Multi-Core Processor GPIO Port 14. Multi-Core Processor registers provide and control an interrupt to the PCI-X host. See the Cavium documentation for more information.

Interrupt Status Register

The **Interrupt Status Register** is an 8-bit read-only register that indicates the status of the interrupts for selected devices. An independent **Interrupt Status Register** is available for each I/O bus space.

Figure 3-6 Interrupt Status Register @ Base + 04h

Bit 7	6	5	4	3	2	1	Bit 0
FAN_FLT_INT	TEMP_INT	EEPROM_INT	TEMP_I2C_INT	SFP3_INT	SFP2_INT	SFP1_INT	SFP0_INT

Bit	Field	Description	Value	Access
0	SFP0_INT	SFP0 Interrupt – When set, indicates a completed I2C transaction on Port 0.	0 = Normal 1 = Interrupt	R/O
1	SFP1_INT	SFP1 Interrupt – When set, indicates a completed I2C transaction on Port 1.	0 = Normal 1 = Interrupt	R/O
2	SFP2_INT	SFP2 Interrupt – When set, indicates a completed I2C transaction on Port 2.	0 = Normal 1 = Interrupt	R/O
3	SFP3_INT	SFP3 Interrupt – When set, indicates a completed I2C transaction on Port 3.	0 = Normal 1 = Interrupt	R/O
4	TEMP_I2C_INT	Temperature Sensor Interrupt – When set, indicates a completed I2C transaction involving the Temperature Sensor.	0 = Normal 1 = Interrupt	R/O
5	EEPROM_INT	EEPROM Interrupt – When set, indicates a completed I2C transaction involving the serial EEPROM.	0 = Normal 1 = Interrupt	R/O
6	TEMP_INT	Temperature/Speed Interrupt – When set indicates an interrupt involving the temperature/fan-speed.	0 = Normal 1 = Interrupt	R/O
7	FAN_FLT_INT	Fan Fault Interrupt – When set, indicates an interrupt involving a fan failure.	0 = Normal 1 = Interrupt	R/O

Peripheral Reset Register

The **Peripheral Reset Register** is an 8-bit read/write register that allows the Multi-Core Processor to force a reset of various on-board devices including the Flash, GbE PHY, and DIMM memory. All peripheral resets are asserted at power-up.

Figure 3-7 Peripheral Reset Register @ Base + 05h

Bit 7	6	5	4	3	2	1	Bit 0
RSVD					DIMM_ RSTN	PHY_ RSTN	FLASH_ RSTN

Bit	Field	Description	Value	Access
0	FLASH_ RSTN	Flash Reset – When set to 0, issues a hardware reset to the Flash device.	0 = Active reset 1 = Normal	R/W
1	PHY_RSTN	PHY Reset – When set to 0, issues a hardware reset to the GbE PHY device.	0 = Active reset 1 = Normal	R/W
2	DIMM_ RSTN	DIMM Modules Reset – When set to 0, issues a reset to all the DIMM modules.	0 = Active reset 1 = Normal	R/W
7-3	RSVD	Reserved		

Transmit Control Register

The **Transmit Control Register** is an 8-bit read/write register that controls transmitter output for each front panel SFPs. Output to each SFP can be configured independently. Setting bits [3:0] in this register disables the respective SFP. To apply the transmitted data as output, set the bit to 0. A hardware reset sets all bits to zero (enable) forcing the transmitter outputs to the enable state on power-up.

Figure 3-8 Transmit Control Register @ Base + 06h

Bit 7	6	5	4	3	2	1	Bit 0
RSVD				CH3 _DIS	CH2 _DIS	CH1 _DIS	CH0 _DIS

Bit	Field	Description	Value	Access
0	CH0 _DIS	Channel 0 Disable - When set, disables transmitter output on SFP0.	0 = Enable 1 = Disable	R/W
1	CH1 _DIS	Channel 1 Disable - When set, disables transmitter output on SFP1.	0 = Enable 1 = Disable	R/W
2	CH2 _DIS	Channel 2 Disable - When set, disables transmitter output on SFP2.	0 = Enable 1 = Disable	R/W
3	CH3 _DIS	Channel 3 Disable - When set, disables transmitter output on SFP3.	0 = Enable 1 = Disable	R/W
7-4	RSVD	Reserved		

Fan and Temperature Status Register

The **Fan and Temperature Status Register** is an 8-bit read/write register that provides the status for the temperature monitor and Pulse Width Modulation (PWM) fan controller. This register provides the status for the die temperature as well as the temperature of the diode (PN) junction of the Multi-Core Processor.

Figure 3-9 Fan and Temperature Register @ Base + 07h

Bit 7	6	5	4	3	2	1	Bit 0
FAN_MASTER	RSVD			SDR	SDL	FAN_FAULT	THERM

Bit	Field	Description	Value	Access
0	THERM	Thermal Overload – When set, indicates that the over temperature set point was exceeded.	0 = Normal 1 = Over temp.	R/W
1	FAN_FAULT	Fan Fault – When set, indicates that the fan has experienced a fault.	0 = Normal 1 = Fault	R/W
2	SDL	Shut Down Local – When set, indicates that the temperature is above the shutdown setpoint.	0 = Normal 1 = High temp.	R/W
3	SDR	Shut Down Remote – When set, indicates that the temperature of the Multi-Core Processor has exceeded the shutdown set point.	0 = Normal 1 = High temp.	R/W
6–4	RSVD	Reserved		
7	FAN_MASTER	Fan Master Control – Identifies the device that has master control of the fan. The device can be either the FPGA or the fan controller.	0 = FPGA 1 = Fan controller	R/W

3.2 I2C Registers

The WANic-58xx contains six I2C compatible devices, which include:

- Four front panel SFP I/O modules
- EEPROM
- Temperature/Fan sensor

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

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

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

The FPGA permits the Multi-Core Processors to simultaneously initiate transfers to the same device and automatically resolves simultaneous transfer requests. The FPGA carries out individual transfers sequentially. When transfers are received simultaneously, an internal arbitrator determine which transfer to send first.

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.

3.2.1 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. 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 Byte (MSB) set to 1.
4. 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.
5. To determine if the transaction is complete, the host can do one of the following:
 - Poll the content of the **I2C Control Register** or the **Interrupt Status Register**.
 - Wait for an interrupt.
6. When the transaction completes, the **I2C Control Register** resets the *I2C_RUN* field (Bit 6), and the *I2C_ERR* field (Bit 7) updates.

3.2.2 Reading Data

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

1. Specify the address of the register that contains the data.
2. Initiate the I2C transaction by issuing the **I2C Control Register**.
3. Determine if the transaction is complete. The host can do one of the following:
 - Poll the content of the **I2C Control Register** (*I2C_RUN* field = 0) or the **Interrupt Status Register** (all bits set to 1).
 - Wait for an interrupt.
4. When the transaction completes, the **I2C Data Register** contains the data that was read from the specified I2C device. Now, the host can 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 are as follows.

I2C Control Register

The **I2C Control Register** is an 8-bit read/write register that initiates a read or write transaction to the specified device (Temperature/Fan Monitor, EEPROM, SFP[3-0]). This register contains I2C transaction status, activity, and error information.

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

Device	Address
Temperature/Fan Monitor	08h
EEPROM	0Ch
SFP0	1Ch
SFP1	18h
SFP2	14h
SFP3	10h

Figure 3-10 I2C Control Register @ Base + Address

Bit 7	6	5	4	3	2	1	Bit 0
I2C_ RUN	I2C_ _ERR	RSVD					I2C WRITE

Bit	Field	Description	Value	Access
0	I2C_ WRITE	I2C Transaction Operation – Initiates either an I2C read or write operation for the specified device. For both read and write operations, the <i>I2C_RUN</i> field (Bit 7) in this register must be set to <i>Run</i> (1).	0 = Read 1 = Write	R/W
5-1	RSVD	Reserved		
6	I2C_ ERR	I2C Error – When set to 1, 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 enabled, initiates an I2C read or write transaction for the specified device.	0 = Disable 1 = Enable (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**.



Values written in both registers must be right justified.

Typically, I2C addresses include an ID bit field that must match the type of I2C device accessed. Since each I2C interface is associated with a single I2C device, the WANic-58xx automatically appends the appropriate ID bits to each I2C transaction.

The **I2C Address Register** hexadecimal (h) addresses for specific devices is as follows:

Device	High Address	Low Address
Temperature and Fan Monitor	None	0Ah
EEPROM	0Dh	0Eh
SFP3	11h	12h
SFP2	15h	16h
SFP1	19h	1Ah
SFP0	1Dh	1Eh

Figure 3-11 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 3-12 I2C Address High Register @ Base + High Address

Bit 15	14	13	12	11	10	9	Bit 8
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** is an 8-bit read/write register that specifies data associated with each I2C transaction.

- 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:

Device	Address
LM83	0Bh
EEPROM	0Fh
SFP3	13h
SFP2	17h
SFP1	1Bh
SFP0	1Fh

Figure 3-13 I2C Data Register @ Base + 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.

4 • Software Features

The Multi-Core Processor on the WANic-58xx provides “embedded Linux-based” boot firmware and application software. The software manages the hardware, configuration, and monitoring of the data processing and also provides services to the host. The software executes on the WANic-58xx, which is accessible through an application programming interface (API). The API provides easy software integration for creating customized applications to configure and to monitor the WANic-58xx. The software makes it easy to configure a variety of signaling, gateway, real-time control, traffic processing, and network interface applications.

4.1 Overview

After the hardware installation is complete and correct, the system is ready to run the software to configure and to manage the hardware. The software ships from the factory with default settings. The WANic-58xx software executes on one or more cnMIPs64 cores, which are specified by the user.

The Software Development Kit (SDK) for the WANic-58xx consists of the following components, most of which are shown in a high-level overview in Figure 4-1:

- User Application (UA) — controls and manages processing for the WANic-58xx
- 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-58xx hardware in a Linux environments. The LSP can be linked with the User Application running on the WANic-58xx and/or PCI-X host.
- 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-58xx hardware.

For your convenience, the WANic-58xx CD-ROM contains the following additional software, which is to be used in accordance with the GNU General Public License as provided in “[Appendix B • GNU General Public License](#).”

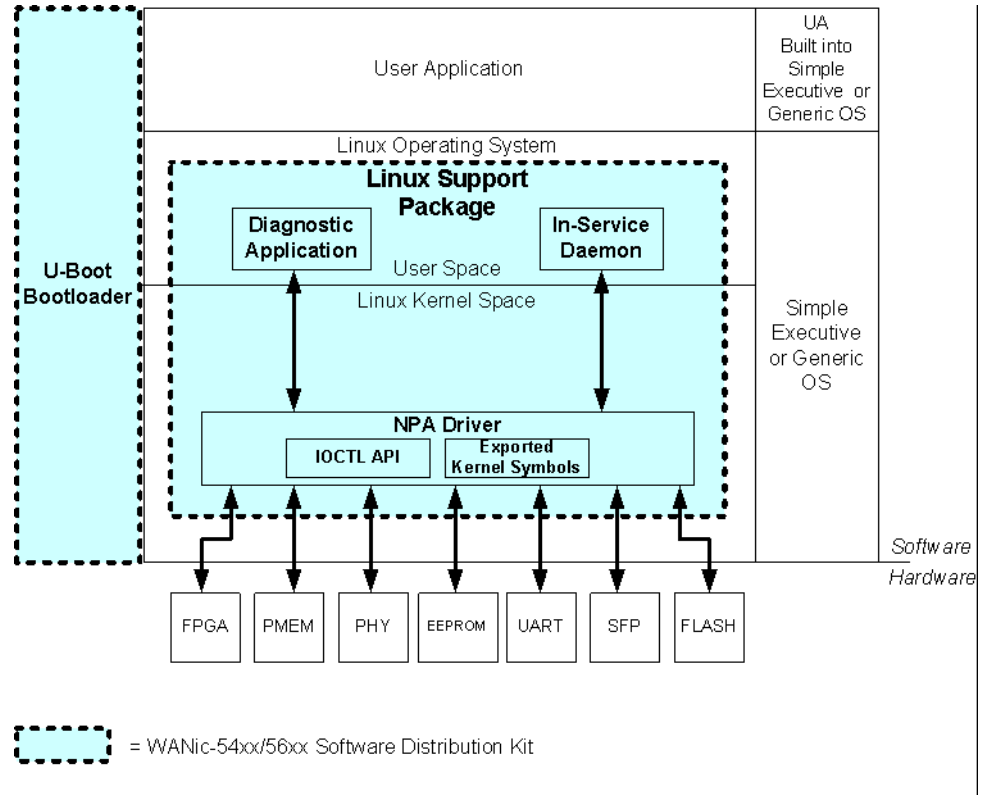
- GNU Tool chain – provides tools for building executables for the Multi-Core Processor.
- GNU Debugger – is a standard debugger for the GNU software that helps diagnose a program that is executing.

WANic-58xx uses one of the following operating systems, which the user must obtain/purchase separately, to create the UA:

- Linux Operating System, Version 2.6.x¹
- OCTEON Simple Executive
- Generic Operating System

1. Where x is an update or patch to Version 2.6.

Figure 4-1 Software Block Diagram



4.2 Cavium Software Development Kit

The WANic-58xx CD-ROM distribution includes the Cavium Software Development Kit (SDK) for rapid development of the OCTEON Multi-Core Processor. The SDK includes drivers, libraries, files, and other tools to facilitate. The SDK is covered by the GPL, v2 license. (See [Appendix B • GNU General Public License](#).)

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 information on your CD-ROM.

After the Cavium Ethernet Driver is loaded, four new Ethernet interfaces are available: eth0, eth1, eth2, and eth3. These four ports correspond to the front panel ports in the following order:

- eth0 -> port 0
- eth1 -> port 1
- eth2 -> port 2
- eth3 -> port 3

All four Ethernet ports function as standard Linux Ethernet interfaces, and can be configured 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 information on your software CD-ROM.

4.2.2 Embedded File System

The Cavium SDK provides the Embedded File System, which is contained in the following directory:

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

The Embedded File System is a RAM-based file system, which uses BusyBox 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, use the Configuration Menu to configure the file system.

4.2.3 Simple Exec Library

The Cavium SDK provides the Simple Exec Library, which is contained in the following directory:

```
$OCTEON_ROOT/executive
```

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

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

Simple Exec 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 the installation information on your software CD-ROM.

4.3 User Application

The User Application (UA) is the component that controls and manages processing. The WANic-58xx U-Boot loads and boots the UA. The UA operation is distributed across the multiple cores of the Multi-Core Processor. The UA can be comprised of the following:

- 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-58xx hardware.

The UA can also interface to the LSP when it runs foreground or background diagnostics for WANic-58xx hardware testing. The UA can assert the LSP API for configuring, monitoring, programming, and testing operations of the WANic-58xx hardware.



NOTE

Always strive to have the UA prevent core spin-locks and idle time during high loading periods among the cores.

4.4 U-Boot Bootloader

The WANic-58xx U-Boot Bootloader is based on the U-Boot open source multi-platform bootloader, which is also used for a wide range of embedded processor architectures.

The WANic-58xx U-Boot bootloader (hereafter referred to as U-Boot) supports interactive commands, environment variables, command scripting, and Flash-located UA read/write. U-Boot is covered by the GPL v2 License. (See [Appendix B • GNU General Public License](#).)

U-Boot executes on Core 0 of the Multi-Core Processor. U-Boot is loaded from Flash and, after reset, U-Boot transfers the execution image to DDR2 SDRAM. Then, U-Boot transfers control to the SDRAM image. U-Boot supports non-volatile configuration retrieval.

U-Boot software performs basic initialization of the WANic-58xx. 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 input.

Once the images are validated and loaded to memory, U-Boot transfers control to the UA software and also passes environmental variables that are relevant to the application images. (Environmental variables are described further in “[Section 4.4.4 IP Configuration Acquisition with DHCP](#).”) U-Boot provides special commands for displaying information and performing various operations. U-Boot commands are described in Table 4-1.

Tuples U-Boot uses *tuples* to define and to store the configuration of the WANic-58xx in NVRAM. (A tuple is a set of values or data set for a specified object or data structure.) U-Boot extends the tuple configuration definitions to U-Boot commands, and includes new tuples for the WANic-58xx as described in “[Section 4.5.2 EEPROM Tuples](#).”

4.4.1 UART Command Interface

U-Boot accepts commands from the UART or SIO on the WANic-58xx and can be configured using this interface. The UART accepts standard U-Boot commands and makes it easy to configure the entire module through the UART Command Interface.

4.4.2 Building U-Boot

U-Boot can be configured for either Flash-based or PCI host RAM-based execution.

Flash-Based Booting

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



NOTE

Make sure the S1 DIP Switch has SW2 set to the **Off** position to select the Flash boot operation.

1. Configure U-Boot to build for Flash-based execution, enter the following command at the prompt:

```
# make octeon_<board_type>_config
```

where *<board_type>* is the following:
 - w5800For example, # make octeon_w5800_config
2. Build the Flash-based U-Boot image using the make command.

```
# make
```
3. Copy the Flash-based boot image into the on-board Flash.
Burn `u-boot-octeon_<board_type>.bin` to the Flash at offset `0x00000000`
where *<board_type>* is the following:
 - w5800See *“Accessing the Flash”* for direction on burning in the Flash.

To build U-Boot for PCI Host RAM boot, perform the following steps:

1. Configure U-Boot to build for RAM-based execution.

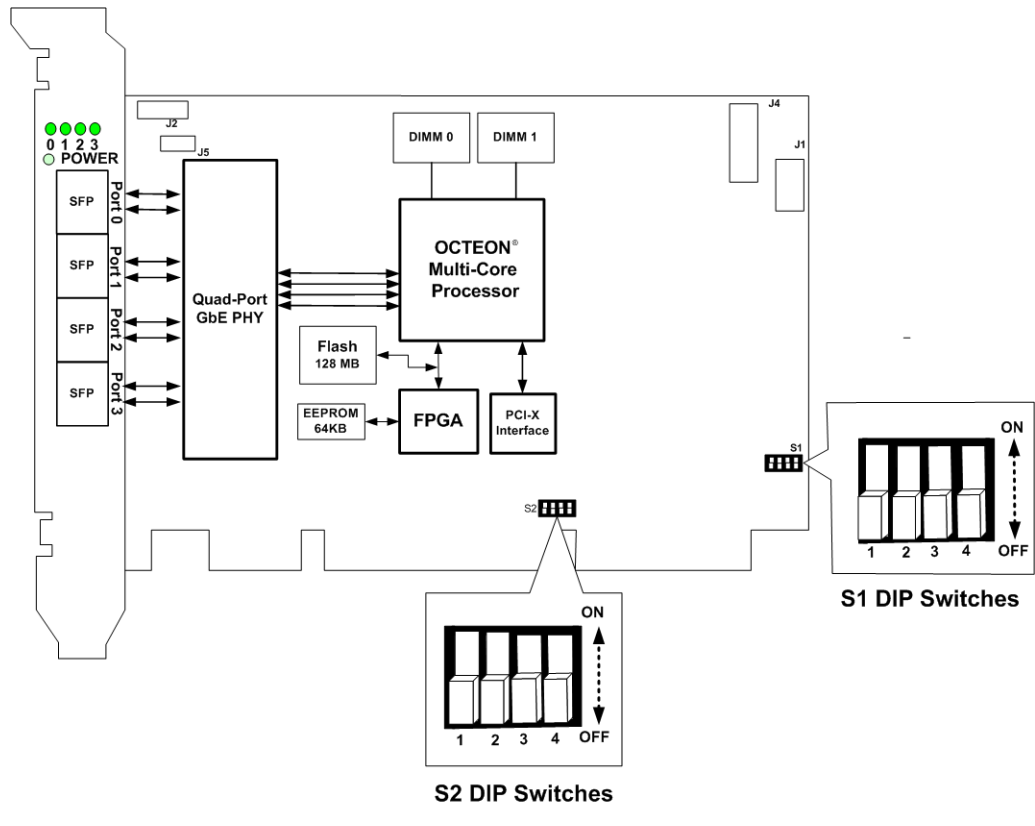
```
# make octeon_<board_type>_ram_debug_config
```

 where *<board_type>* is:
 - w5800
 For example,

```
# make octeon_w5800_ram_debug_config
```
2. Build the RAM-based U-Boot image.

```
# make
```
3. Set the S1 DIP Switch to PCI-X boot and power up the WANic-58xx.
 (See [Section 2.8 “Dual In-Line Package \(DIP\) Switches”](#) for more information on setting the S1 DIP Switch.)
4. Load the image using Diagnostic Utility – PCI Host Tools Menu to boot over the PCI-X. (See [“Chapter 5 • WANic-58xx Diagnostic Utility”](#) for more information on the Diagnostic Utility.)

Figure 4-2 S2 DIP Switch Bank



4.4.3 Boot Progress Status Messages

U-Boot indicates boot progress to both the terminal interface. Boot progress messages display, similar to those shown in Table 4-1, depending on the configuration.

Table 4-1 Boot Progress Status Messages

BPM_SYSTEM_INIT (BPM_OCTEON_SPECIFIC	0x0100)	- System Initialization (startup)
BPM_CS_INIT (BPM_OCTEON_SPECIFIC	0x0101)	- Chip Select Initialization
BPM_TIMER_INIT (BPM_OCTEON_SPECIFIC	0x0102)	- Timer Initialization
BPM_ENV_INIT (BPM_OCTEON_SPECIFIC	0x0103)	- Environment Initialization
BPM_EARLY_BOARD_INIT (BPM_OCTEON_SPECIFIC	0x0104)	- Early Board Initialization
BPM_INIT_BAUDRATE (BPM_OCTEON_SPECIFIC	0x0105)	- Initializing Baudrate (serial ports)
BPM_SERIAL_INIT (BPM_OCTEON_SPECIFIC	0x0106)	- Serial Initialization (serial ports)
BPM_CONSOLE_INIT_F (BPM_OCTEON_SPECIFIC	0x0107)	- Console Initialization
BPM_DISPLAY_BANNER (BPM_OCTEON_SPECIFIC	0x0108)	- Display Banner
BPM_DRAM_CALIBRATE (BPM_OCTEON_SPECIFIC	0x0109)	- DRAM Calibration
BPM_MEM_MAP_CALCULATE (BPM_OCTEON_SPECIFIC	0x010A)	- Memory Map Calculation
BPM_MEMPOST_SHORT (BPM_OCTEON_SPECIFIC	0x020B)	- Memory POST/Short
BPM_MEMPOST_LONG (BPM_OCTEON_SPECIFIC	0x000C)	- Memory POST/Long
BPM_PMEMPOST_SHORT (BPM_OCTEON_SPECIFIC	0x020D)	- Persistent Memory POST/Short
BPM_PMEMPOST_LONG (BPM_OCTEON_SPECIFIC	0x000E)	- Persistent Memory POST/Long
BPM_PHYPOST_SHORT (BPM_OCTEON_SPECIFIC	0x010F)	- PHY POST/Short
BPM_PHYPOST_LONG (BPM_OCTEON_SPECIFIC	0x0110)	- PHY POST/Long
BPM_CSUMPOST_SHORT (BPM_OCTEON_SPECIFIC	0x0211)	- Checksum POST/Short
BPM_CSUMPOST_LONG (BPM_OCTEON_SPECIFIC	0x0412)	- Checksum POST/Long
BPM_CODE_RELOCATE (BPM_OCTEON_SPECIFIC	0x0113)	- Code Relocate (from FLASH to RAM)
BPM_RAM_BASED_EXECUTE (BPM_OCTEON_SPECIFIC	0x0114)	- RAM Based Execution (code)
BPM_CMD_TABLE_RELOC (BPM_OCTEON_SPECIFIC	0x0115)	- Command Table Relocate (from FLASH to RAM)
BPM_FLASH_INIT (BPM_OCTEON_SPECIFIC	0x0116)	- FLASH Initialization (driver used to maintain FLASH)
BPM_MALLOC_INIT (BPM_OCTEON_SPECIFIC	0x0117)	- MALLOC Initialization (memory resource)
BPM_MALLOC_BIN_RELOC (BPM_OCTEON_SPECIFIC	0x0118)	- MALLOC Binary Relocate (memory resource image transfer)
BPM_ENV_RELOC (BPM_OCTEON_SPECIFIC	0x0119)	- Environment Relocate
BPM_BACKPRESSURE_WA (BPM_OCTEON_SPECIFIC	0x011A)	- Backpressure Workaround (Octeon IPD)
BPM_L2_CACHE_F_UL (BPM_OCTEON_SPECIFIC	0x011B)	- L2 Cache Flush and Unlock
BPM_DRAM_CLEAR (BPM_OCTEON_SPECIFIC	0x011C)	- DRAM Clear
BPM_PCI_INIT (BPM_OCTEON_SPECIFIC	0x011D)	- PCI Initialization
BPM_ETHERNET_INIT (BPM_OCTEON_SPECIFIC	0x011E)	- Ethernet Initialization
BPM_PRESTERA_INIT (BPM_OCTEON_SPECIFIC	0x011F)	- Prestera Initialization
BPM_REDHAWK_INIT (BPM_OCTEON_SPECIFIC	0x0120)	- Redhawk Initialization
BPM_EAGLE_INIT (BPM_OCTEON_SPECIFIC	0x0121)	- Eagle Initialization
BPM_PHYLINK_INIT (BPM_OCTEON_SPECIFIC	0x0122)	- PHY Link Initialization
BPM_SPI0_INIT_DELAY (BPM_OCTEON_SPECIFIC	0x0123)	- SPI 0 Initialization Delay
BPM_SPI0_INIT (BPM_OCTEON_SPECIFIC	0x0124)	- SPI 0 Initialization
BPM_SPI0_TSClk_WAIT (BPM_OCTEON_SPECIFIC	0x0225)	- SPI 0 TS Clock Wait

BPM_SPI0_RSCLK_WAIT (BPM_OCTEON_SPECIFIC	0x0226)	- SPI 0 RS Clock Wait
BPM_SPI0_TRAINING_WAIT (BPM_OCTEON_SPECIFIC	0x0227)	- SPI 0 Training Wait
BPM_SPI0_CAL_SEND (BPM_OCTEON_SPECIFIC	0x0228)	- SPI 0 Calibration Send
BPM_SPI0_STAT_SYNC (BPM_OCTEON_SPECIFIC	0x0229)	- SPI 0 Stat Synchronization
BPM_SPI1_INIT_DELAY (BPM_OCTEON_SPECIFIC	0x012A)	- SPI 1 Initialization Delay
BPM_SPI1_INIT (BPM_OCTEON_SPECIFIC	0x012B)	- SPI 1 Initialization
BPM_SPI1_TSCLK_WAIT (BPM_OCTEON_SPECIFIC	0x022C)	- SPI 1 TS Clock Wait
BPM_SPI1_RSCLK_WAIT (BPM_OCTEON_SPECIFIC	0x022D)	- SPI 1 RS Clock Wait
BPM_SPI1_TRAINING_WAIT (BPM_OCTEON_SPECIFIC	0x022E)	- SPI 1 Training Wait
BPM_SPI1_CAL_SEND (BPM_OCTEON_SPECIFIC	0x022F)	- SPI 1 Calibration Send
BPM_SPI1_STAT_SYNC (BPM_OCTEON_SPECIFIC	0x0230)	- SPI 1 Stat Synchronization
BPM_VERSION_NUM_STORE (BPM_OCTEON_SPECIFIC	0x0131)	- Version Number Store
BPM_MAC_ADDR_UPDATE (BPM_OCTEON_SPECIFIC	0x0132)	- MAC Address Update
BPM_MAINLOOP (BPM_OCTEON_SPECIFIC	0x0033)	- Main Loop

4.4.4 IP Configuration Acquisition with DHCP

U-Boot uses the networks 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 assets 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

4.4.5 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 4.5.3 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 “[Section 4.5.2 EEPROM Tuples](#)” for information to create SIPI, PFI, SFI, and LABI tuples.

4.4.6 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 control the Flash-based application boot process. Environment variables can be modified from default values listed in Table 4-2, where h represents a hexadecimal default value.

Table 4-2 Environment Variables

Variable	Default	Description
autoload	yes	Automatic DHCP and load boot script on boot-up.
baudrate	115200	Default U-Boot UART serial baud rate.
loadaddr	0x20000000	Default image load address
swfb_cmd	bootoctlinux	Starts the application.
swfb_flash_addr	0xb8000000	Flash address of the application.
swfb_ram_addr	7000000h	Starting RAM address for the application to be copied and executed. If this value is not specified, a copy operation is not performed and the image executes from the Flash address.
swfb_image_size	1800000h	Application image size.
swfb_cmd_arg	coremask = 0fff	Command argument list passed to the application.
mdio_alloc	Yes	Creates shared named alloc for mdio global lock required when using both npaDriver and Cavium-Ethernet driver together.
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.

4.4.7 Automated UA Executable Load and Boot

Optionally, U-Boot supports loading the automated UA executable image into DDR2 RAM. U-Boot uses TFTP to transfer an image into 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 SDRAM, U-Boot boots the specified cores.



NOTE

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

Table 4-3 Load and Boot Image Default 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	3
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	N/A	N/A
DDR2 RAM Address Boot Location	N/A	N/A
Core Asset Mask	0xFFFE – 15 cores [number limited to (max_cores -1) of part]	0x0001 – Core 0

4.4.8 U-Boot Error Handling

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 1 bank is On or if the 'postall' environment variable is set. See "[Section 2.8.1 S1 DIP Switch](#)" for more information on the S1 DIP Switch.

The WANic-58xx provides the following short and long POST tests:

- Flash Checksum
- RAM
- PHY
- RLDRAM
- DIMM0
- DIMM1

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 continuations 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 4-4.

Table 4-4 POST Failures

Value	Description
FF	No error
00	Flash Checksum Failure
01	RAM Failure
02	PHY Failure
03	SerDes Failure
04	RLDRAM Failure
05	DIMM0 Failure
06	DIMM1 Failure
80	Diagnostic Failure

4.4.9 U-Boot Commands

Table 4-1 lists U-Boot commands in alphabetical order. For more information on these commands, see the “DENX U-Boot and Linux Guide (DULG)” (www.DENX.de).

Table 4-1 U-Boot Commands


Command	Description
?	Displays online help.
askenv	Gets the environmental 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 BOOT/P/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.
echo	Displays the same argument typed on the console.
eeprom	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. the default is /
flinfo	Displays information for all available flash memory banks.
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
loadb	Loads a binary file over serial lines (kermit mode),

Table 4-1 (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	Allocate a named bootmem block
namedfree	Free a named bootmem block
namedprint	Prints a list of named Boolean blocks
netintstat	Obtains network interface status
nm	Memory modify (constant address)
pci	Lists and access PCI Configuration Space
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. It 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 64-bit words from 64-bit address
read64b	Reads 8-bit words from 64-bit address
readcmp	Reads and compares memory to value
readI2C	Reads data from an I2C device
reset	Reboots the system.
run	Runs commands in an environment variable.
saveenv	Saves the environment variables to persistent storage.
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).
tftpboot	Boots the image over the network using the TFTP protocol.
tlv_eeprom	EEPROM data parsing for the WANic card
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
writeI2C	Writes data to an I2C device

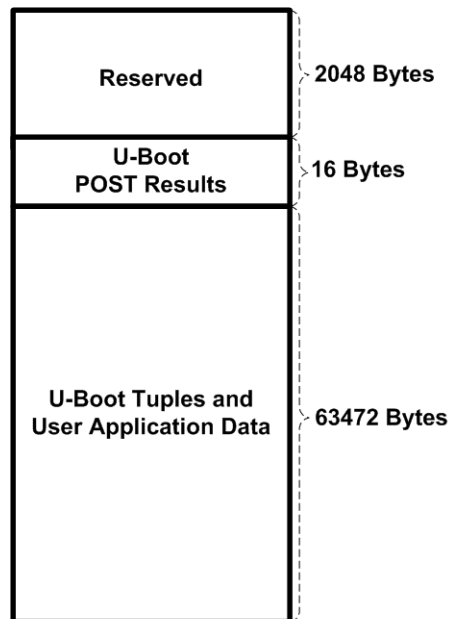
4.5 EEPROM

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 contents of the EEPROM dictates these operations. The EEPROM stores and supports the extended EEPROM tuples in Table 4-2.

4.5.1 EEPROM Mapping

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

Figure 4-3 EEPROM Mapping



4.5.2 EEPROM Tuples

Table 4-2 summarized the new EEPROM tuples for the WANic-58xx U-Boot. These tuples supplement the tuple definitions to U-Boot commands.

Table 4-2 EEPROM Tuples

Tuple	Maximum	Description
CSUMTESTI	Up to four	Validates the Checksum test
LABI	Up to 16	Loads and boot information definitions
MEMTEST	Up to four	Creates a Memory test
PHYTESTI	One	Creates a PHY test
RLDRAMT1	One	Creates an low latency memory RLDRAM test
SCRIPT	Up to four	Creates and updates a script
SCRIPTACTIVE	One	Indicates the active script
SERDESTEST1	One	Creates a SerDes test
SFI	Up to 16	Obtains Secondary File Information
SIPI	Up to 10	Defines the Source IP address
PFI	Up to 16	Obtains Primary File Information
UARTCONI	One	Changes the UART console data rate

Tuple Format

The WANic-58xx 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;
    uint16_t version;
    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 ser #: unknown

BOARD_DESC_TYPE (0x2) tuple found: at addr
0x810
type: 0x2, len: 0x24, csum: 0x1d4, maj_ver: 1, min_ver:
0
Board type: Wxxxx (0x10)
Board revision major:49, minor:0
Chip type (deprecated): NULL (0x0)
Chip revision (deprecated) major:0, minor:0 Board ser #: 7654321
=====
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): 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)
=====
```

Deleting a Tuple

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

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



NOTE

Examples in this section use the WANic-58xx command prompt. The command prompt will vary depending on the model of your WANic-58xx.

Descriptions of WANic-58xx EEPROM tuples follows.

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: 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 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 8-15. (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 `tlv_eeeprom set memtesti [id] [start] [end] [csum location] [csum algorithm][failure action]`

Parameters

<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 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 memtesti 0 0xbfc00000, 0xbfc2fff7, 0xbfc2fff8 3 0;
```

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 0–7.</p> <pre>tlv_eeprom set phyesti 0 0x00ff 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 Multi-Core 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 file name is `vmlinux.64`,
- The TFTP server is specified as `10.71.1.11`.
- The number of retries is 4.

RLDRAM Testing

Name	RLDRAMTI								
Prototype	<code>tlv_eeeprom set rldramti [address] [datasize] [replication type] [display option]</code>								
Parameters	<table><tr><td><code>address</code></td><td>Address</td></tr><tr><td><code>datasize</code></td><td>Size of the data</td></tr><tr><td><code>replication type</code></td><td>Word replication factor</td></tr><tr><td><code>display option</code></td><td>Displays to the screen option.</td></tr></table>	<code>address</code>	Address	<code>datasize</code>	Size of the data	<code>replication type</code>	Word replication factor	<code>display option</code>	Displays to the screen option.
<code>address</code>	Address								
<code>datasize</code>	Size of the data								
<code>replication type</code>	Word replication factor								
<code>display option</code>	Displays to the screen option.								
Description	Creates a test function for the RLDRAM.								
Example	<p>The following example tests the RLDRAM.</p> <pre>tlv_eeeprom set rldramti 0 1024 0</pre> <p>In the above example:</p> <ul style="list-style-type: none">• The data size is 1024• The replication type is 0.								

SerDes Testing

Name	SERDESTESTI				
Prototype	<code>tlv_eeeprom set serdestesti [<i>algorithm</i>] [<i>failure action</i>]</code>				
Parameters	<table><tr><td><i>algorithm</i></td><td>Algorithm</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	<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				
<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 SerDes by detecting the SerDes presence and returning the status of the test.

Example The following example configures a SerDes test.

```
tlv_eeeprom set serdestesti 0 0
```

In the above example:

- The algorithm is 0.
- The failure action is 0.

Script

Name	SCRIPT				
Prototype	<code>tlv_eeprom 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-58xx 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 creating an active script.				
Example	<p>The following example creates a script tuple with two commands:</p> <pre>tlv_eeprom set script 0 dhcp setenv serverip 10.71.1.11</pre> <p>Add subsequent commands to this script tuple as follows:</p> <pre>tlv_eeprom set script 0 tftpboot 21000000 vmlinux.64 10.71/1/11</pre>				

Script Active

Name	SCRIPTACTIVE
Prototype	<code>tlv_eeprom set scriptactive [id]</code>
Parameters	<i>id</i> Script ID
Description	Specifies the active script.
Example	The following example creates a <code>Scriptactive</code> tuple with Script 0 as the active script:

```
tlv_eeprom set scriptactive 0
```

The WANic-58xx executes the following commands from Script 0 during initialization:

```
dhcp
setenv serverip 10.71.1.11
tftpboot 21000000 vmlinux.64
```



NOTE

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

Source IP Address Information

Name SIPI

Prototype `tlv_eeprom set sipi [id] [ethernet portid] [DHCP enable] [DHCP num retries] [IP addr] [subnet mask] [default GW IP addr];`

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 retries</i>	Number of retries
<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-58xx 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_eeprom set sipi 0 octeth0 1 0---
```

The following example, configures a static IP address:

```
tlv_eeprom 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 [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 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>id</i>	Identifier	<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>id</i>	Identifier														
<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 0 bfc80000 80000 0--0</pre> <p>In the above example:</p> <ul style="list-style-type: none">• SFI 0 is configured.• The Flash-based application file address must be specified (<code>bfc80000</code>).• The Flash-based application file size is specified (<code>80000</code>) because the application is to be transferred to RAM.• SIPI 0 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 <code>0</code>.														

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>

4.5.3 EEPROM Tuple Update and Enumeration

The WANic-58xx 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_wnpa58xx
/* 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_wnpa58xx */
    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
};

```

4.5.4 Tuple Format

The WANic-58xx 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;
    uint16_t version;
    uint16_t checksum;
} octeon_eeprom_header_t;
```

The rest 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
```

4.6 Linux Support Package (LSP)

The WANic-58xx provides an LSP. Each of the components of the LSP is covered by the GPLv2 License (see [Appendix B • GNU General Public License](#).)

The LSP supports the following WANic-58xx devices:

- EEPROM
- Flash
- GbE PHYs (when available)
- SFPs or RJ-45s

The LSP can be configured for PCI Host or PCI Target mode:

- In PCI Host mode, the WANic-58xx runs its own Linux kernel.
- In PCI Target mode, the WANic-58xx is controlled by the host PC.

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
- 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-58xx and support select devices.

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

The Flash Driver allows information to be read, deleted, and retrieved from Flash.

The NPA Driver provides the API for communication with the on board WANic-58xx 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 Diagnostic Application provides in-service tests as well as U-Boot and Linux upgrade utilities.

4.6.1 Flash Driver

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`.



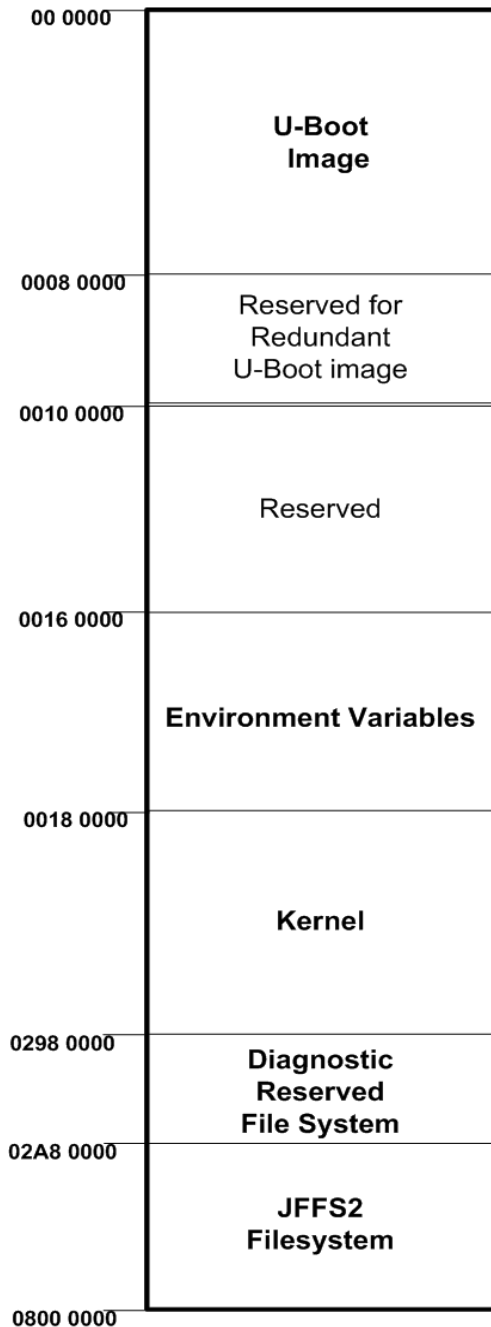
NOTE

All paths are relative to the Linux Kernel source.

Flash Mapping

The WANic-58xx 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 4-4 shows the default factory-installed Flash mapping.

Figure 4-4 Flash Mapping



Flash Driver Functions

The function in Table 4-3 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 4-3 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, _off_t offset, 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://sources.redhat.com.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.erasysize = 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 4-4, and described in Table 4-4.

Table 4-4 Flash Partitioning

Size	Access	Description
1408 KB	Read-Only	U-Boot boot fail-safe 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. For example,

```
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

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 • WANic-58xx Diagnostic Utility](#)” for more information.

Reprogramming the U-Boot Bootloader

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 • WANic-58xx Diagnostic Utility](#)” 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.

4.7 NPA Driver

The NPA Driver is a Linux kernel module that provides the API for communications with the WANic-58xx 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 peer into the kernel's view of the system.

4.7.1 Linux `/proc/` File System

The `/proc/` file system provides access to information about the state of the WANic-58xx hardware and software. Access the `/proc/driver/` and `/proc/net/` directories to obtain information about:

- Basic RGMII Ethernet 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
```

4.7.2 IOCTL Device interface

The WANic-58xx IOCTL device interface supports commands that allow the UA to interface to the hardware. The IOCTL device interface within the NPA Driver supports the following:

- EEPROM tuples
- PHY devices (if available)
- LEDs
- TWSI (I2C access)
- PCI-X
- 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-58xx. Figure 4-5 and Table 4-5 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
```

Figure 4-5 (continued) IOCTL Commands and Structures

```

NPA_LOOPBACK_INFO_t*/
_GET_LOOPBACK,          /**<Gets the loopback state via
NPA_LOOPBACK_INFO_t*/
_NPA_DEV_EX_CMD = 0xF0 /**<Extended device specific commands*/
};

```

4.7.3 IOCTL Commands

The IOCTL device interface performs all device-related operations. The LSP supports the IOCTL interface functions listed in Table 4-5.

Table 4-5 IOCTL Commands

Command	Argument	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.

Table 4-5. (Continued) IOCTL Commands

Command	Argument	Description
<code>_SET_LED_STATE</code>	Integer LED_TYPE_e flags indicate LED state to set	Sets the LED state.
<code>_GET_LED_STATE</code>	Filled with LED_TYPE_e 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.
<code>_PEEK</code>	NPA_REG_RW_OP_t DevID— Device IDs	In PCI Target configuration, peeks at an address.
<code>_POKE</code>	NPA_REG_RW_OP_t DevIDx—Device IDs	In PCI Target configuration, pokes at an address.
<code>_GET_BOARD_INFO</code>	NPA_BOARD_INFO_t structure Returns information about the LSP, FPGA, IPMC and MMC.	Retrieves version information for the board.
<code>_SET_ENVIRONMENT</code>	NPA_ENV_ENTRY_t 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>	NPA_ENV_ENTRY_t 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>	NPA_ENV_ENTRY_t 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>	NPA_PORT_INFO_t 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>	NPA_PORT_INFO_t 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>	NPA_PORT_INFO_t 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>	NPA_PORT_INFO_t 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>	NPA_PORT_INFO_t 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>	NPA_PORT_INFO_t 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>	NPA_PORT_INFO_t structure Resets the statistical counters for the device specified by <i>devId</i> .	Resets the statistic counters.
<code>_SET_LOOPBACK</code>	NPA_LOOPBACK_INFO_t 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 NPA_LOOPBACK_INFO_t.
<code>_GET_LOOPBACK</code>	NPA_LOOPBACK_INFO_t structure Returns the loopback mode of the port specified by port on the device specified by <i>devId</i> .	Get the loopback state via NPA_LOOPBACK_INFO_t.
<code>_NPA_DEV_EX_CMD=0xF0</code>	NPA_DEV_EX_CMD_t 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 */

```

Figure 4-6 (continued) IOCTL Command Types

```
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*/
eNpaOperStateeOperState; /**< Sets or gets the port operational state */
eNpaAdminStateeAdminState; /**< 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_SIZEsizeof(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_SIZEsizeof(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;
```

4.7.4 Proc File Updates

The NPA Driver creates and updates the `/proc/` file system for temperature sensor readings, basic RGMII Ethernet 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
```

4.7.5 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 chipsel
);
.
```

4.8 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`

4.8.1 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      : 1.28.0
Running UBoot Failsafe  : 0.0.0
Running UBoot Normal    : 3.3.0
Running FPGA Version    : 2.2
Running in PCI mode     : host
Running on NPU          : A
```

4.8.2 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 • WANic-58xx Diagnostic Utility

This chapter describes the Diagnostic Utility. It describes how to run tests to verify and to configure the functionality of components on the WANic-58xx.

5.1 Overview

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-58xx Diagnostic Utility is a Linux image that provides the following:

- Transmit/Receive Tests – verify the WANic-58xx Ethernet interfaces.
- Configuration Options – configure the EEPROM to specify boot parameters.
- PCI utilities – provides services for:
 - Booting U-Boot over the PCI-X interface
 - Loading applications over the PCI-X to specified target memory locations
 - Sending U-Boot commands to the target over PCI-X
- 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.



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

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.

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

- WANic-58xx modules 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-58xx, or serial connection to the UART.

Connect to the WANic-58xx, 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-58xx. 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-58xx SDA and the other end to the host terminal. Use a terminal setting of 115200n1.

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 PCI Host Mode. Figure 5-2 display the PCI Host Mode screen. Table 5-2 describes the 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 5-2 Diagnostic Utility Main Menu

```
Running Octeon Model      : CN58xx-SSP Pass 1.1
Running Driver Version   : 1.17.8
Running UBoot Failsafe   : 3.0.1
Running UBoot Normal     : 3.0.1
Running FPGA Version     : 1.2
Running in PCI mode     : host
Running on NPU          : A
setting cpuIndex 0 for eth0
setting cpuIndex 1 for eth1
setting cpuIndex 2 for eth2
setting cpuIndex 3 for eth3
setting cpuIndex 4 for xauio
npa_mem_test namedalloc is not set, skipping DRAM memory tests
*** GE Fanuc WANic 5600 Diagnostics v1.16.0 NPU: A ***
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 Menu 15. USB Storage Test
8. Diagnostic LED test
0. Exit
Menu Choice (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.
8. Diagnostic LED Test	Tests the Diagnostic LEDs.
10. Flash Menu	Displays options to view and update Flash partitioning.

Table 5-2 Diagnostic Utility Main Menu

Menu Item	Description
11. Log File Location	Display the path to the log files.
12. Reserved	
13. Mem Test	Conducts memory read/write for SDRAM. Only SDRAM test is allowed to run when <code>npa_mem_test</code> named allocated space 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-58xx most recent boot, similar to that shown in Figure 5-3.

Figure 5-3 View Test Results Display

```
Menu Choice (0) : 1
PORT status
      PORT PASSED
Diagnostic status
Retrieving data-loop results file...
eth0:  PASSED
eth1:  PASSED
eth2:  PASSED
eth3:  PASSED
Retrieving DDR2 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 LED test results ...
      LED test: Incomplete
Hit enter to continue...
```

EEPROM Menu

Select **2. EEPROM Menu** to display the EEPROM Menu options as shown in Figure 5-4. Use these options to modify the EEPROM contents on the WANic-58xx.

Figure 5-4 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-58xx 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-58xx, the connection may be lost when the Auto Test executes.

Figure 5-5 Run Auto Test

```
Starting data loop thread for eth2
Jan  1 01:18:19 (none) user.info npaDiag: Using peer eth3 for eth2
INFO: [eth2] sched_setaffinity TxRxthread on PID 956 cpu_set 4 ret: 0
Jan  1 01:18:19 (none) user.info npaDiag: [eth2] sched_setaffinity TxRxthread o
INFO: [eth2] sched_getaffinity TxRxthread cur_mask = 4 ret: 0
Jan  1 01:18:19 (none) user.info npaDiag: [eth2] sched_getaffinity TxRxthread c
INFO: Using peer eth2 for eth3
Starting data loop thread for eth3
Jan  1 01:18:19 (none) user.info npaDiag: Using peer eth2 for eth3
INFO: [eth3] sched_setaffinity TxRxthread on PID 956 cpu_set 8 ret: 0
Jan  1 01:18:19 (none) user.info npaDiag: [eth3] sched_setaffinity TxRxthread o
INFO: [eth3] sched_getaffinity TxRxthread cur_mask = 8 ret: 0
Jan  1 01:18:19 (none) user.info npaDiag: [eth3] sched_getaffinity TxRxthread c
Iter:0
  eth0: tx:00006952 rx:00006824
  eth1: tx:00006826 rx:00006952
  eth2: tx:00006668 rx:00006453
  eth3: tx:00006455 rx:00006634

Iter:1
  eth0: tx:00013397 rx:00013269
  eth1: tx:00013270 rx:00013396
  eth2: tx:00013112 rx:00012898
  eth3: tx:00012899 rx:00013077
```


Auto Test Configuration Menu

Select **6. Auto Test Configuration Menu** to display test configuration options as shown in Figure 5-7. Use these menu options to modify the configuration before selecting **5 Run Auto Test**.



CAUTION

A power cycle of the WANic-58xx causes the configuration to be lost.

To test the WANic-58xx, attach a Category 5 Enhanced (Cat 5E) cable to each of the ports as shown in Figure 5-6. Connect Port 0 to Port 1, and Port 2 to Port 3.

Figure 5-6 Test Configuration with Cable Attachment

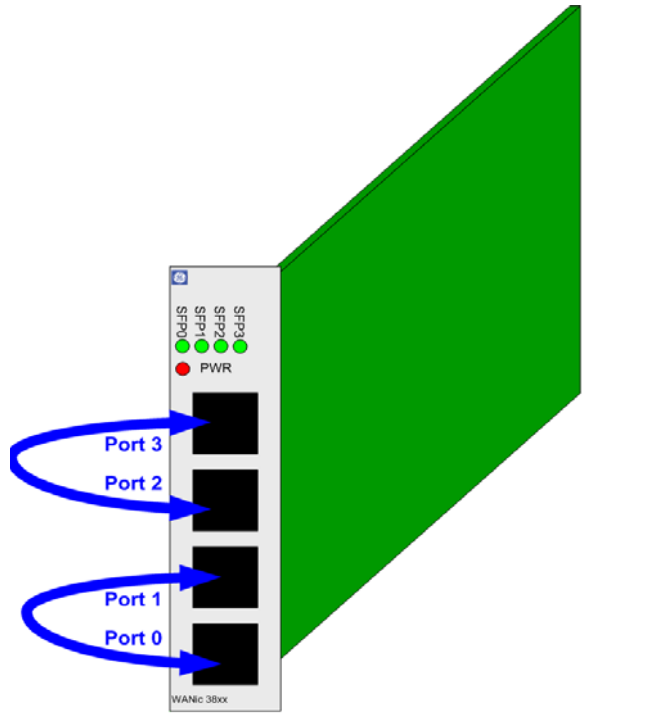


Figure 5-7 Auto Test Configuration Menu

```
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 inot 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

Diagnostic LED Test

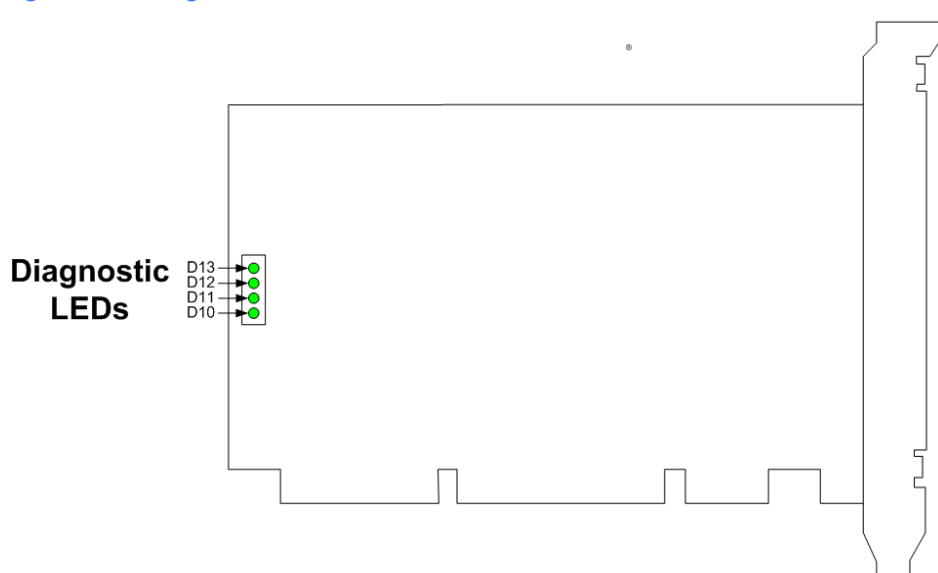
Select **8. Diagnostic LED Test** to verify that the test is toggling the LEDs on and off. This option prompts you with a message as shown in Figure 5-5. View the Diagnostic LEDs, located approximately as shown in Figure 5-8, to verify that the LEDs are turning on and off.

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 5-8 Diagnostic LED Location



Flash Menu

Select **10. Flash Menu** to display and access selected operations to a specified partition in Flash memory. Figure 5-9 shows an example of this menu. For more information on Flash partitioning, see *"Section "Partitioning the Flash"*.



NOTE

The Flash Menu options require decimal values only.

Figure 5-9 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 5-10.

Figure 5-10 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 5-11.

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 5-11 Show Core Temperature

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

Memory Test

Select **13. Mem Test** to run a DDR2 memory test, and to display results as shown in Figure 5-12. The Diagnostic Utility prompts you to select the DDR2.

When you select DDR2 (0), `npa_mem_test` named `alloc` must be created *before* booting Linux.

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 5-13.



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 causes the DDR2 test *not* to run.

Figure 5-12 Memory Test Results

```
Menu Choice <0> : 13
Select memory test <DDR2> Mem = 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 5-13 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
```


A • Hardware Installation and Removal Procedures

This chapter describes how to install or remove the WANic-58xx from a PCI chassis. It also includes directions for insertion and removal of DIMMs and SFPs.

A.1 Installation 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:



CAUTIONS

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 Installation Overview

Installing a WANic-58xx involves the following operations:

- Install DIMMs.
- Install the WANic-58xx.
- Install SFP modules.
- Connect to the power supply.

A.3 Mini-RDIMM Installation and Removal

The DDR SDRAM is implemented as two socketed Mini-RDIMMs.



NOTE

Install the Mini-RDIMMs onto the WANic-58xx *before* installing the board into a chassis.

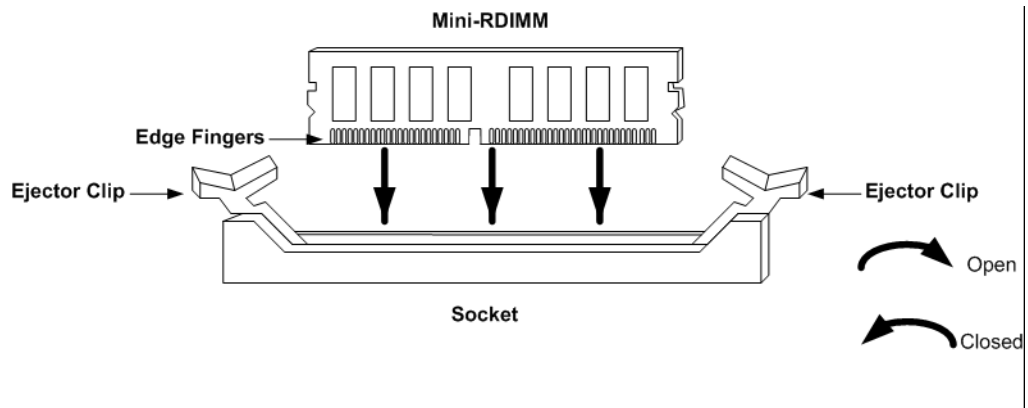
Observe all ESD safety precautions *before* starting this procedure. Failure to follow ESD safety precautions may result in damage to components.

A.3.1 Mini-RDIMM Installation

To install Mini-RDIMMs on the WANic-58xx, 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 WANic-58xx as shown in Figure A-1.

Figure A-1 DIMM 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 a 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-RDIMMs, perform the following steps:

1. Select the Mini-RDIMM to remove.
2. 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.
3. 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-58xx Installation and Removal

The WANic-58xx is inserted in a PCI chassis and requires two adjacent slots to safely accommodate components on the board. Always be sure to follow the safety precautions.

A.4.1 WANic-58xx Installation

To install the WANic-58xx into a PCI 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 module. (*See* previous “CAUTION.”)
5. Locate two adjacent available PCI slot in the chassis.



CAUTION

The WANic-58xx occupies two adjacent slots in the chassis to accommodate components such as the mini-DIMMs, heat sink, and L4 power connector. Make sure you have sufficient space in the chassis *before* inserting the module.

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.

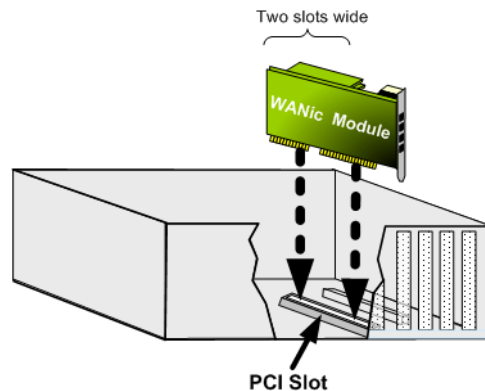


CAUTION

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 envelop off the card with the other hand. Save the package in case you need it for future use.

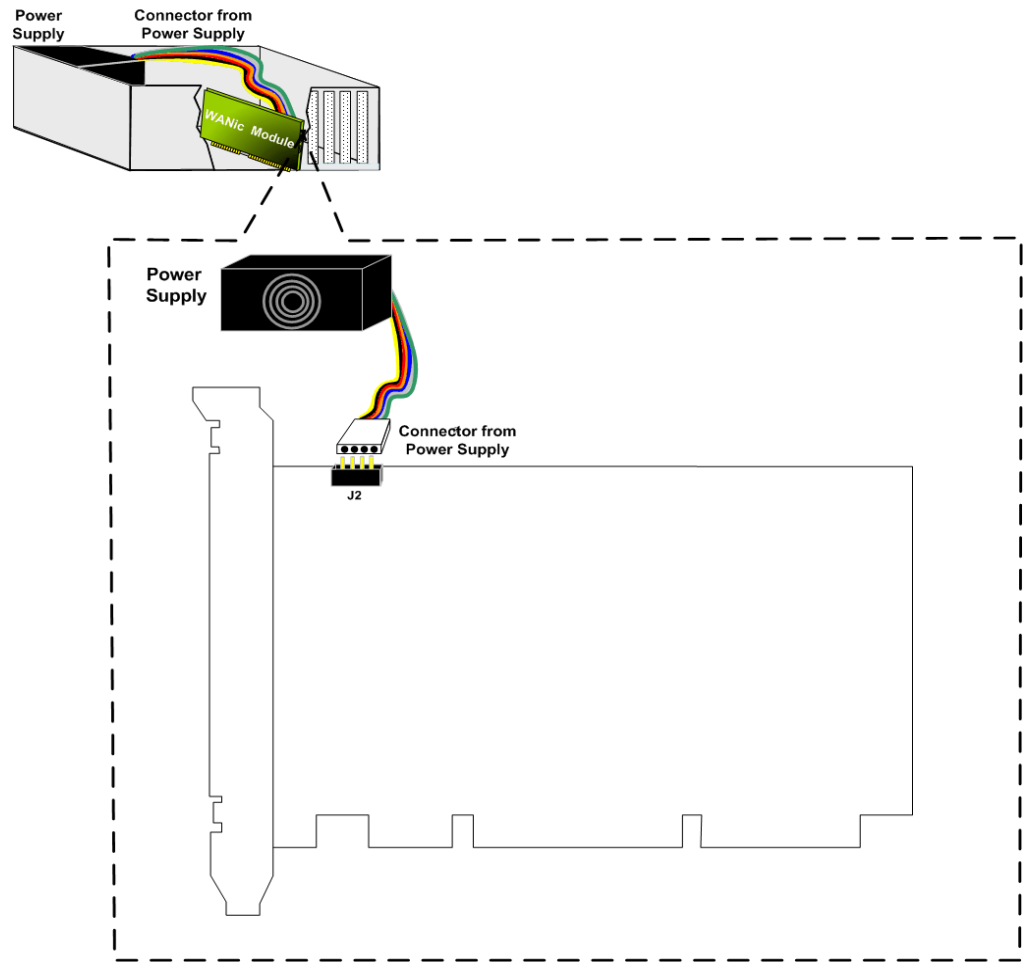
8. Align the card with the appropriate slot and insert the card into the slot. Slide the card into the slot gently, but firmly.

Figure A-2 Installation in a PCI Bus Chassis



9. Attach the screw that secures the card to the card cage frame.
10. Attach the connector from the power supply (for example, a Molex[®] Connector) to the L4 Power Connector on the WANic-58xx as shown in Figure A-3.

Figure A-3 Connecting to the Power Supply



11. Replace the computer chassis cover.
12. Replace the cables.
13. Power on the computer.
14. Run the diagnostic utility as described in *“Chapter 5: WANic-58xx Diagnostic Utility.”*

A.4.2 WANic-58xx Removal

To remove the WANic-58xx, perform the following steps:

1. Turn off the power to the computer.
2. Disconnect the cables from the computer chassis.
3. Remove the cover from the computer chassis.
4. 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.

5. Identify the slot from which you intend to remove the module. Remove the screw that secures the card to the frame.
6. Disconnect the connector from the power supply that is attached to the module.
7. Remove the Mini-RDIMMs from the module.
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 cools 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 the section "[Section A.3.1 Mini-RDIMM Installation](#)" in this chapter. If you are not installing a new module, continue with Step 11.
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 Module installation and Removal

For hot swap support, LSP Version 1.22 must be installed on your WANic-58xx.



CAUTIONS

SFPs must be installed before power-up.

SFP modules can be inserted and removed in the front panel of a WANic-58xx after the WANic-58xx is inserted in a chassis.

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



WARNINGS

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.



CAUTIONS

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

This section provides instructions for inserting an SFP module.



NOTE

For hot swap support, LSP Version 1.22 must be installed on your WANic-58xx.

To insert an SFP module into a socket connector on the WANic-58xx 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

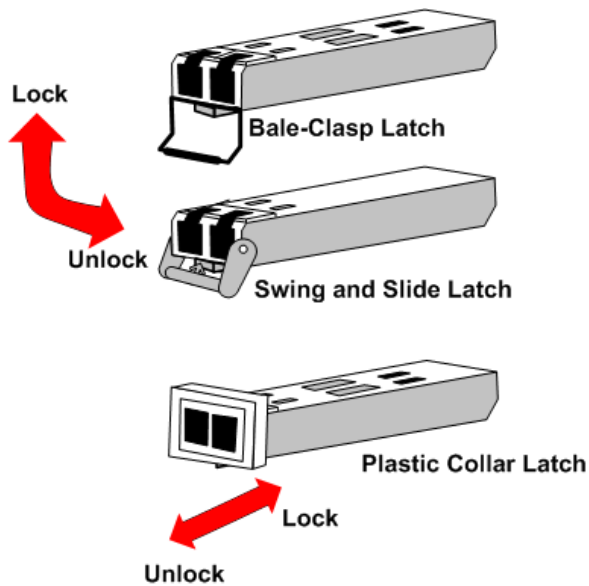
To remove an SFP module from a WANic-58xx module, perform the following steps:

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.

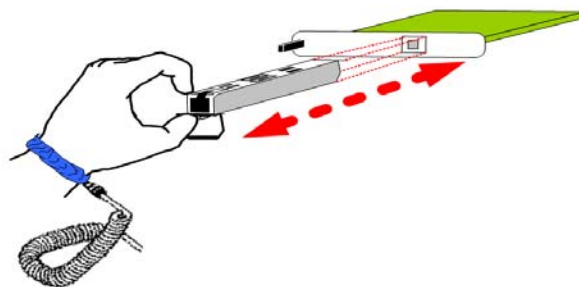
3. Unlock and remove the SFP module. Since SFP modules have various latch designs, follow the instructions provided with the SFP for removing the transceiver. For example, if the module contains a bale-latch, such as those shown in Figure A-4, open the latch on the SFP module by pressing it in the appropriate 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.

4.1 IMITATION 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.

5.1 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-58xx

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 and acronyms, and definitions.

- 1000BASE-T, 1000BASE-X, or 1000BASE-SX** A standard for Gigabit Ethernet connectivity, which provides transmission speeds up to 1000 megabits per second (or one million bits per second.)
1000BASE-T is used for Gigabit Ethernet over twisted-pair copper cable.
1000BASE-X is used for Gigabit Ethernet over fiber optical cable.
1000BASE-SX is used for short wavelength laser over multimode fiber, and has a maximum distance of 550m.
- 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.
- DHCP** Dynamic Host Configuration Protocol.
- DIMM** Dual In-Line Memory Module.
- DIMM Rank** Number of independent sets of DRAMS that can be accessed simultaneous for the full data-bit width of the DIMM to be driven on the bus.
- DIP** Dual In-Line Package.
- EEPROM** Electronically Erasable Programmable Read Only Memory is user-modifiable read-only memory that can be erased and reprogrammed (written to) repeatedly in its entirety, not selectively. A special form of EEPROM is Flash memory.
- ECC** Error Correction Code.
- Egress Traffic** Network traffic that begins inside of a network and proceeds to a destination somewhere outside of the network.
- ESD** Electrostatic Discharge.
- Ethernet** Ethernet is a local area network technology specified in a the IEEE 802.3 standard.
- Flash Memory** Sometimes called *Flash ROM*, Flash memory is a type of constantly-powered nonvolatile memory that can be erased and reprogrammed.
- Gigabits (Gbps)** 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 (GND) 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 (GMII) provides a simple interconnection between MAC and PHY that is independent of the physical media types.
- GPIO** General Purpose Input and Output.
- 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.
IP	Internet Protocol.
KB	Kilo Byte.
LED	Light Emitting Diode.
LFM	Linear feet per minute. A standard measure of airflow.
LOS	Loss of Signal.
MAC	Media Access Control.
MB	Mega Byte.
Mbps	Megabits per second.
MDI	Medium Dependent Interface.
MDIO	Management Data Input/Output.
MPSC	Multi-Protocol Serial Controller.
MTBF	Mean Time Between Failure.
Multi-Core Processor	A multi-core processor is an integrated circuit (IC) to which two or more processors have been attached and integrated for enhanced performance, reduced power consumption, and more efficient simultaneous processing of multiple tasks (see parallel processing).
NMI	Non-Maskable Interrupt.
NSP	Network Service Processor.
NVRAM	Non-Volatile Random Access Memory.
ODT	On-Die Terminations.
PCI	Peripheral Component Interconnect.
PCS	Physical Coding Sublayer.
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.
PMA	Physical Medium Attachment.
PICMG	PCI Industrial Computer Manufacturers Group.
RGMII	Reduced Gigabit Media Independent. RGMII is typically connected to an external physical layer (PHY) chip.
RoHS	Restriction of Hazardous Substances directive.
ROM	Read Only Memory.
SCP	Secure Communications Processor.
SDRAM	Static Dynamic Random Access Memory.
SGMII	Serial Gigabit Media Independent. SGMII use the PCS and PMA sections of the Ethernet MAC.
TFTP	Trivial File Transfer Protocol.

- Tool Chain** Tool Chain generally refers to a C/C++ compiler, an assembler, a linker/loader, associated binary utilities and header files for a specific architecture.
- Xtal** Crystal. Quartz piezoelectric circuit element, which often serves as a frequency reference for Transmit and Receive oscillators.

- Active script tuple 89
 - Auto test 121
- A**
- Boot
 - bus 29
 - commands (U-Boot) 76
 - status messages 70
- B**
- Cable, SDA 43
 - Cavium Ethernet driver 64
 - Checksum tuple 81
 - Command Line Parameters 114
 - Compliance 19
 - Connector
 - JTAG 40
 - RJ-45 39
 - RS-232 (UART) 42
 - SFP 39
 - CSUMTESTI tuple 79
- C**
- DDR2 SDRAM 31
 - DIMMs 31
 - Serial Presence Detect 31
 - timing 31
 - Diagnostic LEDs 40
 - test 123
 - Diagnostic Utility
 - Auto Test Configuration Menu 121
 - cable 121
 - Command Line Parameters 114
 - Diagnostic LED Test 123
 - EEPROM menu 119
 - Flash Menu 124
 - Log File Location 125
 - Main Menu 116
 - memory test 127
 - run auto test 120
 - Show Core Temperature 126
 - view test results 118
 - DIP Switch
 - S1 34
 - S2 35
 - Dust plugs 23, 134
- D**
- EEPROM 32, 78
 - enumeration 93
 - mapping 78
 - programming 119
 - tuple API 96
 - tuples 79
 - Electrostatic discharge (*See ESD*)
 - Emission compliance 20
 - Environment
 - requirements 19
 - variables 73
 - ESD 22, 129
 - Examples
 - intercepts 110
 - Linux applicaton 108
 - LSP 108
 - Simple Exec Applicatio 109
 - Exported Kernel Symbols 107
 - External JTAG (EJTAG) 41
- E**
- F**
- Firmware 63
 - Flash 32
 - accessing 99
 - base address 97
 - boot 68
 - connection to Boot Bus 32
 - device identification 97
 - displaying partitions 99
 - driver 97
 - functions 98
 - mapping 97
 - partitioning 99
 - programming 124
 - reset 32
 - running the file system 100
 - FPGA
 - I2C interface 30
 - Registers
 - Board ID and DIP Switch Register 50
 - Fan and Temperature Status 56
 - I2C Address High 60
 - I2C Address Low Register 60
 - I2C Control 59
 - I2C Data Register 61
 - Interrupt Control Register 52, 53
 - LED Control Register 51
 - Peripheral Reset Register 54
 - Revision Register 49
 - Transmit Control Register 55
 - Free Software Foundation 137
 - Front panel
 - SFP connectors 33, 38
 - SGMII mode 33

G

- Generic Operating System 63
- Gigabit Ethernet PHY 33
 - PHY test tuple 84
 - SGMII mode 33
- GNU General Public License 137
- GPIO interface 28
 - signals 28

H

- Hardware components 25
- Heat sink 45
- Hot swapping SFP modules 134

I

- I2C
 - Interface 30
 - Registers 57
- Installation
 - DIMMs 130
 - optical SFPs 135
 - precautions 129
 - procedure 131
- IOCTL
 - command types 105
 - commands 103
 - device interface 102
 - examples 102

J

- JTAG
 - boundary scan 34
 - connector 40
 - scan chain 40

L

- LABI tuple 79, 82
- LEDs
 - diagnostic 40
 - front panel 38
 - location 38
- Linux Operating System 63
- LP4 Connector 45
- LSP
 - examples 108

M

- Maximum Power 45
- Memory
 - DIMMs 31
 - RLDRAM 31
- MEMTEST tuple 79
- Module
 - installation 131

- removal 133
- mtd_debug 100
- Multi-Core Processor 26
 - Boot Bus 29
 - fan controller 44
 - GPIO interface 28
 - PLL 37
 - RGMI Interface 27
 - RS-232 serial port 42
 - System Management Interface Controller 33
 - temperature monitor 44
 - TWSI interface 27

N

- Notation convention 13
- NPA Driver 101

O

- OCTEON Simple Executive 63
- Optical transceivers 39

P

- PCI interface 29
- PCI_RESET 30
- PCI-X interface 29
- PFI tuple 79, 85
- PHYTESTI tuple 79
- POST error codes 75
- Power supplies 45
- Precautions
 - handling 22
- Primary File Information (PFI) 72
- Primitives, TWSI 96
- PROC file 101, 107

R

- RAM boot 69
- RDIMM 31
- Reading data 58
- Reduced Gigabit Media Independent Interface (RGMI)
27
- Reference clock DDR2 SDRAM 31
- Related specifications and documentation 11
- Removing
 - card 133
 - DIMMs 130
 - dust plugs 135
 - SFP module 136
- Reprogramming
 - Linux Kernel in Flash 100
 - U-Boot 100
- RJ-45 connector 39
 - pin assignment 39
- RLDRAM 31
 - tuple 86

S

- S2 DIP Switch 35
- SCRIPT tuple 79
- SCRIPTACTIVE tuple 79
- Secondary File Information (SFI) 72
- SerDes tuple 87
- SERDESTEST1 tuple 79
- Serial Debug Adapter (SDA) Cable 43
- SFI tuple 79, 91
- SFP module 33
 - connectors 33, 39
 - holding 136
 - inserting 135
 - removing 136
 - storing 136
 - unlocking 136
- SGMII mode 33
- Simple Exec Library 66
- SIPI tuple 79
- Software
 - components 64
- Source IP Information (SIPI) 72
- Static-protective packaging 22, 129
- System Management Interface (SMI) Controller 33

T

- Temperature Monitor 44
- Tuple 67
 - CSUMTESTI 81
 - deteting 80
 - displaying 80
 - EEPROM 79
 - format 79, 95
- TWSI
 - interface 27
 - primitives 27, 96

U

- UA
 - composition 66
 - loading 66
- UARTCONI tuple 79
- UARTs 42
 - command interface 67
- U-Boot
 - building 68
 - commands 76
 - reprogramming 100
- Unlocking the SFP module 136
- User Application *See UA*

V

- Voltage 45

W

- Writing data 57

© 2008– 2011 GE Intelligent Platforms
Embedded Systems, Inc. All rights reserved.

The 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

