Pigeon Point Shelf Manager User Guide

ShMM-500

ShMM-300

PIGEON POINT SYSTEMS

Release 2.3
July 28, 2006

TABLE OF CONTENTS

PART I: Getting Started

Chapter

1

1. Introduction

This chapter provides an overview of the Pigeon Point Systems Pigeon Point Shelf Manager (ShM) and Shelf Management Mezzanine (or ShMM, currently ShMM-300 or ShMM-500) products.

The Pigeon Point Shelf Manager is a shelf-level management solution for AdvancedTCA™ (ATCA) products. The Pigeon Point ShMM, when coupled with a corresponding carrier board, provides the necessary hardware to run the Shelf Manager within an ATCA shelf. This document focuses on aspects of the Shelf Manager and ShMM that are common to any ShMM carrier used in an AdvancedTCA context. Carrier-specific details are separately documented.

The Pigeon Point Shelf Manager is adaptable to manage CompactPCI and (when available) CompactTCA platforms as well. This document focuses primarily on AdvancedTCA contexts, but provides CompactPCI-specific comments where appropriate.

1.1. Overview of Intelligent Platform Management in ATCA

The Pigeon Point products are the first Intelligent Platform Management building blocks designed from the ground up for modular platforms like AdvancedTCA, in which there is a strong focus on a dynamic population of Field Replaceable Units (FRUs) and maximum service availability. The Intelligent Platform Management Interface (IPMI) specification provides a solid foundation for the management of such platforms, but requires significant extension to support them well.

IPMI defines a management infrastructure that is widely used across the PC and server industry. PICMG 3.0, the AdvancedTCA specification, defines the necessary extensions to IPMI. In fact, more than 30% of the 460 pages of PICMG 3.0 are devoted to shelf management, including the definition of 25 new commands, six new FRU Information data structures—several quite complex—and 2 new sensor types.

The strategy for the Pigeon Point Shelf Manager is to fully support these extensions and also map them to other platform architectures such as CompactPCI and CompactTCA.

Figure 1 shows the logical elements of an example AdvancedTCA shelf[1], identified in terms of the ATCA specification and potential sites for incorporation of Pigeon Point products.

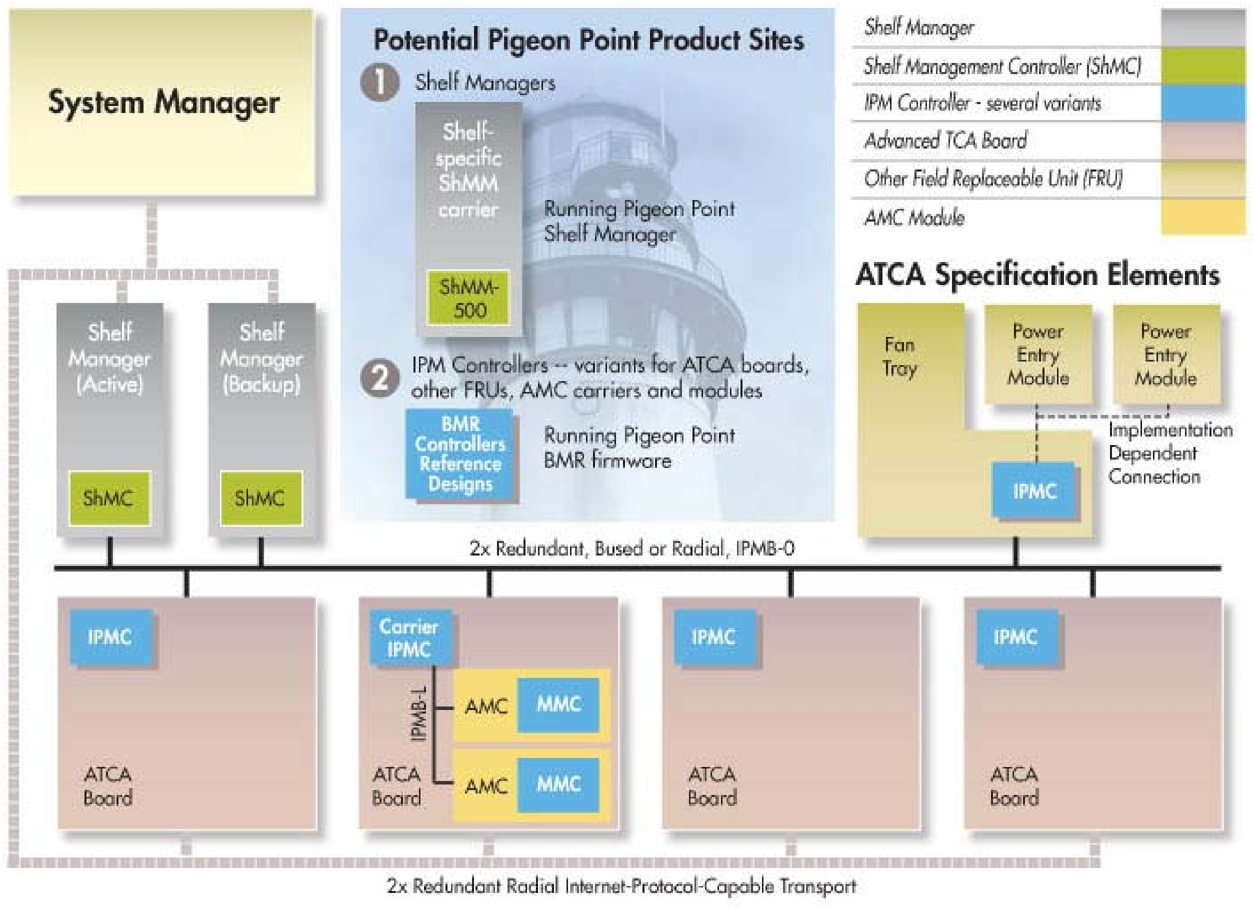
System Manager
Potential Pigeon Point Product Sites
1 Shelf Managers
Shelf-specific ShMM carrier
ShMM-500
Running Pigeon Point Shelf Manager
2 IPM Controllers -- variants for ATCA boards, other FRUs, AMC carriers and modules
BMR Controllers Reference Designs
Running Pigeon Point BMR firmware
Shelf Manager
Shelf Management Controller (ShMC)
IPM Controller - several variants
Advanced TCA Board
Other Field Replaceable Unit (FRU)
AMC Module
ATCA Specification Elements
Fan Tray
Power Entry Module
Power Entry Module
Implementation Dependent Connection
IPMC
Shelf Manager (Active)
Shelf Manager (Backup)
ShMC
ShMC
2x Redundant, Bused or Radial, IPMB-0
IPMC
ATCA Board
Carrier IPMC
IPMB-L
AMC
MMC
AMC
MMC
ATCA Board
IPMC
ATCA Board
IPMC
ATCA Board
2x Redundant Radial Internet-Protocol-Capable Transport

Figure 1 Management Aspects and Potential Pigeon Point Product Sites in an Example AdvancedTCA Shelf

An AdvancedTCA Shelf Manager communicates inside the shelf with IPM Controllers, each of which is responsible for local management of one or more Field Replaceable Units (FRUs), such as boards, fan trays or power entry modules. Management communication within a shelf occurs primarily over the Intelligent Platform Management Bus (IPMB), which is implemented on a dual-redundant basis as IPMB-0 in AdvancedTCA.

The PICMG Advanced Mezzanine Card (AdvancedMC or AMC) specification, AMC.0, defines a hot-swappable mezzanine form factor designed to fit smoothly into the physical and management architecture of AdvancedTCA. Figure 1 includes an AMC carrier with a Carrier IPMC and two installed AMC modules, each with a Module Management Controller (MMC). On-carrier management communication occurs over IPMB-L ("L" for Local).

[1] AdvancedTCA has adopted the term "shelf" for alignment with typical practice in telecommunications applications. Traditionally (for instance, in the CompactPCI specifications), the term "chassis" has been used with essentially the same meaning.

An overall System Manager (typically external to the shelf) can coordinate the activities of multiple shelves. A System Manager typically communicates with each Shelf Manager over Ethernet.

The next two sections address the board and shelf levels of management, highlighting the following Pigeon Point products and their capabilities as well as the relevant AdvancedTCA functionality:

- Pigeon Point Board Management Reference firmware and corresponding hardware reference design, which together implement various types of management controllers.

- Pigeon Point Shelf Manager software and ShMM mezzanine module, which together with an appropriate ShMM carrier board, implement an AdvancedTCA-compliant Shelf Manager and Shelf Management Controller (ShMC).

1.2. Pigeon Point Board Management Reference: Hardware and Firmware

This level includes the local management of full-size 8U AdvancedTCA boards as well as other auxiliary FRUs, such as fan trays or power entry modules. Based on the interfaces specified by IPMI and extended by AdvancedTCA and AdvancedMC, any compliant Shelf Manager can work with any compliant IPM Controller and the FRUs that it represents, including AMCs. This section focuses on controllers based on Pigeon Point technology as a concrete example.

The Pigeon Point BMR reference designs can be implemented as part of any board or other FRU, and executes the corresponding firmware, thereby realizing a compliant IPM Controller. The BMR firmware represents one or more FRUs (via IPMB-0) to the Shelf Manager, including:

- Providing inventory information identifying each such FRU, including its manufacturer and other data.

- Describing and implementing a set of logical sensors (such as for temperature, state of IPMB-0, and operational state for each FRU (activated, deactivated, etc.)).

- Generating events (typically directed to the Shelf Manager) for exceptional conditions detected by any sensor, based on its configured event generation settings.

- Negotiating with the Shelf Manager for resources needed by the FRU(s), including power and interconnects.

BMR firmware running on an AMC Carrier IPMC additionally represents its installed AMCs to the Shelf Manager, including negotiating for power resources on their behalf. Furthermore, a Carrier IPMC negotiates with its AMCs and on-carrier switching resources regarding interconnect configurations. The AMC management architecture is purposely designed to: 1) avoid impacting existing ATCA Shelf Managers and 2) minimize the resources required to implement a Module Management Controller, since board real estate and cost are at a premium on AMCs.

There are Pigeon Point BMR variants for AdvancedTCA and CompactPCI IPM Controllers, as well as for AMC Carrier IPMCs and MMCs. Currently, the principal BMR variants are based on the Atmel AVR ATmega and Renesas H8S/2168 microcontroller families.

Pigeon Point Board Management Starter Kits for each of these BMR variants include all the materials necessary (documentation, schematics, bill of materials, firmware source code and development tools, etc.) for Intelligent FRU developers to integrate a reference design directly into their boards and take immediate advantage of the fully validated BMR firmware.

More details (including product briefs) on the available Pigeon Point BMR variants and corresponding Starter Kits are available at www.pigeonpoint.com/products.htm.

1.3. Pigeon Point Shelf Manager and ShMM

The Pigeon Point Shelf Manager (consistent with AdvancedTCA Shelf Manager requirements) has two main responsibilities:

- Manage/track the FRU population and common infrastructure of a shelf, especially the power, cooling and interconnect resources and their usage. Within the shelf, this management/tracking primarily occurs through interactions between the Shelf Manager and the IPM Controllers over IPMB-0.
- Enable the overall System Manager to join in that management/tracking through the System Manager Interface, which is typically implemented over Ethernet.

Much of the Pigeon Point Shelf Manager software is devoted to routine missions such as powering a shelf up or down and handling the arrival or departure of FRUs, including negotiating assignments of power and interconnect resources. In addition, the Shelf Manager can take direct action when exceptions are raised in the shelf. For instance, in response to temperature exceptions the Shelf Manager can raise the fan levels or, if that step is not sufficient, even start powering down FRUs to reduce the heat load in the shelf.

1.3.1. Pigeon Point Shelf Manager Features

- Executes on the ShMM, a compact SO-DIMM form-factor module, installed on a suitable carrier board for the shelf.
- Conforms to the AdvancedTCA specification.
- Monitors activities within the shelf via the ATCA-specified dual redundant Intelligent Platform Management Bus (IPMB).
- Accepts and logs events posted by any intelligent FRU in the shelf (reflecting exceptions in temperatures, voltages, etc.); posts alerts outside the shelf based on configurable IPMI Platform Event Filters.
- Supports hot swapping of Field Replaceable Units (FRUs), while maintaining full management visibility.

- Interfaces to standard "Telco Alarm" infrastructures, via ShMM carrier-implemented dry contact relays.
- Supports redundant Shelf Manager instances for high availability.
- Integrates a watchdog timer, which resets the ShMM if not periodically strobed; such resets automatically trigger a switchover to the backup ShMM, if configured.
- Includes battery-backed real-time clock for time-stamping events.
- Implements rich set of shelf-external interfaces accessible over Ethernet, including Remote Management Control Protocol (RMCP, required by AdvancedTCA), command line, web browser, Simple Network Management Protocol (SNMP).

The Pigeon Point Shelf Manager can also be used in CompactPCI shelves and is already substantially compatible with the CompactTCA specification, currently in development within PICMG.

1.3.2. Support for Dual Redundant Operation

The Pigeon Point Shelf Manager can be configured with active/backup instances to maximize availability. Figure 2 shows how both instances are accessible to the System Manager, with only the active instance interacting at any given time. Similarly, only the active instance communicates over IPMB-0 with the IPM Controller population in the shelf. The two instances communicate over TCP/IP, with the active instance posting incremental state updates to the backup. (On the ShMM-500, these state updates can occur via USB connections between the ShMMs.) As a result, the backup can quickly step into the active role if necessary.

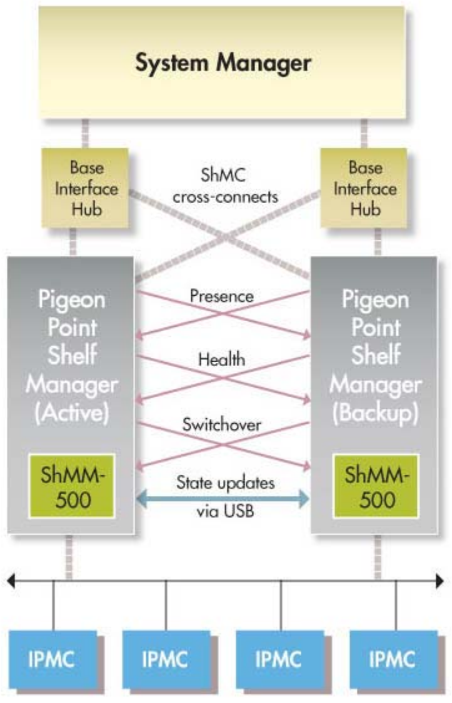

Figure 2: Pigeon Point Shelf Manager Redundancy Support

Three cross-connected signals between the two Shelf Manager instances enhance their coordination:

- Presence: each Shelf Manager instance knows whether the other instance is present in the shelf.
- Health: each instance knows whether the other instance considers itself "healthy."
- Switchover: the backup instance can force a switchover if necessary.

When dual ShMM-500s are configured for state updates via USB, both Ethernet interfaces become available for external communication. This allows support for an ATCA feature that was recently adopted by PICMG: ShMC cross-connects. As shown in Figure 2, ShMC cross-connects allow both ShMMs to be connected with both Base Interface Hubs. This improves system availability because either hubs or ShMMs can switchover independently, if necessary.

1.3.3. System Manager Interface

Another major subsystem of the Pigeon Point Shelf Manager implements the System Manager Interface. "System Manager" is a logical concept that may include software as well as human operators in the "swivel chairs" of an operations center. The Pigeon Point Shelf Manager provides a rich set of System Manager Interface options, which provide different mechanisms of access to similar kinds of information and control regarding a shelf.

One such mechanism is the IPMI LAN Interface. To maximize interoperability among independently implemented shelf products, this interface is required by the AdvancedTCA specification and supports IPMI messaging with the Shelf Manager via the Remote Management Control Protocol (RMCP). A System Manager that uses RMCP to communicate with shelves should be able to interact with any ATCA-compliant Shelf Manager. This relatively low level interface provides essentially complete access to the IPMI aspects of a shelf, including the ability for the System Manager to issue IPMI commands to IPM Controllers in the shelf, using the Shelf Manager as a proxy.

In addition, the Pigeon Point Shelf Manager provides two interfaces oriented towards human users rather than programmatic ones:

- Command Line Interface (CLI): This interface provides a comprehensive set of textual commands that can be issued to the Shelf Manager via either a physical serial connection or Telnet.
- Web-based Interface: This interface enables essentially the same functionality as the CLI, with access to the Shelf Manager via a web browser.

Using either of these mechanisms, the System Manager can access information about the current state of the shelf, including current FRU population, sensor values, threshold settings, recent events and overall shelf health.

Finally, the Pigeon Point Shelf Manager supports Simple Network Management Protocol (SNMP) access to the shelf. This popular management protocol is supported with a custom Management Information Base (MIB) providing Get and Set access to a wide range of information and controls regarding the shelf.

These aspects of the ATCA's System Manager Interface are considered to be the Pigeon Point Shelf-External Interfaces. They are documented separately in a corresponding reference guide.

1.3.4. *Pigeon Point ShMM Shelf Management Mezzanines*

The Pigeon Point Shelf Manager executes on the ShMM, a small (67.60mm x 50.80mm) Shelf Management Mezzanine that conforms to the Small-Outline Dual Inline Memory Module (SO-DIMM) specification. There are currently two ShMM variants: ShMM-300 and ShMM-500. The following table shows the key characteristics of these variants.

Feature	**ShMM-300**	**ShMM-500**
CPU	Texas Instruments TMS320VC5471 (C5471)	AMD Alchemy Au1550
Processor core(s)	47.5 MHz ARM7 and 100 MHz C54x DSP	333 Mhz MIPS-32
SDRAM	16 Mbytes	64 or 128 Mbytes
Flash	8 Mbytes	16 or 64 Mbytes
Ethernet	Dual 10/100 Mbit	Dual 10/100 Mbit
Serial	Two, one with modem controls	Two, one with modem controls
Universal Serial Bus (USB)	No	Host and device ports
Duplex IPMB-0	Yes	Yes
ATCA watchdog timer	Yes	Yes
Real-time clock, optionally battery backed on ShMM carrier	Yes	Yes
General Purpose I/O signals	Nine	Nine
Shelf Manager redundancy and hot swap interface, with on-board CPLD assist	Yes	Yes
High speed interface(s) to on-carrier devices	Serial Peripheral Interface (SPI)	Multiple ports supporting either SPI or SMBus
JTAG interface for processor debug and flash programming	Yes	Yes

Chapter
2

2. Configuration

The Shelf Manager runs on top of Monterey Linux (www.montereylinux.com), a specialized implementation of Linux with separate editions for the ShMM-300 and the ShMM-500. The lowest layer of Monterey Linux is the firmware monitor, which is called ARMboot on the ShMM-300 and U-Boot on the ShMM-500. U-Boot works on multiple processor architectures, not just ARM and is generally upward compatible with ARMboot. In the remainder of this chapter, the firmware monitor is generally referenced as U-Boot, except where it is important to distinguish the two variants.

2.1. Setting up U-Boot

On a power-up/reset of the ShMM, the hardware starts executing the U-Boot firmware in Flash. The firmware performs basic initialization of the ShMM, and unless the user explicitly disables the Autoboot feature (thus forcing the firmware to switch to the maintenance user command interface), commences booting the Linux kernel. Linux is booted from the kernel and root file system images residing in Flash. U-Boot relocates the kernel image to RAM, sets up kernel parameters, and passes control to the kernel entry point.

2.1.1. U-Boot Interface

U-Boot is accessible via the Serial port of the ShMM and requires configuration specific to the intended operational environment. When the ShMM is powered up, text like following will be displayed on the console:

For ShMM-300:
```
ARMboot 1.0.2 (Apr 18 2003 - 14:58:54)

ARMboot code: 10f00000 -> 10f156a4
DRAM Configuration:
Bank #0: 10000000 16 MB
Flash: 8 MB
Hit any key to stop autoboot:  0
ShMM #
```

For ShMM-500:

```
U-Boot 1.1.2 (Apr 27 2005 - 19:17:09)

CPU: Au1550 324 MHz, id: 0x02, rev: 0x00
Board: ShMM-500
S/N: 00 00 00 00 00 00 00 00 00 03 03 03
DRAM:  64 MB
Flash: 16 MB
In:    serial
Out:   serial
Err:   serial
Net:   Au1X00 ETHERNET
Hit any key to stop autoboot:  0
shmm500
```

"ShMM #" (or "shmm500") is the prompt allowing for user commands to be entered. For a complete set of supported commands, type "help".

2.1.2. U-Boot Environment Variables

U-Boot includes a set of environment variables that should be configured prior to use. The following table describes the default set of variables available:

Environment Variable	Description
addmisc	Appends quiet, reliable_upgrade and console settings to bootargs. This variable is normally not modified.
baudrate	Serial port baud rate, default=115200 (=9600 on ShMM-300).

bootargs	Command line to be passed to the Linux kernel. May contain references to other U-Boot environment variables, which will be resolved at run-time. On ShMM-500, the default value is:
	root=/dev/ram rw console=ttyS0,115200 reliable_upgrade=y.
	On ShMM-300, the default value is:
	console=ttyS0,9600 root=/dev/ram0 IPADDR=$(ipaddr) IPDEVICE=$(ipdevice) IP1ADDR=$(ip1addr) IP1DEVICE=$(ip1device) GATEWAY=$(gatewayip) RMCPADDR=$(rmcpaddr) TIMESERVER=$(time_server) TZ=$(timezone) HOSTNAME=$(hostname) RC2=$(rc2) FLASH_RESET=$(flash_reset) PASSWORD_RESET=$(password_reset) START_RC2_DAEMONS=$(start_rc2_daemons) RC_IFCONFIG=$(rc_ifconfig) LOGGING=$(logging)

bootcmd	U-Boot command executed to accomplish auto-booting. Normally, this is something similar to bootm BFB00000 BFC40000 (bootm 0x20000 on ShMM-300), which starts the Linux image stored in Flash.

Environment Variable	Description
bootdelay	Autoboot delay value, in seconds. Default setting: 3.
bootfile	Parameter that specifies what kernel image should be used by the net and nfs boot options
console	Setting for the kernel and init script console port and baud rate. Default is console=ttyS0,115200
ethaddr	MAC address of the primary on-chip Ethernet controller. The value of this variable is set automatically by U-Boot. This address is passed to the kernel Ethernet driver.
eth1addr	MAC address of the secondary Ethernet controller. The value of this variable is set automatically by U-Boot. This address is passed to the kernel Ethernet driver.
flash_reset	Instructs Linux to erase the flash filesystems (/etc and /var), restoring to factory default (y/n). The system startup script sets this variable back to "n" after the flash erase. Default is "n".
gatewayip	Default gateway IP address. This variable can be passed as a part of the kernel command line to automatically configure routing for the network interfaces. Default setting: 192.168.0.1.
hostname	Network host name; default is "sentry"
io_config	Determines if the PSC controllers are configured for the dual-slave-address-configuration (y/n). Default setting: y.
ipaddr	IP address used by the primary on-chip Ethernet interface. This variable is used to configure the network interface specified by ipdevice automatically if the rc_ifconfig variable is set to "y". Note that the system startup script sets the least significant bit of this variable to the least significant bit of the Hardware Address for the ShMM carrier; that is, if the Hardware Address is an even value, the last bit in the IP address is set to 0, otherwise it is set to 1. This is done in the startup script /etc/netconfig to support coordinated IP address configurations on redundant ShMMs. To disable this functionality, simply remove the /etc/readhwaddr file.
ip1addr	IP address used by the secondary Ethernet interface. This variable can be passed as a part of the kernel command line to automatically configure the corresponding kernel network interface.
ip1device	Device corresponding to ip1addr; "eth1" is default.
kernel_start	The absolute starting address of the kernel image in Flash. This variable is set automatically by U-Boot during bootstrap.
logging	Specifies if messages log file should be maintained in ram or flash. Default is "ram", which is the recommended option.

Environment Variable	Description
net	This variable can be used as a replacement for bootcmd as a means of booting a kernel and rfs image from TFTP. Use run net.
netmask	Network netmask, default=255.255.255.0
nfs	This variable can be used as a replacement for bootcmd as a means of booting and running with an NFS mounted root filesystem. See the Monterey Linux User Guide sample NFS project for details
password_reset	Instructs Linux to restore factory default password for user "root" (which is the empty password ""). Default is "n".
post_normal	Determines the list of POST tests that are executed on each boot-up. If not set, compile-time default settings are used. The test names listed in a value of this variable are separated by space characters.
post_poweron	Determines the list of POST tests that are executed after power-on reset only (vs. on each boot-up). If not set, compile-time default settings are used. The test names listed in a value of this variable are separated by space characters.
quiet	Instructs the kernel upon bootup not to print progress messages to the serial console. Default is quiet=quiet.
ramargs	Sets the kernel command line in the bootargs variable as appropriate for the root filesystem to be mounted from a ramdisk.
ramdisk	Specifies what .rfs image should be used by the net and nfs boot options.
ramsize	Size of the system memory, in bytes. Default setting: calculated from the SDRAM configuration encoding in the build-time configuration block
rc_ifconfig	Allows the /etc/rc script to set up the IP address instead of shelfman. Default is "n" (allow shelfman to set up IP addresses).
rc2	Specifies secondary RC script that is to be invoked. This is the carrier-specific startup script. Default is /etc/rc.carrier3 or other appropriate script for given target platform.
reliable_upgrade	Determines if the reliable software upgrade procedure is enabled on the ShMM-500 (y/n). Default setting: y. Setting this variable to n is not currently supported. If the variable is set to n, on the ShMM-500's next boot, it will issue an error message and hang.
rfs_start	The absolute starting address of the root filesystem image in Flash. This variable is set automatically by U-Boot during bootstrap.
rmcpaddr	Default IP address for the RMCP service.
serverip	IP address of the TFTP server

Environment Variable	Description
start_rc2_daemons	Instructs the secondary startup script to start or not start the snmpd/boa and shelfman daemons after bootup. Default is "y".
time_proto	Protocol used to retrieve time from a network time server; possible values are "ntp" and "rdate".
time_server	Time server for synchronization at runtime. If this variable is not specified, time is extracted from the hardware clock at system startup.
timezone	Local time zone in CCC*n* format where *n* is the offset from GMT and optionally negative, while CCC identifies the time zone. The default is UTC0.

2.1.3. Assigning Values to Environment Variables

To assign a value to an environment variable, use the format (in this example and examples below, ShMM-500 users see the command prompt "shmm500" instead of the ShMM-300-oriented prompt that is shown):

```
ShMM # setenv <variable_name> <new_value>
```

For example:

```
ShMM # setenv bootdelay 1
```

Once all of the environment variables have been properly set, you need to save them back out to the Flash so that they will remain after the ShMM is powered down. The "`saveenv`" command is used for this purpose.

```
ShMM # saveenv
```

The **setenv** functionality is also available as a Linux utility with the same usage. To display U-Boot variables at the shell prompt, use the additional **getenv** utility or issue the **setenv** command without parameters.

2.1.4. Configuration Environment Variables for the Shelf Manager

When U-Boot is started for the first time, the following default environment variables are defined.

For ShMM-300:

```
bootcmd=run setup_bootargs; bootm 20000 120000
bootdelay=3
baudrate=9600
ethaddr=00:50:C2:22:xx:yy
eth1addr=00:50:C2:22:xx:zz
serverip=192.168.0.7
netmask=255.255.0.0
```

```
hostname=sentry
gatewayip=192.168.0.1
ipdevice=eth0
ip1addr=192.168.1.3
ip1device=eth1
rc2=/etc/rc.acbfc
ipaddr=192.168.0.2
start_rc2_daemons=y
flash_reset=n
password_reset=n
logging=ram
rc_ifconfig=n
setup_bootargs= setenv bootargs console=ttyS0,9600 root=/dev/ram0
                IP1ADDR=$(ip1addr) IP1DEVICE=$(ip1device)
                IPADDR=$(ipaddr) IPDEVICE=$(ipdevice)
                HOSTNAME=$(hostname) RC2=$(rc2)
                GATEWAY=$(gatewayip) RMCPADDR=$(rmcpaddr)
                TIMESERVER=$(time_server) TZ=$(timezone)
                FLASH_RESET=$(flash_reset) PASSWORD_RESET=$(password_reset)
                RC_IFCONFIG=$(rc_ifconfig)
                START_RC2_DAEMONS=$(start_rc2_daemons)
                LOGGING=$(logging)
```

For ShMM-500:

```
bootcmd=run setup_bootargs; bootm BFB00000 BFC40000
bootdelay=3
baudrate=115200
ethaddr= 00:00:1a:18:xx:yy
eth1addr= 00:00:1a:18:xx:zz
serverip=192.168.0.7
netmask=255.255.0.0
hostname=sentry
gatewayip=192.168.0.1
ipdevice=eth0
ip1addr=192.168.1.3
ip1device=eth1
rc2=/etc/rc.acbfc
ipaddr=192.168.0.2
start_rc2_daemons=y
flash_reset=n
password_reset=n
logging=ram
rc_ifconfig=n
bootfile=sentry.mips.kernel
ramdisk=sentry.mips.rfs
net=tftp 80400000 $(bootfile); tftp 80800000 $(ramdisk); bootm 80400000
80800000
rmcpaddr=192.168.1.15
timezone=EST
bootargs=root=/dev/ram rw console=ttyS0,115200 reliable_upgrade=y
```

Several of these environment variables need be reconfigured with values that are appropriate to the network context in which the ShMM will be used.

2.2. Setting up Ethernet

The ShMM uses two Ethernet ports, one of them being used for shelf-external access. Since RMCP is the only shelf-external interface that is required by ATCA, the shelf-external Ethernet port is referenced as the RMCP port, though the other shelf-external interfaces (HTTP, TELNET, FTP) are accessible via this port as well. The other Ethernet port can be used for communication between redundant Shelf Managers, or potentially for other purposes if the redundancy link is provided by another mechanism.

2.2.1. Usage of the First Ethernet Interface

Since the RMCP Ethernet port will be directly connected to the site network, the IP address should be set up appropriately for that network. For example, if the site uses the IP address range 192.168.0.x, the RMCP Ethernet port should be set to a unique IP address within that range such as 192.168.0.2. In a redundant ShMM setup, it is important to note that only one ShMM (the active ShMM) has the RMCP IP address enabled on the RMCP Ethernet port. The backup ShMM assigns the same IP address to the RMCP Ethernet port, but only enables it when that ShMM assumes the active role. This way, the RMCP IP address maintains availability in a failover situation.

Assigning an Additional IP Address to the First Network Interface

In the default configuration, no IP address is assigned to the first network interface (and the ShMM is not accessible over the network) until the Shelf Manager starts and the RMCP IP address is assigned. However, it may be useful in some cases to assign an IP address to the RMCP network interface and have the ShMM accessible over the network as soon as the operating system is booted. In that case, it is also desirable that when the Shelf Manager is started, for the RMCP IP address to coexist with the originally assigned IP address rather than replacing it.

To achieve this configuration, it is necessary to instruct the Shelf Manager to assign the RMCP IP address not to the first network adapter itself ("eth0") but to its first alias ("eth0:1"). The initial IP address will be assigned in that case to the network adapter itself ("eth0") during the start of the operating system. This initial assignment happens in the initialization script /etc/rc; it is accomplished by: 1) enabling the U-BOOT variable "rc_ifconfig" (setting it to "y"), 2) assigning the original IP address to the U-BOOT variable "ipaddr", for example:

```
setenv rc ifconfig y
setenv ipaddr 192.168.1.240
```

and 3) changing the value of the Shelf Manager configuration parameter RMCP_NET_ADAPTER to "eth0:1".

In a redundant configuration, the U-BOOT variable "ipaddr" is allowed to have the same value on both ShMMs. The actual initial IP address assigned to each of the two redundant ShMMs will be based on the value of "ipaddr" but will be modified depending on the hardware address of the ShMM. The least significant bit of the IP address will be set to the least significant bit of the hardware address. In the example above, the IP address will be 192.168.1.240 for the ShMM with an even hardware address, and will be 192.168.1.241 for the ShMM with an odd hardware address. This modification of the IP address can be turned off by removing the file "/etc/readhwaddr".

RMCP Address Propagation

There is an optional feature of the Shelf Manager that allows the backup ShMM also to be exposed on the external network with an IP address that is different from the RMCP IP address only in the least significant bit. The netmask and default gateway on the backup ShMM will be the same as on the active ShMM. For example, if the RMCP IP address is 192.168.0.2, the backup ShMM will have the corresponding IP address 192.168.0.3, with the same netmask and default gateway. To enable this feature, it is necessary to define the Shelf Manager configuration parameter PROPAGATE_RMCP_ADDRESS as TRUE in the Shelf Manager configuration file.

2.2.2. Usage of the Second Ethernet Interface

Originally, the second Ethernet interface was dedicated for use as a private network between redundant ShMMs and was used to synchronize state information between the active and backup ShMMs. This is still the only usage mode for this interface for ShMM-300. Unlike the RMCP Ethernet port, this second Ethernet interface is always enabled on both the active and backup ShMM, but with a small twist – both the active and backup ShMM specify the same IP address for the redundancy interface, but software assigns the next logical IP address to the ShMM with an odd hardware address. For instance, the default setting for the redundancy Ethernet port is 192.168.1.2. The odd-addressed ShMM will assume the address 192.168.1.3. This way the active and backup ShMM can be identically configured but still assume unique IP addresses for the redundancy Ethernet link.

On the ShMM-500, this second network interface can potentially be used for other purposes, such as for support in the Shelf Manager for ShMC cross connects, in accordance with the PICMG ECN 3.0-2.0-001. In that case, the second network interface connects the Shelf Manager with one of the ATCA network hub boards. Dual USB-based network interfaces can be used for communication between the redundant Shelf Managers (see the next section).

To configure the Shelf Manager to support cross-connects, it is necessary to define the configuration parameter RMCP_NET_ADAPTER2. In that case, if the backplane and hub boards also support cross-connects, the Shelf Manager will use the two network adapters (RMCP_NET_ADAPTER and RMCP_NET_ADAPTER2) for RMCP communication. Two usage models are available for this feature:

- Redundant usage.
- Parallel usage.

2.2.2.1. Redundant Usage of the Two Network Interfaces

The two network interfaces can be used in a redundant way. At any given time, RMCP communication will pass through only one adapter (initially this is RMCP_NET_ADAPTER). However, if the Shelf Manager detects that the adapter currently used for RMCP communication becomes physically disconnected from the network (link broken), it automatically switches to the other adapter. The first adapter is turned off, the RMCP IP address is transparently moved to the other adapter, and three ARP notifications are broadcast to notify other systems about the address change. This change is transparent for the System Manager and does not break existing RMCP connections.

However, if the Shelf Manager detects that the other network adapter is also physically disconnected from the network, it does not perform the IP address switchover described above, but performs a full switchover to the backup Shelf Manager. A full switchover is also performed if the Shelf Manager detects physical disconnection of the network adapter used for RMCP communication, in non cross-connect configurations.

Detection of physical disconnection of the RMCP network adapter is controlled by the Shelf Manager configuration parameter SWITCHOVER_TIMEOUT_ON_BROKEN_LINK. The value of this parameter is the time interval in seconds during which the adapter stays physically disconnected, before the Shelf Manager performs an IP address switchover or full switchover. Detection and switchover is disabled if the value of this parameter is equal to -1.

Another configuration parameter, INITIAL_SLOW_LINK_DELAY, specifies the time interval from the start of the Shelf Manager, during which detection of physical disconnection is not performed. This allows using Ethernet links that are slow to start and need some time after shelf power up to establish the physical connection.

Usually the parameter RMCP_NET_ADAPTER2 is assigned the value "eth1", while the value of the configuration parameter RMCP_NET_ADAPTER (the main network adapter used for RMCP communication) is "eth0", as in the following sample.

```
RMCP_NET_ADAPTER = "eth0"
RMCP_NET_ADAPTER2 = "eth1"
```

However, other configurations are possible. For example, values "eth0:1" and "eth1:1" can be used if additional (permanent) IP addresses need to be assigned to both network interfaces.

The two network interfaces are used in redundant mode if the configuration parameter USE_SECOND_CHANNEL is set to FALSE.

2.2.2.2. Parallel Usage of the Two Network Interfaces

The approach to network interface redundancy outlined in the previous section seems to be insufficient for some configurations where the network connection between the Shelf Manager and the System Manager goes through several switches and may break on an Ethernet segment that is not adjacent to the ShMM. This type of failure cannot be immediately recognized by the Shelf Manager. Checking the accessibility of the System Manager from the Shelf Manager does not seem practical in this case, since the architecture of the System Manager and its usage of IP addresses is not defined in the ATCA specification and the Shelf Manager design should not artificially limit it. The solution in this case should be implemented at the System Manager level.

Some new features introduced in ECN-002 to the ATCA specification (PICMG 3.0 R2.0) facilitate a solution. The command "Get Shelf Manager IP Addresses" allows the System Manager to retrieve the IP addresses exposed by the Shelf Managers, both active and backup. Using this information, the System Manager can check the availability of the Shelf Managers and the current distribution of responsibilities between them (active vs. backup).

In parallel mode, instead of having a single RMCP network address that is switched between the two network interfaces, the Shelf Manager supports RMCP on both interfaces with different IP addresses, as two separate IPMI channels (channels 1 and 2). Each channel has its own set of LAN configuration parameters that includes the IP address, network mask, default gateway, etc. Both addresses are available via the "Get Shelf Manager IP Addresses" command. The System Manager, in this case, is responsible for switching over to a different IP address if the currently used IP address becomes unavailable, or maintaining two parallel RMCP sessions to both addresses. If IP addresses on both interfaces become unavailable, the System Manager can access one of the IP addresses on the backup Shelf Manager and initiate a switchover in a non-standard way (for example, log in to the backup ShMM via telnet and issue the command "clia switchover").

The parallel mode is enabled by setting the configuration parameter USE_SECOND_CHANNEL to TRUE. In this case, the configuration parameter DEFAULT_RMCP_IP_ADDRESS_2 should be set to a non-zero IP address. This IP address becomes the default IP address for the channel 2, and the configuration variables DEFAULT_RMCP_NETMASK2 and DEFAULT_GATEWAY_IP_ADDRESS2 specify the network mask and default gateway IP address for the channel 2, respectively. The values of the above-mentioned configuration parameters take no effect if the correspondent values in the channel configuration parameter file are non-zero. The channel configuration parameters for the channel 2 are stored in the file "/var/nvdata/ch2_param" on the ShMM.

In parallel mode, the IP addresses on both network interfaces are switched over to the backup ShMM as a result of a switchover.

2.2.3. *Using Dual USB Network Interfaces for Redundant Communication*

On the ShMM-500, two additional network interfaces are implemented over the two USB connections. In this configuration, they always connect the two redundant Shelf Managers. These interfaces are named "usb0" and "usb1". The interface "usb0" always exists, while the interface "usb1" exists only if the interface "usb0" is active on the peer Shelf Manager (which means that the peer Shelf Manager is physically installed and running). Also, the interfaces are cross-connected: "usb0" on the first Shelf Manager is connected to "usb1" on the second Shelf Manager, and vice versa.

The Shelf Manager supports usage of the USB network interfaces for communication between the redundant Shelf Managers. To use this feature, it is necessary to define two redundancy network adapters in the Shelf Manager configuration file /etc/shelfman.conf, as follows:

```
REDUNDANCY_NET_ADAPTER = "usb0"
REDUNDANCY_NET_ADAPTER2 = "usb1"
```

One additional consideration relates to the definition of the subnet mask for the redundancy network interfaces. In the legacy case, when only one redundant network adapter is used, two different IP addresses are derived from the redundancy IP address specified in /etc/shelfman.conf. They are assigned to the two endpoints of the redundancy connection and differ only in the least significant bit.

However, when two redundancy network adapters are used, four different IP addresses are used, one for each of the endpoints (two endpoints on each of the two redundant Shelf Managers). To ensure proper operation, the two endpoints on the same Shelf Manager ("usb0" and "usb1") must belong to different logical networks, while "usb0" on one Shelf Manager and "usb1" on the other Shelf Manager must belong to the same logical network. This is achieved by dividing the IP address space into two ranges. These ranges (logical networks) are defined by the subnet mask given by the parameter `REDUNDANCY_NETMASK` from the configuration file /etc/shelfman.conf. If the network mask is 255.255.255.128 then the first range is

192.168.1.0 - 192.168.1.127 and the other is 192.168.1.128 – 192.168.1.255. The "usb0" endpoint on the first Shelf Manager and the "usb1" endpoint on the other Shelf Manager will be in the first range. The "usb1" endpoint on the first Shelf Manager and the "usb0" endpoint on the other Shelf Manager will be in the second range.

The 4 IP addresses in question can be derived from one IP address (for example, the IP address assigned to "usb0" on the Shelf Manager with the even hardware address) and the network mask (for which the recommended value is 255.255.255.128). The rules are as follows:

- The IP address for "usb0" on the Shelf Manager with the even hardware address must always be set even if the file /etc/readhwaddr is present.
- The IP address for "usb0" on the Shelf Manager with the even hardware address is set by the parameter `REDUNDANCY_IP_ADDRESS` from the file /etc/shelfman.conf if the file /etc/readhwaddr is not present.
- To compute the IP address for "usb0" on the Shelf Manager with the even hardware address when the file /etc/readhwaddr is present you should set the least significant bit of `REDUNDANCY_IP_ADDRESS` to 0.
- To compute the IP address for "usb1" on the other Shelf Manager you should toggle the least significant bit in the "usb0" IP address on the first Shelf Manager. This guarantees that "usb0" on Shelf Manager with the even hardware address and "usb1" on the Shelf Manager with the odd hardware address are in the same logical network and are not equal to each other.
- To compute the IP address for "usb1" on the Shelf Manager with even hardware address you should take the IP address for "usb0" on the same Shelf Manager and toggle the least non-zero bit of the network mask. This guarantees that "usb0" and "usb1" on the Shelf Manager with the even hardware address are in different logical networks.
- The last step is to compute the IP address for "usb0" on the Shelf Manager with the odd hardware address. You should either toggle the least significant bit in the IP address for "usb1" on the Shelf Manager with the even hardware address or toggle the least non-zero bit of the network mask in the IP address for "usb1" on the Shelf Manager with odd hardware address. The result will be the same. This guarantees that "usb0" on Shelf Manager with the odd hardware address and "usb1" on Shelf Manager with the even hardware address are in the same logical network and are not equal.

Here is an example of deriving IP addresses for the USB network interfaces, under the assumption that the following definitions are in /etc/shelfman.conf:

```
REDUNDANCY_IP_ADDRESS = 192.168.1.2
REDUNDANCY_NETMASK = 255.255.255.128
```

The least significant non-zero bit in the network mask is the 7th bit (where smaller bit numbers are less significant). To toggle this bit in an IP address it is sufficient to add 128 (if this bit is set to zero in the IP address) or subtract 128 (if this bit is set to 1 in the IP address).

To toggle the least significant bit in an IP address it is sufficient to add 1 if the IP address is even or subtract 1 if the IP address is odd. Since `REDUNDANCY_IP_ADDRESS` is even, the computations are the same whether the file /etc/readhwaddr is present or not.

On the ShMM with the even hardware address the assignment of IP addresses will look like this:

- "usb0": 192.168.1.2 (no changes)
- "usb1": 192.168.1.130 (toggling the least significant non-zero bit of the netmask)

On the ShMM with the odd hardware address the assignment of IP addresses will look like this:

- "usb0": 192.168.1.131 (toggling the least significant bit of the IP address and the least non-zero bit of the netmask)
- "usb1": 192.168.1.3 (toggling the least significant bit of the IP address)

Here is another example of deriving IP addresses for the USB network interfaces, under the assumption that the following definitions are in /etc/shelfman.conf:

```
REDUNDANCY IP ADDRESS = 192.168.1.13
REDUNDANCY_NETMASK = 255.255.255.128
```

Suppose also that the file /etc/readhwaddr is present.

The least significant non-zero bit in the network mask is 7th bit. To toggle this bit in an IP address it is sufficient to add 128 (if this bit is set to zero in the IP address) or subtract 128 (if this bit is set to 1 in the IP address). To toggle the least significant bit in an IP address it is sufficient to add 1 if the IP address is even or subtract 1 if the IP address is odd.

On the ShMM with the even hardware address the assignment of IP addresses will look like this:

- "usb0": 192.168.1.12 (since the file /etc/readhwaddr is present, the least significant bit should be set to zero)
- "usb1": 192.168.1.140 (toggling the least significant non-zero bit of the netmask)

On the ShMM with the odd hardware address the assignment of IP addresses will look like this:

- "usb0": 192.168.1.141 (toggling the least significant bit of the IP address and the least significant non-zero bit of the netmask)
- "usb1": 192.168.1.13 (toggling the least significant bit of the IP address)

2.2.4. Changing the Default ShMM Network Parameters

Configuring a ShMM to work in a specific network environment requires changing the following network parameters:

RMCP IP address
RMCP GATEWAY address
RMCP Netmask

As discussed in section 2.2, changing the second dedicated redundancy Ethernet interface is not required since this is a dedicated private network between redundant ShMMs.

Changing the RMCP network parameters is a two step process. First, the U-BOOT network environment variables need to be updated, then the booted ACTIVE ShMM module network settings need to be updated using the Shelf Manager command line interface (CLIA). Specific steps are shown below:

- Attach a serial port console connection to the ShMM module. This typically will be 9600 Baud, N/8/1 for ShMM-300, and 115200 Baud, N/8/1 for ShMM-500. Reset the ShMM carrier and press the space bar to interrupt the automatic boot-up process. You should see:

```
U-Boot 1.1.2 (Apr 27 2005 - 19:17:09)

CPU: Au1550 324 MHz, id: 0x02, rev: 0x00
Board: ShMM-500
S/N: 00 00 00 00 00 00 00 00 00 03 03 03
DRAM:  64 MB
Flash: 16 MB
In:    serial
Out:   serial
Err:   serial
Net:   Au1X00 ETHERNET
Hit any key to stop autoboot:  0
shmm500
```

- Echo current network settings:

```
shmm500 printenv rmcpaddr netmask gatewayip
rmcpaddr=192.168.0.44
netmask=255.255.255.0
gatewayip=192.168.0.1
shmm500
```

- Change settings and commit to non-volatile storage:

```
shmm500  setenv rmcpaddr 10.1.1.10
shmm500  setenv netmask 255.255.0.0
shmm500  setenv gatewayip 10.1.1.1
shmm500  saveenv
Saving Environment to EEPROM...
shmm500
```

- Boot the ShMM up to full operational state and log in as user root.

```
shmm500 reset


U-Boot 1.1.2 (Apr 27 2005 - 19:17:09)

CPU: Au1550 324 MHz, id: 0x02, rev: 0x00
Board: ShMM-500
S/N: 00 00 00 00 00 00 00 00 00 03 03 03
DRAM:  64 MB
Flash: 16 MB
In:    serial
Out:   serial
Err:   serial
Net:   Au1X00 ETHERNET
Hit any key to stop autoboot:  0
## Booting image at bfb00000 ...
   Image Name:   MIPS Linux-2.4.26
   Created:      2005-05-07  17:35:21 UTC
   Image Type:   MIPS Linux Kernel Image (gzip compressed)
   Data Size:    843144 Bytes = 823.4 kB
   Load Address: 80100000
   Entry Point:  802bc040
   Verifying Checksum ... OK
   Uncompressing Kernel Image ... OK
## Loading Ramdisk Image at bfc40000 ...
```

```
   Image Name:   sentry RFS Ramdisk Image
…
…
…
sentry login: root


BusyBox v0.60.5 (2005.05.07-17:27+0000) Built-in shell (msh)
#
```

- Allow the ShMM to start up. Please note that the settings that were changed in the U-BOOT firmware will not necessarily be propagated to the Linux environment. The reason for this is that the Shelf Manager needs to maintain its own copy of the network configuration data in order to manage failover situations. If this is the first time the Shelf Manager has been booted, or if the flash devices have been reset to factory default prior to bootup, then the Shelf Manager will use the network settings provided by U-BOOT to set up this networking context (and thus the changes you made in U-BOOT will be propagated forward). If not, then the following steps are required to configure the network settings in the Shelf Manager context.

- First check to see if you are the Active Shelf Manager. You only need to make changes on the Active Shelf Manager as it will update the backup with the network configuration changes via the redundancy interface (use the cpld command and look for **active**). If you are not the active ShMM, then connect to the other ShMM device and continue to step 7:

```
# cpld
CPLD word: E806
        0002h - Local Healthy
        0004h - Switchover Request Local
        0800h - Hot Swap Latch Open
        2000h - Active
        4000h - Interrupt Status
        8000h - Reboot Was Caused By Watchdog
#
```

- Get the current IP settings:

```
# clia getlanconfig 1

Pigeon Point Shelf Manager Command Line Interpreter

Authentication Type Support: 0x15 ( None MD5 Straight Password/Key )
Authentication Type Enables:
    Callback level: 0x00
    User level: 0x15 ( "None" "MD5" "Straight Password/Key" )
    Operator level: 0x15 ( "None" "MD5" "Straight Password/Key" )
    Administrator level: 0x15 ( "None" "MD5" "Straight Password/Key" )
    OEM level: 0x00
IP Address: 206.25.139.28
IP Address Source: Static Address (Manually Configured) (0x01)
MAC Address: 00:50:c2:22:50:30
Subnet Mask: 0.0.0.0
IPv4 Header Parameters: 0x40:0x40:0x10
Primary RMCP Port Number: 0x026f
Secondary RMCP Port Number: 0x0298
BMC-generated ARP Control: 0x02
    Enable BMC-generated ARP Response
Gratuitous ARP Interval: 2.0 seconds
Default Gateway Address: 206.25.139.3
Default Gateway MAC Address: 00:00:00:00:00:00
Backup Gateway Address: 0.0.0.0
Backup Gateway MAC Address: N/A
Community String: "public"
Number of Destinations: 16
Destination Type:
    N/A
Destination Address:
    N/A
```

- Change the IP settings:

```
# clia setlanconfig 1 ip 10.1.1.10
Pigeon Point Shelf Manager Command Line Interpreter
IP set successfully
# clia setlanconfig 1 subnet_mask 255.255.0.0
Pigeon Point Shelf Manager Command Line Interpreter
Subnet Mask set successfully
# clia setlanconfig 1 dft_gw_ip 10.1.1.1
Pigeon Point Shelf Manager Command Line Interpreter
Default Gateway Address set successfully
#
```

2.2.5. *Assigning IP Addresses to the Shelf Manager via DHCP*

DHCP (Dynamic Host Configuration Protocol) and DHCP servers can be used to assign IP addresses to the Shelf Manager. The following types of IP addresses can be assigned via DHCP:

- RMCP accessible addresses (for one or both network interfaces);
- Private Shelf Manager addresses (for one or both network interfaces, for both Shelf Managers).

A total of 6 IP addresses can be assigned via DHCP. A specific IP address is designated by a particular value of the Client Identifier that is passed from the Shelf Manager to the DHCP server.

Currently the Client Identifier values are hardcoded as follows:
00:00:00:00:00:00:00:10 - Shelf Manager 1, eth0
00:00:00:00:00:00:00:11 - Shelf Manager 1, eth1
00:00:00:00:00:00:00:20 - Shelf Manager 2, eth0
00:00:00:00:00:00:00:21 - Shelf Manager 2, eth1
00:00:00:00:00:00:00:00 - Logical Shelf Manager eth0
00:00:00:00:00:00:00:01 - Logical Shelf Manager eth1

However, these Client IDs can be redefined for a specific carrier inside the corresponding ShMM carrier-specific module.

To use this feature in the Shelf Manager, it is necessary to define the configuration parameter "USE_DHCP" in the Shelf Manager configuration file /etc/shelfman.conf, as follows:

```
USE_DHCP = TRUE
```

By default, the Shelf Manager uses the first DHCP server that answers the DHCPDISCOVER request to assign the IP addresses. If more than one DHCP servers is present in the network, the configuration parameter PREFERRED_DHCP_SERVER can be used in the Shelf Manager configuration file /etc/shelfman.conf to specify the IP address of the preferred DHCP server, as follows:

```
PREFERRED_DHCP_SERVER = 192.168.1.50
```

In that case, only the preferred DHCP server will be used.

The DHCP server should be configured to provide a unique IP address for each Client Identifier. To avoid IP address expiration, the lease time of each address should be set as "infinite" (time value 0xFFFFFFFF).

The example configuration file below shows how to configure the Linux DHCP server (DHCPD) to provide IP addresses to the Shelf Manager. Fixed predefined addresses are used for that purpose. This file should be located as /etc/dhcpd.conf on the system hosting the DHCP server. Other DHCP servers (such as those on non-Linux operating systems) are configured differently.

```
allow booting;
allow bootp;

option domain-name "tst";
option subnet-mask 255.255.255.0;
option domain-name-servers 192.168.1.100;
option ntp-servers 192.168.1.50;
option routers 192.168.1.253;
option vendor-class-identifier "PPS";

min-lease-time 4294967295;
default-lease-time 4294967295;

use-host-decl-names on;
ddns-update-style ad-hoc;

subnet 192.168.1.0 netmask 255.255.255.0
{
    host client00 {
        option dhcp-client-identifier 0:0:0:0:0:0:0:10;
        fixed-address 192.168.1.140;
    }
    host client01 {
        option dhcp-client-identifier 0:0:0:0:0:0:0:11;
        fixed-address 192.168.1.141;
    }
    host client02 {
        option dhcp-client-identifier 0:0:0:0:0:0:0:20;
        fixed-address 192.168.1.142;
    }
    host client03 {
        option dhcp-client-identifier 0:0:0:0:0:0:0:21;
        fixed-address 192.168.1.143;
    }
    host client04 {
        option dhcp-client-identifier 0:0:0:0:0:0:0:0;
        fixed-address 192.168.1.144;
    }
    host client05 {
        option dhcp-client-identifier 0:0:0:0:0:0:0:1;
        fixed-address 192.168.1.145;
    }
}
```

2.3. Setting up Shelf Manager Configuration File

The Shelf Manager configuration file ("`shelfman.conf`") is located in the `/etc` directory. Each line in the file is either a comment line (starting with '#') or a name = value pair, representing the assignment for the configuration parameter. The name and the value are separated with the equal sign "=".

The configuration parameter name is case-insensitive. Each configuration parameter is one of the following types: Boolean, number, string, or IP-address.

Format of the value conforms to the type of the configuration parameter as follows:

Boolean	A Boolean can be represented by either the strings FALSE or TRUE, or by their numerical representations of 0 or 1 respectively.
Number	A whole (possibly signed) numeric value; hexadecimal notation "0x…" is also supported.
String	A string, quoted or unquoted (double quotes " " are used). Quoted strings may contain blanks; unquoted strings are terminated by the first blank. The maximum string size is specified separately for each string-oriented configuration parameter.
IP-address	The IP address in decimal-dot ("xxx.xxx.xxx.xxx") notation.

It is possible to specify a value of an environment variable as a configuration parameter value, using the notation *$envvar*; in that case, the value of the variable 'envvar' is substituted when the configuration file is read. Here is an example:

```
DEFAULT_RMCP_IP_ADDRESS = $IPADDR
```

After the shelf manager has been brought up for the first time, the IP addresses are stored with the IPMI LAN configuration parameters. The LAN configuration parameters can be accessed or modified via any of the RMCP, CLI, web, or SNMP shelf-external interfaces and take precedence over the shelfman configuration file when the shelf manager is restarted. This is to ensure the persistency of any modifications that are made to the LAN IP Addresses and gateway via those interfaces. If however, the Shelf Manager IP Connection record in the Shelf FRU Information contains an IP address, it will take precedence over all other settings of the shelf-external or RMCP IP address. We recommend that the Shelf FRU Information either not specify this address or set it to 0.0.0.0 to ensure that addresses can be controlled through the shelf manager configuration file and the IPMI LAN configuration parameters.

The following configuration parameters are currently supported:

Name	**Type**	**Default**	**Description**
2_X_SYSTEM	Boolean	None	If specified, this parameter explicitly specifies the current system as CompactPCI (if TRUE) or AdvancedTCA (if FALSE). If not specified, the choice of the system type is made automatically. It is not recommended to specify this parameter, unless it is necessary to override a wrong hardware detection algorithm for the system type.

ALARM_CUTOFF_TIMEOUT	Number	600 (seconds)	The alarm cutoff timeout (time after which the alarm cutoff is deactivated), in seconds.
ALLOW_ALL_COMMANDS_FROM_IPMB	Boolean	FALSE	If set to TRUE, most of the commands allowed from the RMCP interface are allowed from IPMB-0 as well (except for session-related commands). For example, Cold Reset and user management commands are allowed from IPMB-0.
ALLOW_CHANGE_EVENT_RECEIVER	Boolean	TRUE	If set to TRUE, the Event receiver address for the Shelf Manager can be set to an address other than 20h, LUN 0. If set to FALSE, any attempt to change event receiver address for the Shelf Manager is rejected.
ALLOW_CLEARING_CRITICAL_ALARM	Boolean	FALSE	If set to TRUE, the critical alarm condition can be cleared by the clia command “clia alarm clear”.
ALLOW_RESET_STANDALONE	Boolean	FALSE	If set to TRUE, the command Cold Reset is accepted even if the Shelf Manager does not have an available backup, and reboots the Shelf Manager. By default, the command Cold Reset is accepted only in a dual redundant configuration and causes a switchover.
ALTERNATE_CONTROLLER	Boolean	TRUE	Use alternate controller on the Shelf Manager with the address = ShMM hardware address.
AUTO_SEND_MESSAGE	Boolean	TRUE	Automatically convert an RMCP request sent to a non-Shelf Manager IPMB address into a “Send Message” request directed to that address.
CARRIER	String(16)	“PPS”	The name of the specific carrier board on which the ShMM is installed.
CARRIER_OPTIONS	String(256)	Variable	The carrier-specific options; defined separately for each supported carrier. By default, this parameter is set from the environment variable $CARRIER_OPTIONS.
CONSOLE_LOGGING_ENABLED	Boolean	FALSE	Output log messages to the console on which the Shelf Manager was started.

COOLING_FAN_DECREASE_TIMEOUT	Number	0	The minimum timeout between successive decrements of the fan speed during operation of the cooling algorithm in Normal state. Should be a multiple of COOLING_POLL_TIMEOUT; if not, it is rounded up to the next multiple. If the parameter is omitted or set to 0, this timeout is equal to COOLING_POLL_TIMEOUT.
COOLING_FAN_INCREASE_TIMEOUT	Number	0	The minimum timeout between successive increments of the fan speed during operation of the cooling algorithm in Minor Alert state. Should be a multiple of COOLING_POLL_TIMEOUT; if not, it is rounded up to the next multiple. If the parameter is omitted or set to 0, this timeout is equal to COOLING_POLL_TIMEOUT.
COOLING_IGNORE_LOCAL_CONTROL	Boolean	FALSE	Do not use local control capabilities on fan devices; Shelf Manager explicitly manages the fan level.
COOLING_POLL_TIMEOUT	Number	30 (seconds)	The maximum time (in seconds) between successive invocations of the cooling monitoring and management thread.
CPLD_ACTIVE_WORKAROUND	Boolean	TRUE	This flag applies to ShMM-300 only and indicates whether a special workaround is enabled for detecting the loss of the ACTIVE signal in the CPLD. This loss may happen on some platforms when inserting another ShMM carrier. The workaround can be turned off by setting this value to FALSE, if this problem does not exist for a specific platform; turning the workaround off may improve performance of the ShMM-300.
CTCA_FRU_RESET_TIMEOUT	Number	500 (millisecond s)	CompactPCI shelves only. The time in milliseconds during which the Shelf Manager holds the BD_SEL# line low in order to reset a CompactPCI board.

CTCA_HEALTHY_TIMEOUT	Number	0 (seconds)	CompactPCI shelves only. The time in seconds during which the Shelf Manager waits for the HEALTHY# signal to appear when powering on a CompactPCI board. If the HEALTHY# signal does not appear within the specified time, the Shelf Manager will deactivate the board. 0 (the default) stands for "infinity".
CTCA_INITIAL_FAN_LEVEL	Number	15	CompactPCI shelves only. The initial fan speed (in the range 0..15) that Shelf Manager applies to fan trays in CompactPCI shelves. 0 corresponds to the slowest, and 15 to the fastest possible speed.
DEFAULT_GATEWAY_IP_ADDRESS	IP-address	None	The default IP address used for the gateway for shelf-external (RMCP-based) communication, if the corresponding parameter is set to 0.0.0.0 in the IPMI LAN Configuration Parameters for channel 1. If a non-zero gateway IP address is provided in the LAN Configuration Parameters, the value provided in the Shelf Manager configuration file is ignored.
DEFAULT_GATEWAY_IP_ADDRESS2	IP-address	None	The default IP address used for the gateway for shelf-external (RMCP-based) communication on the second network interface, if the corresponding parameter is set to 0.0.0.0 in the IPMI LAN Configuration Parameters for channel 2. If a non-zero gateway IP address is provided in the LAN Configuration Parameters, the value provided in the Shelf Manager configuration file is ignored.

DEFAULT_RMCP_IP_ADDRESS	IP-address	None	The default IP address used for shelf-external (RMCP-based) communication; it is switched over between the redundant instances of the Shelf Manager. This IP address is used only if the corresponding parameter is set to 0.0.0.0 in the IPMI LAN Configuration Parameters for channel 1 and in the Shelf Manager IP Connection record in Shelf FRU Information. If a non-zero IP address is provided in the LAN Configuration Parameters and/or Shelf FRU Information, the value provided in the Shelf Manager configuration file is ignored.
DEFAULT_RMCP_IP_ADDRESS2	IP-address	None	The default IP address used for shelf-external (RMCP-based) communication on the second network interface; it is switched over between the redundant instances of the Shelf Manager. This IP address is used only if the corresponding parameter is set to 0.0.0.0 in the IPMI LAN Configuration Parameters for channel 2. If a non-zero IP address is provided in the LAN Configuration Parameters, the value provided in the Shelf Manager configuration file is ignored.
DEFAULT_RMCP_NETMASK	IP-address	Variable	The network mask for the network adapter used for RMCP communication. This mask is used only if the corresponding parameter is set to 0.0.0.0 in the IPMI LAN Configuration Parameters for channel 1 and in the Shelf Manager IP Connection record in Shelf FRU Information. The default value depends on the class of the default IP address used for the gateway for shelf-external (RMCP-based) communication. (see parameter DEFAULT_RMCP_IP_ADDRESS). For example, for an IP address of class C, this parameter is set to 255.255.255.0.

DEFAULT_RMCP_NETMASK2	IP-address	Variable	The network mask for the second network adapter used for RMCP communication. This mask is used only if the corresponding parameter is set to 0.0.0.0 in the IPMI LAN Configuration Parameters for channel 2. The default value depends on the class of the default IP address used for the gateway for shelf-external (RMCP-based) communication. (see parameter DEFAULT_RMCP_IP_ADDRESS2). For example, for an IP address of class C, this parameter is set to 255.255.255.0.
DEVICE_POLL_TIMEOUT	Number	10 (seconds)	The time (in seconds) between successive polls of the IPMB-0 devices by the Shelf Manager via sending the “Get Device ID” command to them.
EXIT_IF_NO_SHELF_FRU	Boolean	FALSE	If TRUE, the Shelf Manager will exit (probably resetting the ShMM) if no Shelf FRU can be found.
FAN_LEVEL_STEP_DOWN	Number	1	The number of fan steps by which the fan speed is decreased during operation of the cooling algorithm in the Normal state.
FAN_LEVEL_STEP_UP	Number	1	The number of fan steps by which the fan speed is increased during operation of the cooling algorithm in the Minor Alert state.
INITIAL_FAN_LEVEL	Number	15	This parameter specifies the initial fan level that the Shelf Manager applies to fan trays. Usually fan level values are in 0..15 range, where 0 is the slowest, and 15 is the fastest possible fan speed. This parameter has an alias CTCA_INITIAL_FAN_LEVEL for CompactPCI shelves.

INITIAL_SLOW_LINK_DELAY	Number	0	The initial delay, in seconds, before the Shelf Manager starts testing the integrity of the physical network link between the Shelf Manager and the System Manager (the RMCP link; see the description of the configuration parameter SWITCHOVER_ON_BROKEN_LINK). A non-zero delay can be used to accommodate slow network links that need significant time to initialize after shelf power up.
IPMB_ADDRESS	Number	0	The IPMB address of the Shelf Manager, overriding the hardware address. A value of 0 causes the Shelf Manager to read the hardware address from hardware and set IPMB address to hardware address * 2.
IPMB_LINK_ISOLATION_TIMEOUT	Number	-1 (seconds)	In radial shelves, if an IPMB link is disabled due to the isolation algorithm, the link is automatically enabled after this time interval (in seconds). –1 (the default) stands for "forever"
IPMB_RETRIES	Number	3	The number of attempts to send an IPMB request before finally giving up, if no response is received to this request.
IPMB_RETRY_TIMEOUT	Number	2 (seconds)	The amount of time the Shelf Manager waits for a response after sending an IPMB request, before retrying the request.
IPMB_RETRY_TIMEOUT_MSEC	Number	0	The millisecond part of the retry timeout value. If the retry timeout is less than a second, this configuration variable contains the actual timeout, while the value of the configuration variable IPMB_RETRY_TIMEOUT is 0.
LOCAL_SHELF_FRU	Boolean	TRUE	Create a local FRU#1 on the Shelf Manager that exposes the Shelf FRU Information (obtained from the file "/var/nvdata/shelf_fru_info").

M7_TIMEOUT	Number	-1 (seconds)	The maximum time (in seconds) for a FRU to stay in M7 state; after the expiration of this time the FRU is automatically transitioned to M0. –1 (the default) stands for "forever". Setting this parameter to 0 completely prevents FRUs from going into state M7.
MAX_ALERT_POLICIES	Number	64	The maximum number of PEF Alert Policies available.
MAX_ALERT_STRINGS	Number	64	The maximum number of PEF Alert Strings available.
MAX_DEFERRED_ALERTS	Number	32	The maximum number of outstanding PEF alerts.
MAX_EVENT_FILTERS	Number	64	The maximum number of PEF event filters available.
MAX_EVENT_SUBSCRIBERS	Number	64	The maximum number of entities that can simultaneously subscribe to receive event notifications from the Shelf Manager.
MAX_EVENT_SUBSCRIBER_IDLE_TIME	Number	60 (seconds)	The maximum timeout for an event subscriber, in seconds, between the moment when an event arrives and the moment when the subscriber retrieves this event from the Shelf Manager. If this timeout is exceeded, the subscriber is considered dead and is automatically unregistered.
MAX_INCOMING_IPMB_REQUESTS	Number	128	The size of the internal Shelf Manager queue for incoming IPMB requests. Incoming IPMB requests are stored in this queue before processing.
MAX_NODE_BUSY_RETRANSMISSIONS	Number	255	The maximum number of retransmissions of an IPMB command if the receiver always returns the completion code NODE BUSY in response.
MAX_OEM_FILTERS	Number	16	The maximum number of PEF OEM event filters available
MAX_PENDING_EVENT_NOTIFICATIONS	Number	1024	The maximum number of outstanding event notifications for each active subscriber.
MAX_PENDING_IPMB_REQUESTS	Number	192	The maximum number of pending IPMB requests awaiting response.

MAX_SEL_ENTRIES	Number	1024	The maximum number of entries in the System Event Log. CAUTION: This parameter *must not* be increased above the default value on ShMM-300-based Shelf Managers. Large values of this parameter can significantly degrade Shelf Manager performance.
MAX_SESSIONS	Number	32	The maximum number of simultaneous IPMI sessions.
MAX_USERS	Number	32	The maximum number of IPMI users.
MIN_FAN_LEVEL	Number	1	The minimum fan level; the cooling management code will not reduce the fan level of any fan below this value when controlling the fan level automatically
MIN_SHELF_FRUS	Number	2	The minimum number of Shelf FRUs in the shelf that the Shelf Manager must detect to start up successfully.
NORMAL_STABLE_TIME	Number	3600 (seconds)	The time in seconds for which the Shelf Manager preserves the minimum fan level dynamically found in Normal mode (that is, the minimum fan level that does not cause thermal alerts). After this time expires, the cooling algorithm decreases the minimum fan level, if possible, to allow the shelf to decrease the fan level if the thermal load in it has also decreased.
PHYSICAL_SENSORS	Boolean	TRUE	Create IPMI sensors based on physical sensors hosted by ADM1026 and LM75.
POWER_UNLISTED_FRUS	Boolean	TRUE	Allow the FRUs not listed in the power management table in the Shelf FRU Information to be activated and powered up.
PREFERRED_DHCP_SERVER	IP-address	None	This parameter is the IP address of the preferred DHCP server; applies only if USE_DHCP is set. If this parameter is omitted or set to 0, the Shelf Manager accepts address information from any DHCP server that responds to the broadcast discovery request.

PROPAGATE_RMCP_ADDRESS	Boolean	FALSE	If TRUE, the active Shelf Manager propagates the RMCP IP address to the backup Shelf Manager, which configures the network interface specified by the RMCP_NET_ADAPTER variable using that IP address, but with the least significant bit inverted.
REDUNDANCY_ENABLED	Boolean	TRUE	Run the Shelf Manager in redundant mode.
REDUNDANCY_NET_ADAPTER	String(16)	“” (undefined)	The name of the network adapter used for communication between redundant instances of the Shelf Manager. “usb0” is the recommended value if USB network interfaces are used for redundancy.
REDUNDANCY_NET_ADAPTER2	String(16)	“” (undefined)	The name of the second network adapter used for communication between redundant instances of the Shelf Manager (if the dual USB network interface is used for this purpose). “usb1” is the recommended value if USB network interfaces are used for redundancy.
REDUNDANCY_NETMASK	IP-address	0.0.0.0	The netmask to assign to the redundancy IP address; by default (if 0), the netmask is determined automatically from the class of the IP address. “255.255.255.128” is the recommended value if USB network interfaces are used for redundancy.
REDUNDANCY_PORT	Number	1040	The TCP port used for interactions between redundant instances of the Shelf Manager.
REDUNDANT_IP_ADDRESS	IP-address	None	The IP address used for redundant communications. This address actually specifies a pair of IP addresses that differ only in the least significant bit. They are assigned to redundant Shelf Managers according to their hardware addresses.
RESERVATION_RETRIES	Number	10	The maximum number of times the Shelf Manager retries the “Reserve Device SDR” command.
RMCP_NET_ADAPTER	String(16)	“eth0”	The name of the network adapter used for RMCP-based communication.

RMCP_NET_ADAPTER2	String(16)	None	The name of the alternate network adapter used for RMCP-based communications, if cross-connect links are supported by the hardware. "eth1" is the recommended value if the parameter is specified.
SDR_ READ_RETRIES	Number	3	The maximum number of times the Shelf Manager retries the "Read Device SDR" command.
SEL_HIGH_WATERMARK	Number	0	This value is the "high watermark" for the algorithm that controls automatic purging of the SEL; if the actual percentage of free entries in the SEL falls below this value, or the SEL overflows, the Shelf Manager starts a thread that purges old records from the SEL in order of decreasing age.
SEL_LOW_WATERMARK	Number	0	This value is the "low watermark" for the algorithm that controls automatic purging of the SEL; if the thread that purges old records from the SEL starts, it will purge records until the percentage of occupied entries in the SEL falls below this value.
SENSOR_POLL_INTERVAL	Number	1 (seconds)	The time (in seconds) between successive polls of local Shelf Manager sensors by the Shelf Manager.
SHELF_FRU_IN_EEPROM	Boolean	TRUE	If TRUE, the Shelf FRU Information will be retrieved from EEPROMs on the backplane in a carrier-specific way; if FALSE, the Shelf FRU will be obtained from a file on the flash file system.
SHELF_FRU_IPMB_SOURCE1	Number	0	If defined (non-zero), specifies the IPMB address of the first designated source of Shelf FRU Information in the shelf. (Shelf FRU is located at FRU 1.) If this value is defined, the search for the Shelf FRU on the IPMB is limited to the designated sources only.

SHELF_FRU_IPMB_SOURCE2	Number	0	If defined (non-zero), specifies the IPMB address of the second designated source of Shelf FRU Information in the shelf. (Shelf FRU is located at FRU 1.) If this value is defined, the search for the Shelf FRU on the IPMB is limited to the designated sources only.
SHELF_FRU_TIMEOUT	Number	5 (seconds)	The time interval during initialization that the Shelf Manager waits for Shelf FRU Information devices to be detected.
SHORT_SEND_MSG_RESPONSE	Boolean	TRUE	Determines the type of the Send Message response provided by the Shelf Manager: required by the PICMG 3.0 ECR (if TRUE) or compatible with the previous versions of the Shelf Manager (if FALSE).
SWITCHOVER_ON_HANDLE_OPEN	Boolean	FALSE	If TRUE, switchover-related behavior of the Shelf Manager is affected by the state of its Hot Swap handle, as follows: - If the active Shelf Manager goes to the state M1 due to its Hot Swap Handle being open, a switchover to the backup Shelf Manager is initiated; - If the active Shelf Manager goes to the state M5 and there is no available backup Shelf Manager, the active Shelf Manager is not deactivated and stays in M5 indefinitely.
SWITCHOVER_TIMEOUT_ON_BROKEN_LINK	Number	-1(seconds)	This parameter affects when or whether the Shelf Manager initiates a switchover when the physical network link between the Shelf Manager and the System Manager (the RMCP link) is broken. If the link remains broken for at least the number of seconds given in this parameter, a switchover takes place; if the link is restored during this timeout period, no switchover takes place. If the value of this parameter is -1, no automatic switchovers take place on broken RMCP links.
SYSLOG_LOGGING_ENABLED	Boolean	TRUE	Output log messages to the system log.

SYSTEM_MANAGER_TRUNCATES_SEL	Boolean	FALSE	If TRUE, the Shelf Manager algorithm for truncating the SEL automatically is disabled; the System Manager is responsible for truncating the SEL by monitoring the value of the sensor "SEL State" of the type "Event Logging Disabled" on the Shelf Manager, and removing events from the SEL by sending the "Delete SEL Entry" command to the Shelf Manager.
TASKLET_RETRIES	Number	3	The number of times each Shelf manager tasklet (activation, deactivation, getting information) retries before finally giving up.
USE_DHCP	Boolean	FALSE	Requests assignment of RMCP-accessible and private IP addresses for the Shelf Manager from a DHCP server; the configuration parameter PREFERRED_DHCP_SERVER can be used to specify the IP address of the preferred DHCP server.
USE_SECOND_CHANNEL	Boolean	FALSE	This parameter applies only if two network interfaces on the ShMM are used for RMCP communication. If TRUE, the two network interfaces on the ShMM are used in parallel mode; if FALSE, they are used in redundant mode.
VERBOSITY	Number	7	The Shelf Manager verbosity level.
VERIFY_SHELF_FRU_CHECKSUM	Boolean	TRUE	Enable verification of checksums in Shelf FRU Information records; if set to FALSE, Shelf Manager will ignore checksums
WATCHDOG_ENABLED	Boolean	TRUE	Use the hardware watchdog timer supported by the CPLD.

By default, the Configuration file variables will be used automatically when the ShMM is brought up for the first time. The default configuration file imports several environment variables set by U-Boot. They are:

```
$IPADDR     -     Default RMCP IP Address
$IPDEVICE   -     Default RMCP network adapter
$IP1ADDR    -     Default Redundant IP Address
$IP1DEVICE  -     Default Redundant network adapter
$GATEWAY    -     Default gateway used for RMCP communication
```

The environment variables $CARRIER and $CARRIER_OPTIONS are set by the secondary RC script. The name of this carrier-specific startup script is defined by either the U-Boot or the ARMBoot environment variable rc2.

The shelf manager can be reset to factory default parameter values if needed. Please refer to section 7.2 for further details.

2.3.1. Verbosity Level Description

The verbosity level allows for additional output to be sent to either the console or to the Syslog depending on how the configuration parameters CONSOLE_LOGGING_ENABLED and SYSLOG_LOGGING_ENABLED are set. The VERBOSITY configuration parameter is a hexadecimal bit mask, each bit enabling output of a specific type of message:

Verbosity Level

0x01	Error messages
0x02	Warning messages
0x04	Informational messages
0x08	Verbose informational messages
0x10	Trace messages
0x20	Verbose trace messages
0x40	Messages displayed for important commands sent to the IPM Controllers during their initialization
0x80	Verbose messages about acquiring and releasing internal locks

The default debug level is 7, allowing error, warning and informational messages to appear.

2.3.2. Configuring the FRU Information

2.3.2.1. Accessing the Shelf FRU Information

According to the ATCA specification, the Shelf FRU Information should be redundant (at least two copies per shelf) and each copy may be represented by separate IPM controllers, as FRU #1. Some ATCA shelves adopt this approach fully. For such shelves, the default configuration file must be changed : the variable LOCAL_SHELF_FRU must be set to FALSE. This will enable a shelf-wide search for potential sources of Shelf FRU Information on IPMB-0.

On most ATCA shelves implementing this approach, the Shelf FRU Information is accessed via two IPM controllers with well-known IPMB addresses. In this case, it is possible to limit the search for Shelf FRU Information to the two well-known IPMB locations. To do this, two configuration variables SHELF_FRU_IPMB_SOURCE1 and SHELF_FRU_IPMB_SOURCE2 must be defined in the configuration file /etc/shelfman.conf. These variables are of the Number type and contain the IPMB addresses of the two designated IPM controllers that represent Shelf FRU Information. For example, to limit the search for the Shelf FRU information to IPM controllers at 66h and 68h respectively, these variables should be defined as follows (note the use of "0x" prefix for hexadecimal addresses):

- SHELF_FRU_IPMB_SOURCE1 = 0x66
- SHELF_FRU_IPMB_SOURCE2 = 0x68

However, most known ShMM-based shelves provide two SEEPROMs that are connected to the Shelf Manager via the master-only I^{2}C interface, usually with each SEEPROM residing on its own bus behind an

I^{2}C multiplexer. Each of these two SEEPROMs store a copy of the Shelf FRU Information, providing the needed redundancy. Some shelves do not provide even this type of storage; in that case it is possible to store the Shelf FRU Information on the ShMM itself, as a single flash file "/var/nvdata/shelf_fru_info". The redundant Shelf Managers each have their own copy of that flash file, and synchronize them using the redundancy protocol, so even it that case some degree of redundancy is still preserved.

The Shelf Manager configuration (as represented in shelfman.conf) must be aligned with the mechanisms for accessing Shelf FRU Information that are provided by the shelf. The default configuration supports the approach where redundant Shelf FRUs are represented by separate IPM controllers as FRU #1 as well as the "two SEEPROMs" approach. The following key configuration variables are set as follows for that case:

- LOCAL_SHELF_FRU = TRUE
- SHELF_FRU_IN_EEPROM = TRUE

In this case, however, support for SEEPROMs must also be provided by the carrier-specific module in the Shelf Manager.

If none of the above approaches is supported by the shelf, and there are no sources of Shelf FRU Information represented by separate IPM controllers, the system integrator must resort to the flash file as storage for Shelf FRU Information. In that case, the following changes to the default configuration should be done:

- set the variable SHELF_FRU_IN_EEPROM to FALSE
- set the variable MIN_SHELF_FRUS to 1.

The last change is necessary because there will be only one copy of the Shelf FRU Information on each ShMM. The variable LOCAL_SHELF_FRU must retain its default value of TRUE.

The following table summarizes the configuration variable settings that correspond to the various Shelf FRU Information source possibilities described above.

Source of the Shelf FRU Information	**Settings of Configuration Variables**

Non-volatile storage (likely SEEPROMs), accessed via IPM controllers on IPMB-0	LOCAL_SHELF_FRU = FALSE
	SHELF_FRU_IN_EEPROM = does not matter
	MIN_SHELF_FRUS = minimum number of IPM controllers on IPMB-0 providing the Shelf FRU Information (usually 2)
	Optionally:
	SHELF_FRU_IPMB_SOURCE1 = IPMB address of the first designated source
	SHELF_FRU_IPMB_SOURCE2 = IPMB address of the second designated source

SEEPROMs, accessed locally by the	LOCAL_SHELF_FRU = TRUE

Shelf Manager	SHELF_FRU_IN_EEPROM = TRUE MIN_SHELF_FRUS = the number of SEEPROMs providing the Shelf FRU Information (usually 2)

Flash file	LOCAL_SHELF_FRU = TRUE
	SHELF_FRU_IN_EEPROM = FALSE
	MIN_SHELF_FRUS = 1

2.3.2.2. Setting up the Shelf FRU Information

Since the contents of the Shelf FRU Information is crucial for successful management of the shelf, it is necessary to set up the Shelf FRU Information on a fresh shelf before starting the Shelf Manager on it. This procedure consists of the following steps:

- Creating a description of the shelf in a formalized text format ("INF format").
- Compiling the text description using the FRU Information Compiler
- Placing the binary image of the FRU Information into the appropriate storage.

The first two steps are documented separately in the user manual for the FRU Information Compiler. The last step is documented here and depends on where the Shelf FRU Information is stored.

The simplest case is if the Shelf FRU Information is stored on a flash file on the ShMM.. In that case, the binary image file should be downloaded on the ShMM via FTP and copied to the location "/var/nvdata/shelf_fru_info".

The following log represents an example of the above process. It should be performed on an x86 Linux machine (NOT on the ShMM, itself!).

```
# mv shelf_fru.bin shelf_fru_info
# ftp 192.168.1.230
Connected to 192.168.1.230.
220 shmm-230 FTP server (Version wu-2.6.2(1) Sun Dec 15 17:40:37 GMT 2002)
ready.
Name (192.168.1.230:serjio): ftp
331 Guest login ok, send your complete e-mail address as password.
Password:
230 Guest login ok, access restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd /var/nvdata
250 CWD command successful.
ftp> del shelf_fru_info
250 DELE command successful.
ftp> put shelf_fru_info
local: shelf_fru_info remote: shelf_fru_info
227 Entering Passive Mode (192,168,1,230,107,162)
150 Opening BINARY mode data connection for shelf_fru_info.
226 Transfer complete.
129 bytes sent in 5.1e-04 secs (2.3e+02 Kbytes/sec)
```

```
ftp> quit
221-You have transferred 129 bytes in 1 files.
221-Total traffic for this session was 640 bytes in 1 transfers.
221-Thank you for using the FTP service on shmm-230.
221 Goodbye.
#
```

Another case is when the Shelf FRU Information is stored in SEEPROMs that are accessible to the Shelf Manager via the master-only I^{2}C bus. In that case, the binary file should be copied onto the ShMM and then written into the SEEPROMs. The utility "eepromw" can be used for that purpose.

The exact location of SEEPROMs on the master I2C bus is carrier-specific, but typically they are located at address 0xA4 on channels 1 and 2 of the I2C multiplexer, and the multiplexer itself resides at address 0xE0. In that case, the following commands (issued on the ShMM!) can be used to download the file "/var/nvdata/shelf_fru_info" to the SEEPROMs. It is assumed that the file was compiled by the FRU Information Compiler and downloaded onto the ShMM, as in the previous example:

```
#eepromw –c 1 A4 /var/nvdata/shelf_fru_info
#eepromw –c 2 A4 /var/nvdata/shelf_fru_info
```

The general syntax for the "eepromw" utility is as follows:

```
eepromw [-b <multiplexer>] [–c <channel>] <eeprom-address> <file>
```

where:

- <multiplexer> - the address of the I2C multiplexer on the I2C bus (default=0xE0)
- <channel> - the channel on the multiplexer to use (default=0)
- <eeprom-address> - the address of the target SEEPROM
- <file> - path to the file to write onto the target SEEPROM

The reciprocal "eepromr" utility allows the user to read the contents of an SEEPROM into a file on the flash, and has the following parameters:

```
eepromr [-b <multiplexer>] [–c <channel>] <eeprom-address> <file> <count>
```

where:

- <file> - path to the file where data are written from SEEPROM
- <count> - how many bytes to read

The remaining parameters are the same as for "eepromw".

The remaining case, where the Shelf FRU Information resides on separate IPM controllers inside the shelf, is completely shelf-specific and is beyond the scope of this document.

2.3.2.3. Other FRU Information Repositories

The Shelf Manager itself exposes at least one IPM controller (the ShMC at IPMB address 20h). For most carriers, the Shelf Manager also exposes a "physical" IPM controller that represents the resources of the carrier board and has an IPMB address derived from the physical address of the carrier. While the ShMC is exposed only by the active Shelf Manager and is subject to switchover, the "physical" IPM controller is exposed separately by both active and backup Shelf Managers.

For both of these IPM controllers, FRU Information is stored in flash files on the ShMM:

- "./var/nvdata/bmc-fru-information" for the ShMC
- "/var/nvdata/shelfman-fru-information" for the "physical" IPM controller

For some carriers, there may be additional FRUs represented by the ShMC. The location of the FRU Information for these FRUs is carrier-specific.

Reading and writing these FRU Information repositories can be done via the IPMI commands "Read FRU Data", "Write FRU Data", addressed to the appropriate FRUs.

2.4. Configuring Local Sensors

Local sensors on the ShMM can be configured when the ShMM is started. (This capability applies to sensors that are associated with either: 1) the active Shelf Manager or 2) the physical IPM controller that takes its IPMB-0 address from the hardware address of the ShMM carrier slot.) Only sensor attributes that are defined in Sensor Data Records (such as thresholds, hysteresis values, sensor name, linearization parameters, etc.) can be configured at this time. The Sensor Device Records (SDRs) defining these sensors are read from the file "/var/nvdata/user_sdr". This file must contain an array of binary SDRs that are compliant with the IPMI specification. However, these SDRs can contain only partial sensor definition, if only a subset of the attributes of the sensor need to be redefined (see below).

A PPS-supplied SDR compiler utility can be used to produce the binary SDRs from plain text human-readable text files. The SDR compiler can also decode binary SDR data and produce human-readable text file from it. This utility is described in the SDR Compiler User Guide. The current version of the SDR compiler (as distributed with release 2.2 or later of the Shelf Manager) must be used.

In order to take advantage of this sensor configuration facility, you should install the SDR Compiler on either a Linux or an MS Windows system. Then you should create an SDR definition text file according to the format described in SDR Compiler User Guide. You may use standard text editors such as *vi* for Linux or *Notepad* for MS Windows. The standard extensions for SDR definition text files and SDR definition binary files are ".inf" and "bin" respectively. The SDR Compiler is a command-line utility. To produce a binary SDR definition file from a text SDR definition file you should use the utility in the compilation mode. For example,

```
>python sdrc.py test.inf
```

The SDR compiler is written in Python, so you will also need a Python interpreter (version 2.3 or later) to run it. It is available for free downloading, with support for both Windows and Linux at www.python.org.

The SDR compiler creates a binary SDR definition file `test.bin` from the text SDR definition file `test.inf`. You should rename the file `test.bin` as `user_sdr.` You may explicitly specify the name for the binary SDR definition file as a command-line argument. For example,

```
>python sdrc.py test.inf user_sdr
```

After producing the binary SDR definition file, you should place it on your ShMM in the directory "/var/nvdata" under the name "user_sdr". For example, you may use FTP:

```
>ftp 192.168.191
```

```
Connected to 192.168.1.191 (192.168.1.191).
220 shmm+191 FTP server (Version wu-2.6.2(1) Wed Oct 5 21:30:04 GMT 2005)
ready.
Name (192.168.1.191:username):anonymous
331 Guest login ok, send your complete e-mail address as password.
Password:
230 Guest login ok, access restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
 ftp> cd /var/nvdata
250 CWD command successful.
ftp> put user_sdr
local: user_sdr remote: user_sdr
227 Entering Passive Mode (192,168,1,191,246,195)
150 Opening BINARY mode data connection for user_sdr.
226 Transfer complete.
124 bytes sent in 8.5e-05 secs (7.6e+03 Kbytes/sec
ftp> quit
221-You have transferred 124 bytes in 1 files.
221-Total traffic for this session was 1165 bytes in 1 transfers.
221-Thank you for using the FTP service on shmm+191.
221 Goodbye.
```

A newly installed SDR definition file "/var/nvdata/user_sdr" takes effect only after a restart of the Shelf Manager; the Shelf Manager log during that restart could look like the following example:

```
daemon -f shelfman -lcs -cs -sf- eth0 1F
# <I> 03:45:09.041   [152] Pigeon Point Shelf Manager ver. 2.3.0. Built on May
23 2006 11:03:27
<*> 03:45:09.055   [152] Limits: code=(10af4040:10ba95c0), end_data=10c355d4,
start_stack=10c36e64, esp=10c36874, eip=10ba1838
<*> 03:45:09.063   [152] Stack limits: curr=800000, max=fffffff
<*> 03:45:09.067   [152] Data limits: curr=fffffff, max=fffffff
<*> 03:45:09.090   [152] *** Lock log print buffer at 10c10e90 ***
<*> 03:45:09.094   [152] *** Pthread lock log print buffer at 10c14ed0 ***
<I> 03:45:09.380   [152] Enabling the CPLD Active bit workaround
<I> 03:45:09.435   [152] User SDR size = 124
<I> 03:45:09.453   [152] 20 # 120, Type = 1
<I> 03:45:09.467   [152] a0 # 2, Type = 1
…
```

If the SDR definition file "/var/nvdata/user_sdr" is not present, the configurations of the local sensors are unmodified from the default established in the Shelf Manager, and the log looks like the following example:

```
# daemon -f shelfman -lcs -cs -sf- eth0 1F
# <I> 17:11:28.004   [522] Pigeon Point Shelf Manager ver. 2.3.0. Built on May
23 2006 19:03:33
<*> 17:11:28.009   [522] Limits: code=(400000:529030), end_data=10062000,
start_stack=7fff7df0, esp=7fff7758, eip=2ab0d2e4
<*> 17:11:28.010   [522] Stack limits: curr=1ff000, max=7fffffff
<*> 17:11:28.010   [522] Data limits: curr=7fffffff, max=7fffffff
<*> 17:11:28.014   [522] *** Lock log print buffer at 1003c910 ***
<*> 17:11:28.014   [522] *** Pthread lock log print buffer at 10040940 ***
<W> 17:11:28.027   [522] Custom SDR initialization file is absent
…
```

It is important to understand that replacement SDRs must be closely coordinated with SDRs that are defined within the Shelf Manager. Therefore, additions or modifications to the set of replacement SDRs in a shelf should only be undertaken in close cooperation with the shelf supplier.

The following rules apply to SDRs that are used to configure local sensors (referenced as replacement SDRs, below):

1. Every replacement SDR must be a Full Sensor Record (type 01). Compact Sensor Records (type 02) are not supported and are never used by the Shelf Manager for local sensors.

2. The following fields are mandatory in every replacement SDR (note that this list is smaller than the list of mandatory fields for normal SDRs processed by the SDR compiler):

   - *Sensor Owner ID*
   - *Sensor Number*
   - *Sensor Initialization*

3. If a field from a replacement SDR is used (as a result of the operation of other rules) to replace an attribute for a target sensor, and the field is not specified in the replacement SDR, a value zero (0) is used for this field.

4. The tuple (*Sensor Owner ID*, *Sensor Number*) identifies the sensor that is to be configured. The *Sensor Owner ID* is a literal IPMB address; so for sensors on the physical IPM controller two instances of the corresponding replacement SDR must be present, one each for the IPMB addresses associated with each of the two redundant dedicated ShMC slots.

5. The fields *Entity ID* and *Entity Instance*, if specified in the replacement SDRs and different from (0,0), replace the corresponding attributes of the sensor.

6. The *Sensor Initialization* field identifies the specific attributes to be redefined and indicates what fields in the rest of the replacement SDR are applicable. It is specified as a list of symbolic constants; the following constants are defined:

   - THRESHOLDS: indicates that the sensor thresholds are to be replaced with the corresponding threshold values specified in the replacement SDR

   - HYSTERESIS: indicates that the sensor hysteresis values are to be replaced with the corresponding hysteresis values specified in the replacement SDR

   - SENSOR_TYPE: indicates that the sensor type is to be replaced with the corresponding fields *Sensor Type* and *Event/Reading Type* from the replacement SDR

   - EVENTS: indicates that the event masks are to be replaced with the corresponding fields from the replacement SDR

7. The field *Sensor Capabilities* cannot be replaced.

8. If and only if the symbolic constant SENSOR_TYPE has been specified in the *Sensor Initialization* field, the fields *Sensor Type* and *Event/Reading Type* from the replacement SDR replace the corresponding attribute of the sensor (even if the values of these fields are 0).

9. If and only if the symbolic constant EVENTS has been specified in the Sensor Initialization field, the following fields from the replacement SDR replace the corresponding attributes of the target sensor:

   - For Threshold-Based sensors:
     - *Lower Threshold Reading Mask*
     - *Upper Threshold Reading Mask*
     - *Threshold Assertion Event Mask*
     - *Threshold Deassertion Event Mask*
     - *Settable Threshold Mask*
     - *Readable Threshold Mask*

   - For Discrete sensors:
     - *Assertion Event Mask*
     - *Deassertion Event Mask*
     - *Discrete Reading Mask*

10. The fields *Sensor Units 1*, *Base Unit*, *Modifier Unit* cannot currently be replaced.

11. The following sensor attributes are replaced from the replacement SDR fields, if at least one of these fields is specified in the replacement SDR with a non-zero value:
    - *Linearization*
    - *M*
    - *Tolerance*
    - *B*
    - *Accuracy*
    - *Accuracy exp*
    - *R exp*
    - *B exp*
    - *Analog Characteristic Flags*
    - *Nominal Reading*
    - *Normal Maximum*
    - *Normal Minimum*
    - *Sensor Maximum Reading*
    - *Sensor Minimum Reading*

12. If and only if the symbolic constant THRESHOLDS has been specified in the *Sensor Initialization* field, the fields listed below from replacement SDRs specify replacement threshold values for the target sensor. Note, however, that only thresholds supported by the sensor implementation can be redefined. (Some local sensors do not support all possible thresholds.) Here is a list of replaceable threshold types, all of which must be specified in the raw format (with the "0x" prefix):
    - *Upper Non-Recoverable Threshold*
    - *Upper Critical Threshold*

- *Lower Non-Critical Threshold*
- *Lower Non-Recoverable Threshold*
- *Lower Critical Threshold*
- *Lower Non-Critical Threshold*

If and only if the symbolic constant HYSTERESIS has been specified in the *Sensor Initialization* field, the following fields from the replacement SDRs specify the replacement hysteresis values for the target sensor:
- *Positive Hysteresis*
- *Negative Hysteresis*

Hysteresis values must be specified in the raw format (with the "0x" prefix).

13. The field *OEM* from the replacement SDR always replaces the corresponding attribute of the target sensor (even if not specified).

14. The field *Id String* replaces the target sensor name only if specified in the replacement SDR.

The example below illustrates a typical local sensor configuration text definition. It redefines thresholds for the two temperature sensors on the physical IPM controller. It is assumed that the physical IPM controllers have IPMB-0 addresses 10h and 12h in the target shelf.

```
[Full Sensor Record]
Owner Id = 0x10
Sensor Number = 2
Sensor Initialization = THRESHOLDS
Lower Non-Critical Threshold = 0xb0
Lower Critical Threshold = 0xc0
Lower Non-Recoverable Threshold = 0xd0
Upper Non-Critical Threshold = 0x40
Upper Critical Threshold = 0x48
Upper Non-Recoverable Threshold = 0x50

[Full Sensor Record]
Owner Id = 0x10
Sensor Number = 3
Sensor Initialization = THRESHOLDS
Lower Non-Critical Threshold = 0xb0
Lower Critical Threshold = 0xc0
Lower Non-Recoverable Threshold = 0xd0
Upper Non-Critical Threshold = 0x40
Upper Critical Threshold = 0x48
Upper Non-Recoverable Threshold = 0x50

[Full Sensor Record]
Owner Id = 0x12
Sensor Number = 2
Sensor Initialization = THRESHOLDS
Lower Non-Critical Threshold = 0xb0
Lower Critical Threshold = 0xc0
Lower Non-Recoverable Threshold = 0xd0
Upper Non-Critical Threshold = 0x40
Upper Critical Threshold = 0x48
Upper Non-Recoverable Threshold = 0x50

[Full Sensor Record]
Owner Id = 0x12
Sensor Number = 3
Sensor Initialization = THRESHOLDS
Lower Non-Critical Threshold = 0xb0
Lower Critical Threshold = 0xc0
Lower Non-Recoverable Threshold = 0xd0
Upper Non-Critical Threshold = 0x40
Upper Critical Threshold = 0x48
Upper Non-Recoverable Threshold = 0x50
```

2.5. Setting Auxiliary Firmware Revision

The Auxiliary Firmware Revision can be set when the Shelf Manager is started. The Auxiliary Firmware Revision is reported by Get Device Id command targeted to a physical ShMC (at the hardware-specified IPMB-0 address, versus the logical Shelf Manager at IPMB-0 address x20) and is stored in a single flash file "/var/nvdata/aux-fw-revision". If the file "/var/nvdata/aux-fw-revision" is absent, the Auxiliary Firmware Revision is not defined.

According to IPMB v2.0 R1.0, Section 20.1 "Get Device Id Command," the Auxiliary Firmware Revision is a 4-byte data item. The file "/var/nvdata/aux-fw-revision" should contain the eight hexadecimals that represent the 4 bytes in question. No separators are allowed. The first two hexadecimal digits represent the most significant byte of the Auxiliary Firmware Revision.

For example, assume that the file "/var/nvdata/aux-fw-revision" contain the string 'a0b1efcd'. When the Shelf Manager is started and the RMCPTA connection is established, it is possible to obtain the Auxiliary Firmware Revision via RMCP. The parameter `<IPMB-address>` represents the IPMB-address of the alternative controller. Here is a dialogue with the Pigeon Point internal tool RMCPTA, but any RMCP client can be used to make this query:

```
RMCPTA{1}-> TargetFwd <IPMB-address>
RMCPTA{1}-> GetDeviceId
 Completion Code = 0x00 (OK)
 Device ID       = 0x00
 Device Revision = 0x0
 Device Mode     = normal operation ; Device SDR present
 Firmware Rev.   = 2.30
 IPMI Version    = 1.5
 Device Support  = IPMB Req.Gen; FRU; Sensor;
 Manufacturer ID = 0x0400A
 Product ID      = 0x0000
 AUX FW Rev.     = 0xA0B1EFCD (A0 B1 EF CD)
```

2.6. Setting Up the Clock

When the system is brought up for the first time, the clock will not be set and must be initialized. Initially the clock will be set to January 1, 1970. The date can be accessed via the serial console.

```
    # date
    Thu Jan  1 03:16:30 UTC 1970
```

In order to change the date, you should type in the correct date using the `date` application. The format for the date command is MMDDHHMMSSYYYY, where:

```
      MM    -     Month
      DD    -     Day
      HH    -     Hour (use 24 hour notation)
      MM    -     Minute
      SS    -     Second
      YYYY  -     Year
```

For example:

```
    # date 04291628002003
    Tue Apr 29 16:28:00 UTC 2003
```

To make the date persistent, you need to store it using the `hwclock` application.

```
    # hwclock --systohc
```

In some cases, you might get the error message:

```
    mktime: cannot convert RTC time to UNIX time
```

This error can be ignored. It is due to the original date being in an uninitialized state.

2.6.1. Obtaining Date and Time from a Time Server

If the ShMM carrier does not have an RTC battery, it is possible to obtain the system date and time from a time server during system startup and synchronize it periodically thereafter. There are two network time protocols that can be used for that purpose: NTP and RFC 868 (rdate). The specific protocol to be used is selected when configuring the ShMM.

To enable obtaining the network time via the NTP protocol, it is necessary to define the U-Boot variables "time_proto", "time_server" and optionally the additional variable "timezone". The variable "time_proto" determines the adjust time protocol (if this variable is undefined, by default the RFC 868 (rdate) protocol is used). This variable should be set to "ntp" to enable the NTP protocol. The usage of the other variables is identical their usage with RFC 868 (rdate) and is described below.

To enable obtaining the network time via the RFC 868 protocol (rdate) over TCP, it is necessary to define the U-Boot variable "time_server" and optionally the additional variable "timezone". The variable "time_proto" should be left undefined or set to "rdate".

The variable "time_server" contains the IP address of the time server that the Shelf Manager queries for the system time after the startup. This server should support RFC 868 over TCP as required by the "rdate" utility or support NTP as required by the "ntpdate" utility. This variable is propagated to the Linux level as the environment variable TIMESERVER. If this variable is set, the startup script /etc/netconfig starts the script /etc/timesync as a daemon, which runs in an endless loop and queries the time server with a default interval of 300 seconds. To change this interval, edit the script /etc/timesync and change the value of the variable INTERVAL. The TIMESERVER variable can be changed by Shelf Manager if the ntp-server option is received by DHCP. In this case the Shelf Manager overrides the /tmp/timeserverip file which is used by the script /etc/timesync to define the TIMESERVER variable.

The variable "timezone" contains the name of the current time zone followed by its offset from Greenwich Meridian Time (GMT). The offset is positive for time zones to the west of Greenwich and negative for time zones to the east of Greenwich. This variable is propagated to the Linux level as the environment variable TZ. The default value of this variable is UTC0, i.e. Universal Coordinated Time which matches Greenwich time.

The time sent by time servers is GMT time; if the time zone on the Shelf Manager is not set or not set correctly, the time obtained from the time server will be interpreted incorrectly. The three-letter name of the time zone is not used by the Shelf Manager, but is propagated to set the Linux time zone. (For instance, if the time zone name XXX0 is used, the date command will produce output like the following: "Thu Sep 9 21:24:24 XXX 2004".) Daylight saving time is not supported.

Here is an example of a time zone definition for US Eastern Time:

```
timezone = EST5
```

Here the digit 5 specifies that the time zone is 5 hours west of GMT. Any three letters can replace "EST"; they are used to identify the time zone in (for example) the Linux date command output.

2.7. Setting Up and Using ShMM-500 Power On Self Tests

The available Power On Self Test (POST) tests are built into U-Boot. The choice of the available POST tests that are actually executed is controlled by the dedicated U-Boot environment variables "post_normal" and "post_poweron".

The value of the environment variable "post_normal" contains names of tests that are executed on each boot-up. These names are separated by space characters. These tests do not take much time and can be run on a regular basis.

The value of the environment variable "post_poweron" contains names of tests that are executed after power-on reset only (vs. on each boot-up). These names are separated by space characters.

As the POST tests are executed, the results are logged in a textual form in a dedicated area in SDRAM. Results for each particular test have the following form:

```
<4>POST <test name> [<test-specific output>] [PASSED|FAILED]
```

The POST framework provides interfaces for accessing the results of the POST tests in U-Boot and in Linux.

U-Boot supports a `log show` command, which can be used to access the POST test results. This command outputs the contents of the POST log buffer onto the serial console.

```
shmm500 log show
<4>POST uart UART 0 test failed
<4>FAILED
<4>POST crc PASSED
```

In a POST-enabled configuration, the Linux kernel shares its internal message log buffer with the POST log buffer. This causes POST results to be automatically displayed on the serial console during the kernel bootstrap. Interactively, the user can access the kernel log buffer (and thus the POST test results) using the `dmesg` command implemented by `busybox`.

The defined names for POST tests are:

- "memory" (SDRAM tests, recommended for execution on the first boot-up after power-on)
- "crc" (Flash checksum verification, recommended for execution on each boot-up)
- "uart" (UART verification, specific to the Au1550 processor, recommended for execution on each boot-up)
- "ethernet" (Ethernet MACs test, specific to the Au1550 processor, recommended for execution on each boot-up)
- "i2c" (Master-Only I^2C test, recommended for execution on each boot-up)

The names of the tests can be used in values of the "post_poweron" and "post_normal" environment variables.

Chapter

3

3. Using the Shelf Manager

This chapter introduces the overall operation of the Shelf Manager, including operation in a redundant (active/standby) pair.

3.1. ShMM Login

Once the ShMM has been fully booted, you are prompted to login. You can log in as user **root**. With the factory defaults, no password will be requested. We highly recommend that you change passwords during the configuration of the ShMM. The password can be reset to factory defaults if the password is ever forgotten (see Section 5.3 for the ShMM-500 and Section 7.3 for the ShMM-300). Here is a typical log in session for ShMM-500 (on ShMM-300, it will look very similar):

```
demo login: root

BusyBox v0.60.5 (2005.05.07-17:27+0000) Built-in shell (msh)
# ls
bin             dev             etc             lib             mnt
proc            sbin            tmp             usr             var
##
```

3.2. Starting Shelf Manager

The Shelf Manager software is implemented in the executable file "shelfman" in the directory "/bin". During normal use of the Shelf Manager it is invoked automatically by startup scripts. Typical users will never need to invoke it manually as described here. Nevertheless, for the unusual circumstances in which manual invocation may be necessary, the details are described here.

The syntax of a Shelf Manager invocation command line is defined as follows:

```
shelfman [<options>] &
```

or

```
daemon –f shelfman [<options>]
```

The following options are recognized:

```
-h <address>

-v <verbosity>

-c <path>

-cs[-]

-w[-]

-wt

-l[c][s]

-g <ip_address>

-sf[-]

-port <port>

-ph[=<IPMB-addr1>,<IPMB-addr2>].
```

A detailed description of these options is given below.

`-h <address>`

This option overrides the hardware address of the FRU site where the Shelf Manager resides. This hardware address is used as the IPMB address for the Shelf Manager (one which is based on its hardware address, separate from 20h) and is treated as a hexadecimal number. This option can be used if no hardware address is automatically available for the Shelf Manager. If this option is not used, the Shelf Manager obtains the hardware address in a carrier-specific way; the IPMB address is the hardware address multiplied by 2.

`-v <verbosity>`

Set the initial debug verbosity mask (see section 2.3.1). If this option is not present, the value of the parameter VERBOSITY in the file /etc/shelfman/conf determines the initial debug verbosity mask.

`-c <path>`

Path to non-volatile configuration files. The default path is "/var/nvdata".

`-cs[-]`

Enforce checking Shelf FRU Info checksums. If a checksum is invalid, an error message is produced and the Shelf FRU Info is not used. If a checksum is valid, no message is produced. If the option *-cs* is not present in the command line, the value of the parameter VERIFY_SHELF_FRU_CHECKSUM from the file /etc/shelfman/conf determines Shelf Manager behavior in this area.

`-w[-], -wt`

Enable/disable the watchdog timer. If neither of the *–w-* or *-wt* options are present in the command line, the value of the parameter WATCHDOG_ENABLED from the file /etc/shelfman.conf determines Shelf Manager actions in this area.

The -wt option applies the watchdog timer in test mode: that is, the actual timer is not strobed, but a warning message is printed if the interval between subsequent strobes exceeds 500 ms. (In normal operation, if this interval exceeds one second, the Shelf Manager is reset.)

`-l[s][c]`

Set logging destination: use 's' - to choose syslog and 'c' - to choose console. If this option is not specified, the values of the parameters CONSOLE_LOGGING_ENABLED and SYSLOG_LOGGING_ENABLED from the file /etc/shelfman.conf determine Shelf Manager actions in this area.

`-g <ip_address>`

Set the default gateway IP address. This address is only used if no gateway address is set in the LAN Configuration Parameters for channel 1.

`-sf[-]`

The option –sf forces the Shelf Manager to use EEPROMs for Shelf FRU Info storage. The option –sf- forces the Shelf Manager not to use EEPROMs for Shelf FRU Info storage. If neither of these options is present in the command line, the value of the parameter SHELF_FRU_IN_EEPROM from the file /etc/shelfman.conf determines Shelf Manager actions in this area.

`-p <port>`

Set redundancy communication port. If this option is not present, the value of the parameter REDUNDANCY_PORT from the file /etc/shelfman.conf determines Shelf Manager actions in this area.

`-ph[=<IPMB-addr1>,<IPMB-addr2>]`

This option defines IPMB addresses of pseudo-hubs (by default, 0x82 and 0x84). Pseudo-hubs are virtual IPM controllers that are created and emulated by the Shelf Manager and behave like IPM controllers for Base Interface hub boards. This option can be used for testing and when non-intelligent hub boards (non-compliant boards that do not implement an IPM controller) are installed in the shelf.

To run the Shelf Manager from the command line, type the following:

```
    daemon –f shelfman [<options>]
```

For example, here is a typical log of the Shelf Manager starting with options that cause it to print log information both on the console and to the log file and to not use EEPROMs for Shelf FRU Information:

```
# daemon –f shelfman –lcs –sf-

# <I> 11:16:49.133   [168] Pigeon Point Shelf Manager ver. 2.3.0. Built on
May 31 2006 15:29:02
<*> 11:16:49.139   [168] Limits: code=(400000:5132f0), end_data=10061000,
start_stack=7fff7e10, esp=7fff7880, eip=2ab0d2e4
<*> 11:16:49.139   [168] Stack limits: curr=1ff000, max=7fffffff
<*> 11:16:49.139   [168] Data limits: curr=7fffffff, max=7fffffff
<*> 11:16:49.144   [168] *** Lock log print buffer at 1003b710 ***
<*> 11:16:49.145   [168] *** Pthread lock log print buffer at 1003f740 ***
<*> 11:16:49.171   [168] Carrier set to "PPS"
<I> 11:16:49.177   [168] Device GUID: {1A85A1FA-FB3E-11D8-0080-0050C23FBC40}
<I> 11:16:49.255   [168] Redundancy netmask: 0
<I> 11:16:49.256   [168] Input redundancy socket successfully bound to:
192.168.0.192:1040
<I> 11:16:49.257   [168] Output redundancy socket successfully bound to:
192.168.0.192:1041
<I> 11:16:49.262   [168] Active listening thread created successfully
<I> 11:16:49.263   [168] Connecting to: 192.168.0.193:1040
<I> 11:16:49.265   [168] Connect() took 0 seconds, err=-1
<I> 11:16:49.266   [168] *** Running in Active mode (connection to backup
failed: -146) ***
<I> 11:16:49.267   [168] Redundancy protocol initialized successfully
<I> 11:16:49.267   [168] Shelfman: Using outside IP 192.168.1.198
<I> 11:16:49.291   [168] ADM1026 Controller found at I2C address 58,
context=10076ab8
<E> 11:16:49.606   [168] ADM1026: Ext 1 temp sensor reports bad value: -128,
ignored
<E> 11:16:49.607   [168] ADM1026: Ext 2 temp sensor reports bad value: -128,
ignored
<I> 11:16:49.690   [168] Creating 1 fan FRUs (eeprom_size=0),
descr=0x10000484
<I> 11:16:49.693   [168] Registering FRU for FT 0
<I> 11:16:49.696   [168] FT 0 FRU successfully registered as FRU 02
<I> 11:16:49.698   [168] Initializing fans
<I> 11:16:49.700   [168] Registering fan RD facility
<I> 11:16:49.702   [168] Activating fan tray 0, fan level=5, power level=0
<I> 11:16:49.705   [168] Controller 20, FRU 2: ATCA state set to M1,
prev=M0, cause=0, locked=0
<I> 11:16:49.732   [168] SEL truncation thread started successfully
<I> 11:16:49.734   [168] ShM SEL Activation complete
<I> 11:16:49.743   [180] PEF thread: starting
<I> 11:16:49.746   [168] PEF activated successfully
<I> 11:16:49.748   [168] PEF initialized successfully, System Event Sensor
#133
<I> 11:16:49.751   [168] Chassis facility activated successfully
<I> 11:16:49.754   [168] Chassis facility initialized successfully
<I> 11:16:49.776   [168] Controller FC, FRU 0: ATCA state set to M1,
prev=M0, cause=0, locked=0
<I> 11:16:49.783   [168] Operational state for SA FE, FRU 0 is set to M7
from stored SDR!
<I> 11:16:49.788   [168] SDR Repository: broadcasting Get Device ID
<I> 11:16:49.791   [168] SDR Repository registration: res=0
<I> 11:16:49.824   [184] IPMC Stored write thread started
rupgrade_tool: Cannot get upgrade status.
```

```
<I> 11:16:49.921   [168] Controller 20, FRU 0: ATCA state set to M1,
prev=M0, cause=0, locked=0
<I> 11:16:49.923   [168] Controller 20, FRU 1: ATCA state set to M1,
prev=M0, cause=0, locked=0
<I> 11:16:49.964   [168] Shelfman: Running.
<I> 11:16:49.977   [191] Added 20#1 to the fru list, treated as Shelf FRU
Info storage, size = 2048 (data size = 529)
<I> 11:16:49.978   [191] 1 Equal Shelf FRUs have been detected
<I> 11:16:49.981   [191] Added 20#2 to the fru list, treated as FRU Info
storage, size = 176 (data size = 176)
<I> 11:16:49.982   [191] 1 Equal Shelf FRUs have been detected
<I> 11:16:49.985   [194] Create controller for 20, sent GDI, returned NB
<I> 11:16:50.830   [183] Controller 20, FRU 0: ATCA state set to M2,
prev=M1, cause=2, locked=0
<I> 11:16:50.836   [183] Controller 20, FRU 1: ATCA state set to M2,
prev=M1, cause=2, locked=0
<I> 11:16:50.840   [183] Controller 20, FRU 2: ATCA state set to M2,
prev=M1, cause=2, locked=0
<I> 11:16:50.844   [183] Controller FC, FRU 0: ATCA state set to M2,
prev=M1, cause=2, locked=0
<I> 11:16:50.966   [200] SA 0x20 FRU 0 is ACTIVATING
<I> 11:16:50.982   [201] SA 0x20 FRU 1 is ACTIVATING
<I> 11:16:51.005   [200] Tasklet ACTIVATE: postponed until Shelf FRU
Information is found
<I> 11:16:51.008   [188] Added 20#2 to the fru list, treated as FRU Info
storage, size = 176 (data size = 176)
<I> 11:16:51.009   [188] 1 Equal Shelf FRUs have been detected
<I> 11:16:51.060   [201] Tasklet ACTIVATE: postponed until Shelf FRU
Information is found
<I> 11:16:51.063   [202] SA 0x20 FRU 2 is ACTIVATING
<I> 11:16:51.073   [203] SA 0xfc FRU 0 is ACTIVATING
<I> 11:16:51.147   [188] Added 20#2 to the fru list, treated as FRU Info
storage, size = 176 (data size = 176)
<I> 11:16:51.148   [188] 1 Equal Shelf FRUs have been detected
<I> 11:16:51.151   [202] Tasklet ACTIVATE: postponed until Shelf FRU
Information is found
<I> 11:16:51.161   [203] Tasklet ACTIVATE: postponed until Shelf FRU
Information is found
<I> 11:17:04.812   [204] Timeout for obtaining Shelf FRU Information has
expired, check operation called
<I> 11:17:04.815   [204] 1 Equal Shelf FRUs have been detected
<I> 11:17:04.818   [204] Registered potential fan 0x100905c0, sa=0x20
fru_id=2
<I> 11:17:04.825   [185] Move fan 0x100905c0, cnt 2 0->1
<I> 11:17:04.845   [185] IPMC Cooling Management: cooling state switched
from Unknown to Normal
<I> 11:17:04.874   [185] Cooling thread sensor scan: 2 sensors, 0.011166
seconds
<W> 11:17:04.938   [204] Shelf FRU Info found: failed to get Shelf Manager
IP Connection record from Shelf FRU (res = -61)
<I> 11:17:04.941   [204] LAN: setting RMCP IP address to 192.168.1.198
<I> 11:17:04.954   [204] LAN: using default RMCP subnet mask
<I> 11:17:04.959   [204] *** 0 Shelf FRU Info updated ***
<I> 11:17:04.972   [205] SA 0x20 FRU 0 is ACTIVATING (Postponed)
<W> 11:17:04.974   [205] Local FRU 0 at SA 0x20 is not listed in Shelf FRU
Info; still activate
```

```
<I> 11:17:04.977   [193] Controller 20, FRU 0: ATCA state set to M3,
prev=M2, cause=1, locked=0
<I> 11:17:04.981   [193] Controller 20, FRU 0: ATCA state set to M4,
prev=M3, cause=0, locked=0
<E> 11:17:05.011   [210] Enabling E-keyed Ports for SA 0x20 FRU 0: failed to
get Board Connectivity record (res=-22)
<I> 11:17:05.023   [210] SA 0x20 FRU 0 is OPERATIONAL
<I> 11:17:05.025   [206] SA 0x20 FRU 1 is ACTIVATING (Postponed)
<W> 11:17:05.025   [206] Local FRU 1 at SA 0x20 is not listed in Shelf FRU
Info; still activate
<I> 11:17:05.027   [207] SA 0x20 FRU 2 is ACTIVATING (Postponed)
<W> 11:17:05.028   [207] Local FRU 2 at SA 0x20 is not listed in Shelf FRU
Info; still activate
<I> 11:17:05.032   [208] SA 0xfc FRU 0 is ACTIVATING (Postponed)
<W> 11:17:05.032   [208] Local FRU 0 at SA 0xFC is not listed in Shelf FRU
Info; still activate
<I> 11:17:05.034   [209] RMCP: starting server thread for 192.168.1.198:623
<I> 11:17:05.050   [193] Controller 20, FRU 1: ATCA state set to M3,
prev=M2, cause=1, locked=0
<I> 11:17:05.053   [193] Controller 20, FRU 1: ATCA state set to M4,
prev=M3, cause=0, locked=0
<I> 11:17:05.056   [193] Controller 20, FRU 2: ATCA state set to M3,
prev=M2, cause=1, locked=0
<I> 11:17:05.064   [193] Controller FC, FRU 0: ATCA state set to M3,
prev=M2, cause=1, locked=0
<I> 11:17:05.139   [193] Controller 20, FRU 2: ATCA state set to M4,
prev=M3, cause=0, locked=0
<I> 11:17:05.152   [212] SA 0x20 FRU 2 is OPERATIONAL
<I> 11:17:05.164   [193] Controller FC, FRU 0: ATCA state set to M4,
prev=M3, cause=0, locked=0
<I> 11:17:05.190   [213] Set Port State (disable): ipmc=FC, chan=1, it=0,
ports=1, lt=1, ext=0, group=0
<I> 11:17:05.192   [213] Set Port State (disable): ipmc=FC, chan=2, it=0,
ports=1, lt=1, ext=0, group=0
<I> 11:17:05.195   [213] SA 0xfc FRU 0 is OPERATIONAL
<W> 11:17:05.234   [194] Set Port State to SA 0xfc (seq_no = 41): error
reported in response (cc=0xc1)
<W> 11:17:05.237   [194] Set Port State to SA 0xfc (seq_no = 42): error
reported in response (cc=0xc1)
<I> 11:17:05.247   [211] SA 0x20 FRU 1 is OPERATIONAL
```

3.3. Redundant Operation

The active Shelf Manager exposes the ShMC device (address 20h) on IPMB, manages IPMB and the IPM controllers and interacts with the System Manager over RMCP and other shelf-external interfaces. It maintains an open TCP connection with the backup Shelf Manager. It communicates all changes in the state of the managed objects to the backup Shelf Manager.

The backup Shelf Manager does not expose the ShMC on IPMB, does not actively manage IPMB and IPM controllers, nor interact with the System Manager via the shelf-external interfaces (with one exception noted below). Instead, it maintains the state of the managed objects in its own memory (volatile and non-volatile) and updates the state as directed by the active Shelf Manager.

The backup Shelf Manager may become active as the result of a switchover. Two types of switchover are defined:

- cooperative switchover: the active and backup Shelf Managers negotiate the transfer of responsibilities from the active to the backup Shelf Manager; this mode is supported via the CLI 'switchover' command issued on the active or backup Shelf Manager.
- forced switchover: the backup Shelf Manager determines that the active Shelf Manager is no longer alive or healthy, and forcefully takes on the responsibilities of the active Shelf Manager.

The backup Shelf manager recognizes the departure of the active Shelf Manager when the Remote Healthy or Remote Presence low-level signal becomes inactive. The Remote Presence signal monitors the presence of the peer Shelf Manager; this signal going inactive means that the board hosting the peer Shelf Manager has been removed from the shelf. The Remote Healthy signal is set by the peer Shelf Manager during initialization; this signal going inactive means that the remote Shelf Manager has become unhealthy (typically, has been powered off or reset).

Another situation that needs some action from the backup Shelf Manager is when the TCP connection between the Shelf Managers gets closed. This happens when either the communication link between the two Shelf Managers gets broken or the shelfman process on the active Shelf Manager terminates, in a voluntary or involuntary way, or due to a software exception. Also, since the keepalive option is enabled on the TCP connection, it will close shortly after the active ShMM is switched off or reset. In the case of Shelf Manager termination, it is possible that the TCP connection is closed *before* the Remote Healthy signal becomes inactive. So, in order to determine why the TCP connection closed, the backup Shelf Manager samples the state of the Remote Healthy signal immediately and, if it is still active, again after some delay. If the Remote Healthy signal ultimately goes inactive, the backup Shelf Manager concludes that the active Shelf Manager is dead, and initiates a switchover.

Otherwise, if the Remote Healthy signal stays active, the backup Shelf Manager concludes that the communication link between the Shelf Managers is broken. In that case, no switchover is initiated; instead the backup Shelf Manager repeatedly reinitializes itself and tries to establish a connection with the active Shelf Manager, until the communication link is restored. Reinitialization is achieved by rebooting the ShMM and automatically restarting the Shelf Manager after the reboot. Special logic in the Shelf Manager guarantees that it does not try to become active at startup if the peer Shelf Manager is already active.

The Shelf Manager uses a watchdog timer to protect against becoming unresponsive due to infinite loops or other software bugs. In the event the watchdog timer on the active Shelf Manager triggers, that ShMM will be reset, causing the Remote Healthy signal on the backup ShMM to become inactive, thus triggering a switchover.

After a switchover, the formerly backup Shelf Manager performs additional initialization, actualizing its cached state information and collecting any necessary further information from the IPM Controllers on IPMB. The newly active Shelf Manager then exposes the ShMC device (address 20h) on IPMB, and assumes the IP address that was used for RMCP and other shelf-external interactions between the formerly active Shelf Manager and the System Manager. Since the RMCP session information is propagated from the active Shelf Manager to the backup Shelf Manager, RMCP sessions survive the switchover. For the System Manager using RMCP, the switchover is transparent.

The switchover is also transparent for the Web interface and for the SNMP interface (provided that they use the redundancy IP address that is switched over). Command-line interface sessions, since they are initiated locally on the target Shelf Manager, do not survive a switchover and need to be re-established again on the newly active Shelf Manager. The command-line interface support on the backup Shelf Manager is limited, but it does allow the backup Shelf Manager to request a switchover using the 'switchover' command.

The formerly active Shelf Manager after the switchover can cease to exist or reinitialize itself as the backup Shelf Manager. Reinitializing as the backup Shelf Manager requires rebooting the operating system on the formerly active ShMM.

3.4. Operation in Radial Shelves

Some shelves implement radial control of IPMB-0 links in the shelf. In that case, the segment of IPMB-0 leading to each IPM controller in the shelf can be turned on and off individually by the Shelf Manager. This applies individually to both the IPMB-A and IPMB-B portions of IPMB-0. The operation of the Shelf Manager in such shelves is different from the shelves with a simple bused IPMB-0.

The Shelf FRU Information in radial shelves must contain special records that identify the shelf as implementing a radial IPMB-0 topology and specify the routing of the IPMB links to the IPM controllers. These records must conform to the ATCA specification (PICMG 3.0 R2.0), section 3.8.1.1. In addition, the carrier must support radial operation. (Not all carriers do that.)

If both the prerequisites above are satisfied, the Shelf Manager implements isolation of faulty bus segments in the case of a persistent error on IPMB-0. This prevents the failure of an entire bus in the case of a fault on one or several specific IPM controllers. Isolation is performed independently for IPMB-A and IPMB-B. Isolated segments are turned off (which means that the corresponding IPM controller is isolated from the corresponding bus). They are kept in the isolated state for IPMB_LINK_ISOLATION_TIMEOUT seconds, after which they are automatically turned on; if the fault still exists, the persistent IPMB-0 error will occur again and the faulty links will be isolated again. (By default, however, the value of this configuration parameter is -1, which means "never re-enable the link automatically".)

In addition, isolated links can be turned on in one of the following ways:

- manually, via the CLI command "setipmbstate" with parameters indicating the ShMC target address (0x20) and the corresponding link number and bus (A or B).

- automatically, when the corresponding IPM controller is removed from the shelf (that is, when it goes to the state M0).

To find out the current isolation state of radial IPMB-0 links, the CLI command "getipmbstate" can be used. When applied to the ShMC target address (0x20), this command shows the state of all radial IPMB-0 links. For example, the command

```
clia getipmbstate 20
```

can yield a report like this:

```
Pigeon Point Shelf Manager Command Line Interpreter
```

```
20: Link: 1, LUN: 0, Sensor # 10 ("IPMB LINK 1")
    Bus Status: 0x8  (IPMB-A Enabled, IPMB-B Enabled)
    IPMB A State: 0x08 (LocalControl, No failure)
    IPMB B State: 0x08 (LocalControl, No failure)

20: Link: 2, LUN: 0, Sensor # 11 ("IPMB LINK 2")
    Bus Status: 0x8  (IPMB-A Enabled, IPMB-B Enabled)
    IPMB A State: 0x08 (LocalControl, No failure)
    IPMB B State: 0x08 (LocalControl, No failure)

20: Link: 3, LUN: 0, Sensor # 15 ("IPMB LINK 3")
    Bus Status: 0x8  (IPMB-A Enabled, IPMB-B Enabled)
    IPMB A State: 0x08 (LocalControl, No failure)
    IPMB B State: 0x08 (LocalControl, No failure)

20: Link: 4, LUN: 0, Sensor # 16 ("IPMB LINK 4")
    Bus Status: 0x4  (IPMB-A Disabled, IPMB-B Enabled)
    IPMB A State: 0x07 (Isolated, Undiagnosed communication failure)
    IPMB B State: 0x08 (LocalControl, No failure)

20: Link: 5, LUN: 0, Sensor # 17 ("IPMB LINK 5")
    Bus Status: 0x8  (IPMB-A Enabled, IPMB-B Enabled)
    IPMB A State: 0x08 (LocalControl, No failure)
    IPMB B State: 0x08 (LocalControl, No failure)

20: Link: 6, LUN: 0, Sensor # 18 ("IPMB LINK 6")
    Bus Status: 0x8  (IPMB-A Enabled, IPMB-B Enabled)
    IPMB A State: 0x08 (LocalControl, No failure)
    IPMB B State: 0x08 (LocalControl, No failure)

20: Link: 7, LUN: 0, Sensor # 19 ("IPMB LINK 7")
    Bus Status: 0x8  (IPMB-A Enabled, IPMB-B Enabled)
    IPMB A State: 0x08 (LocalControl, No failure)
    IPMB B State: 0x08 (LocalControl, No failure)

20: Link: 8, LUN: 0, Sensor # 20 ("IPMB LINK 8")
    Bus Status: 0x8  (IPMB-A Enabled, IPMB-B Enabled)
    IPMB A State: 0x08 (LocalControl, No failure)
    IPMB B State: 0x08 (LocalControl, No failure)

20: Link: 9, LUN: 0, Sensor # 21 ("IPMB LINK 9")
    Bus Status: 0x8  (IPMB-A Enabled, IPMB-B Enabled)
    IPMB A State: 0x08 (LocalControl, No failure)
    IPMB B State: 0x08 (LocalControl, No failure)

20: Link: 10, LUN: 0, Sensor # 22 ("IPMB LINK 10")
    Bus Status: 0x8  (IPMB-A Enabled, IPMB-B Enabled)
    IPMB A State: 0x08 (LocalControl, No failure)
    IPMB B State: 0x08 (LocalControl, No failure)

20: Link: 11, LUN: 0, Sensor # 23 ("IPMB LINK 11")
    Bus Status: 0x8  (IPMB-A Enabled, IPMB-B Enabled)
    IPMB A State: 0x08 (LocalControl, No failure)
    IPMB B State: 0x08 (LocalControl, No failure)

20: Link: 12, LUN: 0, Sensor # 24 ("IPMB LINK 12")
    Bus Status: 0x8  (IPMB-A Enabled, IPMB-B Enabled)
    IPMB A State: 0x08 (LocalControl, No failure)
    IPMB B State: 0x08 (LocalControl, No failure)

20: Link: 13, LUN: 0, Sensor # 25 ("IPMB LINK 13")
    Bus Status: 0x8  (IPMB-A Enabled, IPMB-B Enabled)
    IPMB A State: 0x08 (LocalControl, No failure)
    IPMB B State: 0x08 (LocalControl, No failure)

20: Link: 14, LUN: 0, Sensor # 26 ("IPMB LINK 14")
    Bus Status: 0x8  (IPMB-A Enabled, IPMB-B Enabled)
    IPMB A State: 0x08 (LocalControl, No failure)
    IPMB B State: 0x08 (LocalControl, No failure)

20: Link: 15, LUN: 0, Sensor # 27 ("IPMB LINK 15")
    Bus Status: 0x8  (IPMB-A Enabled, IPMB-B Enabled)
    IPMB A State: 0x08 (LocalControl, No failure)
    IPMB B State: 0x08 (LocalControl, No failure)
```

The output above indicates that IPMB-0 link 4 for IPMB-A has been isolated as a result of a persistent IPMB failure. To manually re-enable the link, the following CLI command can be used:

```
clia setipmbstate 20 A 4 1
```

3.5. Automatic SEL Truncation

The System Event Log (SEL) on the Shelf Manager is used to store events from all IPM controllers and FRUs in the shelf and can easily exceed its maximum capacity. To prevent overflows, the Shelf Manager can automatically truncate the SEL, removing the oldest entries as SEL approaches its limits.

The automatic truncation algorithm works as follows. The two configuration parameters are defined: SEL_HIGH_WATERMARK and SEL_LOW_WATERMARK. The first parameter relates to the percentage of free entries in the SEL, and the second one relates to the percentage of occupied entries in the SEL. When the percentage of free SEL entries falls below SEL_HIGH_WATERMARK, the truncation thread wakes up and starts deleting the oldest entries from the SEL using the IPMI command Delete SEL Entry. The thread works until the percentage of occupied entries in the SEL falls below SEL_LOW_WATERMARK; then it stops and stays dormant until the percentage of free SEL entries falls below SEL_HIGH_WATERMARK again.

This algorithm is enabled by default, but can be turned off by setting the configuration variable SYSTEM_MANAGER_TRUNCATES_SEL to TRUE. In that case, automatic truncation is turned off, but the Shelf Manager assists the System Manager by indicating the current state of the SEL via the sensor "SEL State" of the type "Event Logging Enabled", as follows (in accordance with the IPMI 2.0 specification):

- The reading returned by that sensor is equal to the percentage of the SEL that is occupied (0 to 100);
- The sensor assumes the state "SEL Almost Full" (5h) when the percentage of free entries in the SEL falls below the high watermark (the value of the configuration parameter SEL_HIGH_WATERMARK). An event is generated in that case and placed in the SEL; Event Data Byte 3 of the event contains the percentage of the SEL that is occupied. No event if generated for this state until the occupancy of the SEL falls below the low watermark; this is done to prevent multiple events if SEL occupancy oscillates around the high watermark.
- A custom state (6h) is defined for the state when the SEL occupancy is below the low watermark (the percentage of occupied entries falls below the value of the configuration parameter SEL_LOW_WATERMARK). There is no event associated with this state; the System Manager can detect this state by polling the sensor value and its asserted states.

Using this information, the System Manager performs truncation of the SEL, presumably using the IPMI command "Delete SEL Entry," sending it over the RMCP session.

Chapter

4

4. Customer Support

If you are having any problems with the Pigeon Point Shelf Manager or ShMM product, please contact your supplier for the Pigeon Point products with questions and problem reports.

If you have any questions about direct purchase of Pigeon Point products (one of the Pigeon Point Board Management Reference designs, for instance), please contact Pigeon Point Systems.

Pigeon Point Systems
PO Box 66989
Scotts Valley, CA 95067-6989
Phone: (831) 438-1565
Fax: (831) 438-3709

PART II: Re-initializing and Re-programming the ShMM-500

Chapter

5

5. Re-initializing the ShMM-500

This chapter describes how to re-initialize the U-Boot environment variables, the filesystem in Flash and the login password—on all the ShMM-500.

5.1. Re-initializing the U-Boot Environment

The U-boot environment variables are stored in the ShMM EEPROM. If you would like to restore the factory defaults for the U-boot environment variables, you must first erase the environment variables stored in EEPROM and reset (or power cycle) the ShMM.

To erase the EEPROM, you need to enter the following command from the U-Boot prompt:

```
shmm500 eeprom write 80400000 0 1000

EEPROM @0x50 write: addr 80400000  off 0000  count 4096 ... done
shmm500
```

Then you need to reset the ShMM.

```
shmm500 reset

U-Boot 1.1.2 (Apr 27 2005 - 19:17:09)

CPU: Au1550 324 MHz, id: 0x02, rev: 0x00
Board: ShMM-500
S/N: 8000041
DRAM:  64 MB
Flash: 16 MB
*** Warning - bad CRC, using default environment

In:    serial
Out:   serial
Err:   serial
Net:   Au1X00 ETHERNET
Hit any key to stop autoboot:  0
shmm500
```

Then save these environment settings. Use the “saveenv” command to store the settings:

```
shmm500 saveenv
```

5.2. Re-initializing the File System

The filesystem is stored within the Flash and can be reset to factory defaults quite easily. U-boot has an environment variable called `flash_reset`. By setting this variable to “y” and then booting up the system, the file system will be re-initialized to factory defaults.

```
shmm500 setenv flash_reset y
shmm500 saveenv
shmm500 boot
```

The `flash_reset` variable is autiomatically set to “n” at system startup after re-initializing the flash. The `boot` command (the second command above) begins booting the Linux kernel. It is during this process that the file system is re-initialized. The following output will be shown on the console.

```
/etc/rc: Mounted /dev/pts
/etc/rc: Flash erase requested via U-BOOT var
/etc/rc: erasing mtdchar1 -> /etc
Erased 1024 Kibyte @ 0 -- 100% complete.
/etc/rc: erasing mtdchar0 -> /var
Erased 1536 Kibyte @ 0 -- 100% complete.
/etc/rc: Mounted /dev/mtdblock3 to /var
/etc/rc: /var/log mounted as FLASH disk
/etc/rc: Started syslogd and klogd
/etc/rc: /var/tmp mounted as RAM disk
/etc/rc: hostname demo
/etc/rc: /dev/mtdblock2 appears to be empty ... restoring from factory
/etc...
```

5.3. Resetting the Login Password

The factory default login for the ShMM is a user id of “`root`” without any password. We highly encourage users to change the password when configuring the Shelf Manager. In the event that the new password is forgotten, the password can be reset to its factory default via the `password_reset` U-Boot variable. By setting this variable to “y” and then booting up the system, the root password will be removed.

```
shmm500 setenv password_reset y
shmm500 saveenv
shmm500 boot
```

The following output will be shown on the console, during boot up.

```
/etc/rc: hostname demo
/etc/rc: Restoring password file to factory default
```

Chapter

6

6. Re-programming the ShMM-500

Reprogramming the ShMM-500 involves reprogramming several firmware images stored on it. The set of images can be conceptually divided into two groups, depending on the method of reprogramming them. These groups are:

- U-Boot, kernel and RFS images. These images are normally reprogrammed using the reliable upgrade procedure. Alternatively, the kernel and RFS can be reprogrammed from the U-Boot prompt by loading them from a TFTP server.

- CPLD image. This image is reprogrammed from the ShMM-500 command line, using a special command-line utility "cpldtool".

Detailed instructions for reprogramming U-Boot, kernel and RFS images are given below. When updates to the CPLD image are necessary (which is expected to be rare), specific step-by-step instructions will be posted for each instance.

6.1. Firmware Reliable Upgrade Procedure

6.1.1. Reliable Upgrade Procedure Overview

Monterey Linux provides a reliable upgrade procedure for the firmware images on a running and functioning ShMM-500. The procedure supports upgrade of the U-Boot firmware, the Linux kernel and the Linux root file system (or an arbitrary combination of these three images). If a software upgrade attempt fails (for instance, due to installation of a faulty U-Boot firmware image that is not capable of booting the ShMM-500 or a Shelf Manager that can't start) the reliable upgrade procedure automatically falls back to the previous version of the firmware in persistent Flash.

The ShMM-500 Flash is divided into two areas. When a stable set of firmware is established in one of these areas, it is designated the persistent area. When new firmware is installed, it goes in the other area, which is initially designated provisional. Once a new set of firmware in the provisional area is validated, that area is designated the persistent area and continues in use until a future upgrade cycle starts the process over.

The reliable upgrade hardware mechanisms ensure that no matter what is installed to the provisional Flash, the ShMM-500 always manages to boot from a software copy that is either fully functional or sufficiently sane to determine that that there has been a failure in the upgrade session and consequently take appropriate corrective actions to revert to the safe software copy in persistent Flash.

At a higher level, the reliable upgrade hardware mechanisms are assisted by a software protocol based on logging of the status of the upgrade session to a non-volatile file in /var/upgrade/status (refer to 6.1.4). The software protocol ensures that the reliable upgrade does not finish until all the required actions, including those defined by custom "hook" scripts that may be needed for a specific application, have all completed successfully.

The Monterey Linux reliable upgrade procedure is described in full detail in Chapter 6 of the Monterey Linux User's Guide: Au1550 Edition ("ML User's Guide", below). Users of the reliable upgrade procedure should review that material as background. That material explains the underlying mechanisms, which provide substantial opportunities for customization to particular requirements. Any users who wish to extend or customize the workings of the reliable upgrade process must become thoroughly familiar with the ML User's Guide coverage of this topic.

The reliable upgrade procedure implemented in Monterey Linux is neutral to the embedded application running on the ShMM-500. The procedure provides a sufficient set of "hooks" allowing a specific application running on the ShMM-500 to ensure that custom actions are carried out at appropriate points of the reliable upgrade. The remainder of this section focuses on provisions for reliable upgrade of the Pigeon Point Shelf Manager firmware that have been implemented using these hooks.

6.1.2. Flash Partitioning

The ShMM-500 provides a hardware mechanism that allows swapping of the lower and upper halves of the Flash in the system memory map under control of software running on the MIPS. This capability is implemented in support of the reliable upgrade procedure for software images in Flash. The reliable software upgrade procedure assumes that the Flash device contains two copies of the software, located in the lower and upper halves of Flash. All ShMM-500s are shipped with this partitioning, in which the Flash device is divided onto two equal parts, each dedicated to holding one copy of the ShMM-500 software.

The U-Boot environment variable reliable_upgrade (refer to 2.1.2) is used by the Linux layers to determine whether or not the reliable upgrade procedure is enabled. But this variable is required to have a value 'y' and the variable may be removed in future releases. This variable is passed to the Linux kernel in the bootargs kernel parameters string (refer to 2.1.2). The Linux Flash MTD layer checks the reliable_upgrade parameter at Flash partitions initialization time and, depending on the parameter value (as well as the size of the Flash device installed on the ShMM-500 module), partitions the Flash device in an appropriate way.

This section assumes that the ShMM-500 is configured to support reliable upgrade, including the two separate Flash regions. The tables below provide summary of the Flash partitions maintained on the ShMM-500 in this configuration. The exact layout of Flash partitions provided by the FOSL depends on the size of the Flash device installed on the ShMM-500 module (16MB, 32MB, or 64MB).

Table 1 provides a summary of the Flash partitions maintained on the ShMM-500 by FOSL for the 16MB Flash devices.

Offset in Flash (in MBytes)	Size (in MBytes)	Device Node	Mounted As (on Startup)	Content
0	0.5	/dev/mtdchar10, /dev/mtdblock10	/var/upgrade	The second half of the /var/upgrade JFFS2 file system
0.5 + (FLASH_SIZE – 16)/2	1.5	/dev/mtdchar5, /dev/mtdblock5	Not mounted	The "other" /var JFFS2 file system
FLASH_SIZE/2 – 62	1	/dev/mtdchar6, /dev/mtdblock6	Not mounted	The "other" /etc JFFS2 file system
FLASH_SIZE/2 – 53	1	/dev/mtdchar7	Not mounted	The "other" Linux kernel image
FLASH_SIZE/2 – 44	0.25	/dev/mtdchar8	Not mounted	The "other" U-Boot firmware image
FLASH_SIZE/2 – 3.754.25	3.75	/dev/mtdchar9	Not mounted	The "other" Linux root file system (rfs) image
FLASH_SIZE/28	0.5	/dev/mtdchar10, /dev/mtdblock10	/var/upgrade	The first half of the /var/upgrade JFFS2 file system
FLASH_SIZE – 7.58.5	1.5	/dev/mtdchar0, /dev/mtdblock0	/var	The /var JFFS2 file system
FLASH_SIZE – 610	1	/dev/mtdchar1, /dev/mtdblock1	/etc	The /etc JFFS2 file system
FLASH_SIZE – 511	1	/dev/mtdchar2	Not mounted	The Linux kernel image
FLASH_SIZE – 412	0.25	/dev/mtdchar3	Not mounted	The U-Boot firmware image
FLASH_SIZE – 3.7512.25	3.75	/dev/mtdchar4	Not mounted	The Linux root file system (rfs) image

Table 1: Flash Partitioning for 16MB Flash reliable_upgrade=y

Table 2 provides a summary of the Flash partitions maintained on the ShMM-500 by FOSL for the 32MB Flash devices:

Offset in Flash (in MBytes)	Size (in MBytes)	Device Node	Mounted As (on Startup)	Content
0	0.5	/dev/mtdchar10, /dev/mtdblock10	/var/upgrade	The second half of the /var/upgrade JFFS2 file system
0.5	1	/dev/mtdchar7	Not mounted	The "other" Linux kernel image
1.5	1	/dev/mtdchar6, /dev/mtdblock6	Not mounted	The "other" /etc JFFS2 file system
2.5	1.75	/dev/mtdchar5, /dev/mtdblock5	Not mounted	The "other" /var JFFS2 file system
4.25	7.75	/dev/mtdchar9	Not mounted	The "other" Linux root file system (rfs) image
12	0.25	/dev/mtdchar8	Not mounted	The "other" U-Boot firmware image
12.25	3.75	/dev/mtdchar11, /dev/mtdblock11	Not mounted	The second half of the `app_jffs` application-specific JFFS2 partition
16	0.5	/dev/mtdchar10, /dev/mtdblock10	/var/upgrade	The first half of the /var/upgrade JFFS2 file system
16.5	1	/dev/mtdchar2	Not mounted	The Linux kernel image
17.5	1	/dev/mtdchar1, /dev/mtdblock1	/etc	The /etc JFFS2 file system
18.5	1.75	/dev/mtdchar0,	/var	The /var JFFS2

Offset in Flash (in MBytes)	Size (in MBytes)	Device Node	Mounted As (on Startup)	Content
		/dev/mtdblock0		file system
20.25	7.75	/dev/mtdchar4	Not mounted	The Linux root file system (rfs) image
28	0.25	/dev/mtdchar3	Not mounted	The U-Boot firmware image
28.25	3.75	/dev/mtdchar11, /dev/mtdblock11	Not mounted	The second half of the `app_jffs` application-specific JFFS2 partition

Table 2: Flash partitioning for 32MB Flash

Table 3 provides a summary of the Flash partitions maintained on the ShMM-500 by FOSL for the 64MB Flash devices:

Offset in Flash (in MBytes)	Size (in MBytes)	Device Node	Mounted As (on Startup)	Content
0	0.5	/dev/mtdchar10, /dev/mtdblock10	/var/upgrade	The second half of the /var/upgrade JFFS2 file system
0.5	1	/dev/mtdchar7	Not mounted	The "other" Linux kernel image
1.5	1	/dev/mtdchar6, /dev/mtdblock6	Not mounted	The "other" /etc JFFS2 file system
2.5	1.75	/dev/mtdchar5, /dev/mtdblock5	Not mounted	The "other" /var JFFS2 file system

Offset in Flash (in MBytes)	**Size (in MBytes)**	**Device Node**	**Mounted As (on Startup)**	**Content**
4.25	15.75	/dev/mtdchar9	Not mounted	The "other" Linux root file system (rfs) image
20	8	/dev/mtdchar12, /dev/mtdblock12	Not mounted	The second half of the `app1_jffs` application-specific JFFS2 partition
28	0.25	/dev/mtdchar8	Not mounted	The "other" U-Boot firmware image
28.25	3.75	/dev/mtdchar11, /dev/mtdblock11	Not mounted	The second half of the `app_jffs` application-specific JFFS2 partition
32	0.5	/dev/mtdchar10, /dev/mtdblock10	/var/upgrade	The first half of the /var/upgrade JFFS2 file system
32.5	1	/dev/mtdchar2	Not mounted	The Linux kernel image
33.5	1	/dev/mtdchar1, /dev/mtdblock1	/etc	The /etc JFFS2 file system
34.5	1.75	/dev/mtdchar0, /dev/mtdblock0	/var	The /var JFFS2 file system

Offset in Flash (in MBytes)	Size (in MBytes)	Device Node	Mounted As (on Startup)	Content
36.25	15.75	/dev/mtdchar4	Not mounted	The Linux root file system (rfs) image
52	8	/dev/mtdchar12, /dev/mtdblock12	Not mounted	The first half of the `app1_jffs` application-specific JFFS2 partition
60	0.25	/dev/mtdchar3	Not mounted	The U-Boot firmware image
60.25	3.75	/dev/mtdchar11, /dev/mtdblock11	Not mounted	The second half of the `app_jffs` application-specific JFFS2 partition

Table 3: Flash partitioning for 64MB Flash

6.1.3. The /var/upgrade File System

As documented in 6.1.2, Monterey Linux mounts a 1 MByte partition as a JFFS2 file system at /var/upgrade. This file system is used to host the reliable upgrade procedure status file (refer to 6.1.4).

It is important to note that the /var/upgrade JFFS2 partition is composed of two non-contiguous Flash blocks (0.5 MBytes each) residing in both the lower and upper halves of the Flash device. Monterey Linux takes advantage of the ability of the Linux MTD and JFFS2 layers to support a file system in non-contiguous Flash sectors in order to implement /var/upgrade this way.

Another feature of the JFFS2 file system that makes /var/upgrade work for the purposes of the reliable upgrade procedure is that the JFFS2 internal structures do not create any dependencies (such as linked lists) based on Flash sector numbers or absolute offsets in Flash. Instead, when mounting a file system on a partition, the JFFS2 scans all the Flash sectors comprising the partition and recreates the logical content of a file system in an internal in-RAM representation. This feature ensures that regardless of which half of the Flash the ShMM-500 has booted from, Linux is able to mount /var/upgrade as a JFFS2 file system and make use of the previous content of the file system.

6.1.4. Reliable Upgrade Procedure Status File

The software reliable upgrade procedure maintains the status of the most recent upgrade procedure session in the file /var/upgrade/status residing in a dedicated file system (/var/upgrade), which is mounted by Linux regardless of which Flash the ShMM has booted from. If the file exists, it contains the status of an upgrade procedure session that either is in progress presently or has recently completed.

/var/upgrade/status is an ASCII format file that contains one or more new line-terminated records, each describing the status of a particular step in the upgrade procedure. The format of a record line is as follows:

<*step*>: <*status*>

where *step* is an integer ranging from 1 to 14 (with Step 14 corresponding to a completed upgrade session) and *status* is a human-readable string describing status of the current step of the upgrade procedure session. Refer to the ML User Guide for a list and explanation of these steps.

The status file is used by the reliable upgrade utility (refer to 6.1.5) to maintain a software protocol atop the reliable upgrade procedure hardware mechanisms to reliably determine the status of the upgrade procedure and proceed as appropriate.

6.1.5. Reliable Upgrade Utility

A special user-space utility is provided that is used for carrying out the reliable upgrade procedure as well as checking the status of the most recent upgrade.

The utility can be called only from the superuser (root) account. Any attempt to run the utility from a non-superuser account is rejected.

As a first step in its execution, the utility checks that the `reliable_upgrade` U-Boot environment variable (refer to 6.1.2), as passed by U-Boot to the Linux kernel in the kernel parameters string, is set to `y`. If this check fails, the utility immediately terminates and exits with an appropriate error code.

If called with any of the –s, -c, or –f options, the utility is being used to carry out the reliable upgrade procedure. While in the upgrade procedure, the utility logs to /var/upgrade/status the status of each action it performs as it proceeds through the steps of the upgrade procedure. If the utility detects a failure, the reliable upgrade procedure is terminated by adding to /var/upgrade/status a record indicating an unsuccessful completion of the upgrade procedure and exiting with an appropriate error code.

The utility prints any informational messages to stdout. Providing a -v specifier to any option that supports it increases the verbosity of the informational messages. The utility prints any error messages to stderr.

The utility has the following synopsis:

- `rupgrade_tool -s {--`*`dst`*`=`*`src`*`}... [--proto=`*`protocol`*`] [-d] [--hook=`*`args`*`] [-v]`
- `rupgrade_tool –c [-v]`
- `rupgrade_tool –f [--hook=`*`args`*`] [-v]`
- `rupgrade_tool –w [-f]`

- `rupgrade_tool –S [-v]`
- `rupgrade_tool -u`
- `rupgrade_tool -h`

where the parameters are defined as follows:

- `-s {--`*`dst`*`=`*`src`*`}... [--proto=`*`protocol`*`] [--hook=`*`args`*`] [-v]`

  Initiate the reliable upgrade procedure. As delivered with Shelf Manager support, this step includes the following actions:

  - obtaining the images to copy: locally or via the network;
  - copying the images to the provisional Flash;
  - terminating the Shelf Manager instance running on the ShMM-500, if any;
  - copying non-volatile data to the provisional Flash;
  - resetting the ShMM-500 and instructing it to boot from the provisional Flash.

  Because of the last step, an invocation of `rupgrade_tool –s` typically does not return and instead resets the ShMM-500. If `rupgrade_tool –s` does return, it indicates that the reliable upgrade procedure has failed and was terminated before proceeding to reset the ShMM-500 in order to boot from the provisional Flash.

  Before the first step of the upgrade procedure is initiated by the utility, it removes the /var/upgrade/status file (refer to 6.1.4). In other words, the status of the previous upgrade procedure session (if any) is lost and overwritten by the status of the new upgrade procedure session as soon as `rupgrade_tool –s` is called.

  There can be one or more --*dst*=*src* specifiers in a call to `rupgrade_tool –s`. Each such specifier defines the name of a to-be-installed upgrade image file and where the file is to be installed in the Flash of the ShMM-500.

  dst defines the destination of a newly installed upgrade image and can be any of the following:

  - u - Upgrade the U-boot image in the provisional U-Boot firmware image partition (/dev/mtdchar3).
  - k - Upgrade the Linux kernel image in the provisional Linux kernel image partition (/dev/mtdchar2).
  - r - Upgrade the root file system image in the provisional root file system image partition (/dev/mtdchar4).

  src specifies an upgrade image file to be copied to the provisional Flash partition specified by *dst*.

  The image upgrade works as follows. For each of the specified *src* images, the image is copied to the ShMM-500 using the specified copy protocol. If no -d specifier is supplied, the image is first copied to the RAM file system of the ShMM-500 (specifically, the copy is to the /tmp directory) and then moved to Flash (that is, copied to the destination partition in Flash and then removed from the RAM file system). If there is a -d specifier supplied in the call to `rupgrade_tool –s`, the intermediate copy to the /tmp directory is skipped and the image is copied directly to its destination in the Flash. Use of this specifier is intended for a scenario where there is insufficient run-time memory on the ShMM-500 for an intermediate copy to the RAM file system.

If no –d specifier is supplied, the reliable upgrade procedure invokes a special script, the main purpose of which is to validate images after they are copied to the /tmp directory. If –d specifier is present, no such validation is performed.

Currently, the script `/etc/upgrade/step4vshm` supplied with the Shelf Manager does not perform specific image validation steps, but does take responsibility for filling in the Flash partitions for which no images are provided in the current call to rupgrade_tool (as would happen in a partial upgrade scenario). These partitions are copied from the current persistent Flash to the provisional Flash. For example, if the current partial upgrade provides only a new RFS image, the script copies the U-Boot and kernel partitions from the persistent Flash to the provisional Flash.

As soon as the first image has been installed to its destination, the utility proceeds to the second image (if there is one), and so on, until all the supplied image files have been successfully installed to Flash. A failure to successfully install an image immediately terminates the upgrade procedure (vs. skipping a failing image and proceeding to the next one).

This approach enables the user to separately upgrade the three parts of ShMM firmware (U-Boot, kernel and RFS image). However, the user should bear in mind that the parts that are not explicitly updated will be copied from persistent Flash.

PPS recommends use of one of the following upgrade approaches:

- Explicitly upgrading all three partitions.
- When fewer than 3 partitions are explicitly upgraded, omitting the –d specifier; in that case, the special script mentioned above will automatically ensure that every upgrade is effectively a full upgrade covering all three partitions.

protocol specifies a file copy protocol used to pull each of the specified *src* files to the ShMM-500 and can be any of the following:

- `no` - No copy is performed. This protocol assumes that all of the specified *src* files were pushed to the /tmp directory prior to start of the reliable upgrade procedure. This protocol choice cannot be used in conjunction with the -d option.
- `cp:`*`dir`* - Simple copy. This protocol assumes that all of the specified *src* files are to be copied from the specified directory in the ShMM-500 local file system by the cp command. This protocol can be useful, for instance, for installation of upgrade images from an NFS mounted file system or even from a JFFS2 file system.
- `ftp:`*`server`*`:`*`dir`*`:`*`user`*`[:`*`pwd`*`]` - Copy from a remote FTP server. This protocol assumes that all of the specified *src* files are to be copied to the ShMM-500 from the FTP server host specified as the host name or the IP address by *server*. All the images must reside in the directory specified by *dir* on the remote FTP server. The FTP connection is made using the account specified by the *user* parameter, with the password specified by the optional *pwd* parameter. If no *pwd* is supplied, the utility will prompt for a password.

A failure in copying an image to the ShMM-500 causes the utility to terminate the upgrade procedure (vs. skipping a failing image and proceeding to the next one).

For each provisional Flash partition upgraded by the -s option, the to-be-upgraded partition is given write permissions after the validity of the image has been checked and right before the *src* image is about to be moved to the Flash. Write permissions are removed from the partition immediately after the full image has been moved to Flash. Combined with the fact that all the partitions containing the U-Boot, Linux kernel, and root file system images are read-only on boot-up of the ShMM-500, this ensures that applications cannot accidentally erase the critical boot-up images.

After all the specified images have been installed to their respective destinations in Flash, the utility invokes a "hook" script that enables custom actions required by an application at the point where the upgrade images have been already installed in Flash but the upgrade procedure has not yet initiated the hardware mechanisms of the reliable upgrade procedure by enabling the ShMM-500's upgrade watchdog timer (WDT). (Refer to ML User's Guide Chapter 6 for background and details on the upgrade WDT.)

The hook script `/etc/upgrade/step4hshm` is supplied with the Shelf Manager. It performs the following actions:

- terminates the Shelf Manager, performing a switchover to the backup ShMM without restarting the shelf; the ATCA watchdog timer is stopped.
- mounts the provisional /etc and /var Flash partitions and erases all files on them
- optionally copies the current contents of the /etc directory to the provisional /etc Flash partition
- optionally copies the current non-volatile Shelf Manager information from the directory /var/nvdata to the provisional /var file system; or optionally copies the whole /var directory to the provisional /var Flash partition.
- temporarily (just for the next boot), sets the boot delay to 0; this is done to minimize the time of the next boot and prevent the reliable upgrade watchdog timer from premature expiration.

This script is invoked as a sub-shell and given a single parameter, which is either the string specified by *args* or an empty string if no *args* is supplied. The parameter defines the mode of operation of the script with respect to copying non-volatile information from the persistent Flash partitions to the provisional Flash partitions, and can take the following values, each of which triggers a corresponding set of actions:

- no parameter supplied – the script erases both the provisional /etc and provisional /var directories, then copies Shelf Manager non-volatile information from the directory /var/nvdata to the provisional /var partition. This is the default mode of operation; in this case, the non-volatile data will be preserved but the Shelf Manager configuration file will be taken from the new RFS image.
- `erase` – the script erases both the provisional /etc and provisional /var directories; they will be restored from the RFS default values during the next boot; the current Shelf Manager non-volatile data and configuration are not preserved.
- `etc_copy` – the script erases both the provisional /etc and provisional /var directories; then it copies the contents of /etc and the non-volatile information from the directory /var/nvdata to

the provisional Flash partitions. In this case, both the non-volatile data and the Shelf Manager configuration file are preserved.

- `copy` - the script erases both the provisional /etc and provisional /var directories, then copies the full contents of the /etc and /var directories onto the provisional partition. In this case, not only the configuration, but also the executable files placed to /var/bin will be copied and will override executable files with the same name from the RFS image. This mode of operation is useful if the directory /var/bin contains some special executables (e.g. a special version of the Shelf Manager or other utilities) that must be preserved across the upgrade.

The script returns 0 on success and non-zero for failure. If a non-zero value is returned, the upgrade procedure is terminated.

The utility starts the upgrade WDT with a 12.8 sec timeout period. This timeout period is considered sufficient for any software that will boot after the reset to proceed to the point where it is able to call `rupgrade_tool -c` (which strobes the upgrade WDT in case it is active) without having to strobe the upgrade WDT in the interim. The utility performs a strobe of the upgrade WDT just before resetting the ShMM-500.

`-c [-v]`

Proceed with the reliable upgrade procedure after the ShMM-500 is booted from the provisional Flash. The invocation of `rupgrade_tool -c` is done from the /etc/rc script. As described below, certain situations discovered by `rupgrade_tool -c` imply a failure in the upgrade procedure and require corrective actions, including those resulting in the need to soft reset the ShMM-500. This means that an invocation of `rupgrade_tool -c` may not return and instead may result in reset of the ShMM-500. If a reset takes place, it reverts the ShMM-500 to the software installed in the persistent Flash.

If the upgrade WDT is active and has fired at any step prior to invocation of `rupgrade -c`, this means that the ShMM-500 already reverted to the software in the persistent Flash. In this scenario, the utility disables the upgrade WDT and returns to the use of persistent Flash and terminates the upgrade procedure.

If the upgrade WDT is active but has not fired, this means that the ShMM-500 successfully booted (up to this point) from the provisional Flash. The utility strobes the upgrade WDT and exits with the return code of 0 indicating that there is an upgrade procedure session in progress.

If the upgrade WDT is not active but the content of the /var/upgrade/status file indicates that the upgrade procedure is still in progress, this means that the ShMM-500 rebooted due to a power-cycle at one of the steps of the upgrade procedure. In this scenario the utility performs the same corrective actions as for the situation when the upgrade WDT is active and has fired (see above).

Finally, if the upgrade WDT is not active and /var/upgrade/status is either not present or indicates that the upgrade procedure has finished (either successfully or unsuccessfully), the utility exits with the return value of 1, indicating that there is no upgrade procedure in progress.

`-f [--hook=`*`args`*`] [-v]`

Complete the upgrade procedure. The invocation of `rupgrade_tool -f` is done from inside the Shelf Manager after the Shelf Manager successfully completes its initialization. If the Shelf Manager is not started automatically, that invocation is done at the end of the /etc/rc script.

As soon as invoked, `rupgrade_tool -f` strobes the upgrade WDT and proceeds with establishing the new persistent Flash and disabling the upgrade WDT.

After disabling the upgrade WDT, the upgrade procedure can invoke a special script to take any actions necessary after the hardware resources associated with a reliable upgrade have been returned to their normal state, but before a completion record has been added to the status file. This feature is not currently used during reliable upgrades of the Shelf Manager. See the ML User's Guide Chapter 6 for details of this upgrade finalization option.

As the last step, the utility updates /var/upgrade/status with a record indicating a successful completion of the upgrade procedure and exits with a value of 0.

`-w [-f]`

Print the current status of the most recent upgrade procedure. Essentially, this option dumps the content of the /var/upgrade/status file to stdout.

`rupgrade_tool -w` returns a value of 0 if the upgrade procedure has completed successfully, 1 if the upgrade procedure has completed unsuccessfully, an appropriate error code if there is no /var/upgrade/status file to be found.

If the -f specifier is supplied, `rupgrade_tool -w` removes the /var/upgrade/status file before exiting.

`-S [-v]`

Strobe the upgrade WDT. `rupgrade_tool -S` is intended as a shell-level interface to strobe the upgrade WDT, for use by newly installed software that is validating its sanity.

`rupgrade_tool -S` returns a value of 0.

`-u`

Undo a successful upgrade session, reverting to the previous persistent Flash device.

`rupgrade_tool -u` causes the ShMM to reboot.

`-h`

Print help to stdout.

6.1.6. Reliable Upgrade Utility Use Scenarios

It is intended that the reliable upgrade utility (refer to 6.1.5) will be used in the following sequence in order to carry out an upgrade of the ShMM-500:

- The user makes a call to `rupgrade_tool -s` to initiate the upgrade procedure. The call can be made either locally from the ShMM-500 serial console or remotely over the network via telnet, rsh, ssh, or any equivalent.
- The user waits for the `rupgrade_tool -s` to reboot the ShMM-500. If the user is connected to the serial console locally, the status of the reboot is obvious from the messages printed by the U-Boot firmware and Linux to the serial console. If the connection to the ShMM-500 is remote, the status of the reboot is less obvious. For instance, a telnet connection will timeout on the reboot of the ShMM-500. The user can either assume that the upgrade procedure has been carried out successfully or wait for a certain amount of time required for the upgrade session to

complete and then make a call to rupgrade_tool –w (again, remotely, over any of the remote shell tools mentioned above) in order to find out the status of the upgrade session. The amount of time to wait depends on the size of the upgrade images and the copy protocol used to pull the images to the ShMM-500 as well as actions performed by the image validation script.

- On the ShMM-500, the startup script /etc/rc unconditionally makes a call to `rupgrade_tool –c`. If the call returns a value of 1, indicating that there is no upgrade in progress or an error code value indicating that the upgrade session has failed, the startup scripts proceed with the normal mode boot-up sequence. If however, a value of 0 is returned, indicating that there is an upgrade session in progress, the startup scripts proceed with validation of the sanity of the newly installed software, calling `rupgrade_tool -S` in the middle of operation to strobe the upgrade WDT in case the validation takes longer than the upgrade WDT timeout period, and finally start the Shelf Manager to perform final validation. The watchdog timer interval is set to 12.8 seconds; so the processing time in the /etc/rc script between the call to `rupgrade_tool –c` and strobing the WDT and between strobing the WDT and starting the Shelf Manager must not exceed 12.8 seconds each.
- During initialization, the Shelf Manager strobes the upgrade WDT once again, before trying to establishing a network connection with the peer Shelf Manager. Establishing a network connection may take up to 6 seconds. After that, and after successfully finishing the initialization (which indicates validity of the new configuration), the Shelf Manager makes a call to `rupgrade_tool –f`, which completes the upgrade procedure.
- The user optionally calls `rupgrade_tool –w` in order to find out the status of the upgrade session. As explained above, this option may be especially useful for a remote upgrade session where the progress of the upgrade cannot be observed directly from the messages printed to the serial console, as is the case for a local upgrade.
- After the completion of the reliable upgrade, the user can revert to the original images if he/she detects that the new images are not acceptable for any reason. To do this, the user calls `rupgrade_tool –u`.

If necessary, the above sequence can be easily automated by developing a simple script designed to run on a remote network host. Alternatively, an operator can carry out the reliable upgrade manually, either locally from the serial console or remotely over the network.

6.1.7. Reliable Upgrade Examples

Example 1: This example shows a reliable upgrade of all three components (U-Boot, kernel and RFS image), copying /etc and /var/nvdata non-volatile directories to the provisional Flash. All images are taken from the local /tmp (which implies that they have already been copied there in some unspecified way). The U-boot image is taken from /tmp/u-boot.bin, the kernel image is taken from /tmp/sentry.kernel, the RFS image is taken from /tmp/sentry.rfs. The upgrade procedure is started from the serial console. Comments are interspersed in the console log to provide additional background on the steps of the upgrade procedure.

First, the rupgrade_tool is started from the command prompt. The parameters show that all three Flash images are to be updated, with the Shelf Manager non-volatile data and configuration file preserved as well.

```
# rupgrade tool -s --k=sentry.kernel --r=sentry.rfs --u=u-boot.bin --hook=etc copy –v
rupgrade tool: PLB is 5
rupgrade_tool: EEPROM page saved
```

```
rupgrade_tool: persistent flash is 0
rupgrade_tool: provisional flash is 1
rupgrade_tool: copying image(s)
```

The upgrade utility attempts to invoke a validation script to check the images in /tmp currently supplied. If any of the specified file designators is not found in /tmp, the utility stops and a message like the following is produced.

```
rupgrade_tool: cannot open /tmp/u-boot.bin for reading.
rupgrade_tool: failed to copy images to flash
```

The utility proceeds to copy the images to the specified destinations in provisional Flash.

```
rupgrade_tool: invoking scripts (step4v*) [--u=u-boot.bin --k=sentry.kernel --r=sentry.rfs --
hook=etc_copy]
rupgrade_tool: copying u-boot.bin from /tmp to /dev/mtdchar8 using 'cp' protocol
rupgrade_tool: copying sentry.kernel from /tmp to /dev/mtdchar7 using 'cp' protocol
rupgrade_tool: copying sentry.rfs from /tmp to /dev/mtdchar9 using 'cp' protocol
rupgrade_tool: invoking scripts (step4h*) [etc_copy]
```

At this point, the step4hshm hook script is invoked; it stops the Shelf Manager and copies non-volatile information to the provisional Flash.

```
/etc/upgrade/step4hshm: Stopping Shelf Manager...
/etc/upgrade/step4hshm: Erasing /var and /etc, copying /var/nvdata...
/etc/upgrade/step4hshm: Operation: copy /etc and /var/nvdata.
/etc/upgrade/step4hshm: Copying completed.
rupgrade_tool: image(s) copy OK
rupgrade_tool: watchdog started
rupgrade_tool: selected provisional flash
rupgrade_tool: reboot
Restarting system.
```

Here, the reliable upgrade procedure resets the ShMM-500. This causes U-boot to start from the provisional Flash.

```
** Resetting Integrated Peripherals

U-Boot 1.1.2 (May 12 2005 - 21:27:13)

CPU: Au1550 324 MHz, id: 0x02, rev: 0x00
Board: ShMM-500
S/N: 08000412
DRAM:  64 MB
Flash: 16 MB
In:    serial
Out:   serial
Err:   serial
Net:   Au1X00 ETHERNET
Hit any key to stop autoboot:  0
## Booting image at bfb00000 ...
   Image Name:   MIPS Linux-2.4.26
   Created:      2005-06-24  13:29:50 UTC
   Image Type:   MIPS Linux Kernel Image (gzip compressed)
   Data Size:    844843 Bytes = 825 kB
   Load Address: 80100000
   Entry Point:  802bc040
   Verifying Checksum ... OK
   Uncompressing Kernel Image ... OK
## Loading Ramdisk Image at bfc40000 ...
   Image Name:   sentry RFS Ramdisk Image
   Created:      2005-06-27   10:51:03 UTC
   Image Type:   MIPS Linux RAMDisk Image (gzip compressed)
   Data Size:    2465924 Bytes =  2.4 MB
   Load Address: 00000000
   Entry Point:  00000000
   Verifying Checksum ... OK

Starting kernel ...
```

```
init started:  BusyBox v0.60.5 (2005.06.15-14:45+0000) multi-call binary
/etc/rc: Mounted /proc
/etc/rc: Mounting filesystems...
/etc/rc: Mounted /dev/pts
/etc/rc: Mounted /dev/mtdblock0 to /var
/etc/rc: Mounted /dev/mtdblock10 to /var/upgrade
```

At this point in the execution of the rc script, it invokes `rupgrade_tool -c` to check whether a reliable upgrade is in progress. The tool returns 0, confirming that an upgrade is in progress. Given that result, the rc script continues with the startup process.

```
etc/rc: Checking the reliable upgrade watchdog timer...activated
/etc/rc: Mounted ram disk to /var/log
/etc/rc: Started syslogd and klogd
/etc/rc: Mounted ram disk to /var/tmp
/etc/rc: Setting hostname shmm+193
```

Since a reliable upgrade is in progress, the watchdog timer is strobed once more in the rc script.

```
/etc/rc: Strobing the reliable upgrade watchdog timer
/etc/rc: Mounted /dev/mtdblock1 to /etc
/etc/rc: Calling /etc/rc.carrier3
Board Hardware Address: 0xFE
/etc/netconfig: /etc/hosts has valid  192.168.1.193 entry
/etc/netconfig: Updating /etc/profile.sentry with IP settings
/etc/netconfig: ifconfig eth0 192.168.1.193
/etc/netconfig: ifconfig eth1 192.168.0.193
/etc/netconfig: route add default gw 192.168.1.253
/etc/netconfig: Starting inetd...
/etc/rc.carrier3: Starting up IPMBs ...
/etc/rc.carrier3: Updating /etc/profile.sentry with specific settings
/etc/rc.carrier3: Starting snmpd...
/etc/rc.carrier3: Starting httpd...
/etc/rc.carrier3: Starting Shelf Manager ...
<I> 02:48:08.463   [171] Pigeon Point Shelf Manager ver. 2.1.1. Built on Aug 27 2005 14:48:57
<*> 02:48:08.469   [171] Limits: code=(400000:5076f0), end data=10062000, start stack=7fff7e30,
esp=7fff78a0, eip=2ab0d2e4
```

The Shelf Manager starts and finalizes the reliable upgrade, calling `rupgrade_tool -f`.

```
eth0: link up
eth1: link up
eth1: going to full duplex

shmm+193 login:root


BusyBox v0.60.5 (2005.05.12-22:46+0000) Built-in shell (msh)
```

Finally, the user checks the status of the reliable upgrade, by calling `rupgrade_tool -w`.

```
# rupgrade_tool -w
Recent upgrade status:
1:PLB is 5
1:EEPROM page saved
2:persistent flash is 0
3:provisional flash is 1
4:copying image(s)
4:invoking scripts (step4v*) [--u=u-boot.bin --k=sentry.kernel --r=sentry.rfs  --hook=etc copy]
4:copying u-boot.bin from /tmp to /dev/mtdchar8 using 'cp' protocol
4:copying sentry.kernel from /tmp to /dev/mtdchar7 using 'cp' protocol
4:copying sentry.rfs from /tmp to /dev/mtdchar9 using 'cp' protocol
4:invoking scripts (step4h*) [etc copy]
4:image(s) copy OK
5:watchdog started
6:selected provisional flash
7:reboot
9:WDT not fired, upgrade in progress.
```

```
11:provisional flash 1, updating EEPROM
12:EEPROM updated
13:upgrade WDT disabled
13:invoking scripts (step13h*) []
14:upgrade completed successfully
#
```

Example 2: This example shows a reliable upgrade of the RFS image only, copying /etc and /var/nvdata non-volatile directories to provisional Flash. The RFS image is taken from an FTP server at the IP address 192.168.1.253. The path to the RFS image on the FTP server is /tftpboot/ru-mips/sentry.mips.rfs. The upgrade procedure is started from a telnet session.

IMPORTANT NOTE: Since only the RFS image is explicitly updated, the U-Boot and kernel images are automatically copied from the persistent flash partition to the provisional flash partition.

The local system must be able to access the FTP server over the network (that is, its network adapter must be up and configured and a route must exist from the ShMM to the FTP server). In the example below, the ShMM is configured with the network address 192.168.1.174 (which is in the same network with the FTP server):

```
# telnet 192.168.1.174
Trying 192.168.1.174...
Connected to 192.168.1.174.
Escape character is '^]'.

BusyBox on shmm+174 login: root


BusyBox v0.60.5 (2005.05.07-17:27+0000) Built-in shell (msh)
```

The parameters to `rupgrade_tool -s` indicate that only the RFS is being upgraded and that the copy protocol is FTP, accessing a specified IP address and file, with user `admin` and no password supplied.

```
# rupgrade_tool -s --r=sentry.mips.rfs --proto=ftp:192.168.1.253:/tftpboot/ru-mips:admin --hook=etc_copy -
v
rupgrade_tool: PLB is 5
rupgrade_tool: EEPROM page saved
rupgrade_tool: persistent flash is 1
rupgrade_tool: provisional flash is 0
rupgrade_tool: copying image(s)
rupgrade_tool: copying sentry.mips.rfs from 192.168.1.253:/tftpboot/ru-mips:admin to /tmp using 'ftp'
protocol
220 hydra FTP server (Version wu-2.4.2-academ[BETA-17](1) Tue Jun 9 10:43:14 EDT 1998) ready.
USER admin
```

The user is asked here for a password to the FTP site; that password is entered manually.

```
331 Password required for admin.
Password:
PASS *****
230 User admin logged in.
TYPE I
200 Type set to I.
PASV
227 Entering Passive Mode (192,168,1,253,9,20)
RETR /tftpboot/ru-mips/sentry.mips.rfs
150 Opening BINARY mode data connection for /tftpboot/ru-mips/sentry.mips.rfs (2465988 bytes).
226 Transfer complete.
QUIT
221 Goodbye.
```

In the next step, a special script step4vshm is invoked, that copies the U-Boot and kernel images from the persistent Flash to the provisional Flash, After that, the upgrade utility proceeds to copy the RFS image to its designated position in provisional Flash.

```
rupgrade_tool: invoking scripts (step4v*) [--r=sentry.mips.rfs --proto=ftp:192.168.1.253:/tftpboot/ru-
mips:admin --hook=etc copy]
/etc/upgrade/step4vshm: Erasing /dev/mtdchar7...Done
/etc/upgrade/step4vshm: Copying Kernel from /dev/mtdchar2 to /dev/mtdchar7...Done
/etc/upgrade/step4vshm: Erasing /dev/mtdchar8...Done
/etc/upgrade/step4vshm: Copying U-Boot from /dev/mtdchar3 to /dev/mtdchar8...Done
rupgrade_tool: copying sentry.mips.rfs from /tmp to /dev/mtdchar9 using 'cp' protocol
```

The step4hshm hook script is invoked, which stops the Shelf Manager and preserves the non-volatile data. The utility then starts the upgrade WDT and reboots.

```
rupgrade tool: invoking scripts (step4h*) [etc copy]
/etc/upgrade/step4hshm: Stopping Shelf Manager...
/etc/upgrade/step4hshm: Erasing /var and /etc, copying /var/nvdata...
/etc/upgrade/step4hshm: Operation: copy /etc and /var/nvdata.
/etc/upgrade/step4hshm: Copying completed.
rupgrade tool: image(s) copy OK
rupgrade tool: watchdog started
rupgrade_tool: selected provisional flash
rupgrade tool: reboot
Restarting system.
Connection closed by foreign host.
```

At this point, the telnet session is closed after a certain inactivity period; after several seconds, it is possible to reconnect to the target again and check the status of the reliable upgrade by invoking `rupgrade_tool -w`.

```
# telnet 192.168.1.174
Trying 192.168.1.174...
Connected to 192.168.1.174.
Escape character is '^]'.

BusyBox on shmm+174 login: root


BusyBox v0.60.5 (2005.05.07-17:27+0000) Built-in shell (msh)
#
# rupgrade tool -w
Recent upgrade status:
1:PLB is 5
1:EEPROM page saved
2:persistent flash is 1
3:provisional flash is 0
4:copying image(s)
4:copying sentry.mips.rfs from 192.168.1.253:/tftpboot/ru-mips:admin to /tmp using 'ftp' protocol
4:invoking scripts (step4v*) [--r=sentry. rfs --hook=etc copy]
4:copying sentry.mips.rfs from /tmp to /dev/mtdchar9 using 'cp' protocol
4:invoking scripts (step4h*) [etc_copy]
4:image(s) copy OK
5:watchdog started
6:selected provisional flash
7:reboot
9:WDT not fired, upgrade in progress.
11:provisional flash 0, updating EEPROM
12:EEPROM updated
13:upgrade WDT disabled
13:invoking scripts (step13h*) []
14:upgrade completed successfully
#
```

Example 3: This example shows an unsuccessful reliable upgrade. Power is turned off after the boot from the provisional Flash, but before the reliable upgrade is finalized. After turning the power back on, the rollback to the persistent Flash occurs. This reliable upgrade is initiated from the serial console. All three images are assumed to be already in /tmp.

```
# rupgrade_tool -s --k=sentry.kernel --r=sentry.rfs --u=u-boot.bin --hook=etc_copy -v
rupgrade_tool: PLB is 5
rupgrade_tool: EEPROM page saved
rupgrade_tool: persistent flash is 0
rupgrade_tool: provisional flash is 1
rupgrade_tool: copying image(s)
rupgrade_tool: invoking scripts (step4v*) [--u=u-boot.bin --k=sentry.kernel --r=sentry.rfs --
hook=etc_copy]
rupgrade_tool: copying u-boot.bin from /tmp to /dev/mtdchar8 using 'cp' protocol
rupgrade_tool: copying sentry.kernel from /tmp to /dev/mtdchar7 using 'cp' protocol
rupgrade_tool: copying sentry.rfs from /tmp to /dev/mtdchar9 using 'cp' protocol
rupgrade_tool: invoking scripts (step4h*) [etc_copy]
Stopping Shelf Manager...

Pigeon Point Shelf Manager Command Line Interpreter

Terminating the Shelf Manager
Erasing /var and /etc, copying /var/nvdata...
Operation: copy /etc and /var/nvdata.
Copying completed.
rupgrade_tool: image(s) copy OK
rupgrade_tool: watchdog started
rupgrade_tool: selected provisional flash
rupgrade_tool: reboot
Restarting system.
```

The reliable upgrade procedure resets the ShMM here and starts U-boot from the provisional Flash.

```
** Resetting Integrated Peripherals


U-Boot 1.1.2 (Apr 11 2005 - 15:16:25)

CPU: Au1550 324 MHz, id: 0x02, rev: 0x00
Board: ShMM-500
S/N: 8000044
DRAM:  64 MB
Flash: 16 MB
In:    serial
Out:   serial
Err:   serial
Net:   Au1X00 ETHERNET
Hit any key to stop autoboot:  0
## Booting image at bfb00000 ...
   Image Name:   MIPS Linux-2.4.26
   Created:      2005-04-11  10:35:08 UTC
   Image Type:   MIPS Linux Kernel Image (gzip compressed)
   Data Size:    843129 Bytes = 823.4 kB
   Load Address: 80100000
   Entry Point:  802bc040
   Verifying Checksum ... OK
   Uncompressing Kernel Image ... OK
## Loading Ramdisk Image at bfc40000 ...
   Image Name:   sentry RFS Ramdisk Image
   Created:      2005-04-22   9:10:41 UTC
   Image Type:   MIPS Linux RAMDisk Image (gzip compressed)
   Data Size:    2400736 Bytes =  2.3 MB
   Load Address: 00000000
   Entry Point:  00000000
   Verifying Checksum ... OK
```

Power is turned off here. After some time, power is turned back on. Assignment of provisional flash has been lost because of the power loss, so the system reverts back to the persistent flash.

```
U-Boot 1.1.2 (Apr 11 2005 - 15:16:25)

CPU: Au1550 324 MHz, id: 0x02, rev: 0x00
```

```
Board: ShMM-500
S/N: 8000048
DRAM:  64 MB
Flash: 16 MB
In:    serial
Out:   serial
Err:   serial
Net:   Au1X00 ETHERNET
Hit any key to stop autoboot:  0
## Booting image at bfb00000 ...
   Image Name:   MIPS Linux-2.4.26
   Created:      2005-04-11  10:35:08 UTC
   Image Type:   MIPS Linux Kernel Image (gzip compressed)
   Data Size:    843129 Bytes = 823.4 kB
   Load Address: 80100000
   Entry Point:  802bc040
   Verifying Checksum ... OK
   Uncompressing Kernel Image ... OK
## Loading Ramdisk Image at bfc40000 ...
   Image Name:   sentry RFS Ramdisk Image
   Created:      2005-04-11  18:27:17 UTC
   Image Type:   MIPS Linux RAMDisk Image (gzip compressed)
   Data Size:    2372311 Bytes =  2.3 MB
   Load Address: 00000000
   Entry Point:  00000000
   Verifying Checksum ... OK

Starting kernel ...

init started:  BusyBox v0.60.5 (2005.02.07-16:45+0000) multi-call binary
hub.c: new USB device AU1550-1, assigned address 2
usb0: ? speed config #1: Ethernet Gadget
usb1: register usbnet usb-AU1550-1, Linux Device
serial#=8000048: not found
/etc/rc: Mounted /proc
/etc/rc: Mounting filesystems...
/etc/rc: Mounted /dev/pts
/etc/rc: Mounted /dev/mtdblock0 to /var
/etc/rc: Mounted /dev/mtdblock10 to /var/upgrade
```

The next step in the rc script is to call `rupgrade_tool -c` to check whether a reliable upgrade is in progress. The check determines that an attempted reliable upgrade failed. The message “restoring ADM1060 EEPROM to RAM” refers to the ShMM-500 system supervisory device (an ADM1060), which supervises the ShMM-500 boot up process and implements some of the hardware aspects of the reliable upgrade support. This message indicates that key variables affecting the boot process are being reverted to their state before the reliable upgrade was attempted.

```
/etc/rc: Checking the reliable upgrade watchdog timer
rupgrade tool: Watchdog not active.
rupgrade tool: restoring ADM1060 EEPROM to RAM
rupgrade_tool: upgrade failed
/etc/rc: Rupgrade -c Ret: 255
/etc/rc: Mounted ram disk to /var/log
/etc/rc: Started syslogd and klogd
/etc/rc: Mounted ram disk to /var/tmp
/etc/rc: Setting hostname shmm+173
/etc/rc: Mounted /dev/mtdblock1 to /etc
/etc/rc: Calling /etc/rc.carrier3
Board Hardware Address: 0xFE
/etc/netconfig: /etc/hosts has valid  192.168.1.173 entry
/etc/netconfig: Updating /etc/profile.sentry with IP settings
/etc/netconfig: Starting inetd...
/etc/rc.carrier3: Starting up IPMBs ...
/etc/rc.carrier3: Updating /etc/profile.sentry with specific settings
/etc/rc.carrier3: RC2 daemons not started by request
```

PART III: Re-initializing and Re-programming the ShMM-300

Chapter 7

7. Re-initializing the ShMM-300

This chapter describes how to re-initialize the U-Boot environment variables, the filesystem in Flash and the login password—all on the ShMM-300.

7.1. Re-initializing the ARMBoot Environment

The ARMboot variables are stored in Flash. If you would like to restore the factory defaults for the ARMboot variables, you must first erase the environment variables stored in Flash and power cycle the ShMM-300.

To erase the Flash, you first need to unprotect the Flash area where ARMboot stores these values.

```
ShMM # protect off 2:0-1
Un-Protect Flash Sectors 0-1 in Bank # 2
ShMM #
```

You then need to erase those sectors.

```
ShMM # erase 2:0-1
Erase Flash Sectors 0-1 in Bank # 2:
Erasing sector  0 ... Erasing sector at 0x  800000
ok.
Erasing sector  1 ... Erasing sector at 0x  802000
ok.
Done.
ShMM #
```

Finally, you need to re-protect the Flash and power off the ShMM-300.

```
ShMM # protect on 2:0-1
Protect Flash Sectors 0-1 in Bank # 2
ShMM #
```

7.2. Re-initializing the File System

The filesystem is stored within the Flash and can be reset to factory defaults quite easily. ARMboot has an environment variable called `flash_reset`. By setting this variable to "y" and then booting up the system, the file system will be re-initialized to factory defaults.

```
ShMM # setenv flash_reset y
ShMM # run bootcmd
```

Make sure that you don't save the environment variables after setting the `flash_reset` variable, otherwise the file system will be restored every time the system is brought up. The `run bootcmd` command (the second command above) will begin booting the Linux kernel. It is during this process that the file system is re-initialized. The following output will be shown on the console.

```
/etc/rc: Mounted /dev/pts
/etc/rc: Flash erase requested via ARMBOOT var
/etc/rc: erasing mtdchar2 -> /etc
Erased 448 Kibyte @ 0 -- 100% complete.
/etc/rc: erasing mtdchar3 -> /var
Erased 2560 Kibyte @ 0 -- 100% complete.
/etc/rc: Mounted /dev/mtdblock3 to /var
/etc/rc: /var/log mounted as FLASH disk
/etc/rc: Started syslogd and klogd
/etc/rc: /var/tmp mounted as RAM disk
/etc/rc: hostname demo
/etc/rc: /dev/mtdblock2 appears to be empty ... restoring from factory
/etc...
```

7.3. Resetting the Login Password

The factory default login for the ShMM-300 is a user id of "`root`" without any password. We highly encourage users to change the password when configuring the shelf manager. In the event that the new password is forgotten, the password can be reset to its factory default via the `password_reset` ARMboot variable. By setting this variable to "y" and then booting up the system, the root password will be removed.

```
ShMM # setenv password_reset y
ShMM # run bootcmd
```

The following output will be shown on the console, during boot up.

```
/etc/rc: hostname demo
/etc/rc: Restoring password file to factory default
```

Chapter

8

8. Re-programming the ShMM-300

From time to time, new software updates will be made available. In order for these updates to be programmed down onto the ShMM-300, an ARM Injector and the Monterey Linux build environment are required (www.montereylinux.com). The ARM Injector will allow programming both the CPLD and the Flash built into the ShMM-300.

Figure 4.1: Pigeon Point Systems ARM Injector

8.1. Verifying Operation of the ARM Injector

Once the desktop Monterey Linux development environment has been set up, the USB ARM Injector device is ready to be plugged in so that you can verify its operation.

1. Plug the ARM Injector USB cable in to both the ARM Injector and to your desktop Linux environment. You should see the ARM Injector LEDs blink twice after about 4-5 seconds.

2. Verify that the injector has powered up, and the resident USB ARM Injector firmware is responding:

```
ML(arm7tdmi):  inj info –a
           - /001/001 0000:0000:00.00 #14a0
     loader - /001/007 116B:414B:100.01 #PPS-000
```

The last line of the output indicates that the ARM Injector has been found and is ready for operation.

8.2. Programming the CPLD

The ShMM-300 includes a CPLD device. This CPLD device is responsible for controlling several key aspects of ShMM-300 operation, such as the hardware level redundancy interface. Here are detailed instructions on reprogramming the CPLD image via the ARM Injector.

1. Unplug the ARM Injector USB cable from your Linux host.
2. Unplug the AUX and JTAG cables from your ShMM-300 carrier board.
3. Make sure a ShMM-300 is plugged into your carrier board.
4. Plug in the ARM Injector USB cable on your host. Wait 10 seconds.
5. Run the following command:

```
ML(arm7tdmi):  inj info -a
            - /001/001 0000:0000:00.00 #10a0
     loader - /001/017 116B:414B:100.01 #PPS-000
```

   You should see the lines above.

6. Run the command:

```
ML(arm7tdmi):  inj ezload –f $CDE_ROOT/etc/injector/fw_cpld.hex 116b
```

7. Wait another 10 seconds, then issue:

```
ML(arm7tdmi):  inj ctl –p off 116b
```

8. Plug the ARM Injector ribbon cable (both AUX and JTAG connectors) into the ShMM-300 carrier board .
9. Turn on the carrier board using the ARM Injector interface:

```
ML(arm7tdmi):  inj ctl –p on 116b
```

10. Issue the commands:

```
ML(arm7tdmi): inj ctl –3 on 116b
ML(arm7tdmi): inj cpld –vui 116b
CPLD  ID: 0x39604093
CPLD UID: 0x68737330
```

   You should see lines like above displayed.

11. Issue the command:

```
ML(arm7tdmi):  inj cpld –x $CDE_ROOT/etc/injector/<cpld_image_name>.xsvf –v 116b
```

Green and Yellow ARM Injector LEDS should blink for 30 seconds.

12. Issue the commands:

```
ML(arm7tdmi): inj ctl –3 off 116b
ML(arm7tdmi): inj ctl –p off 116b
```

13. Disconnect the ARM Injector cables (AUX and JTAG) from the carrier board.

8.3. Programming the Flash

In addition to the CPLD, the Flash also requires programming. The Flash contains the ARMboot firmware, the Linux kernel, and the root file system (containing the shelf manager). Each of these components are separately programmed to the Flash.

1. Load the secondary ARM Injector Firmware (required for programming ShMM-300 Flash):

```
ML(arm7tdmi):  inj fwload –f $CDE_ROOT/etc/injector/fw.hex –v 116b
Loading firmware from file …/ml1.0/etc/injector/fw.hex...done
(transfered 1148 packets, 18089 bytes total)
```

2. Turn off the ARM Injector switch that will control the carrier board power:

```
ML(arm7tdmi):  inj ctl –p off 116b
```

3. Plug the ARM Injector ribbon cable (both AUX and JTAG connectors) into the carrier board.

4. Turn on the carrier board using the ARM Injector interface:

```
ML(arm7tdmi):  inj ctl –p on 116b
```

5. Program the new ARMboot.bin image to the ShMM-300 using the ARM Injector connection:

```
ML(arm7tdmi):  inj install –v –f $CDE_ROOT/arm7tdmi/boot/armboot.bin –a 0x0 116b
=================================================================
Using init script        ml1.0/etc/injector/initscript
Using flash programmer   ml1.0/arm7tdmi/boot/fprog.elf
Using target image file  armboot.bin
=================================================================
WARNING: old contents of the target Flash device will be lost!
Proceed? (y/n) y
...dumping init script
...dumping flash programmer
...section .text (vma = 0x10008074, size = 0x6c0)
...section .data (vma = 0x10010734, size = 0x7c)
...section .fdev_table (vma = 0x100107b0, size = 0x8)
...dumping image data
Progress: [##############################################################] (100%)
...done (transfer rate = 40720 bytes/sec)
```

```
ML(arm7tdmi):[root@linux-c5471 mII]#
```

6. Program a new shmm.kernel image to the ShMM-300

```
ML(arm7tdmi):  inj install –v –f shmm.kernel –a 0x20000 116b
===================================================================
Using init script        ml1.0/etc/injector/initscript
Using flash programmer   ml1.0/arm7tdmi/boot/fprog.elf
Using target image file shmm.kernel
===================================================================
WARNING: old contents of the target Flash device will be lost!
Proceed? (y/n) y
...dumping init script
...dumping flash programmer
...section .text (vma = 0x10008074, size = 0x6c0)
...section .data (vma = 0x10010734, size = 0x7c)
...section .fdev_table (vma = 0x100107b0, size = 0x8)
...dumping image data
Progress: [############################################################] (100%)
...done (transfer rate = 40753 bytes/sec)
ML(arm7tdmi):[root@linux-c5471 mII]#
```

7. Program a new shmm.rfs root filesystem to the ShMM-300.

```
ML(arm7tdmi):  inj install –v –f shmm.rfs -a 0x100000 116b
====================================================================
Using init script        ml1.0/etc/injector/initscript
Using flash programmer   ml1.0/arm7tdmi/boot/fprog.elf
Using target image file  shmm.rfs.192.168.0.54
====================================================================
WARNING: old contents of the target Flash device will be lost!
Proceed? (y/n) y
...dumping init script
...dumping flash programmer
...section .text (vma = 0x10008074, size = 0x6c0)
...section .data (vma = 0x10010734, size = 0x7c)
...section .fdev_table (vma = 0x100107b0, size = 0x8)
...dumping image data
Progress: [############################################################] (100%)
...done (transfer rate = 40750 bytes/sec)
ML(arm7tdmi):[root@linux-c5471 mII]#
```

8. Power off the ShMM-300 carrier board:

```
ML(arm7tdmi):  inj ctl –p off 116b
```

9. Disconnect the ARM Injector device from the ShMM-300 carrier board.

PART IV: Appendices

Appendix

A

A. Appendix A - Revision History

This section records the major revisions in this document, beginning with release 2.1.0.

A.1. Release 2.1.0

- Section 6.1.2: The boundaries of the Flash partitions maintained on the ShMM-500 by FOSL for 16MB Flash devices are corrected.
- Section 6.1.2: The boundaries of the Flash partitions maintained on the ShMM-500 by FOSL for 32MB Flash and 64MB Flash devices are covered.

A.2. Release 2.2.0

- Section 2.2.3: The algorithm for computation of IP address for endpoints of USB network interface is elaborated.
- Section 2.3: New configuration parameters are introduced: ALLOW_ALL_COMMANDS_FROM_IPMB, ALLOW_CHANGE_EVENT_RECEIVER, ALLOW_RESET_STANDALONE, DEFAULT_RMCP_NETMASK, IPMB_LINK_ISOLATION_TIMEOUT, MAX_INCOMING_IPMB_REQUESTS, MAX_OEM_FILTERS, SENSOR_POLL_INTERVAL, SHELF_FRU_IPMB_SOURCE1, SHELF_FRU_IPMB_SOURCE2, SWITCHOVER_ON_HANDLE_OPEN.
- Section 2.3.2.1: A shelf-wide search for potential sources of Shelf FRU Information on IPMB-0 can be limited to the two well-known IPMB-0 locations defined by the configuration variables SHELF_FRU_IPMB_SOURCE1, SHELF_FRU_IPMB_SOURCE2.
- Section 2.4: Local sensors can be configured when the Shelf Manager is started. The Sensor Device Records (SDRs) defining these sensors are read from the file "/var/nvdata/user_sdr".
- Section 3.2: The details of command-line invocations of the Shelf Manager are provided.
- Section 3.4: Shelf Manager operation on radial shelves is described in detail. .

A.3. Release 2.3.0

- Section 2.1.2: The U-Boot environment variable that was previously referenced as “gateway”, is now properly referenced as “gatewayip”.
- Section 2.2.21: The configuration parameter INITIAL_SLOW_LINK_DELAY is explained.
- Section 2.2.2.2: The parallel usage of two network interfaces is introduced. In parallel mode, instead of having a single RMCP network address that is switched between the two network interfaces, the Shelf Manager supports RMCP on both interfaces with different IP addresses, as two separate IPMI channels (channels 1 and 2).
- Section 2.2.5: IP addresses for the Shelf Manager can be assigned by a DHCP server.
- Section 2.3: New configuration parameters are introduced: COOLING_FAN_DECREASE_TIMEOUT, COOLING_FAN_INCREASE_TIMEOUT, CPLD_ACTIVE_WORKAROUND, DEFAULT_GATEWAY_IP_ADDRESS2, DEFAULT_RMCP_IP_ADDRESS2, INITIAL_SLOW_LINK_DELAY, FAN_LEVEL_STEP_DOWN, FAN_LEVEL_STEP_UP, IPMB_RETRY_TIMEOUT_MSEC, MAX_NODE_BUSY_RETRANSMISSIONS, NORMAL_STABLE_TIME, PREFERRED_DHCP_SERVER, SYSTEM_MANAGER_TRUNCATES_SEL, USE_DHCP, USE_SECOND_CHANNEL.
- Section 2.5: The Auxiliary Firmware Revision can be set when the Shelf Manager is started. It is read from the file “/var/nvdata/aux-fw-revision”.
- Section 2.6.1: A Shelf Manager can obtain date and time via the Network Time Protocol.
- Section 3.5: Automatic SEL truncation under control of the System Manager is introduced.