Basic Blade Services Software on ATCA-7540

Programmer's Reference

P/N: 6806871A04B

September 2019

Trademarks

The stylized "S" and "SMART" is a registered trademark of SMART Modular Technologies, Inc. and “SMART Embedded Computing” and the SMART Embedded Computing logo are trademarks of SMART Modular Technologies, Inc. All other names and logos referred to are trade names, trademarks, or registered trademarks of their respective owners. These materials are provided by SMART Embedded Computing as a service to its customers and may be used for informational purposes only.

Disclaimer*

SMART Embedded Computing (SMART EC) assumes no responsibility for errors or omissions in these materials. **These materials are provided "AS IS" without warranty of any kind, either expressed or implied, including but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement.** SMART EC further does not warrant the accuracy or completeness of the information, text, graphics, links or other items contained within these materials. SMART EC shall not be liable for any special, indirect, incidental, or consequential damages, including without limitation, lost revenues or lost profits, which may result from the use of these materials. SMART EC may make changes to these materials, or to the products described therein, at any time without notice. SMART EC makes no commitment to update the information contained within these materials.

Electronic versions of this material may be read online, downloaded for personal use, or referenced in another document as a URL to a SMART EC website. The text itself may not be published commercially in print or electronic form, edited, translated, or otherwise altered without the permission of SMART EC.

It is possible that this publication may contain reference to or information about SMART EC products, programming, or services that are not available in your country. Such references or information must not be construed to mean that SMART EC intends to announce such SMART EC products, programming, or services in your country.

Limited and Restricted Rights Legend

If the documentation contained herein is supplied, directly or indirectly, to the U.S. Government, the following notice shall apply unless otherwise agreed to in writing by SMART Embedded Computing.

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (b)(3) of the Rights in Technical Data clause at DFARS 252.227-7013 (Nov. 1995) and of the Rights in Noncommercial Computer Software and Documentation clause at DFARS 252.227-7014 (Jun. 1995).

SMART Embedded Computing, Inc.

2900 S. Diablo Way, Suite 190

Tempe, Arizona 85282

USA

*For full legal terms and conditions, visit *www.smartembedded.com/ec/legal*

Table of Contents

List of Figures

List of Figures

List of Tables

List of Tables

About this Manual

Overview of Contents

This manual is divided into the following chapters and appendices:

Abbreviations

This document uses the following abbreviations:

Abbreviation	Definition
ADF	Acceleration Driver Framework
AHCI	Advanced Host Controller Interface
AMC	Advanced Mezzanine Card
API	Application Program Interface
ATCA	Advanced Telecommunications Computing Architecture
BBS	Basic Blade Services
BIOS	Basic Input Output System
BLSV	Blade Services
BT	Block Transfer Interface
CPU	Central Processing Unit
FCU	FUF Command-line Utility

Abbreviation	Definition
FPGA	Field Programmable Gate Array
FRU	Field Replaceable Unit
FUF	Firmware Upgrade Facility
GPIO	General Purpose Input/Output
HPI	Hardware Platform Interface
HPM	Hardware Platform Management
IPMB	Intelligent Platform Management Bus
IPMC	Intelligent Platform Management Controller
IPMI	Intelligent Platform Management Interface
IQR	Interrupt Request
KCS	Keyboard Control Style
MAC	Media Access Control
MMC	Multi Media Card
NIC	Network Interface Card
NMU	Network Management Utility
NVM	Non-volatile Memory
OEM	Original Equipment Manufacturer
OS	Operating System
PCI	Peripheral Component Interconnect
PICMG	PCI Industrial Computers Manufacturers Group
PXE	Preboot Execution Environment
QSFP	Quad Small form-factor Pluggable
RPM	Red Hat Package Manager
RTM	Rear Transition Module

Abbreviation	Definition
SATA	Serial ATA
SDR	Sensor Data Record
SEL	System Event Log
SFP	Small Form-factor Pluggable
SIP	Serial Interface Protocol
SMI	Serial Management Interface
SNMP	Simple Network Management Protocol
SOL	Serial over LAN
USB	Universal Serial Bus

Conventions

The following table describes the conventions used throughout this manual. .

Notation	Description
`0x00000000`	Typical notation for hexadecimal numbers (digits are 0 through F), for example used for addresses and offsets
`0b0000`	Same for binary numbers (digits are 0 and 1)
bold	Used to emphasize a word
`Screen`	Used for on-screen output and code related elements or commands. `Sample of Programming used in a table (9pt)`
`Courier + Bold`	Used to characterize user input and to separate it from system output
Reference	Used for references and for table and figure descriptions
File > Exit	Notation for selecting a submenu
<text>	Notation for variables and keys
[text]	Notation for software buttons to click on the screen and parameter description
...	Repeated item for example node 1, node 2, ..., node 12
. . .	Omission of information from example/command that is not necessary at the time
..	Ranges, for example: 0..4 means one of the integers 0,1,2,3, and 4 (used in registers)
|	Logical OR
	Indicates a hazardous situation which, if not avoided, could result in death or serious injury
	Indicates a hazardous situation which, if not avoided, may result in minor or moderate injury

Notation	Description
	Indicates a property damage message
	Indicates a hot surface that could result in moderate or serious injury
	Indicates an electrical situation that could result in moderate injury or death
Use ESD protection 	Indicates that when working in an ESD environment care should be taken to use proper ESD practices
	No danger encountered, pay attention to important information

Summary of Changes

See the table below for manual revisions and changes.

Part Number	Date	Description
6806871A04B	September 2019	Re-branded SMART Embedded Computing
6806871A04A	October 2018	Initial version

Introduction

1.1 Overview

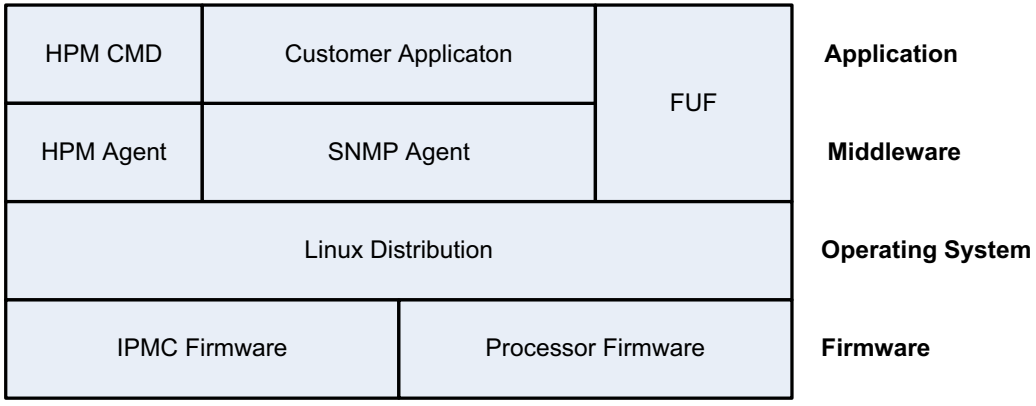
The Basic Blades Services (BBS) software provides packages that are useful on the blade on which the software is installed. BBS software includes packages for hardware management, firmware upgrade, and field upgrade of the programmable devices on the board.

1.2 Software Building Blocks

BBS includes a common set of functionality, which is available for all ATCA blades. It also includes a unique set of functionality, which is tailored particularly for this blade.

The following figure depicts the architecture of the BBS software.

Figure 1-1 BBS Architecture



HPM: Hardware Platform Management
FUF: Firmware Upgrade Facility
SNMP: Simple Network Management Protocol
IPMC: Intelligent Peripheral Management Controller

BBS for the ATCA-7540 consists of the following main software and services:

- Firmware Upgrade Facility (FUF): Provides a uniform way to upgrade firmware on SMART EC blades, regardless of the flash location. FUF can upgrade the Basic Input Output System (BIOS) firmware, the Intelligent Platform Management Controller (IPMC) firmware, and the FPGA via the Intelligent Platform Management Interface (IPMI). Currently, FUF consists of the FUF Command-line Utility (FCU), flash device drivers, and HPM image files.

- Linux Operating System: CentOS Linux 7.5 is the base operating system for the BBS. Various Linux services are started by the BBS.
- Hardware Platform Management: Hardware Platform Management (HPM) in ATCA systems is based on the IPMI specification. IPMI commands can be complex and cumbersome. HPM facilitates the blade and module level hardware management.

Basic Blade Services Software

2.1 Overview

The Basic Blade Services (BBS) software is available in the form of a PXE bootable Linux kernel and initramfs. The base operating system (OS) is CentOS 7.5. BBS software is installed as Red Hat Package Manager (RPM) packages in the initramfs. Tools in the initramfs can be used to create installations on M.2 NVME, AHCI, or SATA drives on the board.

BBS packages are also provided as RPMs that can be installed over a preinstalled CentOS 7.5 installation.

2.1.1 Package Information

The BBS distribution contains software packages listed in the following table.

Table 2-1 Software packages

Software	Package Name
HPM Agent command tool	bbs-hpmagentcmd-atca7540-<version>.rpm
Firmware upgrade utility	bbs-fcu-atca7540-<version>.rpm
Board control driver and related utilities	bbs-boardctrl-atca7540-<version>.rpm
BIOS image	bbs-bios-atca7540-<version>.rpm
Field Programmable Gate Array (FPGA) image	bbs-fpga-atca7540-<version>.rpm
IPMC Firmware	bbs-ipmc-atca7540-<version>.rpm
IPMC Boot Loader	bbs-ipmc-boot-atca7540-<version>.rpm
NMU Package	bbs-nmucmd-atca7540-<version>.rpm
NMU SFP Driver	bbs-nmu-sfp-atca7540-<version>.rpm
Net Device Renamer	bbs-netdevrenamer-atca7540-<version>.rpm
Board Utilities	bbs-board-utils-atca7540-<version>.rpm
Host Name	bbs-hostname-atca7540-<version>.rpm
PCI Tool	bbs-pcitool-atca7540-<version>.rpm

The following RPM commands are useful to review package information.

Table 2-2 RPM Commands

Command	Description
`rpm -qa`	Lists all the installed packages. `Use rpm -qa | grep bbs` to list only BBS packages.
`rpm -ql <package-name>`	Lists the contents of a package, where `package-name` is the name of a specific package. For example: `rpm -ql bbs-fcu-atca7540`.
`rpm -qi <package-name>`	Lists the information about a package, where `package-name` is the name of a specific package. For example: `rpm -qi bbs-fcu-atca7540`.
`rpm -qf <path to file>`	Lists the RPM that provides the file.

2.1.2 Accessing the ATCA-7540 via Serial Console

To invoke the Linux commands or configure BIOS settings, you need to access the ATCA-7540 via the faceplate serial port.

The default serial port settings for using a serial console or a terminal emulator are:

- 115200 baud
- No parity
- Eight data bits
- One stop bit
- Flow control: xon/xoff
- Emulated terminal type: VT100

If you want to access Linux via a Linux shell, refer to the section titled *Login on page 23*.

2.2 BBS Installation

There are two ways to use the software provided.

One way to use the software is to network boot the provided Linux kernel and initramfs. SMART EC content is included in the kernel and initramfs. After booting, tools in the release can be used to create a CentOS 7.5 installation with SMART EC content on an attached disk. Any M.2 NVME, AHCI, or SATA drive on the board, or any attached USB drive can be used as an installation target.

Another way to use the provided software is to install the provided SMART EC RPMs and new and updated drivers over an existing CentOS 7.5 installation. Customers who want to install CentOS in ways not supported by the SMART EC installer should choose this method. After installing the baseline CentOS 7.5 using CentOS provided media, an install script provided in this package can be used to install SMART EC content over a baseline CentOS installation.

The *Release Notes and Quick Start Guide* provided with the software gives details on how to install the BBS using both methods.

Operating System

3.1 Distribution Description

The BBS for the ATCA-7540 blade is based on CentOS 7.5.

3.2 Login

A Linux shell can be accessed via the faceplate serial port. If you use a serial console or terminal emulator, the serial port settings are 115200 baud, no parity, 8 data bits, and 1 stop bit.

The following table lists available default login accounts.

Table 3-1 Login Information

Login Name	Password	Description
bbs	bbsroot	Non-privileged user account
root	root	Privileged user account

3.3 Linux Services

The hpmagent and hostname-atca7540 services are enabled and started at the default multi-user run-level. Should an RTM be present, the netdevrenamer service is also started.

3.4 Network Services Configuration

The following sections describe the default configuration for network services.

3.4.1 ATCA-7540 Ethernet Interfaces

The Ethernet devices, such as eth0, eth1, and eth2 are renamed to more meaningful names on the ATCA-7540, such as base1, base2, fabric1, etc. For renaming of the Ethernet devices of the front blade, the udev rules at the following location are used:

```
/etc/udev/rules.d/70-persistent-net.rules
```

For RTMs, the "eth" network devices are renamed using the netdev-renamer tool.

The following table describes the Ethernet devices on the ATCA-7540.

Table 3-2 Supported Ethernet Devices

Device Name	Description	Speed	Location	IP address	Driver Name
base1, base2	Base Interfaces	1 GbE	Front Board -> Backplane	No IP address assigned	Intel igb
fabric1, fabric2	Fabric Interfaces	KR4 - 40 GbE	Front Board -> Fabric Interface on Backplane	No IP address assigned	Intel i40e
front1, front-sfp	Front Panel Interfaces	1 GbE, 10 Gbe	Front Board	No IP address assigned	Intel igb/i40e
rtm1..n (optional)	RTM Panel Interfaces	1 GbE, 10 GbE	RTM (optional)	No IP address assigned	Dependent on RTM (Intel igb/ixgbe)

3.4.1.1 Front Board Network Devices Setup

Table 3-3 Front Board Network Devices

PCIDev	IF Name	Driver
0000:02:00.0	base1	igb
0000:02:00.1	base2	igb
0000:02:00.2	front1	igb
0000:1a:00.0	front-sfp	i40e
0000:3c:00.0	fabric1	i40e
0000:3c:00.1	fabric2	i40e

The BBS for the ATCA-7540 sets up the network interfaces of the RTM-736x-10G, RTM-736X-10G-SP, RTM-736X-DD, RTM-7360, RTM-7360-S and RTM-7360-L.

3.4.1.2 RTM-736X-10G-SP Configuration Setup

Table 3-4 RTM-736X-10G-SP Configuration Setup

PCI Dev	IF Name	Driver
0000:07:00.0	rtm8	igb
0000:07:00.1	rtm7	igb
0000:07:00.2	rtm6	igb
0000:07:00.3	rtm5	igb
0000:5e:00.0	rtm10	ixgbe
0000:5e:00.1	rtm9	ixgbe
0000:60:00.0	rtm2	ixgbe
0000:60:00.1	rtm1	ixgbe
0000:86:00.0	rtm4	ixgbe
0000:86:00.1	rtm3	ixgbe

3.4.1.3 RTM-736X-10G Configuration Setup

Table 3-5 RTM-736X-10G Configuration Setup

PCI Dev	IF Name	Driver
0000:07:00.0	rtm8	igb
0000:07:00.1	rtm7	igb
0000:07:00.2	rtm6	igb
0000:07:00.3	rtm5	igb
0000:5e:00.0	rtm2	ixgbe
0000:5e:00.1	rtm1	ixgbe
0000:86:00.0	rtm4	ixgbe
0000:86:00.1	rtm3	ixgbe

3.4.1.4 RTM-736X-DD Configuration Setup

Table 3-6 RTM-736X-DD Configuration Setup

PCI Dev	IF Name	Driver
0000:5e:00.0	rtm2	igb
0000:5e:00.1	rtm1	igb

3.4.1.5 RTM-7360/RTM-7360-S Configuration Setup

Table 3-7 RTM-7360/RTM-7360-S Configuration Setup

PCI Dev	IF Name	Driver
0000:60:00.0	rtm1	igb
0000:60:00.1	rtm2	igb
0000:5e:00.0	rtm3	igb
0000:5e:00.1	rtm4	igb
0000:04:00.0	rtm5	igb
0000:04:00.1	rtm6	igb

3.4.1.6 RTM-7360-L Configuration Setup

Table 3-8 RTM-7360-L Configuration Setup

PCI Dev	IF Name	Driver
0000:03:00.0	rtm1	igb
0000:03:00.1	rtm2	igb

3.4.2 ATCA-7540 Fabric Interface

The ATCA-7540 blade is equipped with an Intel XL710 Dual 40 Gb/s dual-port Ethernet controller device which is responsible for handling the fabric interfaces (fabric1 and fabric2).

3.4.2.1 Network Driver - IRQ Assignment

In order to gain maximum network performance, interrupts for the fabric network interfaces should be assigned to the CPU nodes that the interfaces are physically connected to. The IRQs must also not be routed to the first CPU core.

This IRQ assignment for the network devices is performed by the script:

```
/opt/bladeservices/bin/set_irq_affinity
```

3.5 Tools

This section describes the ipmibpar tool which can be used to change the IPMI Boot Parameter list.

3.5.1 IPMIBPAR

The ipmibpar tool is used to change the IPMI Boot Parameter list when Linux is up and running. It supports the following options:

Option	Description
-d	Enables debug output
-a xx	IPMB Address, if not present, local IPMC is used
-i	Gets device ID
-g	Gets IPMI Boot parameter USER area
-s file	Stores IPMI Boot Parameter (USER area), read from file
-h	Provides Help

The following example describes the steps required to change the Boot Order from On-board NVME0 disk to FrontNetwork1.

1. Read the IPMI boot parameter USER area from the IPMC.

```
root@ATCA-7540: ~#  ipmibpar -g
ipmibpar - Version 1.02 - IPMI Boot Parameters
          Copyright 2017 Artesyn Embedded Technologies

Read System Boot Options from USER area (local IPMC)

Hexdump IPMI Boot Parameter:
Size = 1074 (0x432)
0000  2e 04 61 70 65 69 3d 6f 66 66 00 61 70 65 69 5f   <..apei=off.apei_>
0010  75 65 66 69 76 65 72 3d 75 65 66 69 32 32 00 62   <uefiver=uefi22.b>
```

```
0020  6f 6f 74 5f 62 61 73 65 6e 65 74 3d 6f 6e 00 62  <oot_basenet=on.b>
0030  6f 6f 74 5f 66 61 62 72 69 63 6e 65 74 3d 6f 6e  <oot_fabricnet=on>
0040  00 62 6f 6f 74 5f 66 72 6f 6e 74 6e 65 74 3d 6f  <.boot_frontnet=o>
0050  6e 00 62 6f 6f 74 5f 6e 65 74 70 72 6f 74 3d 6c  <n.boot_netprot=l>
0060  65 67 61 63 79 00 62 6f 6f 74 5f 6f 72 64 65 72  <egacy.boot_order>
0070  3d 6e 76 6d 65 30 2c 66 72 6f 6e 74 6e 65 74 31  <=nvme0,frontnet1>
0080  2c 65 66 69 73 68 65 6c 6c 00 62 6f 6f 74 5f 70  <,efishell.boot_p>
0090  72 69 6f 72 69 74 79 3d 6c 65 67 61 63 79 00 62  <riority=legacy.b>
00a0  6f 6f 74 5f 72 74 6d 6e 65 74 3d 6f 66 66 00 62  <oot_rtmnet=off.b>
00b0  6f 6f 74 5f 72 74 6d 73 61 73 3d 6f 6e 00 62 6f  <oot_rtmsas=on.bo>
00c0  6f 74 5f 74 79 70 65 3d 64 75 61 6c 00 62 6f 6f  <ot_type=dual.boo>
00d0  74 5f 75 73 62 3d 6f 6e 00 63 6c 6f 63 6b 5f 73  <t_usb=on.clock_s>
00e0  73 63 3d 6f 66 66 00 63 6f 6e 5f 61 70 3d 6f 66  <sc=off.con_ap=of>
00f0  66 00 63 6f 6e 5f 62 72 3d 31 31 35 32 30 30 00  <f.con_br=115200.>
0100  63 6f 6e 5f 64 62 3d 38 00 63 6f 6e 5f 66 63 3d  <con_db=8.con_fc=>
0110  6f 66 66 00 63 6f 6e 5f 70 61 72 3d 6e 00 63 6f  <off.con_par=n.co>
0120  6e 5f 73 62 3d 31 00 63 6f 6e 5f 74 74 3d 76 74  <n_sb=1.con_tt=vt>
0130  31 30 30 00 63 70 75 30 5f 64 69 73 6d 3d 30 00  <100.cpu0_dism=0.>
0140  63 70 75 31 5f 64 69 73 6d 3d 30 00 63 70 75 5f  <cpu1_dism=0.cpu_>
0150  61 63 70 3d 6f 6e 00 63 70 75 5f 63 31 65 3d 6f  <acp=on.cpu_c1e=o>
0160  6e 00 63 70 75 5f 63 33 3d 6f 66 66 00 63 70 75  <n.cpu_c3=off.cpu>
0170  5f 63 36 3d 61 75 74 6f 00 63 70 75 5f 63 73 6c  <_c6=auto.cpu_csl>
0180  69 6d 69 74 3d 61 75 74 6f 00 63 70 75 5f 63 73  <imit=auto.cpu_cs>
0190  74 61 74 65 73 3d 6f 66 66 00 63 70 75 5f 63 78  <tates=off.cpu_cx>
01a0  61 63 70 69 3d 63 33 00 63 70 75 5f 65 64 3d 6f  <acpi=c3.cpu_ed=o>
01b0  6e 00 63 70 75 5f 68 70 3d 6f 6e 00 63 70 75 5f  <n.cpu_hp=on.cpu_>
01c0  68 74 3d 6f 6e 00 63 70 75 5f 73 73 3d 6f 6e 00  <ht=on.cpu_ss=on.>
01d0  63 70 75 5f 74 6d 3d 6f 6e 00 63 70 75 5f 74 78  <cpu_tm=on.cpu_tx>
01e0  74 3d 6f 66 66 00 63 70 75 5f 76 74 3d 6f 6e 00  <t=off.cpu_vt=on.>
01f0  63 70 75 5f 78 32 61 70 69 63 3d 6f 66 66 00 66  <cpu_x2apic=off.f>
0200  61 69 6c 73 61 66 65 3d 6e 6f 63 68 61 6e 67 65  <ailsafe=nochange>
0210  00 69 6e 66 6f 5f 74 6d 6f 75 74 3d 32 00 69 70  <.info_tmout=2.ip>
0220  6d 69 5f 69 72 71 3d 6f 6e 00 6d 65 6d 5f 64 73  <mi_irq=on.mem_ds>
0230  3d 6f 6e 00 6d 65 6d 5f 68 61 6c 74 3d 6f 6e 00  <=on.mem_halt=on.>
0240  6d 65 6d 5f 6e 75 6d 61 3d 6f 6e 00 6d 65 6d 5f  <mem_numa=on.mem_>
0250  70 73 3d 6f 6e 00 6d 65 6d 5f 72 6d 74 3d 32 00  <ps=on.mem_rmt=2.>
0260  6d 65 6d 5f 73 70 61 72 69 6e 67 3d 6f 66 66 00  <mem_sparing=off.>
0270  6d 65 6d 5f 73 70 65 65 64 3d 61 75 74 6f 00 6d  <mem_speed=auto.m>
0280  65 6d 5f 74 65 73 74 3d 6f 66 66 00 6f 73 62 6f  <em_test=off.osbo>
0290  6f 74 5f 77 64 3d 6f 66 66 00 6f 73 62 6f 6f 74  <ot_wd=off.osboot>
02a0  5f 77 64 5f 61 63 74 69 6f 6e 3d 72 65 73 65 74  <_wd_action=reset>
02b0  00 6f 73 62 6f 6f 74 5f 77 64 5f 74 6f 75 74 3d  <.osboot_wd_tout=>
02c0  35 00 70 63 69 5f 36 34 62 69 74 3d 6f 6e 00 70  <5.pci_64bit=on.p>
02d0  63 69 5f 61 72 69 3d 6f 66 66 00 70 63 69 5f 73  <ci_ari=off.pci_s>
02e0  72 69 6f 76 3d 6f 6e 00 72 74 6d 5f 37 34 38 30  <riov=on.rtm_7480>
```

```
02f0  5f 73 65 6c 31 3d 32 78 34 30 47 00 72 74 6d 5f  <_sel1=2x40G.rtm_>
0300  37 34 38 30 5f 73 65 6c 32 3d 32 78 34 30 47 00  <7480_sel2=2x40G.>
0310  72 74 6d 5f 61 75 74 6f 3d 6f 6e 00 72 74 6d 5f  <rtm_auto=on.rtm_>
0320  63 70 75 30 5f 33 61 3d 61 75 74 6f 00 72 74 6d  <cpu0_3a=auto.rtm>
0330  5f 63 70 75 30 5f 33 62 3d 61 75 74 6f 00 72 74  <_cpu0_3b=auto.rt>
0340  6d 5f 63 70 75 30 5f 33 63 3d 61 75 74 6f 00 72  <m_cpu0_3c=auto.r>
0350  74 6d 5f 63 70 75 30 5f 33 64 3d 61 75 74 6f 00  <tm_cpu0_3d=auto.>
0360  72 74 6d 5f 63 70 75 30 5f 62 69 66 3d 78 38 78  <rtm_cpu0_bif=x8x>
0370  38 00 72 74 6d 5f 63 70 75 31 5f 33 61 3d 61 75  <8.rtm_cpu1_3a=au>
0380  74 6f 00 72 74 6d 5f 63 70 75 31 5f 33 62 3d 61  <to.rtm_cpu1_3b=a>
0390  75 74 6f 00 72 74 6d 5f 63 70 75 31 5f 33 63 3d  <uto.rtm_cpu1_3c=>
03a0  61 75 74 6f 00 72 74 6d 5f 63 70 75 31 5f 33 64  <auto.rtm_cpu1_3d>
03b0  3d 61 75 74 6f 00 72 74 6d 5f 63 70 75 31 5f 62  <=auto.rtm_cpu1_b>
03c0  69 66 3d 78 38 78 38 00 73 61 74 61 3d 6f 66 66  <if=x8x8.sata=off>
03d0  00 73 61 74 61 5f 61 6c 70 6d 3d 6f 6e 00 73 61  <.sata_alpm=on.sa>
03e0  74 61 5f 6d 6f 64 65 3d 69 64 65 00 73 61 74 61  <ta_mode=ide.sata>
03f0  5f 72 61 69 64 77 61 69 74 3d 32 00 74 70 6d 5f  <_raidwait=2.tpm_>
0400  6f 70 65 72 61 74 69 6f 6e 3d 6e 6f 5f 6f 70 65  <operation=no_ope>
0410  72 61 74 69 6f 6e 00 75 73 62 3d 6f 6e 00 76 74  <ration.usb=on.vt>
0420  64 3d 6f 6e 00 76 74 64 5f 69 72 3d 6f 6e 00 00  <d=on.vtd_ir=on..>
0430   7e 82                                                 <~.>

IPMI Boot Parameter:

apei=off
apei_uefiver=uefi22
boot_basenet=on
boot_fabricnet=on
boot_frontnet=on
boot_netprot=legacy
boot_order=nvme0,frontnet1,efishell
boot_priority=legacy
boot_rtmnet=off
boot_rtmsas=on
boot_type=dual
boot_usb=on
clock_ssc=off
con_ap=off
con_br=115200
con_db=8
con_fc=off
con_par=n
con_sb=1
con_tt=vt100
cpu0_dism=0
```

```
cpu1_dism=0
cpu_acp=on
cpu_c1e=on
cpu_c3=off
cpu_c6=auto
cpu_cslimit=auto
cpu_cstates=off
cpu_cxacpi=c3
cpu_ed=on
cpu_hp=on
cpu_ht=on
cpu_ss=on
cpu_tm=on
cpu_txt=off
cpu_vt=on
cpu_x2apic=off
failsafe=nochange
info_tmout=2
ipmi_irq=on
mem_ds=on
mem_halt=on
mem_numa=on
mem_ps=on
mem_rmt=2
mem_sparing=off
mem_speed=auto
mem_test=off
osboot_wd=off
osboot_wd_action=reset
osboot_wd_tout=5
pci_64bit=on
pci_ari=off
pci_sriov=on
rtm_7480_sel1=2x40G
rtm_7480_sel2=2x40G
rtm_auto=on
rtm_cpu0_3a=auto
rtm_cpu0_3b=auto
rtm_cpu0_3c=auto
rtm_cpu0_3d=auto
rtm_cpu0_bif=x8x8
rtm_cpu1_3a=auto
rtm_cpu1_3b=auto
rtm_cpu1_3c=auto
rtm_cpu1_3d=auto
```

```
rtm_cpu1_bif=x8x8
sata=off
sata_alpm=on
sata_mode=ide
sata_raidwait=2
tpm_operation=no_operation
usb=on
vtd=on
vtd_ir=on
```

2. Save the received IPMI Boot Parameter list into a file (for example, bootparam.log) and change the boot order as follows.

   ```
   boot_order=frontnet1,nvme0,efishell
   ```

3. Write the IPMI parameter list file (for example, bootparam.log).

   ```
   ipmibpar -s <filename>
   ```

Firmware Upgrade Facility

4.1 Overview

The Firmware Upgrade Facility (FUF) provides a uniform way to upgrade firmware on SMART EC blades. It consists of a FUF Command-line Utility (FCU), and specially prepared firmware recovery image files.

4.2 Firmware Recovery Image Files

FCU supports firmware images in the HPM.1 format. HPM.1 is a PICMG standard to upgrade firmware images.

By default, the image files for the BIOS, IPMC, and FPGA are loaded as part of the BBS software in `/opt/bladeservices/rom` directory when the blade-specific firmware support packages are installed.

The following describes the image files that are currently supported.

Table 4-1 Firmware Image Files Supported

Filename	Description
`atca-7540-cpu-<version>.hpm`	BIOS image
`atca-7540-ipmc-boot-<version>.hpm`	Firmware for the IPMC booter
`atca-7540-ipmc-<version>.hpm`	Firmware for the IPMC
`atca-7540-glue-fpga-<version>.bin.hpm`	Glue FPGA Bitstream

NOTICE

The BIOS is only released in hpm.1 file format.

4.3 FUF Command-line Utility (FCU)

The FCU displays upgrade capabilities of firmware devices on a board. If the board is managed by an IPMC, then FCU requests IPMC to determine the board type. A board may have multiple devices, which are low-level hardware components, like BIOS, FPGA, IPMC, and so on. The FCU abstracts firmware upgrade operations to a common set of operations for all devices. The FCU user does not need to know the internals of a device. The FCU identifies a device by its name, which is a combination of the board name and the device properties.

For example, the BIOS for the ATCA-7540 has the name "PYLD F/W". FCU provides the following upgrade operations:

- Query the device to return the firmware version and other information
- Query the firmware image to return the firmware version and other information
- Validate the firmware image
- Verify whether the image is applicable on the target device
- Upgrade the device with the given firmware image
- Mark a bank of the device to become active after the next reset or power cycle
- Activate a bank of the device immediately

4.3.1 Query Operation

Using the Query operation, the FCU returns firmware information for a specific device (if used with -d) or information about all firmware devices. The Query operation is exclusive and is not intended to be combined with other operations. The Query operation shows all the banks of a device. One of these banks is the active version, which means that the device was booted with the firmware installed in that bank. The device might have a second bank that contains the rollback version. You can switch to the rollback version with the help of an Activate operation or with a Mark operation in combination with a reboot or power cycle.

If the device supports the Mark operation, the Query operation shows which bank is marked for next use. Furthermore, FCU shows the capabilities of a device. Device capabilities are a set of FCU operations, such as manual, automatic rollback, or self test implemented.

The following example shows the BIOS of the ATCA-7540 blade:

```
root@ATCA-7540:~# fcu --query -d "PYLD F/W"
********************[[[[[REPORT BEGIN]]]]]********************
Operation: Query
 #10 Device    : PYLD F/W
  Bank #1 -       Active Version: 1.00.00000000
```

```
  Bank #0 -    Rollback Version: 1.00.00000000
  Bank marked for next use: #1
********************[[[[[ REPORT END ]]]]]********************
```

In the first line, the number of the device and its name are displayed. In Bank1, the active version is stored and this bank is also marked for next use. Bank0 contains the rollback version.

4.3.2 Show Operation

The Show operation does not access any device. It only operates with the firmware image and it shows the metadata, which is part of the image. It also validates the firmware image by comparing the checksum part of the metadata against the checksum of the raw image. The output of the Show operation is similar to the output of the Query operation.

```
root@ATCA-7540:~# fcu --show --file /opt/bladeservices/rom/atca-7540-cpu-1.0.0.hpm
********************[[[[[REPORT BEGIN]]]]]********************
Operation: Show
 Manufacturer : ARTESYN
 Board        : atca-7540
 #00 Device   : ATCA7540 BIOS Image
  Bank #0 -             Version: 1.00.00000000

********************[[[[[ REPORT END ]]]]]********************
```

Additionally, this operation shows the name of the manufacturer and the name of the board which is compatible with this firmware image. A firmware image does not have multiple banks so you can see only one bank.

4.3.3 Mark Operation

The Mark operation selects the bank that is used after the next reset or reboot. With this operation, you have to specify the name of the device and the bank.

NOTICE

Not all devices support the Mark operation.

4.3.4 Activate Operation

The Activate operation is similar to that of Mark operation. This operation sets a bank to the active state. The bank is activated immediately, which is not the case for the Mark operation. Not every device supports this operation. IPMCs that have the HPM.1 must support the Activate operation on the rollback bank and the deferred bank. If the rollback bank is activated by this operation, a manual rollback is performed.

4.3.5 Upgrade Operation

The Upgrade operation uploads the firmware image to the device. To have a valid firmware image in one bank, the FCU tries to protect the active bank being overwritten by a new firmware image. For devices like the IPMC, this protection is done by the IPMC firmware itself, which communicates with the FCU during the image upload. The IPMC firmware selects the bank to write the new image to. The FCU determines this bank by reading an IPMI sensor, which knows the active bank. For that reason, you have to specify the firmware image with the Upgrade operation.

4.3.6 Verify Operation

The Verify operation checks if the firmware image is applicable on a device of the blade. It prevents you from writing a firmware image into a noncompatible device. The Verify operation is always done before an Upgrade operation. You do not have to specify it explicitly when you perform an upgrade.

4.3.7 Command-Line Options

`fcu --help`
  Usage

`fcu [operations] [operands]`
  Operations

`--query`
  `-q` Set to perform a query operation

`--upgrade`
  `-u` Upgrade the unused version of firmware
  This operation requires the `--file` flag

`--help`
  `-h` Display this help message

`--show`
`-s` Display information about the target which is included in the given upgrade file
This operation requires the `--file` flag

`--verify`
`-v` Set to perform a verification of an upgrade file
This operation does not install the upgrade image
This operation requires the `--file` flag to be set

`--mark`
`-m` Mark the specified bank as next to boot
This operation depends on the `--bank` and the `--device` flag

`--activate`
`-r` Activate the specified bank
This operation depends on the `--bank` and the `--device` flag

`--version`
Display the version of this utility

Operands

`--device=<device name>`
`-d` Device to perform operation on

`--file=<filename>`
`-f` Filename of the firmware file

`--bank=<bankletter>`
`-b` Bank-letter for mark/compare command

`--level=[0-7]`
Severity level of logging, 7 logs everything, default is 5

`--log=ARG`
File name for logging

4.4 Upgrading Firmware Image

This section describes recommended procedures for upgrading firmware devices.

NOTICE

The shown file names and paths are only examples and should be replaced with file names and paths applicable to your configuration.

4.4.1 BIOS Firmware Upgrade of the ATCA-7540 Blade

Follow these steps to upgrade the BIOS firmware on the ATCA-7540 blade:

1. Query the current firmware versions.

```
root@ATCA-7540:/root# fcu --query -d "PYLD F/W"
********************[[[[[REPORT BEGIN]]]]]********************
Operation: Query
 #08 Device    : PYLD F/W
 Bank #1 -       Active Version: 1.0.00000001
 Bank #0 -    Rollback Version: 1.0.00000000
Bank marked for next use: #1
********************[[[[[ REPORT END ]]]]]********************
```

2. Upgrade Bank0 to version 1.0.1

```
root@ATCA-7540:/mnt# fcu -uf /opt/bladeservices/rom/atca-7540-cpu-
1.0.1.hpm
********************[[[[[REPORT BEGIN]]]]]********************
Operation: Upgrade
Result    : Success
********************[[[[[ REPORT END ]]]]]********************
```

3. Check if the new firmware version (here 1.0.1) is in Bank0 and whether the Bank0 is marked.

```
root@ATCA-7540:/mnt# fcu -q -d "PYLD F/W"
********************[[[[[REPORT BEGIN]]]]]********************
Operation: Query
 #08 Device    : PYLD F/W
  Bank #1 -       Active Version: 1.0.00000001
  Bank #0 -    Rollback Version: 1.0.00000001
 Bank marked for next use: #0
********************[[[[[ REPORT END ]]]]]********************
```

4. After the BIOS is upgraded, the payload must be rebooted to use the new image. After confirming that the new BIOS works as expected, the BIOS upgrade steps can then be repeated to upgrade the other BIOS bank.

4.4.2 IPMC Upgrade

The IPMC F/W and IPMC Boot F/W can be updated with the FCU tool.

To upgrade the IPMC F/W on an ATCA-7540 blade:

1. Run the following command:

   ```
   # fcu -uf /opt/bladeservices/rom/atca-7540-hpm.1-ipmc-
   <version>.img
   ```

   You will see the following output on a successful upgrade:

   ```
   ********************[[[[[REPORT BEGIN]]]]]********************
   Operation: Upgrade
   preparation stage started
   capabilities of target and image header successfully compared
   properties of component IPMC F/W retrieved
   properties of component IPMC F/W retrieved
   preparation stage successfully finished
     preparing upload
   upload stage of IPMC F/W started
     initializing upload
     uploading ...100 %
     finishing upload
   upload stage finished
   activation stage of new firmware started
     query self test result
   activation stage successfully finished
   Result   : Success
   ********************[[[[[ REPORT END ]]]]]********************
   ```

2. The upgraded IPMC F/W is active when the command completes.
3. The IPMC Boot F/W can also be updated the same way, and it is also active when the command completes.

4.4.3 FPGA Upgrade

The FPGA on the ATCA-7540 board can also be updated with the FCU tool.

To upgrade the FPGA on the ATCA-7540 blade:

1. Run the following command:

   ```
   fcu -uf /opt/bladeservices/rom/atca_7540_glue_fpga_0.0.11.bin.hpm
   ```

   After a successful upgrade, the following is displayed:

   ```
   ********************[[[[[REPORT BEGIN]]]]]********************
   Operation: Upgrade
   preparation stage started
   capabilities of target and image header successfully compared
   properties of component FPGA retrieved
   properties of component FPGA retrieved
   preparation stage successfully finished
   preparing upload
   upload stage of ATCA7540 FPGA Image started
   initializing upload
   uploading ...100 %
   finishing upload
   upload stage finished
   activation stage of new firmware started
   Result   : Success
   ********************[[[[[ REPORT END ]]]]]********************
   ```

2. Power cycle the board to run the new FPGA image.

Hardware Platform Management

5.1 Overview

Hardware management in ATCA systems is based on the Intelligent Platform Management Interface (IPMI) specification. IPMI commands can be complex and cumbersome. To facilitate blade-level management, SMART EC provides the Hardware Platform Management (HPM) package, which provides a set of commands that are based on IPMI commands. These commands are easier to use than IPMI commands. An HPM command can encapsulate a sequence of IPMI commands, for example, reading the FRU inventory data. An HPM command can be the unifier for OEM IPMI commands that are different on different blade types, for example reading the CPU boot bank. For a catalog of IPMI commands supported by the blade, refer to the respective IPMI manual.

The HPM package consists of:

- HPM daemon, hpmagentd
- Command line utility, hpmcmd
- Script framework for managing shutdown, reboot, and local ekeying events

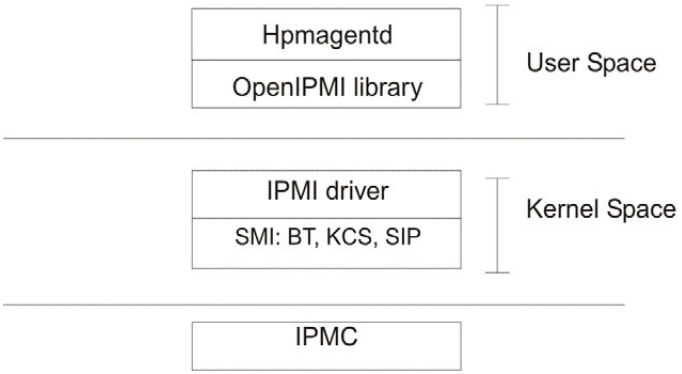
The HPM daemon is responsible for waiting for events from the IPMC to perform a graceful shutdown/reboot of the operating system and to react when the link state of a channel's port is changed.

The utility hpmcmd displays the response of commands on the console in a human-readable format. HPM commands include:

- Retrieving and modifying FRU data
- Reading and controlling status of IPMI-controlled LEDs
- Communicating local slot location information
- Retrieving the event messages from the System Event Log (SEL) of the IPMC

The hpmagentd and hpmcmd make use of OpenIPMI driver to talk to the local IPMC. The following figure shows the software levels that are involved in the HPM architecture:

Figure 5-1 Software Levels of the HPM Architecture



BT Block Transfer Interface
SIP Serial Interface Protocol
SMI System Management Interface
KCS Keyboard Control Style

The System Management Interface (SMI) driver provides the low level interface for talking to the IPMC and could be a Keyboard Control Style (KCS) driver or Block Transfer (BT) driver or other. If you need more information about the software aspects of the blade IPMI controller, refer to the respective IPMI manual.

5.2 hpmagentd—HPM Agent Daemon

5.2.1 Description

The hpmagentd is the service that processes events from the local IPMC. For any incoming event, it calls the respective script which is part of the hpmagentcmd package. Event data is passed to a script by command line arguments. You can modify a script to fulfill your requirements. The following events are handled by the daemon:

Graceful Shutdown - When the IPMC receives a FRU activation request to deactivate a FRU, it redirects the command to the hpmagentd through the IPMI driver. The hpmagentd invokes the shutdown script located in `/opt/bladeservices/bin/hpmshutdown`.

By default this script initiates an immediate shutdown of Linux: `shutdown -h now`

NOTICE

IPMC powers down the processor in any case after a certain time. You may adjust this time with the Graceful Shutdown Timeout parameter of the IPMC, which can be set with a SetSystemBootOption IPMI command.

Graceful Reboot - On receiving a FRU control request to gracefully reboot the payload, the IPMC sends the command to the hpmagentd. The daemon invokes the reboot script which is available at `/opt/bladeservices/bin/hpmreboot`.

Ekeying Events - When the IPMC modifies the link state of a port, it notifies hpmagentd about the change. If hpmagentd is notified of a link down event, it invokes the hpmekeydown script; if it is notified of a link up event, it invokes the hpmekeyup script. These scripts are in `/opt/bladeservices/bin` and are empty by default. hpmagentd passes an argument to these scripts which can be used to determine what actions to take on these events.

The argument starts with the interface. The interface can be BC (Base channel), FC (Fabric channel), or UC (Update channel). The channel number and port numbers of the channel that are affected are appended to the interface. For example, if the ports 1,2,3, and 4 of channel 1 of a base interface are changed, then the argument would be "BC1.1,2,3,4".

5.2.2 Deployment

By default, the HPM daemon is installed in `/opt/bladeservices/bin` directory. Along with the hpmagentd binary, the scripts hpmreboot, hpmshutdown, hpmekeyup, and hpmekeydown are stored in that directory. Additionally, there is an init script hpm to start and stop the daemon, an hpmagent service file for systemd, and the script hpmvar which exports some important variables to `/etc/default/hpmvars` to describe the board.

Synopsis

```
hpmagentd [options]
```

Parameters

`--log =<file name>`	You may specify a log file for the daemon with this option. If you do not use it, then the hpmagentd logs to the syslog.
`-l --level`	Specifies the level of message logging, where level is one of the standard syslog levels. (Refer to *Table 5-1* for Log Levels.)
`-v --version`	Displays the version of the daemon
`-L --disable-led`	Disables the LED management

`-r --reboot-script=<script>`	Use the specified Blade Reboot script. Default script: `/opt/bladeservices/bin/hpmreboot`
`-s --shutdown-script=<script>`	Use the specified Blade Shutdown script. Default script: `/opt/bladeservices/ bin/hpmshutdown`
`-u --ekey-up-script=<script>`	Called when a port is enabled. Default script: `/opt/bladeservices/bin/hpmekeyup`
`-d --ekey-down-script=<script>`	Called when a port is disabled. Default script: `/opt/bladeservices/bin/hpmekeydown`
`-h --help`	Displays the help message
`-i --dont-daemonize`	Run interactively

Table 5-1 Description of Log Levels

Log Level	Description
0	Emergency
1	Alert
2	Critical
3	Error
4	Warning
5	Notice (default)
6	Information
7	Debug

5.2.3 hpm - init.d Script

The hpm init.d script is used to start, stop, and restart hpmagentd. It is invoked by systemd via the hpmagent.service file.

Synopsis

```
hpm { start | stop | restart | status }
```

Parameters

`Start`	Starts `hpmagentd`
`Stop`	Stops `hpmagentd`
`Restarts`	Stops and Starts `hpmagentd` again
`Status`	Reports if `hpmagentd` is currently running or not

5.3 hpmcmd—HPM Command Utility

5.3.1 Overview

The HPM command utility communicates directly with the IPMC through the IPMI driver, which is part of the Linux kernel. It takes care of translating the user-friendly commands into elaborated IPMI commands that the IPMC is able to understand. Those IPMI commands are transferred to the local IPMC. The HPM command utility can be started in interactive mode, where a prompt is displayed and the user enters commands, or it can process a single command.

By default, the hpmcmd binary is installed in `/opt/bladeservices/bin` directory.

Synopsis

```
hpmcmd [options]
```

Parameters

`-c`	Processes a single command
`--help -h`	Displays this help message
`-v`	Verbose mode for some commands Some commands like fruinfoget print more details if this option is given. Commands which do not support the verbose option ignore it.
`-t`	Sends the command to a remote target For more information, refer to *Target Addressing with hpmcmd on page 45*. If this option is not given, then the command goes to the local IPMC.
`-p`	Changes the prompt when hpmcmd runs in interactive mode
`-o`	Prints results to a file

5.3.2 Target Addressing with hpmcmd

Using the -t option, you can send commands to other IPMCs or MMCs, which participate on an IPMB.

syntax: -t <IPMB address>[: MMC address]. The addresses must be set in hexadecimal format.

To send the command to another IPMC type:

```
-t 92
```

To send the command to an MMC, which is attached on another blade in the shelf specify:

```
-t 82:72
```

5.3.3 Command Overview

The following table lists all commands from the hpmcmd program available on the ATCA-7540. You can display this list and a short description about the command using the help command (see *help on page 57*). A detailed description of the commands is given in the section *Supported Commands on page 48*.

Table 5-2 Command Overview

Command	Description
bootbankget	Gets the bootbank to boot from
bootbankset	Sets the bootbank to boot from
bootparamerase	Erases boot parameter value
bootparamget	Gets the boot parameter value
bootparamset	Sets a boot parameter value
chinfo	Retrieves the channel information
cmd	Executes IPMI commands
deviceid	Gets the Device ID
frudata	Allows to get FRU info in hexadecimal numbers
fruinfoget	Gets the string fields from the FRU
fruinv	Gets the FRU size and addressable units
fruread	Reads 'x' number of bytes from the FRU
fruwrite	Writes 'x' number of bytes to the FRU
fwprogevent	Sends a Firmware Progress Sensor Event
help	Gets the list of commands
ipmbaddress	Gets the IPMB address
ipmcstatus	Gets the IPMC status
lancfgget	Gets the LAN configuration parameter
lancfgset	Sets the LAN configuration parameter
ledget	Gets the state of a specific FRU LED

Table 5-2 Command Overview (continued)

Command	Description
ledprop	Get the LED properties for this FRU
ledset	Controls the state of a specific FRU LED
loglevelget	Gets the hpmagentd log level
macaddress	Lists the MAC addresses
partnumber	Gets the board part number
physlotnumber	Gets the board physical slot number
portget	Shows the current state of network interfaces governed by E_Keying events
portset	Enables/Disables the ports in a channel
posttypeget	Gets the posttype to run at boot
posttypeset	Sets the posttype to run at boot
sdr	Shows SDR records
sdr_dump	Shows the SDR records in raw format
sdrinfo	Shows SDR information
sel	Shows SEL records
selclear	Erases all contents from the SEL
selinfo	Shows SEL information
sendamc	Sends an IPMI request to an MMC behind a remote IPMC
sendcmd	Sends an IPMI request to an IPMB address IPMC
solcfgget	Determines which serial output source goes to a particular serial port connector
solcfgget	Selects the serial output source of the serial port connector
shelfaddress	Gets the Shelf Address String
shelfslots	Prints the number of slots in the shelf
slotmap	Prints the slotmap of the shelf

Table 5-2 Command Overview (continued)

Command	Description
slotnumber	Shows the board slot number
solcfgget	Gets SOL configuration parameter
solcfgset	Sets SOL configuration parameter
version	Shows the hpmcmd version
watchdog	Controls Payload WDT functionality

5.3.4 Supported Commands

This section lists the commands supported by hpmcmd. All commands are case insensitive. The examples illustrate the use of hpmcmd in single command mode (-c). If you start hpmcmd without the '-c' option (that is, interactive mode), you can simply enter these commands at the HPM command prompt.

5.3.4.1 bootbankget

This command retrieves the boot bank which is currently marked as active for the CPU specified by the payload_cpu_selector command.

Firmware for the CPU on SMART EC ATCA blades is stored in redundant persistent memory devices. This allows the firmware image in one bank to serve as a backup for other bank. During normal operation, the CPU on a blade determines which bank to boot from, based on a GPIO signal controlled by the IPMC. This bank is considered as the active boot device. You can change the "active" device with the hpmcmd bootbankset command. Active status does not necessarily indicate which device was used on the last boot. It simply represents which device is set to be used on the next boot.

Synopsis

```
bootbankget <payload_cpu_selector>
```

Parameters

`payload_cpu_selector`	An integer between 0 and the number of CPU devices supported on the blade. On the ATCA-7540 the payload_cpu_selector is 0.

Example

```
hpmcmd -c bootbankget 0
BANK1
```

```
hpmcmd -c bootbankget 1
BANK0
```

5.3.4.2 bootbankset

This command sets the boot bank for a particular CPU from which the blade is supposed to boot.

Synopsis

```
bootbankset <payload_cpu_selector> <newBootBank>
```

Parameters

payload_cpu_selector	An integer between 0 and the number of CPU devices supported on the blade. On the ATCA-7540 the payload_cpu_selector is 0.
newBootBank	Can be set to: BANK0, BANK1...

Example

```
hpmcmd -c bootbankset 0 BANK 1
```

5.3.4.3 bootparamerase

This command erases a boot parameter.

Synopsis

```
bootparamerase section [name] [-t ipmbAddr[:mmcAddr]]
```

Parameters

section	Valid values: USER, DEFAULT, TEST, or OS_PARAM
name	Specifies name of the parameter
t	Sends the command to: ipmbAddr:mmcAddr

5.3.4.4 bootparamget

This command gets a boot parameter value.

Synopsis

```
bootparamget section [name] [-t ipmbAddr[:mmcAddr]]
```

Parameters

`section`	Valild values: `USER, DEFAULT, TEST,` or `OS_PARAM`
`name`	Specifies name of the parameter
`t`	Sends the command to: `ipmbAddr:mmcAddr`

5.3.4.5 bootparamset

This command sets a boot parameter value.

Synopsis

```
bootparamset section name=value [-t ipmbAddr[:mmcAddr]]
bootparamset section -f bootparamfile [-t ipmbAddr[:mmcAddr]]
```

Parameters

`section`	Valid values : `USER, TEST,` or `OS_PARAM`
`name=value`	Specifies name of the parameter
`t`	Sends the command to: `ipmbAddr:mmcAddr`

5.3.4.6 cmd

This command allows you to enter the commands understood by the IPMC. Commands are entered as a sequence of hexadecimal numbers as defined in the *IPMI 2.0 Specification* available on the *intel.com* website.

Synopsis

```
cmd    <IPMI command>
```

Parameters

`ipmi command]`	The IPMI command specifies the sequence of hexadecimal bytes. The IPMI command can have a sequence such as: `0f 00 XX ZZ W1 W2 ... Wn` In this example: `XX` specifies netfunc in hexadecimal `ZZ` specifies the command number, as stated in the IPMI/PICMG specification `W1` to `Wn` specifies the data bytes

`ipmi address`	The IPMI address specifies the IPMC that receives the command. It can be the local IPMC or another IPMC on the IPMB. The IPMI address for the local IPMC consists of `<f LUN>`, where `f` is the BMC channel number. The IPMI address for a remote IPMC consists of `<0 SA LUN>`, where `SA` is the slave address.
`netfn cmd`	Identifies the command type
`cmd data`	Specifies the message data associated with the command

Example

GetDeviceId command to the local IPMC:

```
hpmcmd -c cmd f 0 6 1
```

GetDeviceId command to the remote IPMC on address 9a:

```
hpmcmd -c cmd 0 9a 0 6 1
```

5.3.4.7 deviceid

This command retrieves the raw IPMI GetDeviceID command response and decodes the IPMI message.

Synopsis

```
deviceid -t [ipmbAddr[:mmcAddr]]
```

Parameters

`-t [ipmbAddr[:mmcAddr]]`	Sends the command to `ipmbAddress:mmcAddr`

Example

```
root@ATCA-7540:~# hpmcmd -c deviceid

DEVICEID INFORMATION
--------------------
Device Id           =  0x00
Device Revision     =  0x00
Device Mode         =  Normal Operation ; Supports Device SDR
Firmware Revision   =  1.20
IPMI Version        =  2.0
Device Support      =  IPMB Evnt Gen; FRU; SEL; Sensor;
Manufacturer ID     =  0x000065CD
Product ID          =  0x2035
Auxiliary Revision  =  0x00000002
```

5.3.4.8 chinfo

Retrieves information about an IPMI channel.

Synopsis

```
chinfo <channel>
```

Parameters

`channel`	Channel number

Example

```
root@ATCA-7540:~# hpmcmd -c chinfo 0
Channel Medium Type   : IPMB (I2C)
Channel Protocol Type : IPMB-1.0
Session Support       : session-less
Active Session Count  : 0
Protocol Vendor ID    : 001BF2
```

5.3.4.9 frudata

This command dumps the content of the FRU data in hexadecimal format.

Synopsis

```
frudata <fruid> [-t ipmbAddr[:mmcAddr]]
```

Parameters

`fruid`	0 for the ATCA-7540 front board and 1 for the RTM
`-t`	Sends the command to: `ipmbAddr:mmcAddr`

Example

```
root@ATCA-7540:~# hpmcmd -c frudata 0
01 00 00 01 0b 14 00 df 01 0a 19 20 40 90 c7 41
52 54 45 53 59 4e e3 50 43 41 2c 41 54 43 41 2d
37 34 38 30 2f 31 30 35 57 2f 30 47 42 2f 52 31
2e 30 2f 43 46 47 30 30 30 30 c7 45 31 35 32 44
38 43 ca 30 31 30 36 38 34 38 4e 30 31 c2 32 20
c1 00 00 00 00 00 00 14 01 09 19 c7 41 52 54 45
53 59 4e c9 41 54 43 41 2d 37 34 38 30 da 41 54
43 41 2d 37 34 38 30 2f 30 47 42 2f 52 31 2e 30
2f 43 46 47 30 30 30 30 c5 52 2e 31 2e 30 c7 45
```

5.3.4.10 fruinfoget

This command retrieves information from the specified FRU.

Synopsis

```
fruinfoget <fruid> [field] [-v] [-t ipmbAddr[:mmcAddr]]
```

Parameters.

`fruid`	0 for the ATCA-7540 front board and 1 for the RTM
`field`	One of the data fields as described in *Table 5-3*. If no field is specified, it retrieves the entire FRU information.
`-v`	Verbose mode to get point-to-point connectivity information when no specific field is requested
`-t`	Sends the command to: `ipmbAddr:mmcAddr`

Table 5-3 Data Field Descriptions

Field	Description
bmanufacturer	Board area manufacturer
bproductname	Board area product name
bserialnumber	Board area serial number

Table 5-3 Data Field Descriptions (continued)

Field	Description
bpartnumber	Board area part number
pmanufacturer	Product area manufacturer
pproductname	Product area product name
ppartnumber	Product area part number
pversion	Product area version
pserialnumber	Product area serial number
passettag	Product area asset tag

Example

```
root@ATCA-7540:~# hpmcmd -c fruinfoget 0
Common Header:
    Format Version           = 1

Board Info Area:
    Version                  = 1
    Language Code            = 25
     Mfg Date/Time            = June 22 00:00:00 2018 (9453600 minutes
since 1996)
    Board Manufacturer       = ARTESYN
    Board Product Name       = PCA,ATCA-7540/0GB/R1.0/CFG0000
    Board Serial Number      = E187E4C
    Board Part Number        = 0106834S01C
    FRU Programmer File ID   = 9806869F08B

Product Info Area:
    Version                  = 1
    Language Code            = 25
    Manufacturer Name        = ARTESYN
    Product Name             = ATCA-7540
    Product Part / Model#    = ATCA-7540/0GB/R1.0/CFG0000
    Product Version          = R.1.0
    Product Serial Number    = E187E4C
```

```
    Asset Tag                =
    FRU Programmer File ID   = 9806869F08B

Multi Record Area:
    PICMG LED Descriptor Record (ID=0x2f)
        Version = 0
    OEM MAC Addresses Record (ID=0x01)
        Version = 1
    ARTESYN Unknown Record (ID=0x10)
    PICMG Board Point-to-Point Connectivity Record (ID=0x14)
        Version = 1
    AMC Carrier Information Table Record (ID=0x1a)
        Version = 0
    AMC Carrier Activation and Current Management Record (ID=0x17)
        Version = 0
    AMC Carrier Point-to-Point Connectivity Record (ID=0x18)
      Version = 0
    AMC Point-to-Point Connectivity Record (ID=0x19)
        Version = 0
```

5.3.4.11 fruinv

This command retrieves the FRU size and the addressable unit for the specified FRU.

Synopsis

```
fruinv <fruid> [-t ipmbAddr[:mmcAddr]]
```

Parameters

`fruid`	0 for the ATCA-7540 front board and 1 for the RTM
`-t`	Sends the command to: `ipmbAddr:mmcAddr`

Example

```
FrontBoard:
root@ATCA-7540:~# hpmcmd -c fruinv 0
FruSize = 2048
Accessed Units = Bytes
```

```
RTM:
root@ATCA-7540:~# hpmcmd -c fruinv 1
FruSize = 512
Accessed Units = Bytes
```

5.3.4.12 fruread

This command gets nBytes of fruid from the startAddress of the specified FRU.

Synopsis

```
fruread <fruid> <startAddress> <nBytes> [-t ipmbAddr[:mmcAddr]]
```

Parameters

`fruid`	0 for the ATCA-7540 front board and 1 for the RTM
`startAddress`	The starting address for reading the fruid
`nBytes`	Number of bytes to read in decimal
`-t`	Sends the command to: `IpmbAddr:mmcAddr`

Example

```
root@ATCA-7540:~# hpmcmd -c fruread 0 0 100
01 00 00 01 0b 14 00 df 01 0a 19 20 40 90 c7 41
52 54 45 53 59 4e e3 50 43 41 2c 41 54 43 41 2d
37 34 38 30 2f 31 30 35 57 2f 30 47 42 2f 52 31
2e 30 2f 43 46 47 30 30 30 30 c7 45 31 35 32 44
38 43 ca 30 31 30 36 38 34 38 4e 30 31 c2 32 20
c1 00 00 00 00 00 00 14 01 09 19 c7 41 52 54 45
53 59 4e c9
```

5.3.4.13 fruwrite

This command allows writing hexadecimal byte values to fruid starting at startAddr.

Synopsis

```
fruwrite <fruid> <startAddress> <hexval1> [hexval2] [...] [hexval16] [-t
ipmbAddr[:mmcAddr]]
```

Parameters

`fruid`	0 for the ATCA-7540 front board and 1 for the RTM
`startAddress`	The starting address for writing
`hexval1 .. hexvalN`	The hexadecimal value to write
`-t`	Sends the command to: `ipmbAddr:mmcAddr`

5.3.4.14 fwprogevent

This command sends a Firmware Progress Sensor Event to the Shelf Manager SEL. Refer to *IPMI specifications* for details on values.

Synopsis

```
fwprogevent <data1> <data2> <data3>
```

Parameters

`data1`	Stores hexadecimal value as "00" for Error, "01" for Hang, and "02" for Progress
`data2`	Stores hexadecimal value as "00-0D" for Error, "00-19" for Hang or Progress
`data3`	Stores hexadecimal value as "FF", unless an OEM data2 is specified

5.3.4.15 help

This command lists the available commands from the hpmcmd program with a brief explanation about the commands.

Synopsis

```
help
```

5.3.4.16 ipmbaddress

This command retrieves the blade IPMB address.

Synopsis

```
ipmbaddress
```

Example

```
hpmcmd -c ipmbaddress
ipmbaddress is 0x88
```

5.3.4.17 ipmcstatus

This command retrieves the status of given IPMC.

Synopsis

```
ipmcstatus [-t ipmbAddr]
```

Parameters

`-t`	Specifies the target with ipmbAddr

Example

```
hpmcmd -c ipmcstatus
IPMC Mode                =  NORMAL
Payload Control          =  Enabled
IPMC Outstanding Events  =   None
```

5.3.4.18 ledget

This command gets information about a specified LED controlled by the IPMC.

Synopsis

```
ledget <fruid> <led> [-t ipmbAddr[:mmcAddr]]
```

Parameters

`fruid`	0 for the ATCA-7540 Front board and 1 for the RTM
`led`	BLUE for the hot swap LED or LEDN for FRU LED<n>. <n> is a number between 1 and the maximum FRU LEDs supported by the blade.
`-t`	Sends the command to: `ipmbAddr:mmcAddr`

Example

```
hpmcmd -c ledget 0 led1
Current State = OVERRIDE
State Function/(ms) Duration(ms)      Color
---------------------------------------------
Override         Off       Always        Red
```

5.3.4.19 ledprop

This command displays the FRU LED properties under IPMC control.

Synopsis

```
ledprop <fruid>
```

Parameters

`fruid`	0 for the ATCA-7540 front board and 1 for the RTM

Example

```
hpmcmd -c ledprop 0
FRU LEDs under IPMC control:    LED0 = Blue, Default: Blue
LED1 = Amber,Red, Default: Red
LED2 = Amber,Green,Red Default: Green
LED3 = Amber, Default: Amber
```

5.3.4.20 ledset

This command controls the override state of a specific FRU LED.

Synopsis

```
ledset <fruid> <led> <operation> [offms] [onms] [color] [-t
ipmbAddr[:mmcAddr]]
```

Parameters

`fruid`	0 for the ATCA-7540 front board and 1 for the RTM
`led`	BLUE for the hot swap LED or LEDN for FRU LED<n>. <n> is a number between 0 and the maximum FRU LEDs supported by the blade.
`operation`	**ON** = enable override state and turn LED on. **OFF** = enable override state and turn LED off. **BLINK** = enable override state and blink LED; off_duration and on_duration specify the blink duration; the default on and off duration is 300 ms. **LOCAL** = cancel override state and restore LED control to the IPMC, that is, local state. **TEST** = run lamp test for specified on_duration, then restore prior state. The default duration is 5000 ms.
`offms`	Specifies OFF duration in milliseconds. It can have value from 10 ms to 2500 ms in 10 ms increments. It is valid only if the operation is BLINK.
`onms`	Specifies ON duration in milliseconds. It can have value from 10 ms to 2500 ms in 10 ms increments. It is valid only if the operation is BLINK.

`color`	LED0 = BLUE LED1 = RED or AMBER LED2 = GREEN (if supported by IPMC) LED3 = AMBER (if supported by IPMC)
`-t ipmbAddr`	Sends the command to: `ipmbAddr:mmcAddr`

Example

```
hpmcmd -c ledset 0 led1 on
```

5.3.4.21 loglevelget

This command retrieves the current hpmagentd log level. The log level of hpmcmd can be set with the environment variable, HPMCMD_LOG_LEVEL. For example, export HPMCMD_LOG_LEVEL=7 to set debug level. All log messages are sent to the syslog.

Table 5-4 Log Levels

0	Emergency
1	Alert
2	Critical
3	Error
4	Warning
5	Notice
6	Information
7	Debug

Synopsis

```
loglevelget
```

Example

```
hpmcmd -c loglevelget
5
```

5.3.4.22 shelftype

This command retrieves the shelf FRU (IPMB 20) Board Area Product Name (FRU 254).

Synopsis

```
shelftype
```

Example

```
hpmcmd -c shelftype
AXP-1440
```

5.3.4.23 macaddress

This command retrieves the list of available MAC addresses.

Synopsis

```
macaddress [fruid]
```

Parameters

`fruid`	0 for the ATCA-7540 front board and 1 for the RTM

Example

```
FrontBoard:
root@ATCA-7540:~# hpmcmd -c macaddress 0
Front or Rear Panel Interface MAC Addr :  ec:9e:cd:11:7a:f8
Front or Rear Panel Interface MAC Addr :  ec:9e:cd:11:7a:f9
Base Interface             MAC Addr :  ec:9e:cd:11:7a:f6
Base Interface             MAC Addr :  ec:9e:cd:11:7a:f7
Fabric Interface           MAC Addr :  ec:9e:cd:11:7a:fa
Fabric Interface           MAC Addr :  ec:9e:cd:11:7a:fb
Serial over Lan Interface  MAC Addr :  ec:9e:cd:11:7a:fc
Serial over Lan Interface  MAC Addr :  ec:9e:cd:11:7a:fd

RTM:
root@ATCA-7540:~# hpmcmd -c macaddress 1
Front or Rear Panel Interface MAC Addr :  ec:9e:cd:11:77:f2
Front or Rear Panel Interface MAC Addr :  ec:9e:cd:11:77:f3
Front or Rear Panel Interface MAC Addr :  ec:9e:cd:11:77:f4
Front or Rear Panel Interface MAC Addr :  ec:9e:cd:11:77:f5
Front or Rear Panel Interface MAC Addr :  ec:9e:cd:11:77:f6
Front or Rear Panel Interface MAC Addr :  ec:9e:cd:11:77:f7
Front or Rear Panel Interface MAC Addr :  ec:9e:cd:11:77:f8
Front or Rear Panel Interface MAC Addr :  ec:9e:cd:11:77:f9…
```

5.3.4.24 lancfgget

Gets LAN configuration parameter

Synopsis

```
lancfgget <channel> [param]
```

Parameters

`channel`	Channel number
`param`	auth-type-support auth-type-enables ip-addr ip-addr-src mac-addr subnet-mask ipv4-header-params primary-rmcp-port secondary-rmcp-port bmc-generated-arp-control gratuitous-arp-interval default-gateway-addr default-gateway-mac-addr backup-gateway-addr backup-gateway-mac-addr community-string num-destinations destination-type destination-addr vlan-id vlan-prio rmcp-cipher-support rmcp-ciphers rmcp-priv-levels dst-addr-vlan-tags

Example

```
root@ATCA-7540:~# hpmcmd -c lancfgget 1
IP Address         : 192.168.25.10
Subnet Mask        : 255.255.0.0
Default Gateway    : 172.16.0.1
```

5.3.4.25 lancfgset

Sets LAN configuration parameter

Synopsis

```
lancfgset <channel> <param> <value>
```

Parameters

`channel`	Channel number

Example

```
root@ATCA-7540:~# hpmcmd -c lancfgset 1 ip-addr 192.168.25.10
Successful lancfgset Operation
```

5.3.4.26 partnumber

This command retrieves the part number (FRU 0) of the main blade.

Synopsis

```
partnumber
```

Example

```
hpmcmd -c partnumber
```

5.3.4.27 physlotnumber

This command retrieves the physical slot number in which the blade is plugged in.

Synopsis

```
hpmcmd -c physlotnumber
```

5.3.4.28 portget

This command shows the current state of interfaces governed by e-keying. If no channel is specified, portget returns data for all channels in the specified interface. If neither interface nor channel are specified, portget will return data for all interfaces.

Synopsis

```
portget [interface] [channel] [devid]
```

Parameters

`interface`	Valid values: `BASE | FABRIC | UPDATE | AMC`
`channel`	A number from 1 to the maximum number of channels supported for the interface. Node blades usually support two base and two fabric channels, and switch blades support 16 base, 15 fabric, and one update channel.

Example

```
root@ATCA-7540:~# hpmcmd -c portget

  STATE   INTERFACE  CHANNEL  LINKTYPE  LINKEXT  GROUP  PORTS
-------------------------------------------------------------
 ENABLED     BASE       1        BASE       0       0    0
 ENABLED     BASE       2        BASE       0       0    0
 ENABLED   FABRIC       1    RESERVED       1       0    0,1,2,3
DISABLED   FABRIC       1       ETHER       4       0    0,1,2,3
DISABLED   FABRIC       1       ETHER       3       0    0
DISABLED   FABRIC       1       ETHER       4       0    0,1,2,3
 ENABLED   FABRIC       2    RESERVED       0       0    0,1,2,3
DISABLED   FABRIC       2       ETHER       3       0    0
DISABLED   FABRIC       2    RESERVED       1       0    0,1,2,3
DISABLED   FABRIC       2       ETHER       4       0    0,1,2,3
DISABLED   UPDATE       3         OEM       0       0    0
DISABLED   UPDATE       4         OEM       0       0    0
DISABLED   UPDATE       5         OEM       0       0    0
```

5.3.4.29 portset

This command enables and disables ports in a channel. The following table lists the valid values for each parameter.

Synopsis

```
portset <intf> <chan> <grpid> <type> <typeX> <ports> <oper> [devid] [-t
ipmbAddr[:mmcAddr]]
```

Parameters

`intf`	Valid values: `BASE | FABRIC | UPDATE | AMC`
`chan`	A number from 1 to the maximum number of channels supported for the interface. Node blades usually support two base and two fabric channels, and switch blades support 16 base, 15 fabric, and one update channel.
`grpid`	Specifies the group ID. It is always 0 according to the current shelf FRU information.
`type`	Valid values: `BASE | ETHER | EXPRESS | INFINI | STAR | OEM`
`typeX`	Valid values are: `0` (for 1000Base-BX) `1` (for 10GBase-BX4) `2` (for FC-PI)
`ports`	A sequence of ports to act on. For base and update channels, port is always 0. For fabric channels, port can specify up to 4 ports per PICMG 3.1: Option 1: 0 (for port 0) Option 2: 01 (for ports 0,1) Option 3: 0123 (for ports 0,1,2,3) Option 7: 3 (for port 3)
`oper`	Valid values are: `DISABLE or ENABLE`
`devid`	For AMC only: it is an on-carrier device ID that identifies the on-carrier device to which the desired channel is connected
`-t ipmbAddr`	Sends the command to: `ipmbAddr:mmcAddr`

Example

```
hpmcmd -c portset base 1 0 base 0 0 enable
```

NOTE: The portset command issues the command as specified to the IPMC. The command can also be issued to another IPMC in the chassis using the "-t" parameter. An IPMC may not implement certain commands in which case the command would be a no-op. The ATCA-7540 IPMC has no support to enable/disable base and/or fabric channels. This has to be done by sending commands to the ATCA-F140 IPMC.

5.3.4.30 posttypeget

This command retrieves the postType to which the board is currently set to run at boot time, for the specified CPU.

Synopsis

```
posttypeget <payload_cpu_selector>
```

Parameters

`payload_cpu_selector`	The specified CPU is set to postType to run. This should be 0 for the GPP, 1 for the SPP.

Example

```
hpmcmd -c posttypeget 0
LONG
```

5.3.4.31 posttypeset

This command sets the postType to which the board is currently set to run at boot time, for the specified CPU.

Synopsis

```
posttypeset <payload_cpu_selector> <newPostType>
```

Parameters

`payload_cpu_selector`	An integer between 0 and number of CPU devices supported per board
`newPostType`	Valid values: `SHORT | LONG`

5.3.4.32 sdr

This command shows the Sensor Data Records (SDRs).

Synopsis

```
sdr
```

Example

```
root@ATCA-7540:~# hpmcmd -c sdr

recID 1: management controller device locator record
    I2C slave addr:  44
    Channel number:  00
    Power state:     06
    Global init:     0C
    Capabilities:    2D
    Entity Id:       PICMG front board
    Entity instance: 60
    OEM:             00
    Id string:       ATCA-7540


recID 2: full sensor record
    owner is IPMB 88 sensor num 00 on lun 00 channel 00
    logical entity: PICMG front board - instance 60
    Hot Swap Carrier : FRU hot swap : sensor-specific discrete

recID 3: full sensor record
    owner is IPMB 88 sensor num 01 on lun 00 channel 00
    logical entity: RTM - instance 60
    Hot Swap RTM : FRU hot swap : sensor-specific discrete

recID 4: full sensor record
    owner is IPMB 88 sensor num 02 on lun 00 channel 00
    logical entity: PICMG front board - instance 60
    IPMB Physical : IPMC physical link : sensor-specific discrete

recID 5: full sensor record
    owner is IPMB 9C sensor num 06 on lun 00 channel 00
    logical entity: PICMG front board - instance 60
    Version change : 2B : sensor-specific discrete
```

5.3.4.33 sdr_dump

This command shows the SDRs in binary and hexadecimal format.

Synopsis

```
sdr_dump
```

Example

```
root@ATCA-7540:~# hpmcmd -c sdr_dump
SDR Records:
01 00 51 00 12 88 00 cc 2d 00 00 00 30 00 00 00  "..Q....Ì-...0..."
00 00 00 00 7c 00 00 00 00 00 00 00 00 00 00 00  "....|..........."
e0 06 02 00 00 00 00 00 02 00 00 00 00 00 00 00  "à..............."
```

5.3.4.34 sdrinfo

This command shows the SDR information.

Synopsis

```
sdrinfo
```

Example

```
root@ATCA-7540:~# hpmcmd -c sdrinfo
SDR Information:
LUN 0 has 079 sensors; dynamic sensor population
LUN 1 has 000 sensors
LUN 2 has 000 sensors
LUN 3 has 000 sensors
```

5.3.4.35 sendamc

This command allows to send any of the commands supported in the IPMI specifications to a remote AMC or MMC of a remote IPMC IPMB-L.

Synopsis

```
sendamc <IPMBaddress> <MMCaddress> <netfn> <cmd> <data0> ... <datan>
```

Parameters

`IPMBaddress`	Destination IPMB address in hex digits
`MMCaddress`	Destination MMC address in hex digits
`netfn`	IPMI request net function in hex digits
`cmd`	IPMI request command in hex digits
`data0-datan`	IPMI request data bytes, if any, in hex digits

5.3.4.36 sendcmd

This command allows a user to send any of the commands supported in the IPMI specifications to a remote IPMC.

Synopsis

```
sendcmd <IPMBaddress> <netfn> <cmd> <data0> ... <dataN>
```

Parameters

`IPMBaddress`	Destination IPMB address in hex digits
`netfn`	IPMI request net function in hex digits
`cmd`	IPMI request command in hex digits
`data0 ... dataN`	IPMI request data bytes, if any, in hex digits

Example

```
hpmcmd -c sendcmd 90 06 59
07 59 C1
```

5.3.4.37 shelfaddress

This command retrieves the shelf address string from the shelf FRU.

Synopsis

```
shelfaddress
```

Example

```
hpmcmd -c shelfaddress
01
```

5.3.4.38 shelfslots

This command retrieves the total number of blade slots in the shelf.

Synopsis

```
shelfslots
```

Example

```
hpmcmd -c shelfslots
14 slots
```

5.3.4.39 slotmap

This command displays a slotmap table for the shelf where the blade is installed.

Synopsis

```
slotmap
```

Example

```
hpmcmd -c slotmap
------------------------------------------------------------
Physical Slot: 01 02 03 04 . 05 06 07 08 09 10 . 11 12 13 14
Logical Slot : 01 03 05 07 . 09 11 13 04 06 08 . 10 12 14 02
IPMB Address : 82 86 8A 8E . 92 96 9A 88 8C 90 . 94 98 9C 84
------------------------------------------------------------
```

5.3.4.40 slotnumber

This command retrieves the logical slot number of the slot where the blade is plugged in.

Synopsis

```
slotnumber Parameters
```

Example

```
hpmcmd -c slotnumber
4
```

5.3.4.41 sel

This command shows the SEL records.

Synopsis

```
sel
```

Example

```
root@ATCA-7540:~# hpmcmd -c sel
0x01A2: Event: at: Sep 21 10:30:31 2018; from:(0xee,0,0);
sensor:(0xd7,30); event:(0x6f,asserted): a0 65 00
0x01A3: Event: at: Sep 21 10:30:31 2018; from:(0xee,0,0);
sensor:(0xda,34); event:(0x6f,asserted): a0 01 00
0x01A4: Event: at: Sep 21 10:30:31 2018; from:(0xee,0,0);
sensor:(0x08,31); event:(0x6f,asserted): 00 ff ff
0x01A5: Event: at: Sep 21 10:30:31 2018; from:(0xee,0,0);
sensor:(0x08,32); event:(0x6f,asserted): 00 ff ff
```

5.3.4.42 selinfo

This command shows the SEL information.

Synopsis

```
sel
```

Example

```
hpmcmd -c selinfo
root@ATCA-7540:~# hpmcmd -c selinfo
SEL version: 1.5
Number of log entries: 1023
Free space: 0 bytes
Events have been dropped due to lack of space in the SEL
Last addition timestamp: Sep 4 12:06:55 2018
Last erase timestamp: Sep 3 19:00:00 2018
Supported operations:
 - Reserve command supported
```

5.3.4.43 selclear

This command erases all contents of the SEL.

Synopsis

```
selclear
```

Example

```
hpmcmd -c selclear
```

5.3.4.44 solcfgget

This command gets the Serial over LAN (SOL) configuration parameter.

Synopsis

```
solcfgget <channel> [param]
```

Parameter

`channel`	Channel number
`param`	enable authentication char-settings retry non-volatile-bit-rate volatile-bit-rate payload-channel payload-port

Example

```
root@ATCA-7540:~# hpmcmd -c solcfgget 1
Enabled                          : true
Force Encryption                 : false
Force Authentication             : false
Privilege Level                  : Admin
Character Accumulate Interval (ms): 5
Character Send Threshold         : 1
Retry Count                      : 1
Retry Interval                (ms): 50
Non-Volatile Bit Rate        (kps): 115.2
Volatile Bit Rate            (kps): 115.2
Payload Port                     : 623
```

5.3.4.45 solcfgset

This command sets the SOL configuration parameter.

Synopsis

```
solcfgset <channel> <param> <value>
```

Parameters

channel	Channel number

`param`	force-encryption true|false force-authentication true|false privilege-level user|operator|administrator|oem char-accumulate-interval 1-1275 (ms) char-send-threshold 0-255 rety-count 0-255 retry-interval 0-2550 (ms) non-volatile-bitrate 9.6|19.2|38.4|57.6|115.2 volatile-bitrate 9.6|19.2|38.4|57.6|115.2 port 0-255

5.3.4.46 version

This command displays the version of the hpmcmd software.

Synopsis

```
version
```

Example

```
hpmcmd -c version
3.14.9
```

5.3.4.47 watchdog

This command is used to handle the payload BMC watchdog.

Synopsis

```
watchdog set <tmr_use> <tmr_action> <pre_timeout> <flags> <lsb_val>
<msb_val> [user]
watchdog set default
watchdog get
watchdog start
watchdog stop
watchdog reset
```

Parameters

`set`	Possible values are shown in the following table.

Value	Description
tmr_use	dont_stop stop
tmr_action	no_action hard_reset power_cycle power_down
pre_timeout	0-255
flags	clear dont_clear
lsb_val	0-255
msb_val	0-255
user	sms oem

Network Management Utility

6.1 Overview

The Network Management Utilities (NMU) tool can be used for the supervision of SFP+ devices and to enable/disable the laser of SFP+ modules. The following command displays all SFPs that are connected to the board:

```
nmucmd -c show trx
```

In order to use this command, the nmu_sfp driver must be loaded.

The nmu_sfp driver can be loaded using the command:

```
modprobe nmu_sfp
```

The Show command displays transceiver information, which is retrieved from the EEPROM of the SFP. Therefore, the SFP+ device must be compliant to the SFF-8472 specification. With the trx command, you can modify transceiver settings that are accessible in the EEPROM.

6.2 Installing nmucmd

6.2.1 NMU Application

The nmucmd tool can be found at `/opt/bladeservices/bin/`.

6.2.2 nmu_sfp driver

The nmucmd tool requires access to the EEPROM data of the SFP+ devices. This access is provided by the nmu_sfp kernel driver module.

The nmu_sfp driver depends on the Intel i40e device driver for the Intel XL722 NIC device and the ixgbe device driver for the Intel 82599 NIC devices (for example, on RTM-ATCA-7xxx-10G).

NOTICE

Make sure that the nmu_sfp driver is loaded after the Intel network device drivers, such as i40e and ixgbe. When removing, remove nmu_sfp first and then Intel network device drivers.

Caution

You might get kernel crashes if you do not follow this sequence.

6.3 Syntax of nmucmd Tool

```
root@ATCA-7540:~# /opt/bladeservices/bin/nmucmd -c help
-----------------------------------------------------------------
         Command  -  Description
-----------------------------------------------------------------
            help  -  list of commands.
            show  -  Prints information about different network
                     entities.
             trx  -  Modify transceiver settings
         version  -  Shows the version of this tool.

root@ATCA-7540:~# /opt/bladeservices/bin/nmucmd -c show

HELP: show switch
    Dumps switch information

SYNOPSIS:
    show switch

HELP: show trx
    Dumps transceiver information

SYNOPSIS:
    show trx [switch] [port] [property]
    where:
              switch  -  name of the switch
                port  -  number of the port
            property  -  vendor-name
                         vendor-part-number
                         vendor-revision
```

```
vendor-serial-number
vendor-date
connector-type
ethernet-code
fibre-ch-link-length
fibre-technology
fibre-media
encoding
bitrate
tx-state
rx-state
tx-fault
internal-temperature
internal-supply-voltage
laser-bias-current
capability-tx-fault
capability-soft-tx-fault
capability-tx-disable
capability-soft-tx-disable
capability-rx-los
capability-soft-rx-los
```

show command

The nmucmd tool is based on the NMU API, which is part of the Blade Services (BLSV) API. In the NMU architecture, an SFP is part of the hierarchical organization among switches and ports. A blade may have multiple switches and a switch may have multiple ports. An SFP is always connected to a port. There is no switch chip on the ATCA-7540 blade, but the NMU defines a virtual one which can be seen using the following command:

```
nmucmd -c show switch
```

For more information, see *Show switch/NIC device type on page 78*.

Using the following command, you can get the information about a SFP+ type:

```
nmucmd -c show <switch name> [portnumber] [property]
```

When the portnumber is specified in the command, the port specific SFP+ data is shown. If portnumber is not specified, all SFPs that are associated with the switch are displayed.

The property field is optional and using the following command, you can get a list of the possible properties such as, vendor name, temperature, serial number, and so on:

```
nmucmd -c help show
```

Without the property option, all properties of a SFP+ are displayed. For a user, it is difficult to find out which switch and port number combination is associated with which network interface. So, nmucmd supports specifying either the switch and port number or only the name of the network interface.

trx command

With the trx command, you can modify SFP+ settings. Using the following command, you can get a list of settings:

```
nmucmd -c help trx
```

For more information, see *Enable/disable tx-state of specified port on page 81*.

6.4 Examples

```
show bitrate of device i40e at port 1:
nmucmd -c show trx i40e 1 bitrate
    bitrate              : 10300
    MBd
```

6.4.1 Show switch/NIC device type

```
show switch/NIC type:
nmucmd -c show switch
i40e: Intel 40G NIC
```

6.4.2 Show data of all ports

```
show data of all ports assigned to the i40e device:
nmucmd -c show trx
Transceivers of switch i40e:
  Transceiver info at port 1:
    eth-device-name      : rtm10
    vendor-name          : Intel Corp
    vendor-part-number   : FTLX8571D3BCV-IT
    vendor-revision      : A
    vendor-serial-number : AT4062Q
```

```
  vendor-date          : 01/20/15
  connector-type       : LC
  ethernet-code        : 10G BASE-SR
  encoding             : 64b/66b Line Code
  bitrate              : 10300 MBd
  tx-state             : Disabled
  rx-state             : Lost
  tx-fault             : Ok
  internal-temperature : 33.41 degree Celsius
  internal-supply-voltage : 3320700 uV
  laser-bias-current      : 176 uA
  rx-input-power          : 0.20 uW (-36.99 dbm) type=average
  tx-output-power         : 0.00 uW
  capability-tx-fault        : supported
  capability-soft-tx-fault   : supported
  capability-tx-disable      : supported
  capability-soft-tx-disable : supported
  capability-rx-los          : supported
  capability-soft-rx-los     : supported
Transceiver info at port 2:
  eth-device-name      : rtm9
  vendor-name          : AVAGO
  vendor-part-number   : AFBR-57L5APZ
  vendor-revision      :
  vendor-serial-number : A9071001S1
  vendor-date          : 03/07/07
  connector-type       : LC
  ethernet-code        : 1000BASE-SX
  fibre-ch-link-length : intermediate
  fibre-technology     : Shortwave laser w/o OFC
  encoding             : 8B/10B Line Code
  bitrate              : 2100 MBd
  tx-state             : Disabled
  rx-state             : Lost
  tx-fault             : Ok
  internal-temperature : 31.49 degree Celsius
```

```
    internal-supply-voltage : 3311800 uV
    laser-bias-current      : 0 uA
    rx-input-power          : 0.00 uW type=OMA
    tx-output-power         : 0.00 uW
    capability-tx-fault        : supported
    capability-soft-tx-fault   : supported
    capability-tx-disable      : supported
    capability-soft-tx-disable : supported
    capability-rx-los          : supported
    capability-soft-rx-los     : supported
```

6.4.3 Show data of a specified port

show all data assigned to i40e - port6:

```
nmucmd -c show trx i40e 6
  Transceiver info at port 6:
    eth-device-name      : rtm3
    vendor-name          : FINISAR CORP.
    vendor-part-number   : FTLX8571D3BCV
    vendor-revision      : A
    vendor-serial-number : AR60NXN
    vendor-date          : 02/04/14
    connector-type       : LC
    ethernet-code        : 10G BASE-SR
    encoding             : 64b/66b Line Code
    bitrate              : 10300 MBd
    tx-state             : Disabled
    rx-state             : Lost
    tx-fault             : Ok
    internal-temperature : 35.56 degree Celsius
    internal-supply-voltage : 3268500 uV
    laser-bias-current      : 0 uA
    rx-input-power          : 0.00 uW type=average
    tx-output-power         : 44.50 uW (-13.52 dbm)
    capability-tx-fault        : supported
    capability-soft-tx-fault   : supported
    capability-tx-disable      : supported
```

```
    capability-soft-tx-disable : supported
    capability-rx-los          : supported
    capability-soft-rx-los     : supported
```

6.4.4 Show tx-state of specified port

```
show tx-state of device i40e at port 1:
nmucmd -c show trx i40e 1 tx-state
    tx-state             : Enabled
```

6.4.5 Enable/disable tx-state of specified port

```
set tx-state of device i40e at port 1:
nmucmd -c trx i40e 1 tx-state on
nmucmd -c show trx i40e 1 tx-state
    tx-state             : Enabled
nmucmd -c trx i40e 1 tx-state off
nmucmd -c show trx i40e 1 tx-state
    tx-state             : Disabled
```

6.4.6 Show device name of specified port

```
show eth device at i40e port 2:
nmucmd -c show trx i40e 2 eth-device-name
….eth-device-name       : rtm3
```

6.4.7 Display SFP+ temperature

```
show SFP temperature of device i40e at port 1:
nmucmd -c show trx i40e 1 internal- temperature
internal-temperature : 22.88 degrees Celsius
```

6.4.8 Display used bitrate

```
show bitrate of device nmu at port 1:
nmucmd -c show trx i40e 1 bitrate
bitrate: 10300 MBd
```

Board Control Module

7.1 Overview

Board control is a Linux kernel module, which provides access to the glue logic FPGA registers of the ATCA-7540 blade. The boardctrl driver also makes information about the board, like the board serial number, available in the directory `/proc/boardinfo`.

The boardctrl driver is a prerequisite for the firmware upgrade tool FCU.

After loading the driver, you will find data provided in the `/proc/boardinfo` directory. The information is shown in the following table.

Table 7-1 Files Maintained by boardctrl

File	Description	Sample output
bios_version	Shows the BIOS version	`1.0.1`
board_ version	Shows the board version as provided by the BIOS	`Board Version : R.1.0`
cpld	Shows some FPGA information (CPU presence, FPGA version, and so on)	`Board Name : ATCA-7540` `FPGA version : 14`

summary	Shows the summary of the board state (FPGA registers) and BIOS provided information	`root@ATCA-7540:~# cat /proc/boardinfo/summary`
		`Board Vendor : ARTESYN`
		`Board Name : ATCA-7540`
		`Board Version : R.1.0`
		`Board Serial Number: E187CD0`
		`BIOS Vendor : ARTESYN`
		`BIOS Version : 1.0.3`
		`BIOS Release Date : 18-Sep-2018`
		
		`Memory Module:`
		`Device/Bank : CPU0_DIMM_A1/NODE 1`
		`Size : 8192 Mbyte`
		`Data Width : 64 Bit`
		`Manufacturer : Micron`
		`….`

7.2 Board Control Tool

The board control module provides an ioctl (input/output control) interface which can be used by userland applications. As an example, fpgatool uses this interface.

7.2.1 FPGATOOL

This is a tool to read/write to the FPGA register set.

NOTE: Some of the registers are write-protected and cannot be overwritten by userland applications.

The fpgatool can be found at `/opt/bladeservices/bin/fpgatool`.

Synopsis

```
root@ATCA-7540:~# fpgatool
usage: fpgatool <command> <parameter> Available commands: read write dump info version
```

Example

```
read from register 0x50 (LED Control register) root@ATCA-7540:~# fpgatool read 0x50
Offset: 0x50  Value: 0x00 - 00000000
write to register 0x50 (LED Control register)
root@ATCA-7540:~# fpgatool write 0x50 0x01
Old Value: 0x00 - New Value: 0x01
dump complete register set:
root@ATCA-7540:~# fpgatool dump
```

Related Documentation

A.1 SMART Embedded Computing Documentation

The documentation listed is referenced in this manual. Technical documentation can be found by using the Documentation Search at *https://www.smartembedded.com/ec/support/* or you can obtain electronic copies of SMART EC documentation by contacting your local sales representative.

Document Title	Publication Number
ATCA-7540 Installation and Use Guide	6806871A01
ATCA-7540 Quick Start Guide	6806871A02
ATCA-7540 Safety Note Summary	6806871A03