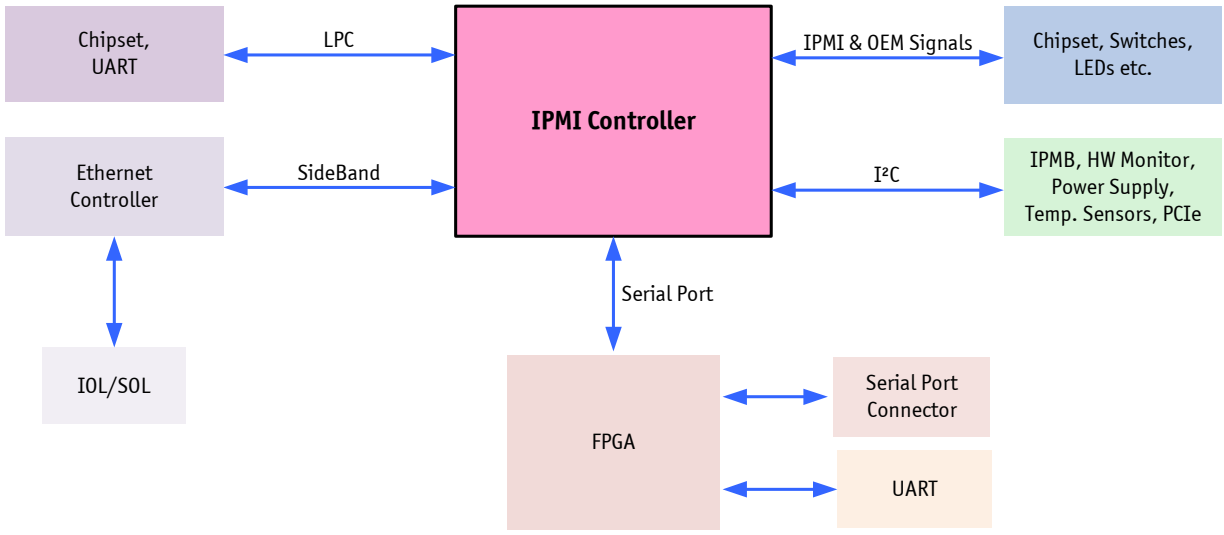
kontron

# » User Guide «




## IPMI Firmware

Doc. ID: 1055-7522, Rev. 1.0

Date: December 20, 2013

# Revision History

| Revision | Brief Description of Changes | Date of Issue |
|---|---|---|
| 1.0 | Initial issue | 20-Dec-2013 |

# Imprint

Kontron Europe GmbH may be contacted via the following:

MAILING ADDRESS

Kontron Europe GmbH
Sudetenstraße 7
D - 87600 Kaufbeuren Germany

TELEPHONE AND E-MAIL

+49 (0) 800-SALESKONTRON
sales@kontron.com

For further information concerning other Kontron products, please visit our Internet website: www.kontron.com.

# Disclaimer

# Contents

# 1 IPMI Overview

The Kontron IPMI implementation fully supports the Intelligent Platform Management Interface specification. The IPMI functionality operates under an autonomous management controller even if the board is held in reset or power-down mode by a management card within a system designed for high availability.

While the IPMI implementation is fully compliant with IPMI v2.0 and has been designed to operate with any system management software (SMS) that respects this specification, can be easily integrated with the Service Availability Forum — Hardware Platform Interface (SAF HPI) specification.

More information about Service Availability can be found on the following website:
http://www.saforum.org/

For further information concerning IPMI refer to the following website:
http://www.intel.com/design/servers/ipmi/

For board-specific IPMI information, refer to the IPMI chapter of the user guide provided with the CompactPCI board or the AMC module.

# 2 IPMI Controller Configuration

## 2.1 IPMI in a CompactPCI Chassis

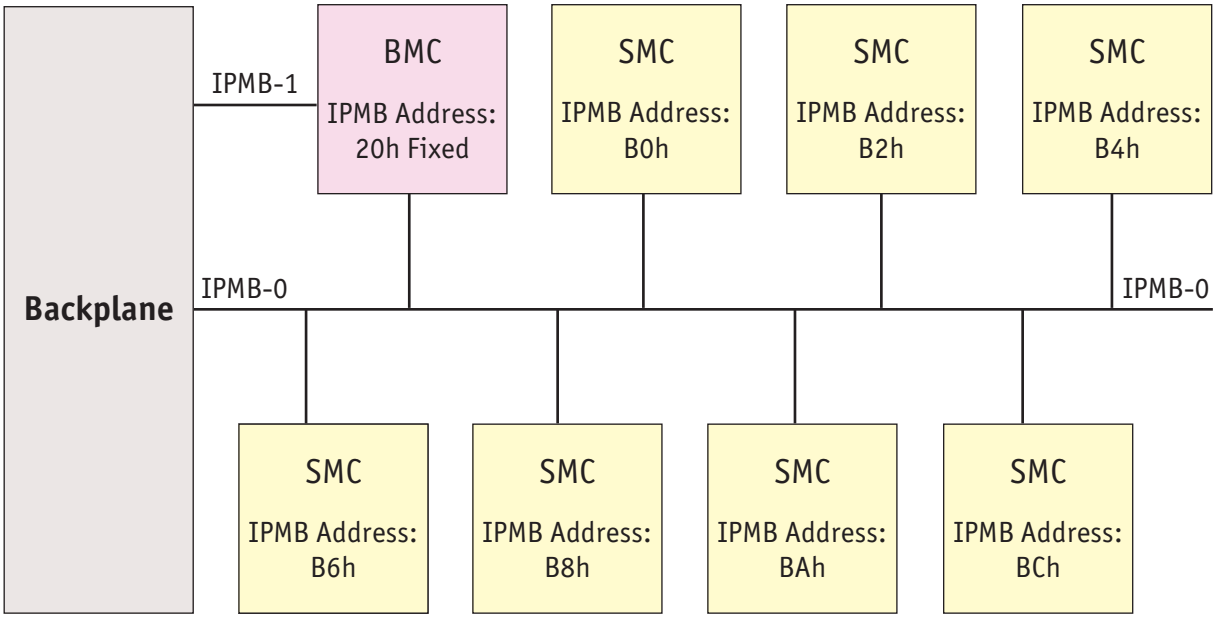
IPMI is an extensible and open standard that defines autonomous system monitoring. It is autonomous because every management controller within a CompactPCI chassis monitors its own sensors and sends critical events through a dedicated bus to the BMC that logs it into a non-volatile System Event Log (SEL). The IPMI implementation includes a device SDR repository module that allows the user's system management software to detect all system components and build a database of all management controller sensors.

Kontron's IPMI implementation in the CompactPCI environment is compliant with the PICMG 2.9 R1.0 specification. This specification defines the pinout of the J1 and J2 CompactPCI connectors as well as the addressing scheme.

Each CompactPCI board is equipped with an IPMI controller acting either as a BMC or as an SMC. There should be only one BMC in the chassis, or at least on the IPMB segment. The BMC may reside either on a board, or on an external system management card, or in a shelf management controller (ShMC). The specification allows all of these variants. As a BMC in the system slot, the board supports dual-ported IPMB (IPMB-0 to the SMCs and IPMB-1 to the external segments via the CompactPCI backplane connector). The BMC administrates the SEL and the SDRR for the complete system. In a CompactPCI chassis, there can be several SMCs. The SMC administrates the sensor and FRU data of the board and makes it available to the BMC. Each SMC can be connected to the BMC via a dedicated bus (IPMB-0).

**Table 1: BMC Mode vs. SMC mode**

| BMC MODE | SMC MODE |
|---|---|
| Fixed IPMB address: 20h | IPMB address is calculated based on the Geographical Adress (GA), e.g. slot number: B0h (slot1), B2h (slot 2), ... BEh (slot 8) |
| System Event Log (SEL) storing all events sent to BMC via IPMB | Events are sent to BMC (20h) via IPMB by default and are also stored in the local System Event Log (SEL) |
| Sensor Data Record Repository (SDRR) may contain all Sensor Data Records (SDR) of all IPMI controllers in the chassis | No SDRR |




The IPMB address of the SMCs is determined by the geographic address of the slot.

To use the IPMI resources in a CompactPCI system requires an initial setup for IPMI operation. The following actions must first be performed to achieve operable IPMI functionality.

### 2.1.1 IPMI Firmware Setup

To select the BMC or the SMC mode, the **kIpmi** uEFI Shell command is used. Upon every board reset, the uEFI BIOS forwards the user settings (BMC or SMC mode) to the IPMI controller. The IPMI controller's factory default setting is SMC mode.

### 2.1.2 IPMI Setup for the Rack

For a working IPMI configuration, the SDRR of the BMC must be filled with all sensor data records of all IPMI controllers in the rack. After every system start the BMC uses the SDRR to initialize all sensors of all boards in the rack. The SDRR setup must be done by a management tool, e.g. the open-source tool "ipmitool", after system modification. Then the command is:

```
ipmitool sdr fill sensors
```

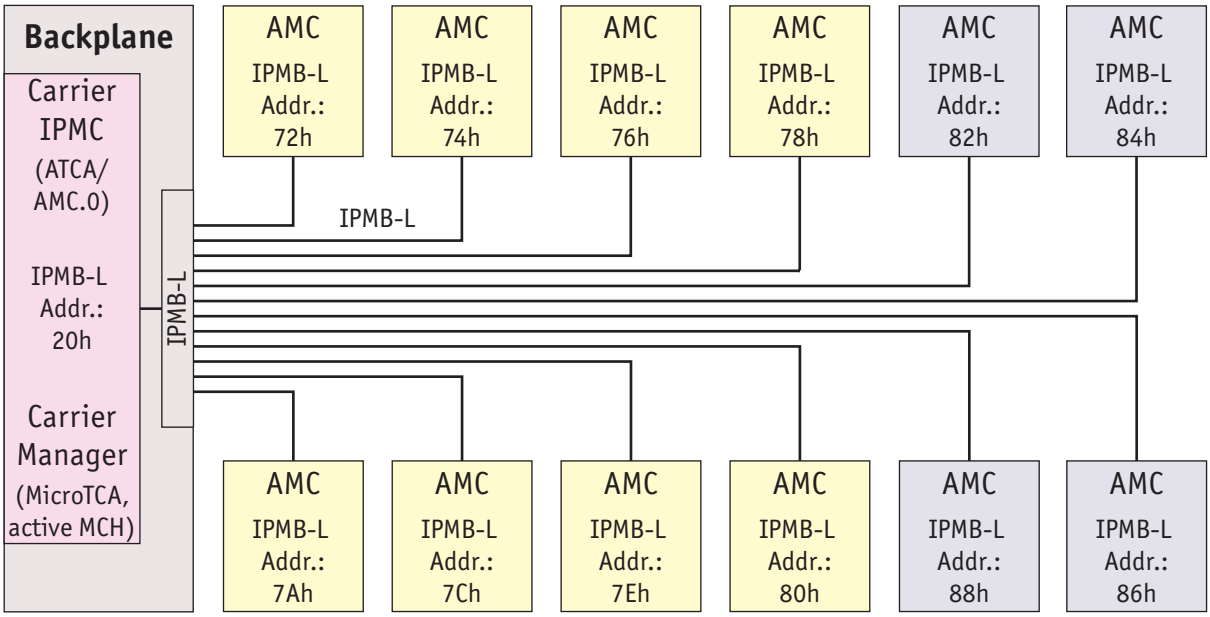
This will only work if the IPMI controller configured as BMC is addressed. This addressing is the default if the “ipmitool” is running on the payload side of the board where the BMC is residing.

## 2.2 IPMI in a an MicroTCA/ATCA Environment

The Module Management Controller (IPMI controller of an AMC module) is a crucial component of any AMC module. Besides acting as a regular IPMI management controller (sensor monitoring, event logging, etc.), it also provides an interface to all necessary data related to module power requirements and implemented interfaces (E-Keying). Further, it plays an active role in the module hot swap state management.

The MMC provides access to various sensors which permit the monitoring of:

- System power voltages: +12V (PWR), +3.3V (MP)
- AMC module voltages: 5V, +3.3V
- Temperatures: CPU and PCH die as well as airflow near AMC edge-connector
- Power Good, LAN links, IPMB link, board reset, POST code, boot error, CPU States (processor hot, THERMTRIP, ...), IPMB-L state, Health error, IPMI watchdog, etc.




The IPMB-L address of the AMC modules is determined by the geographic address of the slot.

### 2.2.1 IPMI in AdvancedMC / AdvancedTCA Environment

The IPMI Controller of the carrier (Carrier IPMC) communicates with the MMC using the local IPMB (IPMB-L) bus. In an ATCA/AMC environment, it is the Carrier IPMC that actually turns on/off the module (payload) power. However, before the Carrier IPMC enables the module payload power, various criteria must be satisfied by both the carrier and the module, including power requirements and capabilities, matching interfaces, current module hot swap state, and any other special conditions as specified by the Shelf Manager policy.

### 2.2.2 IPMI in a MicroTCA Environment

The first element of a MicroTCA Controller Hub (MCH) is its MicroTCA Carrier Management Controller (MCMC) with its carrier manager function as the central authority in a MicroTCA carrier to monitor and control the AMC modules. The Carrier Manager function makes use of IPMB-L links to each AMC, as well as presence detect, module, and power enable signals. The MicroTCA shelf manager also monitors events and controlls the cooling units.

# 3 IPMI Firmware Code

## 3.1 IPMI Firmware Code

### 3.1.1 Structure and Functionality

The IPMI firmware code is organized into a boot code and an operational code, both of which are stored in a flash device. Upon an IPMI controller reset, the IPMI controller first executes the boot code which:

- Performs a self-test to verify the status of the IPMI controller's hardware including its memory, and
- Calculates a checksum of the operational code.

After successful verification of the operational code checksum, the firmware will execute the operational code. Only the operational code is upgradable in the field.

### 3.1.2 uEFI BIOS—IPMI Controller Interaction

For communication between the uEFI BIOS and the IPMI controller, the KCS interface is used. During the boot process, the uEFI BIOS sends various IPMI commands to the IPMI controller, such as:

- An OEM command which reports the version of the uEFI BIOS
- An OEM command which reports the operational status of the uEFI BIOS
- The standard IPMI command **`Set Watchdog Timer`** to stop a possibly running IPMI watchdog timer
- The standard IPMI command **`Set SEL Time`** to set the event log time to the time which is kept by the RTC (CompactPCI)
- The standard IPMI command **`Set ACPI Power State`** to set the state **`ACPI legacy on`** (CompactPCI)

### 3.1.3 Firmware Upgrade

The IPMI's operational code can be upgraded via the open-source tool “ipmitool” or via uEFI BIOS commands. The upgrade tool/commands allow download and activation of new operational code and also rollback to the “last known good” operational code. Additionally, the status and the firmware version of the redundant firmware copies can be checked.

For local or remote firmware upgrade, the following IPMI interfaces are available:

- KCS (locally, requires active payload, but fast)
- IPMB (remote, independent of the payload state)
- LAN (remote, via IOL, requires also active payload)

During the download process, the currently running operational code operates as usual until the activation command is issued. During the activation process, the IPMI controller is off-line for about 20 seconds while the boot code is reorganizing the firmware storage. Afterwards, it starts the new operational code. If this doesn't succeed, after a timeout the boot code performs an automatic rollback to the “last known good” operational code.

#### 3.1.3.1 Firmware File Formats

Firmware images for upgrade are provided in the following format:

- Firmware images in the PICMG defined HPM.1 file format,
  e.g. FW-IPMI-<BOARD>-<REL>-FWUM.hpm,
  for usage with **“ipmitool hpm ..”** commands.

where:

<*BOARD*>identifies the board family of the IPMI firmware (e.g. B400h for the CP6005)

<*REL*>identifies the release (version) of the IPMI firmware.

#### 3.1.3.2 Firmware Upgrade via “ipmitool hpm”

Firmware upgrade using an HPM.1 file requires at least “ipmitool” version 1.8.10.

The firmware upgrade procedure starts with downloading the HPM.1 file using, for example, the following command:

```
ipmitool hpm upgrade <HPM.1_FWFile>.hpm
```

The next step is the activation of the newly downloaded IPMI firmware. This is done using:

```
ipmitool hpm activate
```

Detailed information about the currently active firmware versions or the redundant copies can be obtained using the commands mentioned below.

To obtain detailed version information of the active IPMI firmware, use the following command:

```
ipmitool hpm compprop 1 1
```

To obtain the version of the rollback copy (only valid if a newly downloaded firmware is already activated), use the following command:

```
ipmitool hpm compprop 1 3
```

To obtain the version of the newly downloaded IPMI firmware (only valid after download and before activation), use the following command:

```
ipmitool hpm compprop 1 4
```

To obtain detailed information about the IPMI HPM.1 upgrade capabilities, use the following command:

```
ipmitool hpm targetcap
```

To perform a manual rollback to the previously good firmware image, use the following command:

```
ipmitool hpm rollback
```

### 3.1.4 Firmware Upgrade via “uEFI kIpmi hpm”

The new IPMI firmware must be available from the uEFI Shell using, for example, a USB flash drive. The firmware upgrade procedure starts with downloading the HPM.1 file using, for example, the following command:

```
fs0:\<path>> kIpmi hpm upgrade <HPM.1_FWFILE>.hpm
```

After successful download, the new IPMI firmware is automatically activated. Information about the currently active firmware versions of the rollback firmware version copies can be obtained using the command:

```
fs0:\<path>> kIpmi info
```

Version information of the HPM.1 file can be obtained via the following command:

```
fs0:\<path>> kIpmi hpm info <HPM.1_FWFILE>.hpm
```

To perform a manual rollback to the previously good firmware image, use the following command:

```
fs0:\<path>> kIpmi hpm rollback
```

### 3.1.5 Firmware Upgrade via “uEFI kUpdate”

Firmware updates are typically delivered on a ZIP archive containing only the firmware images. The content of the archive with the directory structure must be copied on a data storage device with FAT partition.

The update procedure via is performed via the **kUpdate -u** uEFI Shell command. To select a specific firmware image from the ZIP archive, the **kUpdate -s** uEFI Shell command is used. When using the **kUpdate** command, the structure of ZIP archive must not be altered.

To perform automatic upgrade during boot-up, use the **kBoardConfig AutoUpdate** uEFI Shell command.

For further information on the uEFI Shell commands, refer to the uEFI BIOS chapter in the respective manual.

### 3.1.6 Setting the SEL Time (CompactPCI)

The IPMI controller does not have its own hardware real-time clock. Therefore, after start-up, restart or upgrade of the IPMI controller, its software clock first must be supplied with the current time. The IPMI controller uses the time when handling event messages which otherwise will have an out-of-date time stamp.

Every time when the uEFI BIOS starts up, it supplies the IPMI controller with the payload’s current time.

Restarts of the IPMI controller without a following uEFI BIOS reboot will result in invalid date and time indication e.g. after a firmware upgrade. In order to apply correct timestamps to the SEL records, issue the IPMI command **`Set SEL Time`**. This may be done by application software on the payload side via the KCS interface or by a remote IPMI controller via the IPMB-0.

# 4 FRU Data

## 4.1 Structure and Functionality

The FRU data contains information about the board such as the part number and the serial number. See PICMG Specification 2.9 for complete details on the FRU data structure.

The IPMI controller provides 4 kB non-volatile storage space for FRU information. For further information regarding the FRU data, refer to IPMI — Platform Management FRU Information Storage Definition v1.0, Document Revision 1.1.

Full low-level access to read or write the board's FRU Information is provided by regular IPMI FRU Device commands. Care must be taken when writing FRU information directly using standard IPMI commands. Invalid FRU information may disturb a shelf management software which uses the FRU data.

## 4.2 FRU Version Identification

The FRU data fields, as defined in the IPMI specification, are used to record the version of the FRU installed. The revision number is incremented for each new release of FRU data.

Example of board FRU ID: “STD_R10”

Example of product FRU ID: “STD_R10”

## 4.3 Board-Specific FRU Data

The following FRU data areas and data fields are provided:

FRU Board Info Area

- Manufacturing date / time
- Board manufacturer(C7): “Kontron”
- Board Product Name(C6): “CPxxxx” or “AMxxxx” 1)
- Board Serial Number(CF): “123456789012345” 1)
- Board Part Number(C9): “1234-5678” 1)
- FRU File ID(C7): “STD_R10”1)

FRU Product Info Area

- Product manufacturer(C7): “Kontron”
- Product Name(C6): “CPxxxx” or “AMxxxx” 1)
- Product Part Number(C2): “00” 1)
- Product Version(D9): “000000000000000000000000” 2)
- Product Serial Number(D9): “000000000000000000000000” 2)
- Asset Tag(D9): “_____” 2)
- FRU File ID(C7): “STD_R10”1)
- CustomData(D5): “MAC=CC:CC:CC:CC:CC:CC” 1)

1) Example value / field will be modified during the manufacturing process.

2) Field is free for user. Please note that changes need special care (checksums).

## 4.4 FRU Data Update

Typically, an update of the FRU data is not necessary because the board’s correct FRU data is installed at the factory. If an update of the FRU data is required, it can be done via regular IPMI FRU device commands. The correct FRU data must be prepared at the factory. Please contact Kontron for further assistance.

## 4.5 E-Keying (AMC)

E-Keying has been defined in the AMC.0 R2.0 Specification to prevent module damage, prevent malfunction, and verify bay connection compatibility. Therefore, the FRU data of an AMC module contains PICMG-defined records which describe the module’s AMC interoperability:

- Module Current Requirements Record
- Clock Configuration Record, for the PCI Express reference clock and Telecom clocks
- AMC Point-to-point Record, describing module’s AMC port capabilities

The IPMI commands **`Set AMC Port State`** and **`Get AMC Port State`** defined by the AMC.0 specification are used by the carrier or MCH for either granting or rejecting the E-keys (i.e. enabling or disabling of AMC ports during E-Keying).

The respective DIP Switch on the AMC module can be used to forcibly disable some AMC ports and to configure the PCI Express reference clock if required. Please refer to the AMC module's user guide for further information.

# 5 XMC Card Support in a CompactPCI Environment

The presence or absence of an XMC card is reported by the "XMC present" IPMI sensor. If an XMC card is present, the card's FRU data EEPROM is readable/writable. The size of the EEPROM must be smaller or equal to 256 bytes because of 8-bit EEPROM addressing. Please note that the XMC FRU size is always reported as 256 bytes and writing to locations that are higher than the real capacity should be avoided. The FRU data of the XMC card can be read under Linux using **`ipmitool fru print 1`**.

# 6 Hot Swap and Shutdown

## 6.1 Hot Swap and Shutdown of a CompactPCI Board

### 6.1.1 Hot Swap Handle and Hot Swap LED

To perform the actions required for hot swapping of the board, a hot swap state machine with the following M-states generated by the IPMI controller is used:

- M0:Board Not Installed
- M1:Board Inactive
- M2:Board Activation Request
- M3:Board Activation in Progress
- M4:Board Active
- M5:Board Deactivation Request
- M6:Board Deactivation in Progress

For further information of the hot swap state machine, please refer to the PICMG® AMC.0 R2.0, Advanced Mezzanine Card Base Specification, Chapter 3.6.

The blue Hot Swap LED (HS LED) of an inserted board in a powered rack is normally used to indicate the board's operational status so as to facilitate hot-swapping of the board:

- **Hot Swap LED On**

  The payload is inactive:
  - The board may be activated by closing the hot swap handle, or
  - The board may be extracted. The M-state is 1.

» When the payload power is off e.g. after a shutdown via an IPMI chassis command and the handle is still closed, the M-state is 1.

» **Hot Swap LED Blinking**

Changing from active state to inactive state or vice versa.

The M-state is 2, 5 or 6. Do not extract the board and do not actuate the hot swap handle during these states.

Blinking pattern:

» long on, short off: the IPMI firmware is in M-state 2 and starts the payload

» long off, short on: the IPMI firmware is in M-state 5 or 6 and shuts down the payload. Wait until the HS LED stops blinking and remains on to extract the board.

» **Hot Swap LED Off**

The payload is active.

Don't extract the board now. Normally the extraction is impossible because the hot swap handle is closed and locked. The M-state is 3 or 4.

Normally the logical states “active” and “inactive” of a payload are identical to the physical states “handle open” and “handle closed” or “payload power off” and “payload power on”.

If, however, the power is switched on or off using IPMI chassis commands or the payload is shut down by the OS, then the position of the hot swap handle and the power state may become asynchronous. In this case the blue HS LED is switched on indicating that the payload power is switched off although the handle is closed. Such actions are not part of the hot swap process and are governed by their own functionality which is not within the scope of this document.

### 6.1.2 The Hot Swap and Shutdown Processes

Hot swap, as defined here, is the purposely initiated process to remove and replace an active board in a powered system. To accomplish this requires that the hot swap process provides for an orderly transition of the payload from the active to inactive state and vice versa. This is necessary to preclude improper system operation and possible loss of data. The board has all the necessary features including hardware and IPMI software to support hot swapping. On the software side, however, not all available OSs support hot swapping, not even partially. Three possible cases for hot swapping based on OS capabilities are described as follows.

**Case 1: Involves an OS which does not support ACPI**

After payload power-on, the starting uEFI BIOS will inform the IPMI controller by sending the IPMI command **`Set ACPI Power State`** / **`Set Legacy on.`** If no further **`Set ACPI Power State`** commands are sent and the hot swap handle is opened, the IPMI controller will immediately disable the payload power.

In this event, the application/operator is responsible for the termination of all payload processes prior to opening the hot swap handle and initiating removal/replacement of the board to avoid improper operation or loss of data.

**Case 2: Involves an OS which emulates ACPI support**

An OS which does not really support ACPI, such as VxWorks, is able to obtain “Graceful Shutdown” support from the IPMI controller by performing in the following way.

After start-up, such an OS must manipulate the chipset in a way that prevents an immediate power-off when the “power button” is logically activated.

Then it must send the IPMI command **`Set ACPI Power State / S0/G0 working`** to the IPMI controller to enable this to process later on an **`S3/G2 soft off`** command.

During application operation the system must cyclically read the “Hot Swap Sensor” (sensor #0) using the IPMI command **`Get Sensor Reading`**. This allows the tracking of the board's state. After the board has once reached M-state 4 (sensor reading is 10h) the leaving of this announces that the handle was opened. Now the time has come to terminate all processes.

After all critical processes have been terminated, the OS must send the IPMI command **`Set ACPI Power State / S3/G2 soft off`** to the IPMI controller which will set the power off immediately.

**Case 3: Involves an OS which supports ACPI**

When an OS is started which supports ACPI, the IPMI command **`Set ACPI Power State / S0/G0 working`** is sent to the IPMI controller. This indicates that the OS has reprogrammed the chipset in such a manner that a “power button” signal does not lead to an immediate power-off but only causes an event that can be detected by the OS.

When the handle is opened, the IPMI controller asserts the “power button” signal to notify the OS. The OS then shuts down all processes and afterwards causes the transmission of the IPMI command **`Set ACPI Power State / S3/G2 soft off`** to the IPMI controller which then switches the power off.

## 6.2 Hot Swap and Shutdown of an AMC Module

As a hot-swappable field replaceable unit (FRU), the board also follows the same stringent carrier grade RASM feature set, namely — Reliability, Availability, Serviceability, Maintainability. When offered in combination with AdvancedTCA platforms, TEM (Telecom Equipment Manufacturers) clients literally conserve valuable system AdvancedTCA system slots. The board supports full hot swap capability as per PICMG 3.0. It can be removed from or installed in the system while it is on (without powering down the system). Please refer to the PICMG 3.0 specification for additional details.

During hot swap of a working module, the payload side has to be shut down automatically on command of the MMC and the end of shutdown has to be signalled back to the MMC. Because the board supports ACPI, an OS on the payload side which supports this too makes shutdown very easy. If the OS doesn't support ACPI, there is a special method to be used.

### 6.2.1 Method 1: The Payload OS Supports ACPI

Requirements:

- The ACPI daemon must be active.
- An ACPI power button event must result in a system shutdown.

Hot swap operation sequence processed by MMC and OS:

- On command of the carrier controller, the MMC simulates the pressing and release of the power button to force an ACPI event.
- The ACPI daemon detects this ACPI event and initiates the shutdown of the payload software system.
- At the end of the shutdown, the payload hardware system reports the sleep state to the MMC by setting the appropriate signal line.
- The MMC detects the sleep state and reports this to the carrier controller (“quiesced”) so that the hot swap processing can be continued and finished.

By default the MMC waits endlessly for the sleep state. Please note that some shelf managers or MCHs use a timeout to simply switch off a module which needs too much time to reach sleep state. As this might be an undesirable situation, refer to the appropriate manual for further assistance. In any event, if an endless wait is to be avoided, it is possible to set a timeout time for the module's MMC after which the system will be switched off unconditionally. For the setting of the timeout refer to refer to the IPMI Chapter of the AMC module’s user guide.

### 6.2.2 Method 2: The Payload OS Does Not Support ACPI

Requirements:

- At system start on the payload side, the Kontron shutdown daemon “grnsd” must be started. It is included in the Linux board support packages for the board. This daemon communicates cyclically with the MMC for the exchange of states, commands and acknowledges. For this, it uses the **OEM Module Quiescence Feedback** command. In principle, it plays the same role as the ACPI daemon of Method 1 above.

Hot swap operation sequence processed by MMC and OS:

- On command of the carrier controller the MMC sets a “shut down request” flag.
- The “grnsd” daemon recognizes this request in the response to its cyclical **OEM Module Quiescence Feedback** command and initiates the shutdown of the payload software system.
- At the end of the shutdown process, the “grnsd” daemon informs the MMC by setting the appropriate flag when calling the **OEM Module Quiescence Feedback** command.
- The MMC reports this to the carrier controller so that the hot swap processing can be continued and finished.

By default the MMC waits endlessly for this information. If an endless wait is to be avoided, it is possible to set a timeout time after which the system will be switched off unconditionally. For the setting of the timeout refer to the IPMI Chapter of the AMC module's user guide.

# 7 LAN Functions

## 7.1 Overview

The IPMI controller supports IPMI over LAN (IOL) and Serial over LAN (SOL). Common for both types of communication is the use of the RMCP/RMCP+ protocol for the packing of the data to be transferred. The RMCP/RMCP+ protocol uses the TCP port 623 by default.

While IOL serves to transport IPMI commands and their responses via Gigabit Ethernet, SOL serves to transport any serial data via Gigabit Ethernet. In each case, the IPMI controller serves as a protocol encoder and decoder. IOL is able to use both RMCP and RMCP+ protocols. SOL works only with the RMCP+ protocol.

Please note that IOL and SOL need the Ethernet device to be powered. Therefore, the board (payload) must be fully powered.

For information on IOL/SOL support, refer to the user guide provided with the CompactPCI board/AMC module.

## 7.2 Setting Up the Ethernet Channel

There are two methods to configure the LAN settings (IOL/SOL) for the Ethernet channels:

- By use of the **kIpmi net** uEFI Shell command in the uEFI BIOS
- By use of the open-source tool "ipmitool" or IPMI commands

The setup methods are compatible, i.e. both methods show the parameters which are set by the other one.

When the MAC addresses are set, the ones which are programmed into the hardware must be re-used. This is a restriction. The IP addresses of a channel being used by "normal" payload traffic and IOL/SOL traffic may differ but need not differ as long as port 623 is not used in parallel by payload and IOL/SOL.

## 7.3 Basic Setup from uEFI Shell

With the **kIpmi net** command from the uEFI Shell some basic settings such as IP address, sub-net mask and gateway address can be set up for the Ethernet channels.

## 7.4 Setup by “ipmitool” or IPMI Commands

The open-source tool “ipmitool” offers commands for the setup of the Ethernet channels. All possible options are shown by issuing:

```
ipmitool lan set
```

If “ipmitool” is not usable, the LAN parameters can be set by using standard IPMI commands as defined in the IPMI specification.

To show the current LAN parameters for a channel, “ipmitool” offers the command:

```
ipmitool lan print <channel>
```

## 7.5 Setup of User Accounts and Password

The open-source tool “ipmitool” offers commands for the listing and manipulation of user accounts for the SOL/IOL channels. An overview can be obtained by issuing:

```
ipmitool user
```

The predefined user accounts for a channel can be listed using the following command:

```
ipmitool user list <channel>
```

For every channel, the firmware has these predefinitions in non-volatile memory:

```
ID  Name            Callin  Link Auth  IPMI Msg   Channel Priv Limit
1                   false   true       true       USER
2   admin           false   true       true       ADMINISTRATOR
```

Please note that the **ADMINISTRATOR** password is preset with **admin**.

Changed accounts and passwords stay valid after payload power-off.

The accounts must be activated using the following command:

```
ipmitool user enable <user number>
```

## 7.6 IPMI Over LAN

IPMI over LAN (IOL) is used to allow the IPMI controller to communicate with the IPMI controller via LAN using the RMCP or the RMCP+ protocol. The data transferred are IPMI commands and the responses to them.

To enable LAN support after parameter setup the following command must be issued:

```
ipmitool lan set <channel> access on
```

Please note that the following commands must use the IP address which belongs to the enabled channel.

The open-source tool “ipmitool” can serve as a control program and user interface for this. “ipmitool” allows the issuing of generic IPMI commands such as:

```
ipmitool -I lan -H 192.168.3.189 -U admin -P admin -A PASSWORD raw 6 1
```

or to call complex functions like “mc info”:

```
ipmitool -I lan -H 192.168.3.189 -U admin -P admin -A PASSWORD mc info
```

This uses many generic IPMI commands to get the information needed.

## 7.7 Serial Over LAN

Serial over LAN (SOL) connects the COM1 or /dev/ttyS0 respectively of the board's payload side to an Ethernet channel. The IPMI controller resides between this serial interface and one of the Ethernet channels. It serves as an encoder and a decoder for the RMCP+ protocol used and controls the data stream. Outside the CompactPCI board/AMC module, for example, the open-source tool “ipmitool” can be used to drive the SOL session, i.e. it offers a console function to communicate via Ethernet with the board’s serial interface.

The IPMI firmware supports only “straight password authentication” SOL sessions with default privilege level USER.

Opening an SOL session requires special parameters as shown below:

```
ipmitool -I lanplus -H 192.168.3.189 -U admin -P admin -L USER -C 0 sol
activate
```

The serial interface can be used as a connection, for example:

- to a user program on the board’s payload
- to the uEFI BIOS. Refer to the Main Setup menu, Console Redirection function in the uEFI BIOS chapter of the respective User Guide for further information. The serial parameters can be set via this function.
- to a Linux login console. This can be activated after payload start, for example, by the command:

  ```
  getty -h 115200 /dev/ttyS0
  ```

SOL supports and requires serial hardware handshake. This should be activated for the serial port. Otherwise the transmitted data might get lost. In any case the same serial parameters for the payload side serial interface and the IPMI controller's serial interface must be used.

The parameters for the IPMI controller's serial interface can be set by using the following command:

```
ipmitool sol set
```

This command shows all options that can be set.

Further options are listed after issuing the following command:

```
ipmitool sol help
```

# 8 OS Support / Tools

## 8.1 ipmitool

A very useful all-in-one Linux tool is “ipmitool” (http://ipmitool.sourceforge.net ). It provides a user-friendly interface to many IPMI features and extensions, for example, to get sensor readings, change sensor thresholds or to access other IPMI controllers via IPMB.

# 9 Terminology and Acronym Definitions

The following table provides descriptions for terms and acronyms used in this guide. The descriptions are derived primarily from the IPMI specifications.

**Table 2: Terminology and Acronym Definitions**

| TERM or ACRONYM | DESCRIPTION |
|---|---|
| BMC | Baseboard Management Controller |
| BSP | Board Support Package |
| FRU | Field Replaceable Unit |
| FWH | Firmware Hub memory location where a complete uEFI BIOS code is stored |
| $I^2C$ | Inter-Integrated Circuit |
| IPMB | Intelligent Platform Management Bus |
| IPMB-0 | Intelligent Platform Management Bus which connects all SMCs with the BMC or the shelf manager |
| IPMI | Intelligent Platform Management Interface |
| IOL | IPMI over LAN |
| KCS | Keyboard Controller Style Interface<br>(Interface on the host system payload to communicate with the IPMI controller) |
| MP | Management Power (power supply for the IPMI controller) |
| PICMG | PCI Industrial Computer Manufacturer Group |
| PWR | Payload Power (power supply for the board) |

**Table 2: Terminology and Acronym Definitions (Continued)**

| TERM or ACRONYM | DESCRIPTION |
|---|---|
| SDR | Sensor Data Record (data structure that defines an IPMI sensor) |
| SDRR | Sensor Data Record Repository<br>(located in the BMC and may contain all SDRs of the chassis' boards that are administrated) |
| SEL | System Event Log (located in the BMC and keeps track of all events in the chassis) |
| SMBIOS | System Management BIOS |
| SMC | Satellite Management Controller |
| SMS | System Management Software (designed to run under the OS) |
| SOL | Serial over LAN |

# 10 Related Publications

The following publications contain information relating to the Kontron IPMI firmware.

**Table 3: Related Publications**

| PRODUCT | PUBLICATION |
|---|---|
| IPMI | IPMI Specification V2.0 |
| IPMI | IPMI- Platform Management FRU Information Storage Definition v1.0,<br>Document Revision 1.1 |
| PICMG | CompactPCI System Management Specification PICMG 2.9 Rev. 1.0<br>CompactPCI Hot Swap Specification PICMG 2.1 Rev. 2.0<br>PICMG® AMC.0 R2.0, Advanced Mezzanine Card Base Specification, Nov. 15, 2006 |
| CPxxxx /AMxxxx | User Guide for the respective product |
| MicroTCA | PICMG® MTCA.0 Micro Telecommunications Computing Architecture R1.0 |
| AMC | PICMG® AMC.0, Advanced Mezzanine Card Specification R2.0<br>PICMG® AMC.1, PCI Express R2.0<br>PICMG® AMC.2, Gigabit Ethernet R1.0<br>PICMG® AMC.3, Storage Interfaces R1.0 |

CORPORATE OFFICES

**Europe, Middle East & Africa**

Oskar-von-Miller-Str. 1
85386 Eching / Munich
Germany
Tel.: + 49 (0) 8165 / 77 777
Fax: + 49 (0) 8165 / 77 219
info@kontron.com

**North America**

14118 Stowe Drive
Poway, CA 92064-7147
USA
Tel.: + 1 888 294 4558
Fax: + 1 858 677 0898
info@us.kontron.com

**Asia Pacific**

17 Building,Block #1, ABP.
188 Southern West 4th Ring Road
Beijing 100070, P.R.China
Tel.: + 86 10 63751188
Fax: + 86 10 83682438
info@kontron.cn