Chassis Manager
User Manual

Revision history:

- 19.11.2019 REV 1.3: - Fixed typos and inconsistencies
- 16.04.2018 REV 1.2: - Added environmental information
- 23.01.2017 REV 1.1: - Added information on PEF (Platform event Filtering), PET (Platform Event Trap), BIT (built in test) and extraction of configuration file
- 01.02.2014 REV 1.0: - Initial Release

Table of Contents

List of Figures

List of Tables

1 Product description and functions

The purpose of the Chassis Manager (ChMC) is to watch over the basic health of the System Platform, to report anomalies, and takes corrective action when needed. The ChMC can retrieve inventory information or sensor readings, can perform basic recovery operations such as power cycle or resets.

The Chassis Manager communicates with other Field Replaceable Units (FRUs) inside the VITA 46.11 System Platform by sending IPMI messages over I2C buses (IPMB).

Beside managing intelligent FRUs attached to the IPMB, the Samway's Chassis Manager provides dedicated inputs for external sensors that could be used for monitoring chassis parameters like: backplane voltages and currents, temperatures, fans speeds, digital signals (PSU FAIL, switch buttons,...). The chassis manager also provides configurable digital outputs that could be used to turn on LEDs or shut down PSUs in case of failures. (Figure 1)

The Chassis Manager uses Sensor Data Records to describe the monitored parameters. For any measured parameter up to 6 thresholds can be defined: lower non critical, lower critical, lower non-recoverable, upper non critical, upper critical and upper non-recoverable. The measured values are retrievable at any time via the RS232 serial interface or Ethernet. In addition, limits and system parameters can be changed at any time with the unit in service. As a result, the Chassis Manager – and hence the connected system - can be controlled and monitored online via any computer with an Internet connection.

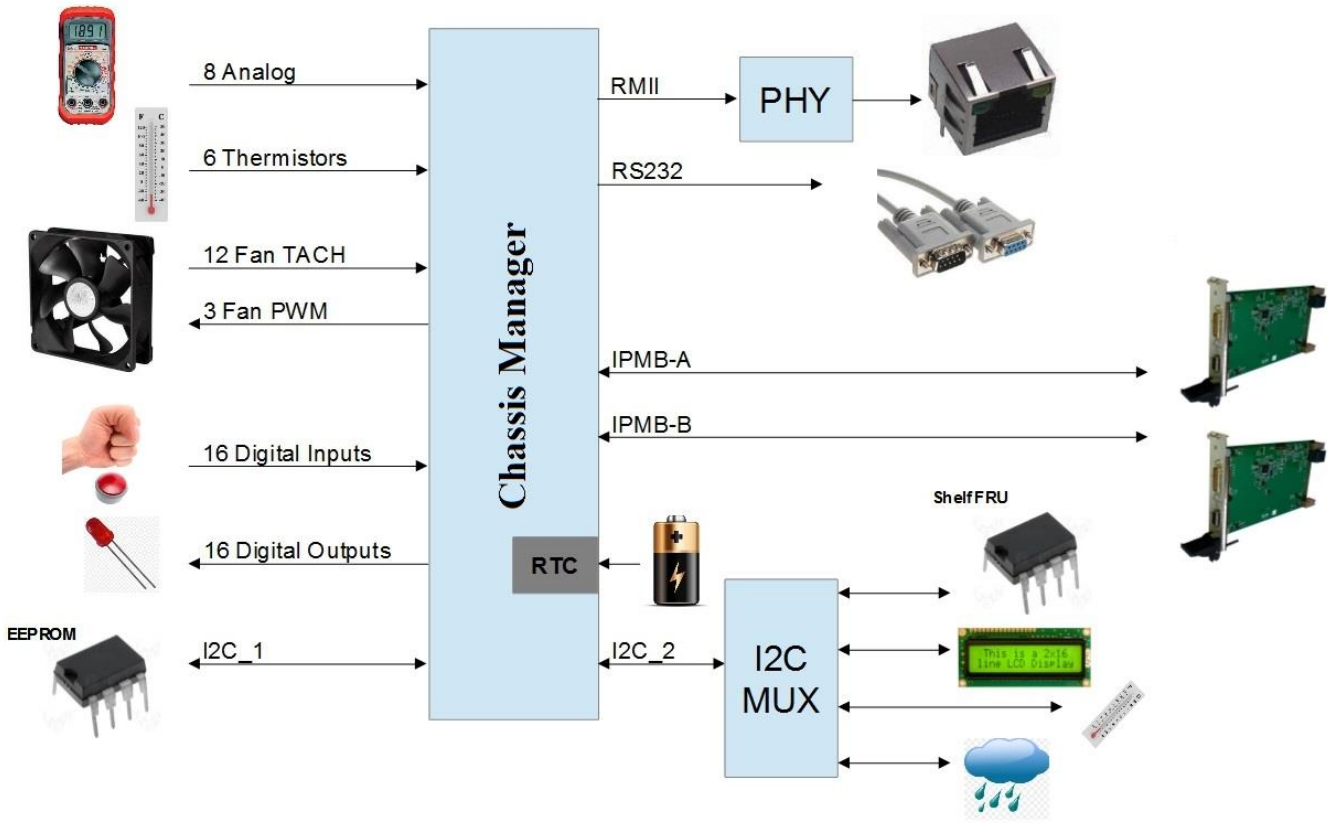

Figure 1: Chassis Manager Interface

2 BIT (Built In Test)

The ChMC is equipped with 3 different BIT functions:

- **P-BIT**: power on BIT. It is ran at every power on and verifies the functionality of the main hardware components of the ChMC
- **C-BIT**: continuous BIT. Runs all the time and verifies that all monitored values are ok
- **I-BIT**: interactive BIT. Started by request in order to verify the functionality of the main hardware components of the ChMC

2.1 P-BIT

This set of tests runs at every power on of the ChMC and its purpose is to verify that all the hardware components are operating correctly. After all checks are done the result of the test can be :

- System OK!
- Errors Detected!

In case errors were detected the ChMC will print a report of the checks that failed as a cumulative error mask. The possible error codes are given in the table bellow:

Error MASK	Error details
0x00000001	Error in setting the core clock frequency for the micro-controller
0x00000002	Wrong Hardware ID
0x00000004	Internal flash memory error
0x00000008	Calibration error for the Analog to Digital converter. Analog measurements may be flawed
0x00000010	Sensor Event Log (SEL) is full. New events can no longer be stored until the SEL is erased.
0x00000020	Local I2C Bus error.
0x00000040	External I2C 1 Bus error.
0x00000080	External I2C 2 Bus error.
0x00000100	Default Factory settings file detected. Default MAC address will be used.
0x00000200	Default User settings detected.
0x00000400	Wrong User settings version detected. The default Settings will be used instead.
0x00000800	Wrong checksum for the User settings. The default Settings will be used instead.
0x00001000	EEPROM Not Acknowledge
0x00002000	Default Sensor list detected
0x00004000	Not Acknowledge of the IO Expander for the auxiliary signals
0x00008000	Not Acknowledge of the IO Expander for the digital outputs
0x00010000	Not Acknowledge of the IO Expander for the digital outputs
0x00020000	Not Acknowledge of the I2C Multiplexer
0x00040000	IPMB A Bus Error
0x00080000	IPMB B Bus Error
0x00100000	Error in the initialization of the Locator memory
0x00200000	Error in the initialization of the Time zone variable
0x00400000	External Flash memory error

Error MASK	Error details
0x00800000	External RAM memory error
0x01000000	Not Acknowledge of the Local Temperature sensor
0x02000000	Not Acknowledge of the Elapsed Time Meter
All other values	Not used.

Table 1: BIT error mask

***For example**:*

The message: ***Errors Detected! ERROR MASK:0x00002300*** means that the Default Factory settings, User settings and Sensor lists are detected.

2.2 C-BIT

The Continuous BIT consist in monitoring and filtering the events sent by all the available sensors: both local sensors of the ChMC and also sensors of the managed FRUs.

The events that trigger a C-BIT error are configured using PEF (Platform event Filter). This way the C-BIT is very flexible and can be setup up differently for each Chassis.

In the default configuration all assertion events of Voltage, Temperature and Fan sensors generate a C-BIT error.

For PEF filters intended for C-BIT error generation, the OEM action will have to be enabled in the filter action field.

All events that trigger C-BIT errors can be later reviewed using the bit CLI command.

The status of the C-BIT and the error log can be reset using a CLI command. For more details refer to the bit command.

The C-BIT can have the following status:

- System Ok!
- Errors Detected! Details in the c_bit log.

2.3 I-BIT

The Interactive BIT is started by request using the bit cli command. It consists in a set of tests intended for validating the hardware components used by the ChMC for normal operation. The I-BIT can have the following status:

- System Ok! : I-BIT finished and all tests were passed successfully
- NA: I-BIT was never started so there is no data available
- Errors detected!

In case the I-BIT discovered issues they will be displayed as a cumulative error mask. The error codes are the same as for the P-BIT and are presented in a table at the P-BIT subchapter.

3 Supported IPMI Commands

The Chassis Manager was developed based on the IPMI v1.5 and ANSI/VITA 46.11 specifications.

IPM Device "Global" Commands	NetFn	CMD
Get Device ID	App	01h
Cold Reset	App	02h
Get Self Test Results	App	04h
Get ACPI Power State	App	06h
Send Message	App	34h
Get Channel Authentication Capabilities	App	38h
Get Session Challenge	App	39h
Activate Session	App	3Ah
Set Session Privilege Level	App	3Bh
Close Session	App	3Ch
Get Session Info	App	3Dh
Chassis Commands	**NetFn**	**CMD**
Get Chassis Capabilities	Chassis	00h
Get Chassis Status	Chassis	01h
Chassis Control	Chassis	02h
Event Commands	**NetFn**	**CMD**
Set Event Receiver	S/E	00h
Get Event Receiver	S/E	01h
Platform Event	S/E	02h
Sensor Device Commands	**NetFn**	**CMD**
Get Device SDR Info	S/E	20h
Get Device SDR	S/E	21h
Reserve Device SDR Repository	S/E	22h
Set Sensor Hysteresis	S/E	24h
Get Sensor Hysteresis	S/E	25h
Set Sensor Threshold	S/E	26h
Get Sensor Threshold	S/E	27h
Set Sensor Event Enable	S/E	28h
Get Sensor Event Enable	S/E	29h
Get Sensor Reading	S/E	2Dh
FRU Device Commands	**NetFn**	**CMD**
Get FRU Inventory Area Info	Storage	10h
Read FRU Data	Storage	11h
Write FRU Data	Storage	12h
SDR Device Commands	**NetFn**	**CMD**
Get SDR Repository Info	Storage	20h

Reserve SDR Repository	Storage	22h	
Get SDR	Storage	23h	
SEL Device Commands	**NetFn**	**CMD**	
Get SEL Info	Storage	40h	
Reserve SEL	Storage	42h	
Get SEL Entry	Storage	43h	
Add SEL Entry	Storage	44h	
Delete SEL Entry	Storage	46h	
Clear SEL	Storage	47h	
Get SEL Time	Storage	48h	
Set SEL Time	Storage	49h	
VITA 46.11 Commands	**NetFn**	**Group ID**	**CMD**
Get VSO Capabilities	Group Extension	VSO(03h)	00h
Get Chassis Address Table Info	Group Extension	VSO(03h)	01h
Get Chassis Identifier	Group Extension	VSO(03h)	02h
Chassis Identifier	Group Extension	VSO(03h)	03h
FRU Control	Group Extension	VSO(03h)	04h
Set IPMB State	Group Extension	VSO(03h)	09h
Set FRU State Policy Bits	Group Extension	VSO(03h)	0Ah
Get FRU State Policy Bits	Group Extension	VSO(03h)	0Bh
Set FRU Activation	Group Extension	VSO(03h)	0Ch
Get Device Locator Record ID	Group Extension	VSO(03h)	0Dh
Get Chassis Manager IPMB Address	Group Extension	VSO(03h)	1Bh
Set Fan Policy	Group Extension	VSO(03h)	1Ch
Get Fan Policy	Group Extension	VSO(03h)	1Dh
FRU Control Capabilities	Group Extension	VSO(03h)	1Eh
FRU Inventory Device Lock Control	Group Extension	VSO(03h)	1Fh
FRU Inventory Device Write	Group Extension	VSO(03h)	20h
Get Chassis Manager IP Addresses	Group Extension	VSO(03h)	21h
Get FRU Address Info	Group Extension	VSO(03h)	40h
Get FRU Persistent Control	Group Extension	VSO(03h)	41h
Set FRU Persistent Control	Group Extension	VSO(03h)	42h
FRU Persistent Control Capabilities	Group Extension	VSO(03h)	43h
Get Mandatory Sensor Numbers	Group Extension	VSO(03h)	44h

Table 2: List of Supported Commands

4 Chassis Discovery

4.1 IPMC Discovery

The starting point of the discovery process is the *Chassis FRU Information*.

Using the *Chassis Address Table Record* the ChMC determines the potential addresses of all the other IPMCs in the chassis. The list of potential addresses will also be filled out with the address of event generators.

For a Tier 1 IPMC the discovery process will be initiated only if it is instantiated in the Chassis Address Table.

Tier 2 IPMCs will always be discovered, as they make their presence known by sending events.

The Discovery process uses a set of three commands: *Get Device ID, Get VSO Capabilities* and *Get Mandatory Sensor Numbers*. The purpose of the process is to determine if an IPMC is present and if it is compliant to VITA 46.11. Only IPMCs that satisfy both conditions are managed by the ChMC.

The discovery process is an on going task and allows management for IPMCs/FRUs that appear at a later time.

4.2 System IPMB Discovery

The ChMC supports both IPMB buses: A and B. IPMB A is always enabled and used to communicate to the other IPMCs.

At each startup the ChMC scans the *Chassis FRU Information* and looks for the *Chassis IPMB Descriptor Record*. If the *IPMB_B Supported* bit is set, or the record is missing, the ChMC also enables its IPMB B for chassis communication.

If the chassis supports IPMB B,its usage for communicating to an IPMC is decided by the discovery process of that address. If the IPMC sets the *Number of supported IPMB Interfaces* to 01b in the response of the Get VSO capabilities request that is part of the discovery process, the ChMC will send a *Set IPMB State* command to the IPMC to enable IPMB B.

4.3 IPMC/FRU Polling

After a VITA 46.11 compliant IPMC/FRU has been successfully discovered, the ChMC starts an on going periodic polling task. The ChMC uses the on going polling for managing the IPMC by obtain information about its FRU operational State and availability.

Depending on the Tier level of the IPMC, the polling process is different.

For a Tier 2 IPMC the polling process consists in sending:

- Get *Sensor Reading* request for the FRU State Sensor. This request is used to determine any change of the FRU Operational States of the IPMC.
- *Get Device SDR Info.* This request is used to determine any changes in a dynamic SDR repository that would require a rebuild of the SDR repository.

For a Tier 1 IPMC the polling process consists in sending:

- Get *Sensor Reading* requests for all the mandatory sensors: FRU State Sensor, System IPMB Link Sensor, FRU Health Sensor, FRU Voltage Sensor, FRU Temperature Sensor, Payload Test Results Sensor, Payload Test Status Sensor. As Tier 1 IPMCs do not send events, these requests are used to determine the values for all monitoring parameters.

5 Event Handling

5.1 Sensor Event Log (SEL)

The ChMC acts as an event receiver for its chassis and saves all events in the Sensor Event Log (SEL).

The SEL is implemented using a non volatile Flash memory and can hold information for up to 65534 events.

Depending on the configuration after the log reaches full capacity the ChMC can behave in two ways:

1. If the SEL is a linear implementation or if ageing has been disabled for the circular implementation: A warning message will be displayed and all new events will be disregarded. New Events will be logged only after the sel is cleared.
2. If the SEL is circular and ageing is enabled: The oldest events will be erased in order to make room for new ones.

For each event the log provides the following information:

- time stamp
- number and name of the sensor that generated the event
- event type: - for threshold sensors: UNR,UC,UNC,LNC,LC,LNR
  - for discrete sensors: 1 (Asserted), 0 (De-Asserted)
- sensor value that triggered the event and threshold value (only for threshold sensors)

Example 1. Sensor event log

```
 %>sel print startup
-------------------------------------------------------------------------------
                            Sensor  Event Log

Rec.ID. dd.mm.yyyy hh:mm:ss  Sensor No. Name             Event  Ev.Dir  Value  Threshold
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
0x0001  01.01.2012 00:00:00     97       ChMC Power ON     1 (Asserted)
0x0002  01.01.2012 00:00:00     2        V0                LNC     As      0.00     2.86
0x0003  01.01.2012 00:00:00     4        V2                LC      As      0.00     11.38
0x0004  01.01.2012 00:00:00     6        V4                LC      As      0.00     2.16
0x0005  01.01.2012 00:00:00     8        V6                LC      As      0.00     0.24
0x0006  01.01.2012 00:00:00     3        V1                LC      As      0.00     4.76
0x0007  01.01.2012 00:00:00     5        V3                UC      As      2.58     -11.34
0x0008  01.01.2012 00:00:00     7        V5                LC      As      0.00     0.66
0x0009  01.01.2012 00:00:00     9        V7                LC      As      0.00     12.39
```

Each ChMC restart is marked by an event 1 (Asserted) for the *"ChMC Power On Sensor".*
For more details and syntax refer to the ***sel*** command section of this user manual.

5.2 Platform event Filtering (PEF) and Platform Event Trap(PET)

The ChMC supports PEF and PET compatible with IPMI rev. 1.5. This document will not describe the parameters used by the ChMC for configuring PEF and PET. For more details on each parameter and a simple setup procedure for generating a trap please refer to the *ChMC PEF & PET Getting Started* application note.

5.2.1 Platform Event Filtering

Using PEF the ChMC can determine which platform event message should be used to generate a SNMP trap. The filtering algorithm is based on 2 elements: an event filter and an alert policy. Optionally an alert string can be assigned for each alert policy/event filter match. Besides the filtering elements the algorithm also has other parameters that are used for configuring its behavior.

The ChMC supports two methods for changing the PEF parameters: command line interface (CLI) commands (accessible over the local serial port or remotely over telnet) or IPMI commands that can be sent remotely using ipmitool.

5.2.2 Platform Event Trap

For events that match a PEF event filter that supports alerting actions, the ChMC will send a specific format of SNMP trap called Platform Event Trap (PET). The trap format is defined by the IPMI PET specification and will not be discussed by this document.

The PET data is generated using the parameters received from PEF and the event data. Usually PEF defines the alert string and event severity parameters that will be used by the trap.

In rare cases when these fields are not defined in the event filter the ChMC uses the event data and the SDR of the sensors to fill out the trap parameters.

For events that do not have alert strings assigned, the ChMC will generate a string containing the name of the sensor. If the event is generated by a threshold sensor the parsed value that triggered the event and the threshold code will also be added to the alert string.

If the event severity parameter is not defined by PEF, the ChMC will also fill out this info based on the event data. For threshold sensors assertion events for:

- Lower Non Critical and Upper Non Critical thresholds will generate Non Critical event severity
- Lower Critical and Upper Critical thresholds will generate Critical event severity
- Lower Non-Recoverable and Upper Non-Recoverable thresholds will generate Non-recoverable event severity.

De-assertion events of threshold sensors or events of other sensor types will generate traps with the event Severity parameter set to 0x02 (Information).

5.2.3 Event Filtering for C-BIT

The on board Continuous BIT is also based on platform event filtering. For generating a C-BIT error the ChMC can filter both local sensor events coming from the embedded system monitor and also sensor events generated by the monitored FRUs. In order to generate a C-BIT error a PEF filter has to have the OEM action enabled in the actions field.

6 Controlling a VPX Chassis

The ChMC supports the *Chassis Control* command so it can be used to control the Chassis as a whole unit. The actions that can be performed are:

- *Power Down*: All FRUs are deactivated using a *Set FRU Activation (Deactivate FRU)* request. This action is performed when the ChMC receives a *Chassis Control (Power Down)* request,or a *Chassis Control (Soft Shutdown)* or it the **chassis_control power_down** CLI command is used.
- *Power Up*: All FRUs that are in the M1, FRU operational state state, are activated using a *Set FRU Activation (Activate FRU)* request. This action is performed when the ChMC receives a *Chassis Control (Power Up)* request or it the **chassis_control power_up** CLI command is used.
- *Power Cycle*: All FRUs that are in the M4 or M5, FRU operational state, are deactivated using a *Set FRU Activation (Deactivate FRU)* request and are re-activated using a *Set FRU Activation (Activate FRU)* request after they reach M1. This action is performed when the ChMC receives a *Chassis Control (Power cycle)* request or it the **chassis_control power_cycle** CLI command is used.
- *Cold Reset*: All FRUs that are in the M4 or M5, FRU operational state, are reset using a *FRU Control (Cold Reset)* request. This action is performed when the ChMC receives a *Chassis Control (Hard Reset)* request or it the **chassis_control reset** CLI command is used.
- *Issue Diagnostic Interrupt*: All FRUs that are in the M4 or M5, FRU operational state, will receive a *FRU Control (Issue Diagnostic Interrupt)")* request. This action is performed when the ChMC receives a *Chassis Control (Pulse Diagnostic Interrupt)* request or it the **chassis_control int** CLI command is used.

7 Monitoring environmental parameters

The ChMC uses Sensor Data Records(SDRs), compliant to IPMI 1.5, to describe the monitored chassis parameters.

The chassis monitoring and control can be set up using a configuration file. The config files can be opened, altered and saved using a GUI Config Tool.

Using the Config Tool the used can:

- specify up to **6** thresholds for all monitored parameters
- decide the way the software treats a limit infringement(events and outputs can be enabled or disabled for each individual threshold)
- change the name for every sensor
- enable/disable the monitoring of a sensor by loading/removing its SDR

7.1 Voltage monitoring

- Monitoring of up to 8 voltages (+3.3V,+5V, +12V, -12V and 4 user defined voltages).
- Any limit infringement is an internal event, and can control any of the 16 digital Outputs.
- Voltage monitoring parameters can be changed via the CLI or by upgrading the config file.
- CLI Commands for the voltage monitoring: ***voltage , local_sensor***

7.2 Temperature monitoring

- Monitoring of up to 6 analog temperature sensors (10 Kohm NTC thermistors with ß=3950)
- Temperature range from –20°C to +100°C

- Temperature measurement accuracy +/-3°C (max.)
- Monitoring of up to 8 TMP75 I2C sensors connected to SDA3/SCL3 (Digital Temp connector on the carrier card)
- Any limit infringement is an internal event, and can control any of the 16 digital Outputs.
- Any temperature sensor can be used in the fan control algorithm(The user can choose for each temperature sensor the fan control groups the sensor is active for.)
- Commands for the temperature monitoring: ***temp , local_sensor***

7.3 Fan monitoring and control

- Monitoring of up to 12 fans.
- Control of PWM fans
- Speed control via 3 fan control groups
- Any limit infringement is an internal event, and can control any of the 16 digital Outputs.
- Commands for the fan monitoring: ***fan, local_sensor***

7.3.1 Fan control

The Speed of the fans can be controlled using one of the 3 fan control groups. Each group controls an independent PWM1 signal using a user-defined temperature-speed characteristic.

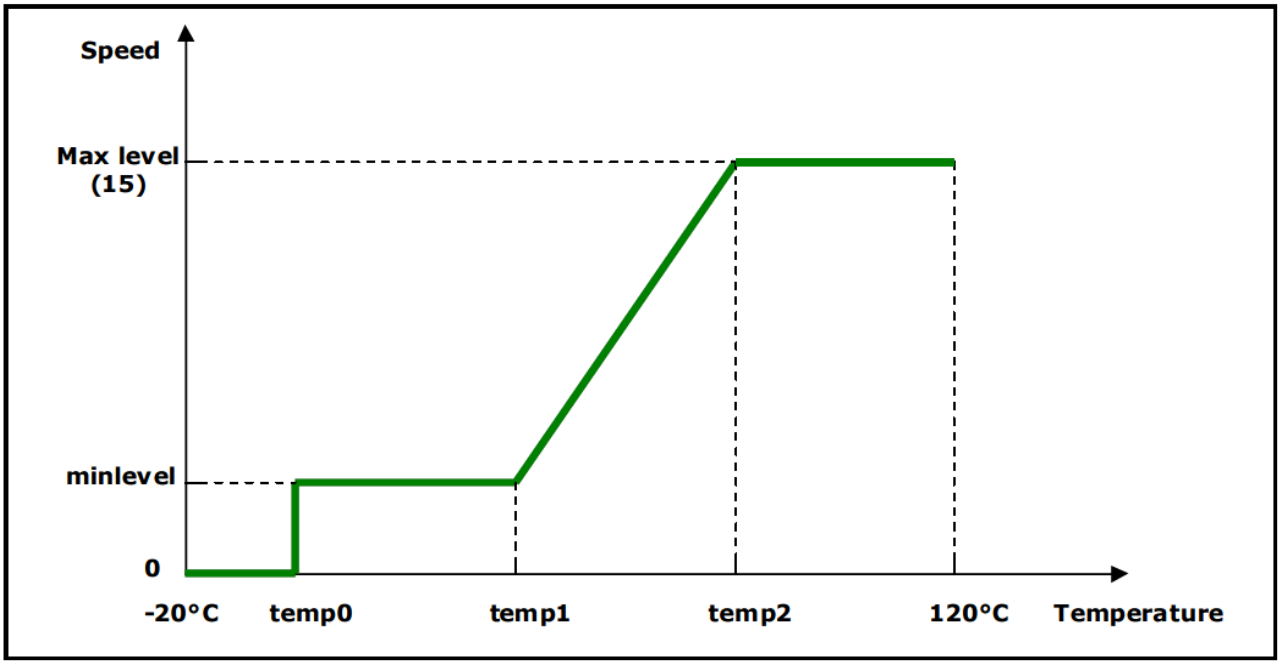

Figure 2: Fan Control Algorithm

The user can define the temperature-speed characteristic using 4 parameters: min level (the minimum level at which the fans operate) and 3 temperature thresholds (temp0, temp1, temp2).

The speed of the fans is split in 15 equal levels. At level 0 the fans are stopped and at level 15 the fans are running full speed.

1 The ChMC can output 3 PWM signals with frequencies between 1KHz and 125KHz. The frequency of PWM1 is equal to that of PWM2, but the duty factors of the 2 PWM signals are independent.

The 3 temperature thresholds split the operating range of the fan in 4 control regions:

Temperature T	Fans Behavior:
T< temp0	Stopped
temp0 < **T** < temp1	Running at min level
temp1<**T**<temp2	Running at a speed level proportional to **T**
temp2<**T**	Running at full speed

Table 3: Fan Control Algorithm

The Fan control algorithm can use any of the installed temperature sensors. The user can choose the active temperature sensor for each fan control group. The temperature that drives the algorithm is the maximum value of all the temperature sensors active for the respective fan control group. If no temperature sensors are active for a particular fan group, the fan control algorithm sets the fans to run at the maximum level.

The 3 fan control groups are independent, each have their own minlevel, temp0, temp1, temp2 parameters and each control a different PWM signal.

The automatic fan control algorithm can also be disabled. In this case the fan speed is set at a fixed manual level. Each fan control group has its on own manual fan level which can be changed using the CLI via RS232 or Telnet.

Commands useful for fan control:***fancontrol, pwm, temp***

7.4 Digital Inputs

- 16 digital inputs
- Logic level: 5V TTL.
- Each individual input can be declared as active low or active high
- Any active input generates an internal event, and can control any of the 16 digital outputs
- Any input can be setup as shutdown switch or push-button or NVMRO input
- Commands for inputs: ***input, local_sensor***

7.5 Digital Outputs

- 16 digital outputs
- All outputs support startup sequencing
- Each individual output can be declared as active low or active high
- Manual setting and clearing of outputs
- Reset signal behavior supported
- Each output can be driven by more than one source. Potential output drivers are:
  - Internal Events : threshold violations for threshold sensors, active levels for discrete sensors
  - External Events: active inputs
- Each output can perform an **AND** or an **OR** function on the driving signals:
  - **AND** outputs are active if all the drivers are active
  - **OR** outputs are active when at least one of the drivers is active
- Commands for outputs: ***output, local_sensor***

7.6 Accessing Local Sensor Information

For each monitored chassis parameter the ChMC has a local software sensor attached. At each startup the ChMC scans the config file and instantiates only the sensors that have SDRs

defined.

For accessing local sensor information there are two options:

- displaying information for multiple sensors
- accessing each sensor individually using its number

For accessing all the local sensors the command ***local_sensor*** is used. Also the user can access information regarding sensors of a particular type using one of the commands: ***voltage, current, temp, fan, input, output.***

All the commands that display sensor information use the same header:

```
----------------------------Sensor List-----------------------------------
--no--Name-------------Type--Value--Unit---State--------------------------
```

The header displays:

- **no** → sensor number; a unique number that identifies a particular sensor.
- **Name** → sensor's name
- **Type** → sensor type : threshold or discrete
- **Value** → current value
- **Unit** → measuring unit
- **State** → current state for threshold sensors. Depending on the sensor's current value its state could be:
  - Lower Non-Recoverable
  - Lower Critical
  - Lower Non-Critical
  - Upper Non-Critical
  - Upper Critical
  - Upped Non-Recoverable

The ***local_sensor*** command can also be used to display a a more detailed description for a single sensor:

- name
- type
- value
- threshold values
- sensor state
- outputs assigned to threshold events
- fan control groups the sensor is assigned to (only for temperature sensors)
- active level for discrete sensors
- output function for output sensors
- active drivers for output sensors

For the complete syntax of the commands refer to the CLI chapter.

7.7 Local Sensor Numbers

This section refers to the local sensors implemented on the Chassis Manager. The correspondence between the sensor numbers and monitored parameters is defined in a table presented bellow.

! Depending on the monitored system the ChMC can be set up differently. Not all the sensors described bellow are always available.

! The ChMC monitors only the parameters for which a SDR has been uploaded. The monitored parameters set can be changed by uploading a new SDR set. The SDRs are encapsulated in a configuration file. Besides the SDRs, the config file also hosts several other ChMC parameters.

Sensor Number	Monitored System Parameter
1	reserved
2	V0 Voltage
3	V1 Voltage
4	V2 Voltage
5	V3 Voltage
6	V4 Voltage
7	V5 Voltage
8	V6 Voltage
9	V7 Voltage
26	Temperature sensor 1
27	Temperature sensor 2
28	Temperature sensor 3
29	Temperature sensor 4
30	Temperature sensor 5
31	Temperature sensor 6
37	Tachometer signal for Fan1
38	Tachometer signal for Fan2
39	Tachometer signal for Fan3
40	Tachometer signal for Fan4
41	Tachometer signal for Fan5
42	Tachometer signal for Fan6
43	Tachometer signal for Fan7
44	Tachometer signal for Fan8
45	Tachometer signal for Fan9
46	Tachometer signal for Fan10
47	Tachometer signal for Fan11
48	Tachometer signal for Fan12
64	Digital Input 1
65	Digital Input 2
66	Digital Input 3
67	Digital Input 4
68	Digital Input 5
69	Digital Input 6
70	Digital Input 7
71	Digital Input 8
72	Digital Input 9
73	Digital Input 10
74	Digital Input 11
75	Digital Input 12
76	Digital Input 13
77	Digital Input 14
78	Digital Input 15
79	Digital Input 16
80	Digital Output 1
81	Digital Output 2
82	Digital Output 3

83	Digital Output 4
84	Digital Output 5
85	Digital Output 6
86	Digital Output 7
87	Digital Output 8
88	Digital Output 9
89	Digital Output 10
90	Digital Output 11
91	Digital Output 12
92	Digital Output 13
93	Digital Output 14
94	Digital Output 15
95	Digital Output 16
97	ChMC Power On
117	I2C Temperature 1 (8 bit address 0x90)
118	I2C Temperature 2 (8 bit address 0x92)
119	I2C Temperature 3 (8 bit address 0x94)
120	I2C Temperature 4 (8 bit address 0x96)
121	I2C Temperature 5 (8 bit address 0x98)
122	I2C Temperature 6 (8 bit address 0x9A)
123	I2C Temperature 7 (8 bit address 0x9C)
124	I2C Temperature 8 (8 bit address 0x9E)

Table 4: Local Sensor Numbers

8 Ethernet interface

The integrated 10/100Mbps Ethernet interface allows the ChMC to be linked to any existing network. The interface supports DHCP, SNMP, TFTP, HTTP and TELNET protocols via TCP/IP and UDP.

All monitored system parameters can be displayed via a standard browser (HTTP protocol).

The Command Line Interface (CLI) is accessible via TELNET, allowing remote control of the ChMC.

The use of standard protocols avoids the need for special software or drivers and so achieves platform-independence. The TCP/IP protocol supports up to 10 simultaneous connections.

! The factory default setting for the ChMC is DHCP enabled so it negotiates automatically all the necessary addresses. If a fixed IP address is desired, DHCP must be disabled and the address has to be set manually. For all these operations the ***lanconfig*** command needs to be used.

Terminal settings:

- Local echo: *off*
- Local line editing: *off*
- Backspace key: *Control-H*

8.1 WEB

The Chassis Manager includes a built in WEB server that provides a simple way to access the management information.

8.1.1 Overview

The WEB page can be designed as a graphical representation of the monitored System Platform, thus providing a very intuitive way of obtaining system / board information.(Figure 2 exemplifies the web page for a 5 slot ATCA system) . The available information includes:

- Board / Carrier /chassis Field Replaceable Unit (FRU) information file (Manufacturer's Name, Part number, Serial Number, Board Connectivity Records)
- Sensors information: value, name,measuring unit, status,threshold and hysteresis values
- System Event Log (SEL) : sensor events
- ChMC attributes: MAC address, Serial number, Firmware version

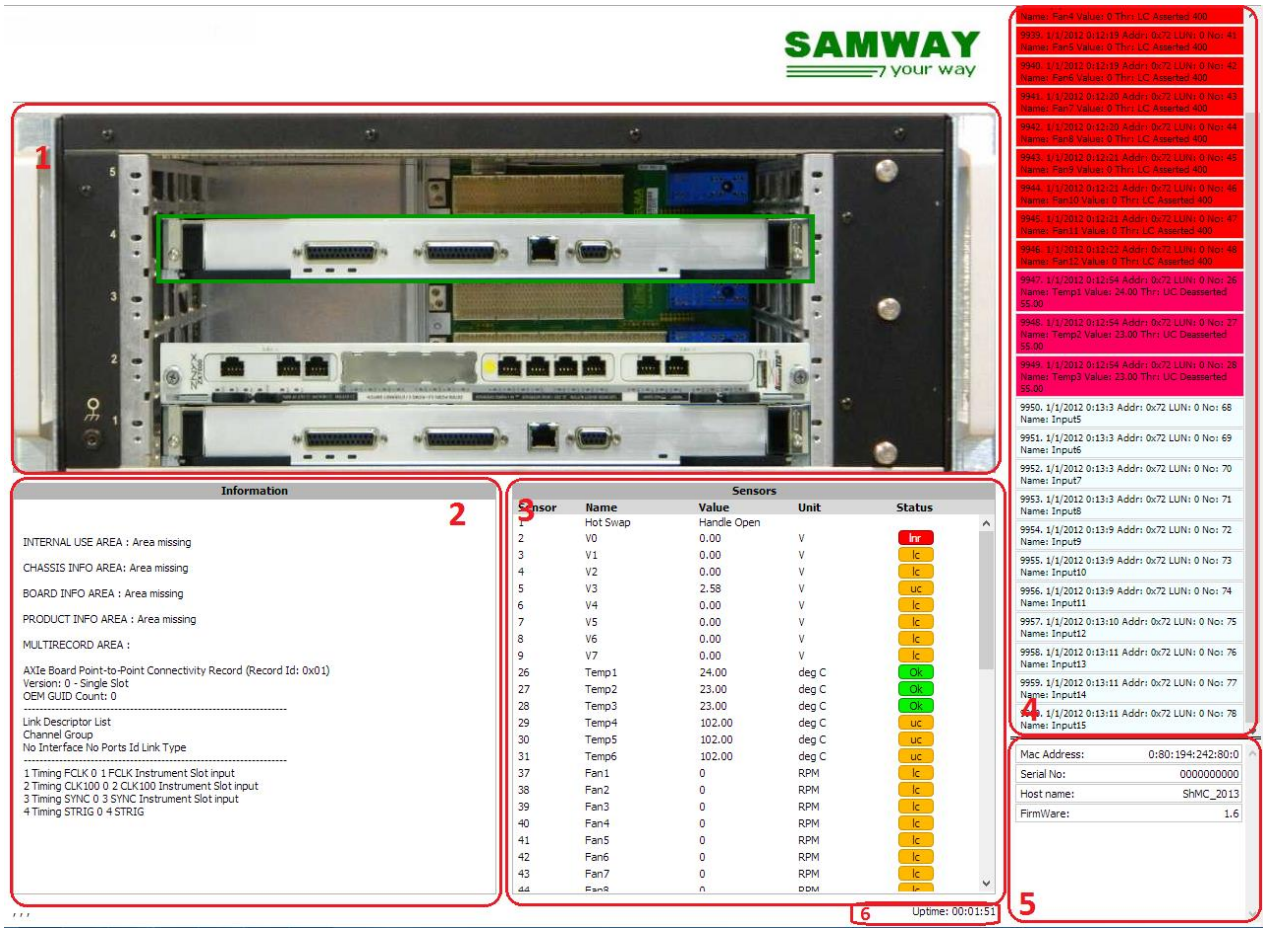

Figure 3: Web Interface overview

The default Layout for the Web Page is composed of several areas:

1. *Dynamic System Platform Representation*:: The images are populated or removed depending on the hot-swap state of the boards. Both system platform and boards are treated as objects and can be selected using a mouse click. The information in panes *2. FRU info* and *3. Sensor info* is changed depending on the selected object.
2. *FRU info*: displays the FRU information for the selected object(board or system platform)
3. *Sensor info:* displays the values, names and status for all the sensors of the selected object. All the sensors are considered objects and can be selected using a mouse click.
4. *System Event Log (SEL)*: displays all the sensor events received by the Chassis manager starting from the moment the Web page has been loaded.
5. *Info Area*: displays more details for the selected sensor (threshold and hysteresis values) or, if no sensor is selected, displays ChMC attributes: MAC address, Serial number, Firmware version.
6. *Uptime :* displays the amount of time the ChMC has been operational. It is reset at each ChMC restart.

8.1.2 Architecture

The Web Server uses a Representational State Transfer (REST) based architecture and Extensible Markup Language (XML) files.

REST is an architectural style that abstracts the architectural elements within a distributed hypermedia system. REST ignores the details of component implementation and protocol syntax in order to focus on the roles of components, the constraints upon their interaction with other components, and their interpretation of significant data elements. REST has emerged as a predominant web API design model.

XML is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. The design goals of XML emphasize simplicity,generality, and usability over the Internet.

The ChMC WEB page is composed of:

- objects : board and system images; sensors
- information display areas

When an object is selected by a mouse click, the web browser requests predefined XML files using predefined commands. The information contained by the XML files returned by the ChMC is parsed in the information display windows.

The XML files could also be requested by an application software in case a custom managing solution is required. The format of the XML files is described in the following chapter.

8.1.3 REST Commands

"/settings"

For this request the ChMC returns an XML file containing the following tags:

- <mac_addr>
- <serial_no>
- <host_name>
- <firmware> : firmware version
- <uptime> : the amount of time the ChMC has been operational
  - <H>: hours
  - <M>: minutes
  - <S>: seconds

Request: GET **/settings**

IP Address/**settings** (ex: 192.168.16.1/settings)

Response:

```
<?xml version="1.0"?>
<settings>
<mac_addr>0:80:194:242:80:0</mac_addr>
<serial_no>0000000000</serial_no>
<host_name>ChMC_2013</host_name>
<firmware>1.6</firmware>
<uptime>
        <H>0</H>
```

```
            <M>31</M>
            <S>21</S>
    </uptime>
    </settings>
```

"/frustatus"

For this request the ChMC returns an XML file that contains the following tags:

- <boot_cnt>: boot index. The index is incremented at each ChMC start-up.
- <sel_cnt>: System Event Log (SEL) index. The index is incremented when an event is added to the SEL. This field is used to detect if new events have occurred.
- <fru_list>: a list with all the modules(FRUs) present in the system(including the ChMC)
- <fru_addr>: the IPMB address of the module
- <fru_id>: active FRU ID. The Module Management Controller (MMC) is always represented by FRU ID 0 and is always present for active modules. Additional FRU IDs can also be active.

Request: GET /**frustatus**

IP Address/**frustatus** (ex: 192.168.16.1/frustatus)

Response:

```
<?xml version="1.0"?>
<fru_status>
<boot_cnt>315</boot_cnt>
<sel_cnt>9791</sel_cnt>
<fru_list>
        <fru_addr addr="0x00">
                <fru_id>0</fru_id>
        <fru_addr addr="0x82">
                <fru_id>0</fru_id>
        </fru_addr>
        <fru_addr addr="0x84">
                <fru_id>0</fru_id>
        </fru_addr>
        <fru_addr addr="0x86">
                <fru_id>0</fru_id>
        </fru_addr>
        <fru_addr addr="0x10">
                <fru_id>0</fru_id>
        </fru_addr>
</fru_list>
</fru_status>
```

"/sel/start_index/end_index"

This command retrieves multiple SEL event records. The command use two indexes (start,end) to define the desired number of records. The ChMC returns an XML file that contains all the event records that have an index between the start index and the end index. The XML file uses the following tags:

- <rec id> : the index of the current record
- <tmp> : the time when the event was triggered
- <addr>: address of the card that launched the event
- <lun>: the LUN on which the sensor resides
- <no>: sensor number
- <name>: sensor name
- <type>: code representing the sensor type
- <sta>: sensor state, available only for discrete sensors

- <ev_type>: threshold that triggered the event for threshold sensors: UNR (upper non-recoverable), UC(upper critical), UNC(upper non-critical), LNC (lower non-critical), LC(lower critical), LNR(lower non-recoverable)
- <ev_dir> Asserted, DeAsserted
- <val>: sensor value
- <thr>: threshold value

Request: GET /**sel**/*start_index/end_index*

IP Address/**sel**/*start_index/end_index* (ex: 192.168.16.1/sel/1/2)

Response:

```
<?xml version=”1.0”?>
<sel>
<rec id="1">
        <tmp>1325376000</tmp>
        <addr>0x72</addr>
        <lun>0</lun>
        <no>97</no>
        <name>ChMC Power ON</name>
        <type>192</type>
        <sta>0</sta>
</rec>
<rec id="2">
        <tmp>1325376000</tmp>
        <addr>0x72</addr>
        <lun>0</lun>
        <no>2</no>
        <name>V0</name>
        <type>2</type>
        <ev_type>LNC</ev_type>
        <ev_dir>Asserted</ev_dir>
        <val>0.00</val>
        <thr>2.86</thr>
</rec>
</sel>
```

“/fruinfo/*FRU_Address*/*FRU_Id*”

This command retrieves a **text file** containing the FRU information for the desired FRU Id on the specified card. The command uses two parameters:

- *FRU_Address*: the IPMB address of the card(hexadecimal value)
- *FRU_Id*: used to distinguish between multiple FRUs located on the same card. The FRU Id for the Module Management Controller (MMC),(the card itself) is 00.

Request: GET /**fruinfo**/*FRU_Address/FRU_Id*

IP Address/**fruinfo**/*FRU_Address/FRU_Id*(ex: 192.168.16.1/fruinfo/0x82/0)

Response: *text file*

"/sensor/FRU_Address/FRU_Id"

This command retrieves all the sensor associated to a FRU Id of a card. The command uses two parameters:

- *FRU_Address*: the IPMB address of the card(hexadecimal value)
- *FRU_Id*: used to distinguish between multiple FRUs located on the same card. The FRU Id for the Module Management Controller (MMC),(the card itself) is 00.

If this parameter is missing the ChMC will return the sensors associated to FRU Id 0
The ChMC responds to this request using an XML file that contains the following tags:

<name>:sensor name
<value>:sensor value
<unit>:sensors unit of measurement
<state>:sensor state (lnr, lc, lnc, unc, uc, unr)

Request: GET /**sensor**/*FRU_Address*/*FRU_Id*
IP Address/***sensor***/*FRU_Address*/*FRU_Id*(ex: 192.168.16.1/sensor/0x82)
Response:

```
<?xml version="1.0"?>
<sensor_list>
        <sensor no="1">
                <name>Hot Swap</name>
                <value>Handle Open </value>
        </sensor>
        <sensor no="2">
                <name>V0</name>
                <value>0.00</value>
                <unit>V</unit>
                <state>lnr</state>
        </sensor>
</sensor_list>
```

"/sdr/FRU_Address/sensor_No"

This command retrieves additional information for a specific sensor on the desired card. The command uses two parameters:

- *FRU_Address*: the IPMB address of the card(hexadecimal value)
- *sensor_No*: sensor number for the desired sensor

The ChMC responds to this request using an XML file that contains the following tags:

- <name>
- <entity_id> : entity id for the sensor's owner
- <entity_instance> :entity instance for the sensor's owner
- <unr>: upper non-recoverable threshold value, if the unr threshold is enabled
- <uc>: upper critical threshold value, if the uc threshold is enabled
- <unc>: upper non-critical threshold value, if the unc threshold is enabled
- <lnc>: lower non-critical threshold value, if the lnc threshold is enabled
- <lc>: lower critical threshold value, if the lc threshold is enabled
- *<lnr>: lower non-recoverable threshold value, if the lnr threshold is enabled*
- <hyst_pos>: positive going hysteresis value
- <hyst_neg>: negative going hysteresis value
- <nominal_reading>

- <normal_maximum>
- <normal_minimum>
- <maximum_reading>
- <minimum_reading>

Request: GET **/sdr**/*FRU_Address/sensor_No*

IP Address */**sdr**/FRU_Address/sensor_No(ex: 192.168.16.1/sensor/0x10/5)*

Response:

```
<?xml version=”1.0”?>
<sensor no="5">
        <name>V3</name>
        <entity_id>0x01</entity_id>
        <entity_instance>0x61</entity_instance>
        <uc>-11.34</uc>
        <lc>-12.64</lc>
        <hyst_pos>0.07</hyst_pos>
        <hyst_neg>0.07</hyst_neg>
        <nominal_reading>-14.00</nominal_reading>
        <normal_maximum>-14.00</normal_maximum>
        <normal_minimum>-14.00</normal_minimum>
        <maximum_reading>2.58</maximum_reading>
        <minimum_reading> -14.00</minimum_reading>
</sensor>
```

“/picture/FRU_Address”

This command retrieves the pictures for the boards and system platform.

- *FRU_Address*: the IPMB address of the card(hexadecimal value). For the picture of the System platform the request uses FRU_Address = 0.

Request: GET /**picture**/*FRU_Address*

IP Address */**picture**/FRU_Address*(192.168.16.1/picture/0x80; 192.168.16.1/picture/0)

Response: picture file

8.1.4 How it all works

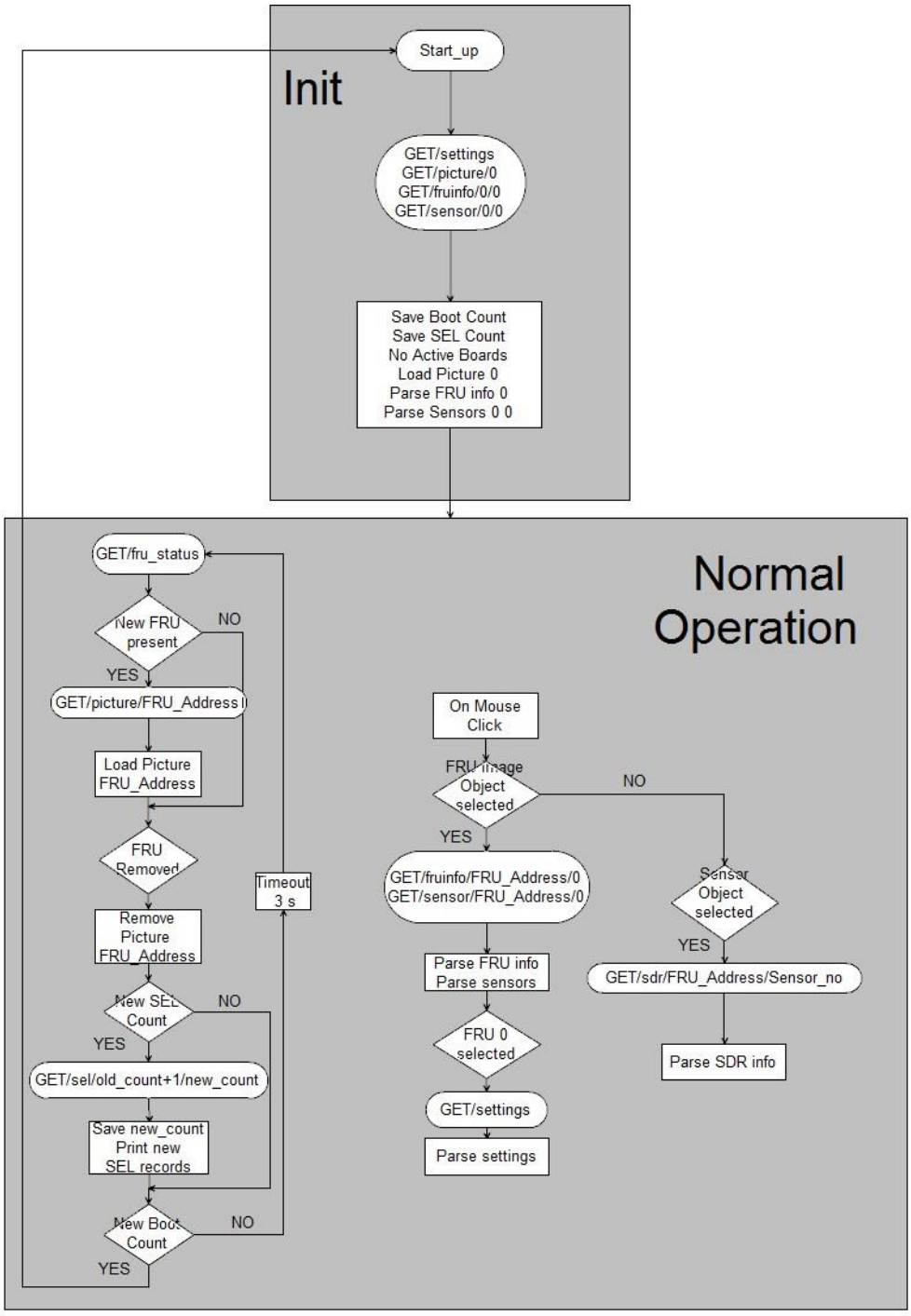

Figure 4: Default Web Page Functional Diagram

Figure 3 describes the functional diagram of the default Web Page. The web page has 2 operational states: Initialization and Normal Operation.

At start-up, and each time it is re-initialized, the web page is empty. The Web page requests and saves the ChMC settings (**GET/settings**): MAC Address, Serial Number, Firmware Version,Uptime. The page also saves some parameters from the response of the frustatus request: Boot count, SEL count.

Next, the web page request information for the system platform(FRU 0): picture, FRU info,

sensors. (**GET/picture/0, GET/fruinfo/0/0, GET/sensor/0/0**). Once the information is received, the web page displays it.

After the system platform is initialized the init phase ends. By default all boards are considered absent during this phase.

The normal operation phase of the web page is split in two processes:

- A periodical process that check to see if new boards have been inserted, or if boards have been removed
- A mouse click listener that changes the active object

The periodic process issues a **GET/frustatus** request at every 3 seconds. By comparing the FRU list in the response and the FRUs loaded, the page determines if boards have been inserted or removed. For all the new boards the web page sends a **GET/picture/FRU_Address** request and loads the received picture. For all the boards that have been removed, the web page also removes the picture.

Using the response for the **frustatus** request the web page also checks if new SEL events have occurred. If the SEL count from the response is different from the one saved by the page, a **GET/sel/old_count+1/new_count** request is sent. Using the response the web page displays all the new SEL events.

The **frustatus** request is also used to detect restarts of the ChMC. If the Boot count in the response is different than the saved one, the WEB page is restarted: all pictures are removed and all the internal variables are reinitialized.

The WEB page uses a mouse click listener to determine if the current selected object has been changed. The WEB page uses two types of objects:

- FRU images
- FRU sensors

At each mouse click the WEB page determines if the active object has been changed.

If the active FRU image object has been changed, the WEB page request the FRU info and sensors for the new active object: **GET/fruinfo/FRU_Address/0**, **GET/sensor/FRU_Address/0**. Using the responses of the requests, the web page updates the information visible in the *FRU Info* and *Sensor Info* panels. By default, at start-up, the active image object is the system platform (FRU 0).

If the active sensor object has been changed, the WEB page request the detailed information or the new active sensor: **GET/sdr/FRU_Address/Sensor_No.** Using the response for the sdr request, the WEB page updates the information in the *Info Area* panel. By default, at start-up no sensor object is selected and the Info are panel displays ChMC parameters: MAC address, Serial No., Firmware version.

After the first sensor object becomes active, the *Info Area* panel will display sensor information. To return to displaying ChMC parameters, a click on the system platform image is necessary.

8.2 Telnet

The Telnet interface supports two operating modes: - Command Line Interface (CLI) and Serial Over Lan (SOL).

The CLI is accessible if the operator is logged on as "user" or "admin" profile.

If the operator is logging into the ChMC as "serial" profile the SOL mode is activated. Whenever the ChMC enters in SOL mode the behavior of RS232 port will change. All the data received on the serial port will be transmitted in telnet packets over Ethernet to all telnet connections opened on "serial" profile. The data received from the telnet connections opened in "serial" mode is transferred to the serial port. This feature allows remote control of a CPU card inside the system. For this the CPU terminal screen should be redirected to the serial port and the serial port of the CPU needs to be connected with the serial port of the ChMC.

Default access settings:

login: serial
password: SERIAL

In SOL mode the SysRq key could be send by sending a Telnet break.

Note: If SOL mode is used it is recommended to make the SOL mode as default using "sol" command.

8.3 SNMP

The ChMC supports SMNP v1, v2c and SNMP v3 with authentication and encryption. The software supports MD5 for authentication, while for privacy supports DES and AES.
For application where data security is an issue, SNMP v3 with authentication and encryption could be used. For such applications the telnet and the web interfaces could be disabled to prevent not secured connections to the ChMC. To disable the Telnet or Web interface use the "eth" command or write the SNMP variables telnet and web to 0 in the control node.
When the SNMP is configured to use authentication in version 3 (see "snmp" command) and the authentication password is not empty the SNMP v1 and V2C are disabled. Only SNMP v3 authenticated messages are accepted.

The default SNMP v3 passwords are:
"SnmpUserAuth" – for SNMP user authentication
"SnmpUserPrivacy" – for SNMP user privacy
"SnmpAdminAuth" – for SNMP admin authentication
"SnmpAdminPrivacy" – for SNMP admin privacy

The authentication and encryption is disabled by default on both user and admin profiles so that the ChMC accepts SNMP v1, SNMP v2c and unauthenticated SNMP v3 messages. Te enable authentication use "snmp" command.

The SNMP data is structured in some categories:

.system

- contains information about the ChMC:name, part number, serial number, MAC address and software version

.temp

- contains the number of temperature sensors and for each sensor, the name, value, state, thresholds and hysteresis.

.voltage

- contains the number of voltage sensors and for each sensor, the name, value, state, thresholds and hysteresis

.current

- contains the number of current sensors and for each sensor, the name, value, state, thresholds and hysteresis

.fan

- contains the number of fan sensors, the speed level of each group of fans and for each sensor, the name, value, state, thresholds and hysteresis

.input

- contains the number of inputs and for each input, the name and state

.output

- contains the number of outputs and for each output, the name and state

.control

- this category allows different configurations to be done to the ChMC. The RS232 speed and the user and admin passwords could be read or changed, the Telnet and Web interfaces could be enabled /disabled. The configuration settings could be saved and the ChMC could be restarted.

8.4 RMCP

The ChMC supports Remote Management Control Protocol (RMCP). The RMCP connections requires authentication. The supported authentication protocol is MD5. Two user names are accepted for RMCP connections: "user" – with "User" privilege level and "admin" with "Administrator" privilege level. The user names and privilege levels are fixed, they cannot be changed through IPMI commands. The password used are the same passwords configured for Telnet, CLI, SNMP v1 and v2c access. The default passwords are: "USER" – for "user" profile and "ADMIN" - for admin profile.

9 RS232 serial interface

The ChMC provides an RS232 serial interface through which the commands of the Command Line Interface (CLI) can be sent.

On Windows systems, we recommend the use of "TeraTerm" or "Hyperterminal" as the terminal programs.

Default Terminal settings:

- 19200 bits per second
- data bits: 8
- parity: none
- stop bit: 1

The baud rate of the RS232 Serial Interface can be changed. The available baud rates are 9600, 19200, 38400,115200. To change the speed of the Serial Interface use the ***scispeed*** command

In addition, the ***xmodem*** command can be used via the serial interface for file transfer.

! Use a 1:1 serial cable for direct connection to the serial port of a PC.
When using ***xmodem*** in "Hyperterminal" the transfer of the desired file can take up to 10 seconds to start.

10 Command Line Interface (CLI)

The Command Line Interface (short-form: CLI) is available via both Telnet (**3 Ethernet interface**) and the RS232 serial interface (**4 RS232 serial interface**).

The user can read or newly configure and save system parameters via the CLI.

Access is divided into 2 profiles and is password-protected.

"user" profile:
System parameters can only be read in this profile – the exception to this write-protect is the ***lanconfig*** command for setting the IP, subnet and gateway addresses.

"admin" profile:
Full access to all system parameters is granted. All available CLI commands can be executed. To avoid possible damage or malfunctions, the access data for this profile must only be available to trained personnel with appropriate knowledge and competence relating to the system in which the ChMC is used!
The profiles can be changed using the ***logout*** command.

The measured values are available at any time via the RS232 serial interface and via Telnet. In addition, limits and system parameters can be changed at any time with the unit in service. As soon as you have established a connection, you will be prompted to log in.

Default access settings:

login: user
password: USER

login: admin
password: ADMIN

! The passwords can be changed using the ***passw*** command. The passwords can also be disabled by changing them to an empty string.

General syntax conventions

Command [parameter1 | parameter2 | parameter3 [*value*]]

[ ] = optional
A command on its own with no entry of other parameters returns all available current values associated with the command.
The parameters separated by "|" rule out each other.
If the command line contains a *value*, this is assigned to the corresponding parameter and saved temporarily in the RAM. The change is active immediately. If the new value is desired to be valid after the reboot, the environment variables must be saved with the "***saveenv"*** command. Changes not confirmed with "***saveenv"*** *command* are lost after a reboot.

10.1 at command

Syntax: at

Functions:
Displays a list with all the IPMB addresses known to the ChMC. This list is populated with the addresses defined by the Address Table record in the chassis FRU Information file and also newly discovered addresses. Those additional addresses belong to IPMCs that have sent messages to the Chassis Manager and due to unknown reasons have not been instantiated in the Address Table record of the Chassis.

10.2 bit command

Syntax: bit [(c_bit log|clear) | (i_bit start)]

Functions:

Displays the status of the Built In Tests performed: P-BIT (Power On BIT), C-BIT (Continuous BIT), I-BIT (Interactive BIT).

The command can be used to display a log of events that triggered a C-BIT error : ***bit c_bit log***. Or to clear the c_bit log: ***bit c_bit clear***

The I-BIT can be started using : ***bit i-start start*** .Once started I-BIT can be stopped by pressing the ESC key.

10.3 clear command

Syntax: clear fru_history

This command is active only for the ATCA/AXIe version

Functions: The command is used to clear the history for the discovered FRUs.

At each startup the discovered boards are compared to a history record from the previous ChMC startup. If any of the FRUs was previously operational, but fails to be discovered correctly its state will be set to M7 – communication lost. There could be two possible reasons for this situation:

- the board has a problem and fails to communicate
- the board has been removed by an operator while the chassis was powered off.

In the first case the M7 state is correct because the board really has a problem, while in the second case the M7 is incorrect as the board is no longer present.

This command was implemented for situations similar to the second example when the setup was changed while the chassis was off and the startup FRU history is no longer correct.

10.4 cooling command

Syntax: cooling [ipmb_addr fru_id [*fanlevel* [**localon**]]]

Functions:

The command displays the minimum, maximum, normal, override and local fan level (see PICMG 3.0 Get Fan Speed Properties, Get/Set Fan Level commands for the meaning of the returned parameters) for discovered cooling units.

All FRUs that have the Entity_ID set to 1Eh (Cooling Unit) in the Management Controller Device Locator Record (SDR type 12h) or FRU Device Locator Record (SDR type 11h) will be treated as cooling units.

If the command is used without any parameter it will display the fan level information for all available cooling units.

For displaying fan level information only for a particular cooling unit the ipmb_addr and fru_id parameters can be used.

The command may also be used to set the fanlevel for a cooling unit by specifying an override level. By default, when the level is changed, the local control is disabled. If local control is desired the "localon" keyword can be used to keep it enabled even after the fanlevel change .

10.5 current command

Syntax: current

Functions:
Displays information about all current sensors implemented locally on the Chassis Manager. For each installed sensor the command displays:

- sensor number
- sensor name
- sensor type
- value
- measuring unit
- state

10.6 date command

Syntax: date[dd.mm.yyyy]

Functions: displays or sets the current date

Example 1: Check out of date

```
%>date

 Date [dd.mm.yyyy] = 04.09.2012
```

Example 2: Setup of date

```
%>date 13.10.2013
 Done!
```

10.7 define command

Syntax: define (hot_swap [en | di]) |
(startup_delay | shutdown_timeout [*seconds_value***]) |**
(localipmbaddr [hexvalue])

Functions: The "define" command is used to configure various parameters.

- **hot_swap:** Configures if the Chassis Manager is hot swappable or not.
- **startup_delay:** Defines a timeout that should expire, before the Chassis Manager starts to monitor the local sensors attached directly to it.
- **shutdown_timeout**: Defines a maximum timeout between the moment a shutdown is initiated for a FRU and the moment it is turned off.
- **Localipmbaddr :** Defines the IPMB address of the local System monitor. By default the ipmb address of the System monitor is 0xFC.

10.8 display_status command

Syntax: display_status

Functions:
When the LCD or TFT displays are used, this command shows the connection status: the display is operating correctly or failure in communication.

10.9 etime command

Syntax: etime [reset]

Functions:

The Elapsed time meter is an optional component and is not assembled on the standard version of the ChMC. In the standard release of firmware this command is not supported.

Some custom ChMC variants have the part assembled and on those the command is used to query the cumulative operating hours of the unit.

10.10 eth command

Syntax: eth [telnet | web [on | off]]

Functions:

Enables/disables the telnet and web interfaces. Telent and WEB interfaces provides no security therefore this interfaces may be disabled is SNMP v3 with authentication and privacy is used.

10.11 exit command

Syntax: exit

Functions:

The command terminates a telnet connection.

10.12 extract command

Syntax: extract configfile

Functions:

The command is used to extract the configuration file from the ChMC. Due to safety reasons all the passwords in the extracted file will be changed to the default ones : user, admin and serial CLI access profile access passwords, SNMP user and admin profiles authentication and privacy passwords and the community name for the SNMP traps.

10.13 fan command

Syntax: fan
Functions:
Displays information about the fan speed sensors implemented locally on the Chassis Manager. This command does not show the fans implemented under other, external IPMC. For each installed sensor the command displays:

- sensor number
- sensor name
- sensor type
- current value
- measuring unit
- current state

Example 1. fan sensor values read out

```
%>fan
----------------------------Sensor List----------------------------
--no--Name------------Type--Value--Unit---State-----------------
* 37  Fan1            Thr   8800   RPM    Ok
* 38  Fan2            Thr   2700   RPM    Ok
* 39  Fan3            Thr   2700   RPM    Ok
* 40  Fan4            Thr   2600   RPM    Ok
* 41  Fan5            Thr   4800   RPM    Ok
* 42  Fan6            Thr   8800   RPM    Ok
```

10.14 fancontrol command

Syntax: fancontrol [*group_no* **[auto] | [manual [level** *l_value***]] |**
[maximum level *l_value***] | [minimum level** *l_value***] |**
[temp0|temp1|temp2 *t_value***]] |**
[sensor]

- group_no : 1 - 3
- manual, maximum and minimum levels value : 0 - 15
- t_value: -20..100 Celsius degrees

Functions:

The command is used to display or modify the parameters of the fan speed algorithm for ChMC controlled Fans.

This command influences only the algorithm for the fans controlled directly by the ChMC using its 3 PWM signals and has no influence on external cooling units instantiated by other IPMCs.

For more details about the algorithm that is used to control the fan speeds refer to 1.4.1 Fan control.

Each fancontrol group represents a PWM signal. The fan control algorithm uses 15 distinct fan levels to represent the 0-100% duty cycle of the PWM signal. Fan level 0 is equivalent to 0% and fanlevel 15 is equivalent to 100% duty cycle for the PWM signal.

10.14.1 Viewing the fancontrol status

Syntax: fancontrol

If the command is used without any parameter it displays the current state of all the fan control groups and their respective parameters.

Example 1. Viewing the fancontrol status

```
%>fancontrol
-------------------------------------------------------------------------
Fan   Control   Current    Manual     Minimum   Temp0  Temp1  Temp2  Maximum
Group  Method  Fan level  Fan Level  Fan level                         Temp
-------------------------------------------------------------------------
  1    Manual     7          7          3        0      30     60      24
  2    Auto       3          3          3        0      30     60      24
  3    Auto       3          3          3        0      30     60      24
```

The fans can be controlled automatically using a control algorithm or manually by specifying a fan level. By default all fan groups are controlled automatically.

10.14.2 Setting a fangroup for auto control

Syntax: fancontrol *group_no* **auto**

For controlling the fans speed automatically a set of the installed temperature sensors is used. Every fan group has his own distinct set. The fan speed is controlled in accordance with the algorithm described in chapter 1.4.1 Fan control and the maximum temperature of the sensors in its set.

Example 2. Setting fancontrol group 2 to auto

```
%>fancontrol 2 auto
 Done! Fan Group 2 is controlled automatically!
```

10.14.3 Setting a fangroup for manual control

Syntax: fancontrol *group_no* **manual**

When a fan control group is controlled manually, its fan level is specified by the manual level parameter.

Example 3. Setting fancontrol group 1 to manual control

```
%>fancontrol 1 manual
```

```
 Done! Fan Group 1 is controlled manually using the manual Fan level value!
```

10.14.4 Changing the manual fancontrol level

Syntax: **fancontrol** *group_no* **manual level** *level_value*

The value for the manual fan level can be choosen between the 15 allowed levels. For level 0 the fans are stopped (PWM duty cycle 0%) and for level 15 the fan are running at full speed (PWM duty cycle 100%)

Example 4. Changing the manual fan level for fan group 3

```
%>fancontrol 3 manual level 13
 Manual Fanlevel = 13
```

Using the fancontrol command the user can change the value for all the fan control algorithm's parameters : minimum level, temp0, temp1, temp2. Each fan group has its own parameters.

10.14.5 Changing the minimum level parameter of the fancontrol algorithm

Syntax: **fancontrol** *group_no* **minimum level** *level_value*

The minimum level parameter determines the speed of the fans when the maximum temperature of the sensors is bellow the temp0 parameter.

Example 5. Changing the minimum fan level for fan group 1

```
%>fancontrol 1 minimum level 1
 Minimum Fanlevel = 1
```

10.14.6 Changing the temp0,temp1,temp2 parameters of the fancontrol algorithm

Syntax: **fancontrol** *group_no* **temp0|temp1|temp2** *temp_value*

The **temp**x parameters are used by the fancontrol algorithm. For more details refer to 1.4.1 Fan control.

Example 6. Changing the temp1 parameter for fan group 1

```
%>fancontrol 1 temp1 34

 Temp1=34
```

10.14.7 Viewing the temperature sensors associated with each fan group

When in auto mode the fan level is controlled by the fan control algorithm, and the maximum temperature in the temperature sensors set that is used for that particular fan group.

Syntax: **fancontrol sensor**

Example 7. Viewing the temperature sensors associated with each fan group

```
%>fancontrol sensor
 Sensors assigned to Fan Group 1:
   Sensor 26 Temp1 : 23.00 deg C
   Sensor 30 Temp5 : 24.00 deg C
   Sensor 31 Temp6 : 24.00 deg C
 Sensors assigned to Fan Group 2:
   Sensor 26 Temp1 : 23.00 deg C
   Sensor 30 Temp5 : 24.00 deg C
   Sensor 31 Temp6 : 24.00 deg C
 Sensors assigned to Fan Group 3:
   Sensor 27 Temp2 : 23.00 deg C
   Sensor 28 Temp3 : 23.00 deg C
   Sensor 29 Temp4 : 23.00 deg C
   Sensor 31 Temp6 : 24.00 deg C
```

10.15 fru command

Syntax: fru

Functions:

Displays a list of all discovered FRUs.

10.16 frubuffer command

Syntax: frubuffer

Functions:

Prints out the contents of the memory that is used to store the FRU information Files for all discovered FRUs. This command displays raw (unparsed) data.

10.17 fruinfo command

Syntax: frubuffer impb_address [fru_id]

Functions:

The command displays the FRU information for the desired FRU. If the fru_id parameter is missing, the command displays the information for the IPMC (FRU Id 0) identified by the requested ipmb address.

10.18 help command

Syntax: help [-v]

Functions:

Displays a list of all available commands or a more detailed description if the verbose attribute is used (-v).

10.19 hostname command

Syntax: hostname [<name>]

Functions:

Displays or changes the Hostname of the Chassis Manager

10.20 info command

Syntax: info ipmb_addr

Functions:

Displays information about an IPMC specified by its address. The information is a parsing of data obtained using IPMI requests.

Example 1.

```
%>info 0xfc

 Name: System Monitor
 Device ID: 0x01
 Device provides SDRs
 Device Revision: 3
 Device Available:  Normal Operation
 Firmware Revision: 2.0
 IPMI Version: 1.5
 Additional Device Support:
   IPMB Event Generator
   IPMB Event Receiver
   FRU Inventory Device
   SEL Device
   SDR Repository Device
```

```
Manufacturer ID: 28688d = 07010h
Product ID: 0x0110
```

10.21 input command

Syntax: input

Functions:
Displays information about all the input sensors. For each sensor the command displays:

- sensor number
- sensor name
- sensor type
- current value

Example 1. Input sensors values read out

```
%>input
-----------------------------Sensor List-----------------------------
--no--Name-------------Type----Value--Unit---State------------------

*  64  Input1           Input    1   (Asserted)
*  65  Input2           Input    0   (De-Asserted)
*  66  Input3           Input    0   (De-Asserted)
*  67  Input4           Input    0   (De-Asserted)
*  68  Input5           Input    0   (De-Asserted)
*  69  Input6           Input    1   (Asserted)
*  70  Input7           Input    1   (Asserted)
*  71  Input8           Input    1   (Asserted)
*  72  Input9           Input    1   (Asserted)
*  73  Input10          Input    1   (Asserted)
*  74  Input11          Input    0   (De-Asserted)
*  75  Input12          Input    0   (De-Asserted)
*  76  Input13          Input    0   (De-Asserted)
*  77  Input14          Input    1   (Asserted)
*  78  Input15          Input    1   (Asserted)
*  79  Input16          Input    1   (Asserted)
```

10.22 ipmb_status command

Syntax: ipmb_status

Functions:

Displays information about the state of the IPMB.

10.23 ipmc command

Syntax: ipmc

Functions:

Displays a list with all discovered IPMCs populated with the following information: ipmb address, operational state, device name, firmware version, IPMI version, Manufacturer IANA number, and part number.

10.24 lanconfig - command

Syntax:

Syntax:
lanconfig [dhcp [hostname | clientid] [on | off]] |
[ip | mask| gateway [*address***]] |**
[settings]]

Functions:
Readout or setting of network parameters. When used without parameters the command return the IP, mask and gateway addresses of the LAN interface

- **dhcp –** displays the current dhcp state -enabled/disabled and dhcp configuration options.
- **hostname** – if turned on will enable the DHCP option 12 hostname.
- **clientid** – if turned on will enable the DHCP option 61 client id. The format of the client ID option follows the requirements form PICMG HPM.3 R1.0 specification
- **settings** – interface settings: dhcp or local control for addresses

! The configuration changes done using this command can only be applied at startup so in order to make the changes effective, they must be saved with **saveenv** and the ChMC restarted, either using the reboot command or using a power cycle.

Example 1: Readout of the IP address
```
%>lanconfig ip

IP=193.155.166.51
```

Example 2: Changing the IP address
```
%>lanconfig ip 196.100.100.1

IP=196.100.100.1
```

Example 3: Enable dhcp
```
%>lanconfig dhcp on
 DHCP on
```

10.25 local_sensor command

Syntax: local_sensor [*sensor_no* **[activelevel 0 | 1] |**
[(assert [*ms_timeout***]) | deassert] |**
[debounce *debounce_value***]**
[fancontrol *fan_mask]* **|**
[fan_group 0(unknown),1,2,3**] |**
[hysteresis *h_code h_value***] |**
[ignore_fan en|di]
[input_flags clr | shutdown_push | nvmro] |
[input_type on_off | shutdown] |
[lcd en | di | time *t_value***] |**
[output *out_code hex_value***] |**
[output_flags clr | bite_nc | bite_c |
bite_nr | reserved1 | reserved2 |] |
fan_mux | shutdown_ready] |
[output_type OR | AND] |
[seq (order *o_value***) | (delay** *d_value)]* **|**
[threshold *th_code th_value***|disable] |**
[user_permission allow | deny]]

- sensor_no : the local sensor number that identifies a particular sensor located on the Chassis Manager (see **2.2 . Sensor Numbers**)
- *ms_timeout : mili seconds timeout 20..65530*
- *debounce_value :0..255*
- *fancontrol_mask defined in Table 6: Fancontrol Mask*
- *hysteresis code is defined in Table 4: Hysteresis Code*
- *t_value :0..15 seconds*
- *hex_value defined in Table 5: Hex Outputs*
- *o_value: 0..16 ; 0 -sequencing disabled*
- *d_delay:3..10000 ms*

- threshold_code is defined in Table 3: Threshold_Code

Threshold	threshold_code
Lower Non-Recoverable	lnr
Lower Critical	lc
Lower Non-Critical	lnc
Upper Non-Critical	unc
Upper Critical	uc
Upper Non-Recoverable	unr

Table 5: Threshold_Code

Hysteresis	*hysteresis_code*
Negative going Hysteresis Value	neg
Positive going Hysteresis Value	pos

Table 6: Hysteresis_Code

hex_outputs bit	Value
15	Output 16
14	Output 15
13	Output 14
12	Output 13
11	Output 12
10	Output 11
9	Output 10
8	Output 9
7	Output 8
6	Output 7
5	Output 6
4	Output 5
3	Output 4
2	Output 3
1	Output 2
0	Output 1

Table 7: Hex_Outputs Bit Mask

fancontrol_mask bit	Bit value	Details
7 .. 3	0	Reserved
2	Sensor active state for PWM 3	1: sensor active 0: sensor inactive
1	Sensor active state for PWM 2	1: sensor active 0: sensor inactive
0	Sensor active state for PWM 1	1: sensor active 0: sensor inactive

Table 8: Fancontrol_Mask

Functions:

The local_sensor command can be used either to access information about the Chassis Manager local sensors, or to change different local sensor parameters.

If the command is used without any parameter, it will return basic information about all the active sensors. For each installed sensor the command displays:

- local sensor number
- sensor name
- sensor type
- value
- measuring unit (only for threshold sensors)
- state (only for threshold sensors)

Example 1. Read all the active sensors

```
%>local_sensor
----------------------------Sensor List----------------------------
--no--Name------------Type--Value--Unit---State------------------
*  2  +3.3V           Thr   3.27   V      Ok
*  3  +5V             Thr   5.04   V      Ok
*  4  +12V            Thr   0.00   V      Lower Critical
*  5  -12V            Thr   -11.88 V      Ok
* 26  Temp1           Thr   27.00  deg C  Ok
* 27  Temp2           Thr   27.00  deg C  Ok
* 45  Fan9            Thr   2100   RPM    Ok
* 46  Fan10           Thr   2100   RPM    Ok
* 47  Fan11           Thr   2100   RPM    Ok
* 48  Fan12           Thr   2100   RPM    Ok
* 64  Input1          Input   1  (Asserted)
* 65  Input2          Input   0  (De-Asserted)
* 66  Input3          Input   0  (De-Asserted)
* 67  Input4          Input   0  (De-Asserted)
* 68  Input5          Input   0  (De-Asserted)
* 80  Output1         Output  1  (Asserted)
* 81  Output2         Output  1  (Asserted)
* 82  Output3         Output  0  (De-Asserted)
* 83  Output4         Output  0  (De-Asserted)
```

If only the sensor number parameter is entered, the command will print a detailed description of the sensor identified by that particular sensor number. The command parses the SDR of the desired sensor and displays various information about the sensor:

- name
- type
- value
- sensor units
- sensor state
- sensor maximum and minimum values
- threshold values
- outputs assigned to threshold events
- fan control groups the sensor is assigned to (only for temperature sensors)
- active level for discrete sensors
- input type for input sensors
- output type for output sensors
- active drivers for output sensors

Example 2. Get detailed information about the 5V sensor

```
%>local_sensor 2
------------------Sensor Details-------------------
* Name: V0
* Type: Threshold
* Value: 3.21
* Sensor Units: V
* State: Lower Non-Recoverable
* Sensor Maximum Reading: 5.00
* Sensor Minimum Reading: 0.00
* Upper non-recoverable threshold: 4.12
    Assigned Outputs:  None
```

```
* Upper critical threshold: 3.80
    Assigned Outputs:  None
* Upper non-critical threshold: 3.49
    Assigned Outputs:  None
* Lower non-critical threshold: 2.86
    Assigned Outputs:  None
* Lower critical threshold: 2.55
    Assigned Outputs:  None
* Lower non-recoverable threshold: 2.23
    Assigned Outputs:  None
* Assertion Events logged for: unr uc unc lnc lc lnr
* De-Assertion Events logged for: unr uc unc lnc lc lnr
* Positive-going threshold hysteresis value: 0.02
* Negative-going threshold hysteresis value: 0.02
* Debounce: 0
```

The local_sensor command can also be used to change different sensor parameters. Using this command the following parameters can be changed:

- temperature sensors : thresholds, hysteresis, fan control active mask, outputs assigned to threshold events
- fan speed sensors: thresholds, hysteresis, outputs assigned to threshold events
- input sensors: active level, outputs assigned to active level,input type
- output sensors: active level, output drivers logic function, input type
- voltage: thresholds, hysteresis, outputs assigned to threshold events
- current: thresholds , hysteresis, output assigned to threshold events

10.25.1Changing the activelevel

Syntax: **local_sensor** sensor_no **activelevel** *0|1*

The activelevel can be changed only for discrete sensors.
The active value can be either 0 (low) or 1 (high).

Example 7. Setting the activelevel to high for input 2 (sensor 65)

```
%>local_sensor 65 activelevel 1
 Operation Successful!
```

10.25.2Asserting/Deasserting an output

Syntax: **local_sensor** *sensor_no* (assert [*ms_timeout]*) | deassert

ms_timeout = 20..65530 (mili seconds)

This command is used to assert or deassert one of the 16 digital outputs. Only the ***admin*** can control the outputs. The ***user*** has its permission denied when he tries to use this command.

For asserting reset signals the *ms_timeout* parameter can be used. In this case the output is first asserted and after the timeout defined by the parameter expires, the output will is de-asserted. The timeout between the assertion and de-assertion of the output can have any value between 20 ms and 65 seconds.

Example 11. Asserting Output 4(sensor 83)

```
%>local_sensor 83 deassert
 Done!
```

10.25.3 Changing the debounce option

Syntax: **local_sensor** sensor_no debounce *debounce_value*

- sensor_no : identifies the sensor that will be set up to drive the led.
- debounce_value: 0-254

Enables or Disables the debounce option for a particular sensor. Debounce is available only for threshold,discrete and input sensors.

The debounce parameter establishes the number of times a sensor is allowed to be out of spec before it is reported as failing. If debounce is 0 the sensor is reported as failing the first time it is out of spec.

Example 17. Set debounce value for sensor 2

```
%>local_sensor 2 debounce 2
 Done!
```

In this case, sensor 2 has to fail 3 times before it is reported as failing.

10.25.4 Changing the fancontrol mask

Syntax: **local_sensor** sensor_no **fancontrol** *fancontrol_mask*

- *fancontrol_mask defined in* Table 6: Fancontrol Mask

The fancontrol parameter is used only for temperature sensors and determines if the sensor is active for any of the fan control groups. For more details on fan control refer to 1.4.1 Fan control .

The fancontrol_mask has 8 bits, but only the least significant 3 are used. The fancontrol_mask is inputed as a hex value. A sensor that is active for all the fancontrol groups has a fancontrol_mask of 0x07.

Example 8. Setting temp 3(sensor 28) active for PWM 1 and 3

```
%>local_sensor 28 fancontrol 0x05
 Operation Successful!
```

10.25.5 Changing the fan to pwm assignment

Syntax: **local_sensor** sensor_no *[fan_group 0(unknown),1,2,3]*

By default the fans are not assigned to any of the PWM signals. So there is no way to distinguish between a fan failing and it being turned off by the firmware. In this case when the fans stop spinning a fan error is signaled regardless if the stop was commanded by the firmware or not.

In order to distinguish between a failure and a firmware commanded stop of the fans, each of them has to be assigned to the PWM it is commanded by. To do so knowledge of the system level wiring is required.

Example 8. assigning fan 1 to PWM 3

```
%>local_sensor 37 fan_group 3
 Operation Successful!
```

10.25.6 Changing the hysteresis

Syntax: **local_sensor** sensor_no **hysteresis** hysteresis_code *value*

- hysteresis code is defined in Table 4: Hysteresis Code

Hysteresis can be modified only for threshold sensors.
Sensors described by full sensor records support 2 hysteresis values.
The *hysteresis_code* parameter represents a abbreviation of the hysteresis that needs to be changed.

Hysteresis values can be changed only for sensors that support hysteresis.
In other cases the following warning message will be displayed:

```
Sensor does not support Hysteresis!
```

Example 4. Changing the positive going hysteresis of 3.3V (sensor 2) a

```
%>local_sensor 2 hysteresis pos 0.04
Operation Successful!
```

10.25.7 Ignoring a fan sensor

Syntax: **local_sensor** sensor_no *ignore_fan en|di*

In some cases fan sensors have to be ignored so that their failure does not trigger any corrective action. This can accomplished by turning on the ignore_fan parameter. This parameter is available only for fan sensors.

Example 8. ignoring fan 1

```
%>local_sensor 37 ignore_fan en
 Operation Successful!
```

10.25.8 Changing the input flags

Syntax: **local_sensor** sensor_no **input_flags clr | shutdown_push | nvmro**

This command is available only for the digital input sensors. It can be used to define specific functionality for an input:

- **shutdown_push**: if the input type is set to shutdown, this defines the input behavior as push button.
- **Nvmro**: Non volatile memory read only . When this input is asserted all non volatile memory is set to read only and all configuration altering commands and procedures are disabled. When an input of this type is asserted the configuration of the ChMC can not be altered in any way.

All input flags are cleared using the **clr** parameter.

10.25.9 Changing the input type

Syntax: **local_sensor** sensor_no **input_type on_off | shutdown**

The *input_type* parameter is available only for input sensors and determines the associated sensor behavior:

- on_off – regular digital input
- shutdown – input used to control the chassis shutdown process. For more details on this

process refer to the 3.2 Shutting down a CPCI system chapter.

Example 10. Setting input 1 as a shutdown input

```
%>local_sensor 64 input_type shutdown
 Operation Successful!
```

10.25.10 Changing the lcd display option

Syntax: **local_sensor** sensor_no **lcd en | di | time** *t_value*

t_value : 0..127 seconds

The ChMC can display local sensors information on a attached LCD or TFT: name, value, state.

The **en** and **di** parameters are used to enable or disable the display of the selected sensor.

The *t_value* parameter selects the amount of time the current sensor is visible, before advancing to the next enabled sensor.

10.25.11 Changing the outputs linked to a sensor event

Syntax: **local_sensor sensor_no output** [*threshold_code*] *hex_outputs*

- *threshold_code is defined in* Table 3: Threshold Code
- *hex_outputs is defined in* Table 5: Hex Outputs

The outputs assigned to sensor events parameter can only be set individually for each sensor and each event.

Outputs can be linked to thresholds infringements for threshold sensors or to the active level for discrete sensors.

For threshold sensor the command uses the threshold code parameter for identifying the event to which the outputs defined by *hex_outputs* are linked.

In the case of discrete sensor the outputs are automatically linked with the active level so the t*hreshold_code* parameter is not used.

If a bit of *hex_outputs* is 1 the corresponding output will be linked to the corresponding event. For example if outputs 1, 5 and 8 need to be linked with an event *hex_output* = 0x0091.

Example 5. Assigning outputs 1 and 2 to the lower critical threshold of fan 2(sensor 38)

```
%>local_sensor 38 output lc 0x0003
 Operation Successful!
```

Example 6. Assigning outputs 7 and 14 to the active level of input 6(sensor 69)

```
%>local_sensor 69 output 0x2040
 Operation Successful!
```

10.25.12 Changing the output flags

Syntax: **local_sensor** sensor_no **output_flags clr | bite_nc | bite_c | bite_nr | reserved1 | reserved2 | fan_mux | shutdown_ready**

This command is available only for the digital output sensors. It can be used to define specific functionality for an output.

- fan_mux : the output set to this function will act as a multiplexer signal for the fan monitoring function. In cases where the ChMC needs to monitor more than 12 fans, an aditional multiplexing board can be used that increases the monitoring capabilities to up to 24 fans. That board multiplexes between 2 sets of 12 fans based on this output signal.
- Shutdown_ready: available only for the CPCI firmware version.

10.25.13 Changing the output type

Syntax: **local_sensor** sensor_no **output_type** *AND | OR*

The *output_type* parameter is available only for output sensors and determines the logic function the sensor performs on its driving signals.

An **AND** output is asserted when all its driving signals are asserted.

An **OR** output is asserted when at least one of its drivers is asserted.

Example 9. Setting the logic function of output 1 to *AND*

```
%>local_sensor 80 output_type AND
 Operation Successful!
```

10.25.14 Changing the sequencing options

Syntax: **local_sensor** sensor_no **seq (order** *o_value***) | (delay** *d_value***)**

o_value: 0..16; 0 = sequencing disabled
d_value: 3..10000 ms

The ChMC supports startup sequencing for the digital outputs.

By default, at startup, all regular outputs are deaserted and all outputs activated for sequencing are asserted. The sequencing outputs are then deasserted one by one according to the sequence order and delays.

To enable an output for sequencing its order in the sequence has to be different than 0.

The sequencing procedure starts deasserting the selected outputs, one by one, starting from the lowest order. The algorithm advances to the next output in the sequence after the delay of the current one has passed.

Example 9. Setting *output 3 to be deasserted at 10 ms after output 1*

```
%>local_sensor 80 seq order 1
 Operation Successful!
%>local_sensor 80 seq delay 10
 Operation Successful!
%>local_sensor 82 seq order 2
 Operation Successful!
```

10.25.15 Changing a threshold's value | Disabling a Threshold

Syntax: **local_sensor** sensor_no **threshold** threshold_code *value| disable*

- threshold_code is defined in Table 3: Threshold Code

The sensors used to monitor the system's parameters are described by full sensor records and support up to 6 thresholds. The threshold can be enabled or disabled by this command.

To activate a disabled threshold all you have to do is set an appropriate value for it. To disable an active threshold you have to use this command with the disable parameter.

The *threshold_code* parameter represents an abbreviation of the threshold that needs to be changed.

The value change operation will be successful only if the new value for the threshold is compliant to the monotony rule for thresholds:

lnr < lc < lnc < unc < uc < unr

The new value is compared only with the values of active thresholds.

Example 3. Changing the upper critical threshold of temp 3 (sensor 28)

```
%>local_sensor 28 threshold uc 45
 Operation Successful!
```

Example 4. Disabling the upper critical threshold of temp 3 (sensor 28)

```
%>local_sensor 28 threshold uc disable
 Threshold disabled!
```

10.25.16 Changing the user's permission to assert/deassert an output

Syntax: **local_sensor** *sensor_no* **userpermission** allow|deny

By default only the admin can assert/deassert the digital outputs. The user is denied when he tries to change the state of the outputs.

This command is used to allow or deny the user to change a particular output.

Example 11. Allow the user to change output 4(sensor 83)

```
%>local_sensor 83 userpermission allow
 Done!
```

10.26 logout command

Syntax: logout

Function:
Logs out the current user and permits a new log in.

10.27 ntpconfig command

Syntax: ntpconfig [serverip | zoneoffset <value>]

Function:

Displays or changes the NTP configuration.

10.28 output command

Syntax: output

Functions:

Displays information about all the output sensors. For each sensor the command displays:

- sensor number
- sensor name
- sensor type
- value

Example 1. Output sensors values read out

```
%>output
-----------------------------Sensor List-----------------------------
--no--Name-------------Type----Value--Unit---State------------------
*   80  Output1          Output   1   (Asserted)
*   81  Output2          Output   1   (Asserted)
*   82  Output3          Output   0   (De-Asserted)
*   83  Output4          Output   0   (De-Asserted)
*   84  Output5          Output   0   (De-Asserted)
*   85  Output6          Output   1   (Asserted)
*   86  Output7          Output   0   (De-Asserted)
*   87  Output8          Output   1   (Asserted)
*   88  Output9          Output   0   (De-Asserted)
*   89  Output10         Output   0   (De-Asserted)
*   90  Output11         Output   0   (De-Asserted)
*   91  Output12         Output   1   (Asserted)
*   92  Output13         Output   0   (De-Asserted)
*   93  Output14         Output   0   (De-Asserted)
*   94  Output15         Output   0   (De-Asserted)
*   95  Output16         Output   0   (De-Asserted)
```

10.29 passw command

Syntax: passw [snmp auth | priv]

Function:

Changes the log-in or snmp passwords for the current user. For log-in password, if instead of the new password an empty string is entered, password checking for the respective user is disabled.

10.30 pef command

Syntax: pef [(action *no_pref_hex_byte*) **| (alert_policy** *index* **[3 x** *no_pref_hex_bytes*]) **|**
(alert_str *index* **[***string***]) | alert_str key** *index* **[2 x** *no_pref_hex_bytes***] |**
(control *no_pref_hex_byte*) **| clear_all |**
(event_filter *index* **[up to 20 x** *no_pref_hex_bytes*])**|**
(guid [17 x *no_pref_hex_bytes*]) **| (id [soft]** *record_id*) **|**
(postpone *no_pref_hex_byte*) **| (set_in_progress 0|1)]**

Function:

Changes or displays all parameters related to PEF (Platform event filtering). For a more detailed description of the syntax of this command please refer to the *ChMC PEF & PET Getting Started* application note.

10.31 pwm command

Syntax: pwm [*pwm_no* **freq** *freq_value***]**

- pwm_no : 1 - 3
- freq_value : 1 – 125 Khz

Functions:

If used without parameters displays the status of all the PWM signals.
The command can also be used to change the frequency of one of the 3 PWM signals.

Example 1. Status reading for pwm signals

```
%>pwm

 Pwm1 Freq: 25 Khz, Duty Cycle:28%
 Pwm2 Freq: 25 Khz, Duty Cycle:100%
 Pwm3 Freq: 25 Khz, Duty Cycle:100%

*Pwm1 and Pwm2 share the same frequency, but can have different duty cycles
```

The pwm command can also be used to change the frequency of the PWM signals.

PWM1 and PWM2 share the same frequency.
If the frequency is changed for one signal it will change for both.
Even tough the frequency is the same for the two PWM signals the duty cycles are independent .

Example 2. Changing PWM1 frequencies to 25 KHz

```
%>pwm 1 freq 25
 Done!
 *Pwm1 and Pwm2 share the same frequency, but can have different duty cycles.
```

10.32 reboot command

Syntax: reboot

Function:
Restarts the ChMC.

10.33 restore command

Syntax: restore

Function:
Restores all parameters to the default values. For the restore to be complete a reboot is necessary.

10.34 saveenv command

Syntax: saveenv

Function:
Saves the parameters that were changed. If the modified parameters are not saved, they will be lost at the next reboot.

10.35 scispeed command

Syntax: scispeed 9600 | 19200 | 38400 |115200

Functions:
Changes the baud rate at which the CLI for the ChMC and the bootloader framework operate. For the change to become valid the environment has to be saved using the saveenv command and the ChMC has to be restarted, either using the reboot command or by using the reset key. The baud rate is changed at the next power up.

Example 1 :

```
 %>scispeed 9600
Baud rate changed to 9600.Save Environment and reboot.
```

10.36 sdrbuffer command

Syntax: sdrbuffer

Function:
Displays all SDRs in raw hexadecimal format

10.37 sel command

Syntax:
Linear log: **sel clr |count | print [***start_record_no* **[***end_record_no***]]**

Circular log: **sel (ageing en| di) | clr | info | (print [(startup [***length***]) | (***index* **[***length***]))**

Functions:

The syntax of the command is different depending on the type of log implemented: linear log on older firmware releases and legacy distributions or circular log on newer releases of firmware.

The circular log has an ageing feature that prevents the sel to fill up and stop recording new events. Once the sel is full, if ageing is enabled, the odest events are deleted in order to make room for new ones.

Linear SEL

The command can display the number of sel entries, display a particular set of these entries or clear the sel.

To print the whole log use : ***sel print***

To print all the records starting with a particular one use : ***sel print*** *record_no*

To print all records in a give interval: ***sel print*** *start_record_no* *end_record_no*

Example 1: Readout of SEL

```
 %>sel print
------------------------------------------------------------------------------
                             Sensor  Event Log

Rec.ID. dd.mm.yyyy hh:mm:ss  Sensor No. Name             Event  Ev.Dir  Value  Threshold
------------------------------------------------------------------------------
------------------------------------------------------------------------------
0x0001  01.01.2012 00:00:00     97      SHMC Power ON     1 (Asserted)
0x0002  01.01.2012 00:00:00     2       V0                LNC     As     0.00     2.86
0x0003  01.01.2012 00:00:00     4       V2                LC      As     0.00     11.38
0x0008  01.01.2012 00:00:00     7       V5                LC      As     0.00     0.66
0x0009  01.01.2012 00:00:00     9       V7                LC      As     0.00     12.39
0x000A  01.01.2012 00:00:00     2       V0                LC      As     0.00     2.55
0x000C  01.01.2012 00:00:00     64      Input1            1 (Asserted)
0x000D  01.01.2012 00:00:00     65      Input2            1 (Asserted)
0x000E  01.01.2012 00:00:00     66      Input3            1 (Asserted)
0x000F  01.01.2012 00:00:00     67      Input4            1 (Asserted)
0x0019  01.01.2012 00:00:00     77      Input14           1 (Asserted)
0x001A  01.01.2012 00:00:00     78      Input15           1 (Asserted)
0x001B  01.01.2012 00:00:00     79      Input16           1 (Asserted)
```

Circular SEL

The command can display the number of sel entries, display a particular set of these entries or clear the sel.

To print all the events of the current startup use : ***sel print startup***

To print a number of records records starting with a particular one use : ***sel print*** *record_no no_of_records*

Example 2: SEL clearing

```
 %>sel clr
```

Done! Sel is empty!

10.38 sendipmb command

Syntax: **sendipmb** ipmb_addr netFn_RsLun cmd rawhexdata...

Function:

Sends a raw IPMI commands over IPMB to the specified ipmb_addr slave address.

Example 1: Get device id request to IPMC 0x82

```
%> sendipmb 0x82 0x18 0x01
```

10.39 sensor command

Syntax: sensor all | (ipmb_address [(**no** *sensor_no* [-v])])

Function:

Parses sdr info for all the sensors, or a particular one on a IPMC specified by its IPMB address and displays info. The command displays sensor information in a user friendly manner.

10.40 shelfaddr command

Syntax: shelfaddr [set [-h] [-cpci] *value***]**

Functions:

Displays or changes the chassis Address.The chassis address field is used to uniquely identify a chassis when the DHCP clientID option is used.

If the "-cpci" options is used the address is a two bytes value as defined in PICMG 2.9.

If the "-cpci" options is not used the address is a maximum 20 characters text, as defined in PICMG 3.0.

The PICMG 3.0 defined address is stored into the Address table record, if this record is present in the chassis FRU Information. Whenever the Address table record is present into the chassis FRU Information, the chassis Address field of this record is used. If this field is empty or the record is not present, the PICMG 2.9 address is used (only for CPCI systems).

10.41 snmp command

Syntax: snmp [admin | user | [auth [md5 | none] | privacy [des | aes | none]]]

Functions:

Configure the SNMP v3 authentication and privacy settings for admin and user profiles.

10.42 sol command

Syntax: sol [on|off]

Functions:

Displays or changes the current status of the Serial Over Lan (SOL) configuration. Whenever the SOL mode is activated the behavior of RS232 port will change. All the data received on the serial port will be transmitted in telnet packets over Ethernet to all telnet connections opened on "serial" profile. The data received from the telnet connections opened in "serial" mode is transferred to the serial port.

10.43 temp command

Syntax: temp

Functions:

Displays information about all the installed temperature sensors. For each installed sensor the command displays:

- sensor number
- sensor name
- sensor type

- current value
- measuring unit
- current state

Example 1. temperature sensor values read out

```
%>temp
----------------------------Sensor List-----------------------------------
--no--Name-------------Type--Value--Unit---State---------------------------
* 26  Temp1            Thr   26.00  deg C  Ok
* 27  Temp2            Thr   26.00  deg C  Ok
* 28  Temp3            Thr   26.00  deg C  Ok
* 29  Temp4            Thr   Not Present!
* 30  Temp5            Thr   Not Present!
* 31  Temp6            Thr   Not Present!
```

10.44 tftp command

Syntax: tftp configfile | firmware | web | *tftp_server_ip_address filename*

Functions:

Uploads a new configuration file, firmware image or web page by TFTP.

10.45 time command

Syntax: time[hh:mm:ss]

Functions: displays or sets the current time

Example 1: Check out of time

```
%>time

 Time [hh:mm:ss] = 16:52:27
```

Example 2: Setup of time

```
%>time 16:53:00
 Done!
```

10.46 uptime command

Syntax: uptime

Function:

Displays the amount of time which has past since the last ChMC power up.

10.47 version command

Syntax: version

Function:

Displays information about the ChMC : Part Number,Software Version,MAC Address, and Serial Number.

10.48 voltage command

Syntax: voltage

Functions:
Displays information about all the installed voltage sensors. For each installed sensor the command displays:

- sensor number
- sensor name
- sensor type
- current value
- measuring unit
- current state

Example 1. voltage sensor values read out

```
%>voltage

-----------------------------Sensor List-----------------------------
--no--Name-------------Type----Value--Unit---State------------------

*    2  V0               Thr     1.00    V      Lower Non-Recoverable
*    3  V1               Thr     2.30    V      Ok
*    4  V2               Thr     4.55    V      Lower Critical
```

10.49 xmodem command

Syntax: xmodem configfile | localfru | shelffru | shmcfru | web |tft

Functions:

Sends via RS232 a configuration file, a fru file for a local FRU, the chassis, or the ChMC or the web page. After the command is entered, the ChMC goes into data receive mode and waits for the data to be sent . You can then start the file transfer with your terminal program and select XMODEM as the protocol.

! When using "**xmodem**" in "Hyperterminal" the transfer of the desired file can take up to 10 seconds to start.
Only the admin can use the xmodem command.

Example 1: Sending of the config file

```
%>xmodem configfile
 Please upload the file...
%>...Done!
```

11 TFT Touch Screen Display

The TFT displays information about the sensors implemented locally on the Chassis Manager. It will display the sensor name, value and state.

For each local sensor implemented on the Chassis Manager, the user can configure it will be displayed or not and the amount of time its information will be shown.

By default the enabled sensors are displayed one by one is an auto play mode. The user can choose to pause and navigate through the sensor list using the on-screen buttons. Optionally if the touch feature is not desired, external buttons can be used.

Figure 5: TFT Display

The TFT interface has the following structure:

1. Stop/pause button
2. Show previous sensor
3. Show next sensor
4. Sensor name
5. Icon associated with the sensor type
6. Sensor unit of measurement (if available)
7. Sensor value: integer value for analog sensors or "HIGH"/"LOW"(for discrete sensors)
8. Sensor state ("Ok", "Not Ready", "Not Present", "Lower Critical", etc)
9. Index of the current sensor in the list of sensors selected to be displayed

12 Update protocol

All update procedures are accomplished by sending binary files to the ChMC. The files can be sent sent over the RS232 interface using the xmodem protocol, or over Ethernet using tftp.

12.1 Update over RS232 using xmodem

For connecting to the RS232 Command Line Interface(CLI) you will have to use a terminal program. On Windows OS we recommend the use of **Tera Term**.

By default the terminal settings are:

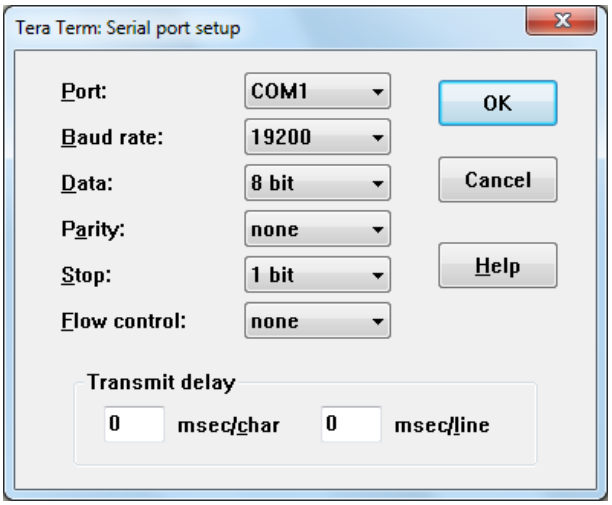

Figure 6: Default Terminal Settings

The **Port** setting should describe the physical port of the computer on which the ChMC is connected.

The default Baud rate is 19200,but it can be changed using the configuration file to : 9600, 19200,38400 or 115200.

At start-up the Command Line Interface (CLI) of the ChMC will request a login. The default login data for the admin account:

login: **admin**
password: **ADMIN**

The default password can be changed using the Configfile.

If the RS232 connection is used, the files are uploaded using the xmodem protocol. The update is started by using the CLI command "**xmodem**" with the parameter that corresponds to the file type uploaded.

%> xmodem firmware | configfile | web | shmcfru

After typing the command you will be prompted to upload the file. At this point you have to send the file using your terminal program.

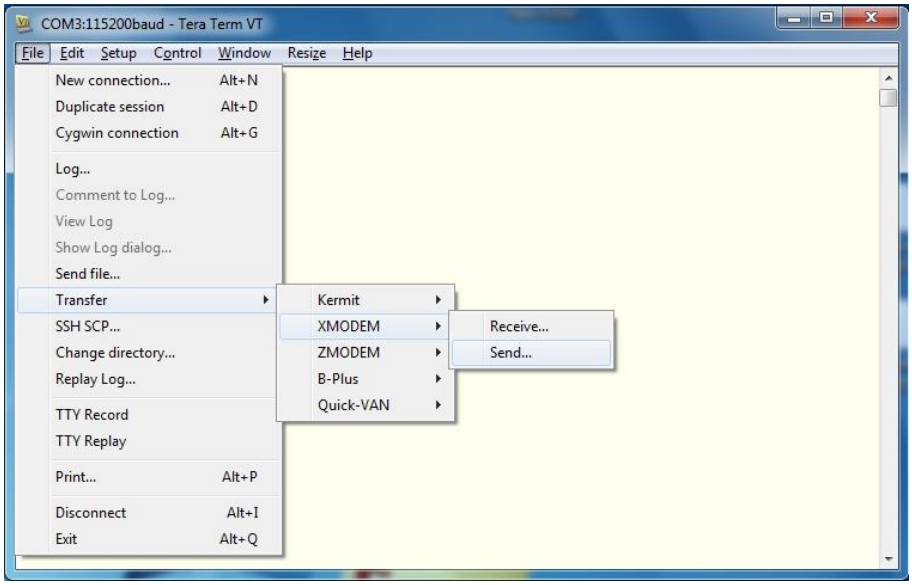

Figure 7: Xmodem send file for TeraTerm software

After the transfer is complete a confirmation message is displayed: **%>Done!**

12.2 Update over Ethernet using tftp

The update operations can also be done remotely using a Ethernet connection and a tftp server program. On Windows OS we recommend the use of Tftpd32 / Tftpd64 as a tftp server.

For setting up the server you will have make sure the Current Directory points to the location of the file you want to upload and the Server interface is set to the Ethernet connection to the ChMC.

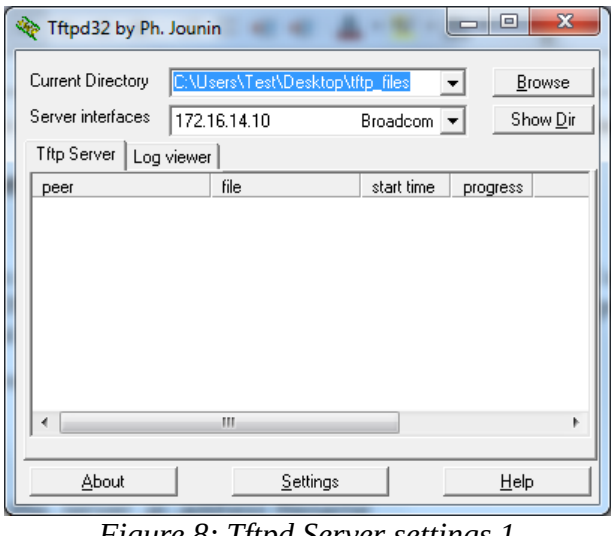

Figure 8: Tftpd Server settings 1

The Tftpd software can be used as a server for multiple Ethernet protocols. In our case we are using it as a tftp server, so you have to make sure that this option is enabled in the settings dialog. For changing the settings of the tfpt server the dedicated tab has to be

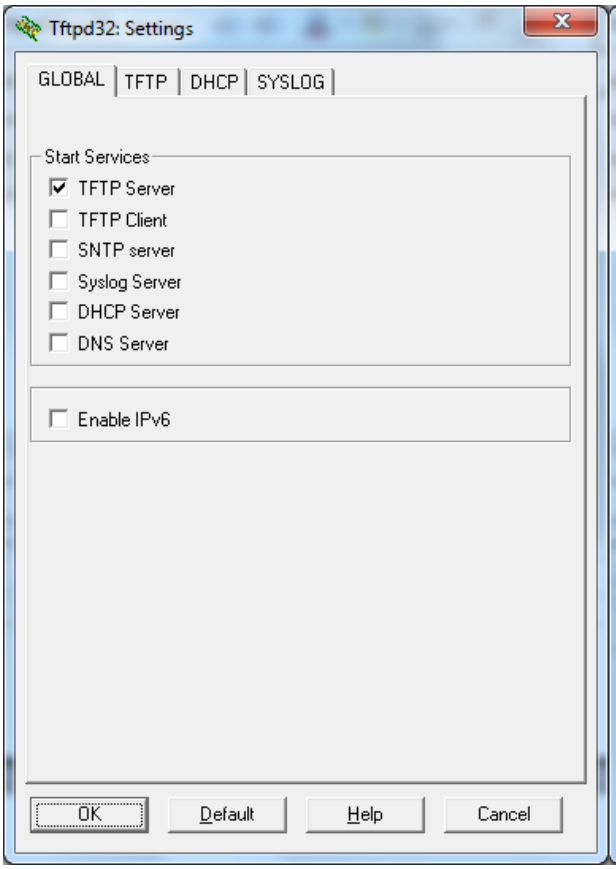

Figure 9: Tftpd Server settings 2

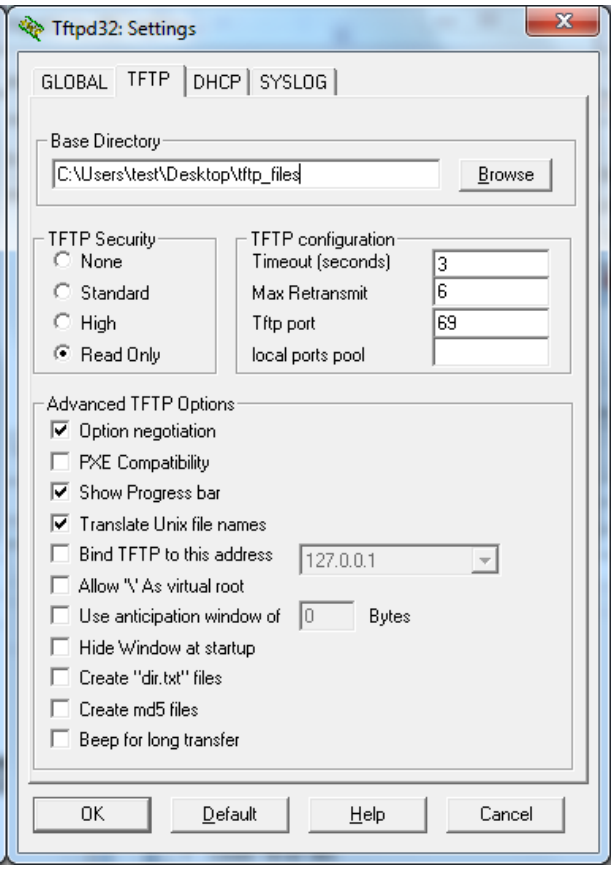

Figure 10: Tftpd Server settings 3

used.

The update process is started from the ChMC CLI. To connect to the CLI remotely a telnet capable terminal program can be used. On Windows OS, we recommend Tera Term. The connection settings are:

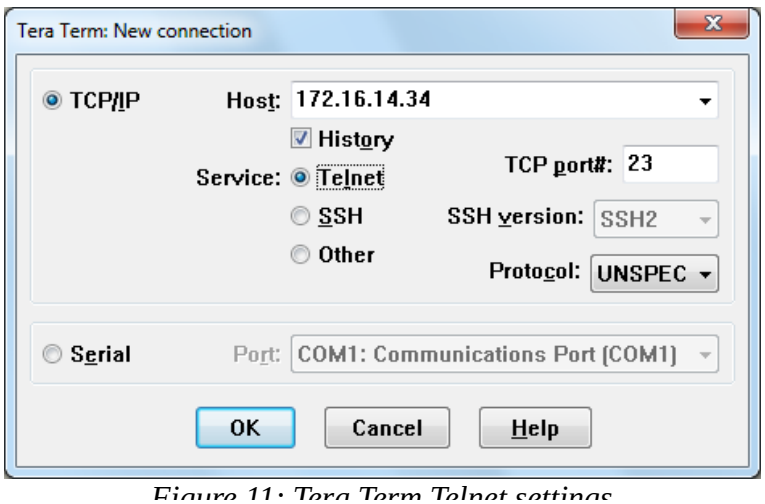

Figure 11: Tera Term Telnet settings

To successfully connect to the ChMC its ip address has to be entered as Host. At start-up the CLI of the ChMC will request a login.

The default login data for the admin account:

login: **admin**
password: **ADMIN**

The default password can be changed using the Configfile.
The file transfer is started using the CLI command tftp:

%>**tftp firmware | web | configfile** *tftp_server_ip_address filename*

12.3 Updating the firmware

12.3.1 RS232 firmware update

For updating the firmware over RS232 the Bootloader framework will be used. The update procedure is:

1. Connect to the RS232 interface using a terminal program.
2. Stop the bootloader at startup by pressing "x" before the timeout expired(3 seconds).
3. Log in using the admin profile, by entering the same credential as for the main firmware.
4. Use the *xmodem* command and the *firmware* parameter. In the bootloader the cursor is different that the one used in the main firmware: # instead of **%>**. The bootloader xmodem command allows only the firmware parameter.
   #xmodem firmware
5. Upload the firmware image: ***.firm**
6. After the transfer is complete a confirmation message is displayed:Done!
7. Launch the new firmware the *run* command: #**run**

12.3.2 Tftp firmware update

The Tftp firmware update procedure:

1. Connect to the telnet interface using a terminal program.
2. Log in using the admin profile.
3. Set up the tftp server to point to the folder that holds the firmware image file and to use the Ethernet interface connected to the ChMC.
4. Use the *tftp* command,the *firmware* parameter, the ip of the tftp server and the firmware update file name (*.firm):
   %>***tftp firmware 172.16.14.10 test.firm***
5. After the transfer is complete a confirmation message is displayed: File transferred!
6. The firmware will be updated by the bootloader at the next start-up. To perform a restart the *reboot* command can be used: %>**reboot**
7. The bootloader doesn't support telnet, so the current connection will be lost. After the update is done a new telnet connection can be opened.

12.4 Updating the configuration file

12.4.1 RS232 configfile update

The RS232 config file update procedure is:

1. Connect to the RS232 interface using a terminal program.
2. Log in using the admin profile.
3. Use the *xmodem* command and the *configfile* parameter.
   %>**xmodem configfile**
4. Upload the configuration file: ***.config**
5. After the transfer is complete a confirmation message is displayed:Done!
6. The configuration will be updated at the next start-up. To perform a restart the *reboot* command can be used: %>**reboot**.
7. If the configuration file changes the scispeed parameter, at this point the baud rate has to be changed for the terminal program. After restarting the ChMC will use the new scispeed parameter

and the characters will not be displayed correctly by the terminal software until its baud rate is also updated.

12.4.2 Tftp config file update

The Tftp config file update procedure:

1. Connect to the telnet interface using a terminal program.
2. Log in using the admin profile.
3. Use the *tftp* command,the *confgifile* parameter, the ip of the tftp server and the config file update file name (*.config):
   %>***tftp configfile 172.16.14.10 test.config***
4. After the transfer is complete a confirmation message is displayed: File transferred!
5. The configuration will be updated at the next start-up. To perform a restart the *reboot* command can be used: %>**reboot**.
6. The bootloader doesn't support telnet, so the current connection will be lost. After the update is done a new telnet connection can be opened.
7. The confif file can be used to change the lan parameters of the ChMC. In this case the ip of the board can change, so this has to be also considered when reconnecting over telnet.

12.5 Updating the web page

12.5.1 RS232 web page update

The RS232 web page update procedure is:

1. Connect to the RS232 interface using a terminal program.
2. Log in using the admin profile.
3. Stop all connections to the ChMC web page.
4. Use the *xmodem* command and the *web* parameter.
   %>**xmodem web**
5. Upload the web page : ***.web**
6. After the transfer is complete a confirmation message is displayed:Done!
7. At this point the web page update has been completed and the new web page is operational.

12.5.2 Tftp web page update

The Tftp web page update procedure:

1. Connect to the telnet interface using a terminal program.
2. Log in using the admin profile.
3. Stop all connections to the ChMC web page.
4. Use the *tftp* command,the *web* parameter, the ip of the tftp server and the web page update file name (*.web):
   %>***tftp web 172.16.14.10 test.web***
5. After the transfer is complete a confirmation message is displayed: File transferred!
6. At this point the web page update has been completed and the new web page is operational.

13 Extracting the configuration file

Only an admin can extract the configuration file. The extraction mechanism is available only over RS232 and is based on the XMODEM protocol.

Due to safety reasons all the passwords in the extracted file will be changed to the default ones : user, admin and serial CLI access profile access passwords, SNMP user and admin profiles authentication and privacy passwords and the community name for the SNMP traps.

The RS232 config file extraction procedure is:

1. Connect to the RS232 interface using a terminal program.
2. Log in using the admin profile.
3. Use the *extract* command and the *configfile* parameter.
   %>**extract configfile**
4. Start the **reception** of the configuration file in your terminal program.
5. The end of transfer will be signaled by the terminal program.

14 Restore to factory defaults procedure

Only an admin can perform a system restore. In order to restore all parameters to their default value the following steps need to be followed:

- Login using the admin account (for more details refer to 5. Command Line Interface (CLI))
- Use the restore command

! When using restore the ChMC disregards all the changes applied to the SDRs, Sensor options and User Settings and uses a predetermined set for all these parameters. This predetermined set of parameters may be different then the one loaded on the ChMC when it was shipped out.

To go back to a particular setup you have to use xmodem configfile, and upload a Configuration file that contains the required setup.

15 Electrical and Environmental Parameters

Power supply: 3.3VDC

Current consumption: 500mA

PWM: Open drain (100mA max). Pull-up required on fans.

Digital Input/Output signal level: Depending on VIO setting (3.3V or 5V)

Digital Output type: open drain with 10K pull-up to VIO.

Digital Output max loading: 25mA on single output but all 16 outputs shall not exceed 200mA.

Operating Temperature: -40..85 deg C

Storage Temp