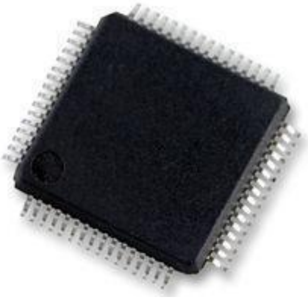
VPX IPM Controller Data Sheet

Key features

- ***Compliant to IPMI 2.0***
- ***Compliant to VITA46.11-2022***
- ***SOSA Aligned***
- ***HPM.1 firmware upgrade***
- ***Easy to integrate***
- ***Evaluation Board Available***
- ***Reference Designs Provided***
- ***GUI software for configuration***
- ***Cost Effective:***
- ***No upfront costs for standard version***
- ***No royalties***
- ***Analog inputs for voltage, current or temperature measurements***
- ***External I2C for temperature measurements and communication with payload***
- ***LEDs and payload power control outputs***
- ***Implemented on 64pins TQFP microcontroller***
- ***Runs on top of FreeRTOS operating system***



Rev 1.6 02.10.2023

Table of Contents

1 Description 3
2 Supported IPMI Commands..4
3 Board Configuration 6
    3.1 FRU Information 6
    3.2 Sensor Information 7
    3.3 VPX Test Card Ios 8
    3.4 I/O Electrical Characteristics.. 8
4 Configuring Sensors 9
5 Payload Signals 12
6 Payload shutdown protocol..12
7 Command Line Interface (CLI)...14
8 List of CLI commands 14
    8.1 bit command...14
    8.2 channel command...14
    8.3 firewall command 14
    8.4 help command...16
    8.5 info command 16
    8.6 ipmb command...16
    8.7 logout command...16
    8.8 nvm command 16
    8.9 payload_reset 17
    8.10 payload_signals 17
    8.11 reboot command...17
    8.12 saveenv command 17
    8.13 sel command...17
    8.14 sensor command...17
    8.15 settings command..18
    8.16 uptime command 18
    8.17 user command..18
    8.18 version command...18
    8.19 xmodem command...19
9 Update Procedure 19
    9.1 Updating the Firmware 19
    9.2 Updating the FRU and SDR files 21
10 Order code..22

Index of Tables

Table 1: List of Supported Commands 4
Table 2: List of Supported Sensors 7
Table 3: VPX Test Card J1 pinout 8
Table 4: VPX Test Card J2 pinout 8
Table 5: Absolut maximum ratings.. 8
Table 6: Payload available Signals 12

1 Description

The Intelligent Protocol Management Controller (IPMC) Software allows quick development of boards without prior IPMI knowledge. It provides all the mandatory IPMI functionality required by VITA46.11 specification and supports also optional features like Firewall and NVM Write protection Commands.

There are two purchase options for IPMC software: software delivered on pre-programmed microcontrollers or IPMC software source code license. Both are accompanied by reference schematics and a complete set of GUI compilers for the SDR and FRU files.

The IPMC Software is written in "C" and runs on top of FreeRTOS operating system. It is compliant to VITA46.11-2022, IPMI 2.0, PICMG HPM.1 and is aligned to SOSA specification.

The IPMC Software supports a predefined set of sensors: temperature, voltage, current, fan, GPIO or OEM sensors. This are selected and configured through a standard IPMI SDRs (**S**ensor **D**ata **R**ecord) file. Beside required IPMB-0 interface (two IPMI busses A and B), the IPMC supports also an I2C SSIF interface, a serial IPMI interface which supports Basic Mode IPMI communication and one serial CLI debug interface.

The software could be easily upgraded in the field, through the Chassis Manager, using HPM.1 protocol.

For IPMC Software evaluation a VPX Test card is available.

The IPMC Software is a cost-effective solution that enables very fast development of VPX boards. There are no royalties. The standard version of the IPMC Software can be customized to fulfill any requirements.

Illustration 1: IPMC Evaluation Board

2 Supported IPMI Commands

The VPX IPMC was developed based on the IPMI v2.0 and ANSI/VITA 46.11 specification and is aligned to SOSA specification.

IPM Device "Global" Commands	NetFn	CMD
Get Device ID	App	01h
Cold Reset	App	02h
Get Self Test Results	App	04h
Master Write Read	App	52h
Firewall	**NetFn**	**CMD**
Get NetFn Support	App	09h
Get Command Support	App	0Ah
Get Command Sub-function Support	App	0Bh
Get Configurable Commands	App	0Ch
Get Configurable Command Sub-Functions	App	0Dh
Set Command Enables	App	60h
Get Command Enables	App	61h
Set Configurable Command Sub-function Enables	App	62h
Get Configurable Command Sub-function Enables	App	63h
Get OEM NetFn IANA Support	App	64h
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	
VITA 46.11 Commands	**NetFn**	**Group ID**	**CMD**
Get VSO Capabilities	Group Extension	VSO(03h)	00h
FRU Control	Group Extension	VSO(03h)	04h
Get FRU LED Properties	Group Extension	VSO(03h)	05h
Get LED Color Capabilities	Group Extension	VSO(03h)	06h
Set FRU LED State	Group Extension	VSO(03h)	07h
Get FRU LED State	Group Extension	VSO(03h)	08h
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
FRU Control Capabilities	Group Extension	VSO(03h)	1Eh
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
Get FRU Hash	Group Extension	VSO(03h)	45h
Get Payload Mode Capabilities	Group Extension	VSO(03h)	46h
Set Payload Mode	Group Extension	VSO(03h)	47h
Get Write Protect Capabilities	Group Extension	VSO(03h)	48h
Get Write Protect Enables	Group Extension	VSO(03h)	49h
Set Write Protect Enables	Group Extension	VSO(03h)	4Ah
Get Control Bits Capabilities	Group Extension	VSO(03h)	4Eh
Get Control Bits	Group Extension	VSO(03h)	4Fh
Set Control Bits	Group Extension	VSO(03h)	50h
Get Bridged NetFn Support	Group Extension	VSO(03h)	51h
Get Bridged Command Enables	Group Extension	VSO(03h)	52h
Set Bridged Command Enables	Group Extension	VSO(03h)	53h
Get Bridged Command Sub-function Enables	Group Extension	VSO(03h)	54h
Set Bridged Command Sub-function Enables	Group Extension	VSO(03h)	55h
Set Bridged NetFn Policy	Group Extension	VSO(03h)	56h

Get Bridged NetFn Policy	Group Extension	VSO(03h)	57h

Table 1: List of Supported Commands

3 Board Configuration

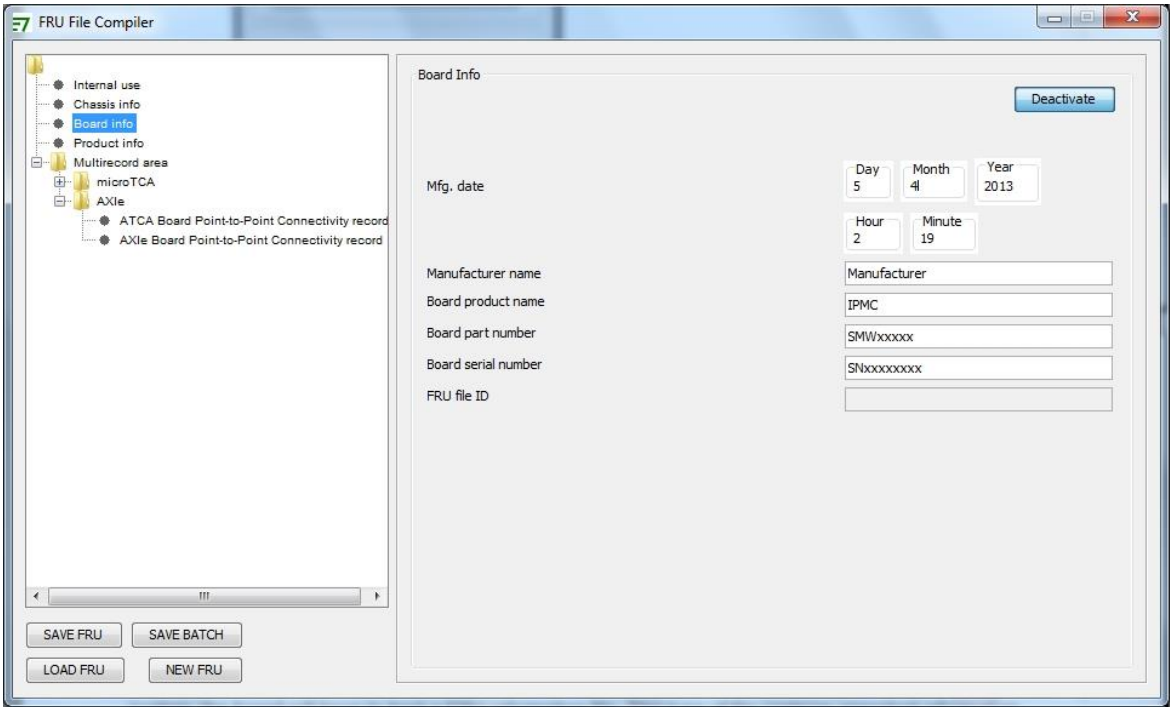
The VPX test card configuration is comprised by two files: FRU information and Sensor information (SDR repository).

3.1 *FRU Information*

The VPX test card will be used in an IPMI environment. In order to interact to the other FRUs in the system, the board will have to host an FRU information file. This type of file contains important information concerning the board:

- Manufacturer's name
- Part Number
- Serial Number
- Revision
- Manufacturing data
- Information related to the communication protocols implemented over the Base, Fabric, Timing and Local Bus interfaces.
- other IPMI related information

All the required information can be saved in the FRU file format using the GUI FRU compiler.



Board Info

Mfg. date: Day 5, Month 4, Year 2013, Hour 2, Minute 19

Manufacturer name: Manufacturer

Board product name: IPMC

Board part number: SMWxxxxx

Board serial number: SNxxxxxxxx

FRU file ID:

Illustration 4: GUI FRU Compiler Example

3.2 Sensor Information

By default, the VPX IPMI software supports a predefined set of sensors. Each supported sensor has a unique sensor number and, if implemented, it shall have a Sensor Data Record (SDR) definition. Using the set of SDRs the IPMI software knows what sensors are implemented on a card and knows how to monitor them.

The pre-programmed microcontrollers with IPMC software support up to 5 analog sensors, which can be any type of sensor with an analog output, for example: voltage sensor (resistor divider or IC), temperature sensor (thermistor or IC), PHT sensor, light sensor etc; and 4 digital (I2C) temperature sensors equivalent with TMP100 (ex: TMPx75, LM75).

A list of all the available sensors and details regarding MCU ports and test card ports is available in the below table:

IPMI Sensor Number	Sensor Type	MCU Port	Test Card Port	Test Card Sensor Config
20	Analog input	PIO0_23	VITA46 [P0]/ -12V_AUX	Voltage
21	Analog input	PIO0_16	VITA46 [P0]/ VS1	Voltage
22	Analog input	PIO0_15	VITA46 [P0]/ +12V_AUX	Voltage
23	Analog input	PIO0_31	VITA46 [P0]/ VS2	Voltage
24	Analog input	PIO1_0	VITA46 [P0]/ VS3	Voltage
32	Digital [I2C] temperature Addr: 0x90h [8 bit]	PIO0_19 [SCL] PIO0_27 [SDA]	J2 [pin 1]/ PRIVATE_SCL J2 [pin 2]/ PRIVATE_SDA	TMP100NA U4
33	Digital [I2C] temperature Addr: 0x92h [8 bit]	PIO0_19 [SCL] PIO0_27 [SDA]	J2 [pin 1]/ PRIVATE_SCL J2 [pin 2]/ PRIVATE_SDA	-
34	Digital [I2C] temperature Addr: 0x94h [8 bit]	PIO0_19 [SCL] PIO0_27 [SDA]	J2 [pin 1]/ PRIVATE_SCL J2 [pin 2]/ PRIVATE_SDA	-
35	Digital [I2C] temperature Addr: 0x96h [8 bit]	PIO0_19 [SCL] PIO0_27 [SDA]	J2 [pin 1]/ PRIVATE_SCL J2 [pin 2]/ PRIVATE_SDA	-

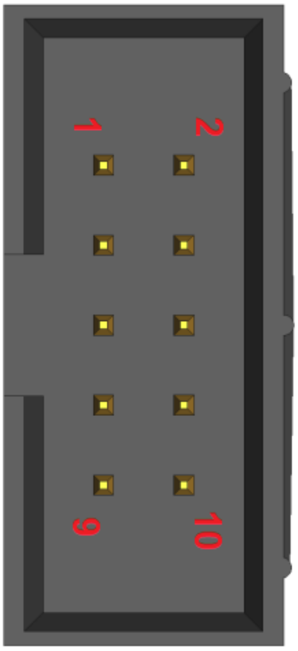
Table 2: List of Supported Sensors

3.3 VPX Test Card Ios

The VPX test card comes with the following IO ports: VITA46 P0 port, RS-232 DSUB9 (COM1) for CLI, and 2 2x5 pin 100MIL shrouded IO connectors (J1/J2) with the following pinout:

PAYLOAD_TXD	PAYLOAD_RXD
BP_SYSRESET*	BP_GDISCRETE1
VCC	GND
NVMRO	SSIF_ALERT
SSIF_SCL	SSIF_SDA

Table 3: VPX Test Card J1 pinout



SHDN_REQ#	SHDN_RDY#
PP_OFF#	SPARE_GPIO
VCC	GND
PWR_GOOD	PP_RST
Private_SCL	Private_SDA

Table 4: VPX Test Card J2 pinout

3.4 I/O Electrical Characteristics

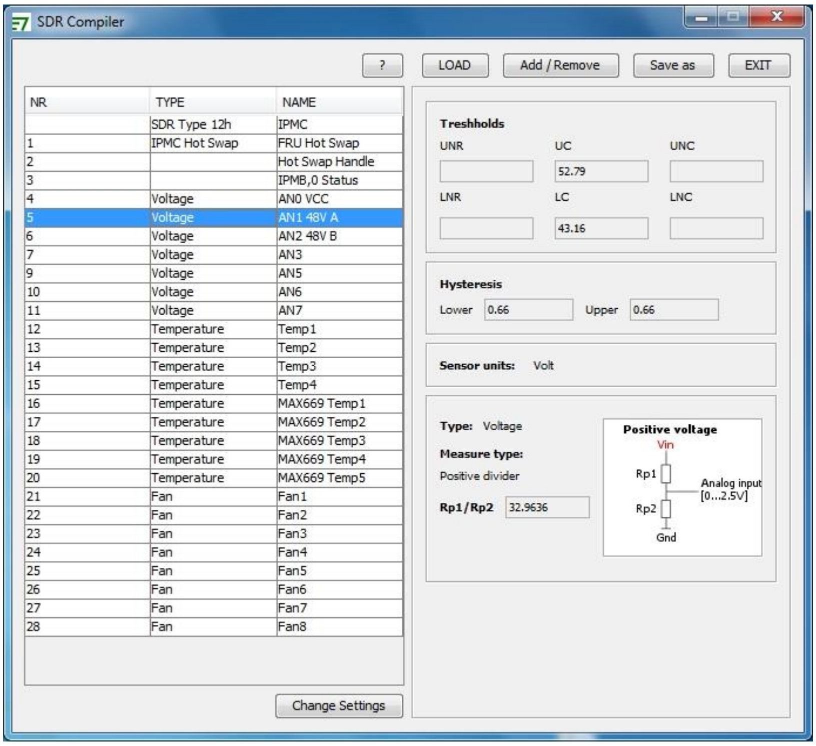
Absolut maximum ratings				
Symbol	Parameter	Min	Max	Unit
Vcc	Main IO supply; Vcc =+3.3V_AUX	-0.3	3.96	V
PIOx_x	Any I/O pin of the microcontroller	-0.3	Vcc +0.3V	V
VS1*	VS1 rail on VITA46 P0 conn	-0.3	16.5	V
VS2*	VS2 rail on VITA46 P0 conn	-0.3	16.5	V
VS3*	VS3 rail on VITA46 P0 conn	-0.3	9.25	V
-12V_AUX*	-12V_AUX rail on VITA46 P0 conn	-14	2.58	V
+12V_AUX*	+12V_AUX rail on VITA46 P0 conn	-0.3	16.5	V
**Applies to test card only*				

Table 5: Absolut maximum ratings

4 Configuring Sensors

The IPMC uses standard, IPMI compliant SDR records to monitor the board parameters.

The SDR repository of a board will be a software image of the hardware sensors. For each board there may be a different set-up as the requirements are different. So, in order to allow a quick and simple set-up, a GUI SDR compiler is provided.



NR	TYPE	NAME
	SDR Type 12h	IPMC
1	IPMC Hot Swap	FRU Hot Swap
2		Hot Swap Handle
3		IPMB,0 Status
4	Voltage	AN0 VCC
5	Voltage	AN1 48V A
6	Voltage	AN2 48V B
7	Voltage	AN3
9	Voltage	AN5
10	Voltage	AN6
11	Voltage	AN7
12	Temperature	Temp1
13	Temperature	Temp2
14	Temperature	Temp3
15	Temperature	Temp4
16	Temperature	MAX669 Temp1
17	Temperature	MAX669 Temp2
18	Temperature	MAX669 Temp3
19	Temperature	MAX669 Temp4
20	Temperature	MAX669 Temp5
21	Fan	Fan1
22	Fan	Fan2
23	Fan	Fan3
24	Fan	Fan4
25	Fan	Fan5
26	Fan	Fan6
27	Fan	Fan7
28	Fan	Fan8

Treshholds

UNR	UC	UNC
	52.79	

LNR	LC	LNC
	43.16	

Hysteresis

Lower 0.66 Upper 0.66

Sensor units: Volt

Type: Voltage

Measure type:

Positive divider

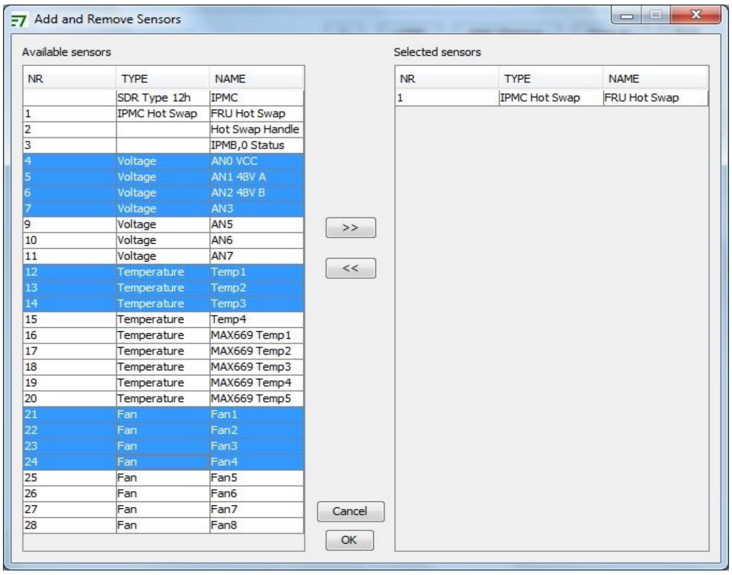
Rp1/Rp2 32.9636

Illustration 7: SDR Compiler Main Interface

Using the GUI compiler, a subset of all supported sensors can be defined by a simple select operation.

After the SDR set has been defined, all the sensors can be customized:

- threshold and hysteresis values can be changed for analog sensors.



Available sensors

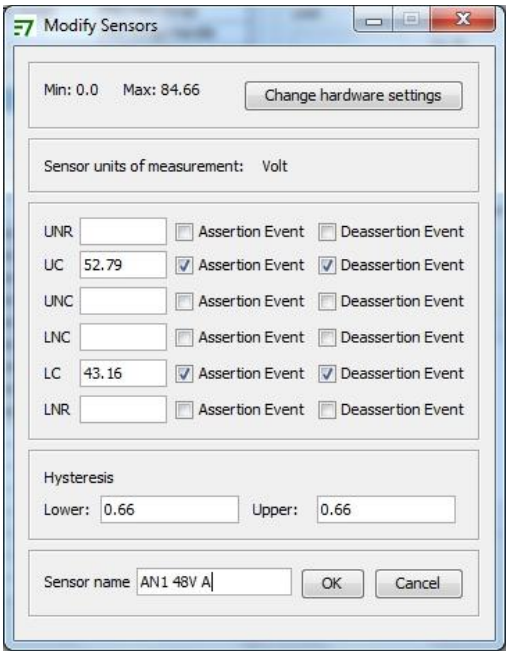
NR	TYPE	NAME
	SDR Type 12h	IPMC
1	IPMC Hot Swap	FRU Hot Swap
2		Hot Swap Handle
3		IPMB,0 Status
4	Voltage	AN0 VCC
5	Voltage	AN1 48V A
6	Voltage	AN2 48V B
7	Voltage	AN3
9	Voltage	AN5
10	Voltage	AN6
11	Voltage	AN7
12	Temperature	Temp1
13	Temperature	Temp2
14	Temperature	Temp3
15	Temperature	Temp4
16	Temperature	MAX669 Temp1
17	Temperature	MAX669 Temp2
18	Temperature	MAX669 Temp3
19	Temperature	MAX669 Temp4
20	Temperature	MAX669 Temp5
21	Fan	Fan1
22	Fan	Fan2
23	Fan	Fan3
24	Fan	Fan4
25	Fan	Fan5
26	Fan	Fan6
27	Fan	Fan7
28	Fan	Fan8

Selected sensors

NR	TYPE	NAME
1	IPMC Hot Swap	FRU Hot Swap

Illustration 10: Selecting a subset of sensors

- names can be changed for all sensors.



Min: 0.0 Max: 84.66

Sensor units of measurement: Volt

UNR		□ Assertion Event	□ Deassertion Event
UC	52.79	☑ Assertion Event	☑ Deassertion Event
UNC		□ Assertion Event	□ Deassertion Event
LNC		□ Assertion Event	□ Deassertion Event
LC	43.16	☑ Assertion Event	☑ Deassertion Event
LNR		□ Assertion Event	□ Deassertion Event

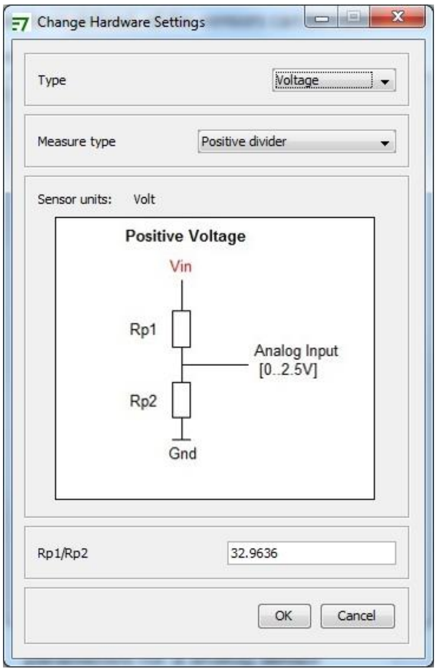
Hysteresis

Lower: 0.66 Upper: 0.66

Sensor name AN1 48V A

Illustration 13: Window for changing parameters for a analog sensor

- for the analog inputs the raw SDR formulas of a few hardware circuits have been implemented. This feature allows easy integration of common analog set-ups: positive voltage divider, negative voltage divider, and gain block. For these common circuits only the divider/gain value must be inputted, and the raw conversion formula will be computed automatically by the software. For more complex circuits, the raw formula can be inputted manually.

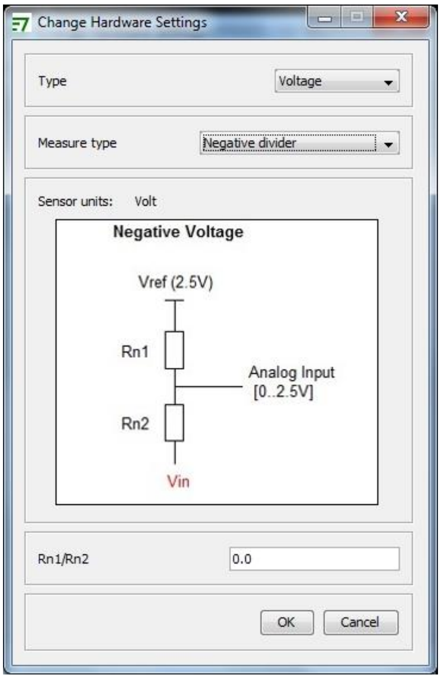

Change Hardware Settings

Type: Voltage

Measure type: Positive divider

Sensor units: Volt

Rp1/Rp2: 32.9636

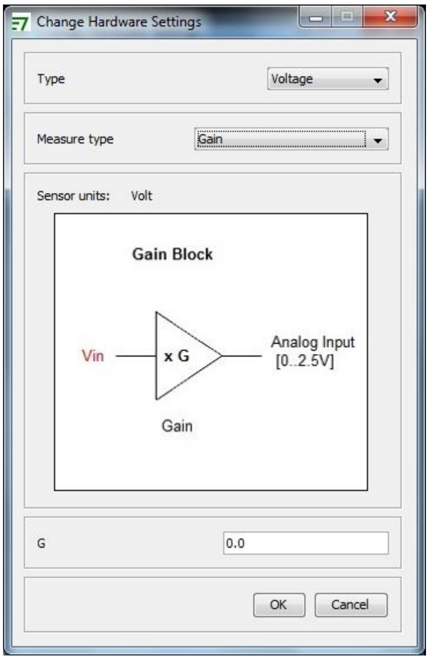

Change Hardware Settings

Type: Voltage

Measure type: Negative divider

Sensor units: Volt

Rn1/Rn2: 0.0

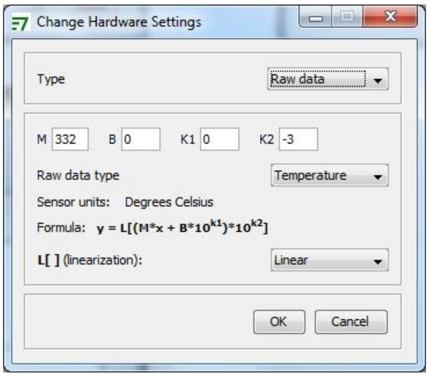

Change Hardware Settings

Type: Voltage

Measure type: Gain

Sensor units: Volt

G: 0.0


Change Hardware Settings

Type: Raw data

M 332 B 0 K1 0 K2 -3

Raw data type: Temperature

Sensor units: Degrees Celsius

Formula: $y = L[(M*x + B*10^{k1})*10^{k2}]$

L[ ] (linearization): Linear

Illustration 16: Various Embedded Formula selection screens

5 Payload Signals

Signal Name	Type	Active level	Description
PP_RST#	Output	Low	Payload reset signal
Private_SCL	In-Out		I2C serial clock
Private_SDA	In-Out		I2C serial data
SHDN_REQ#	Output	Low	When active, signals the payload that a shutdown has been requested
SHDN_RDY#	Input	Low	Asserted by payload when is ready to shut down. Pull signal down if not used
PP_OFF#	Output	Low	When asserted turns off payload power
PowerGood	Input	High	Asserted by payload when all voltages are good. Pull signal high if not used

Table 6: Payload available Signals

6 Payload shutdown protocol

The Payload Signals are used by the IPMC to communicate to the payload. As part of this communication, the IPMC implements a protocol for shutting down / powering on the payload when the IPMC is deactivated / activated.

At start-up the IPMC fist check the state of PP_OFF# signal. If it is low, pulled by pull-down resistor if PCA9536 is not configured, or signal is driven low by PCA9536, the IPMC asserts PP_RST signal. After this PP_OFF# signal is released and PowerGood is monitored after 10ms timeout. When PowerGood is sampled high, the PP_RST is deasserted.

The shutdown protocol uses 2 output signals (SHDN_REQ#, PP_OFF#) and 1 input (SHDN_RDY#) signal. These signals are implemented on an optional PCA9536 I/O expander. If the shutdown option is not needed, the I/O expander may not be placed into design.

If the payload shutdown I/O expander is implemented the shutdown function could be further disabled/enabled in software, using the *settings payload_sd en* | *di* command. (Changes done to the settings have to be saved to the non-volatile area using the *saveenv* command to become permanent). By default, the protocol is disabled.

When the IPMC is activated the PP_OFF# signal will be deaserted so the payload will start receiving power.

When the I/O expander is implemented, and shutdown protocol is disabled, the IPMC cannot be deactivated. Otherwise, before completing the deactivation, the IPMC will shut down the payload according to the diagram bellow:

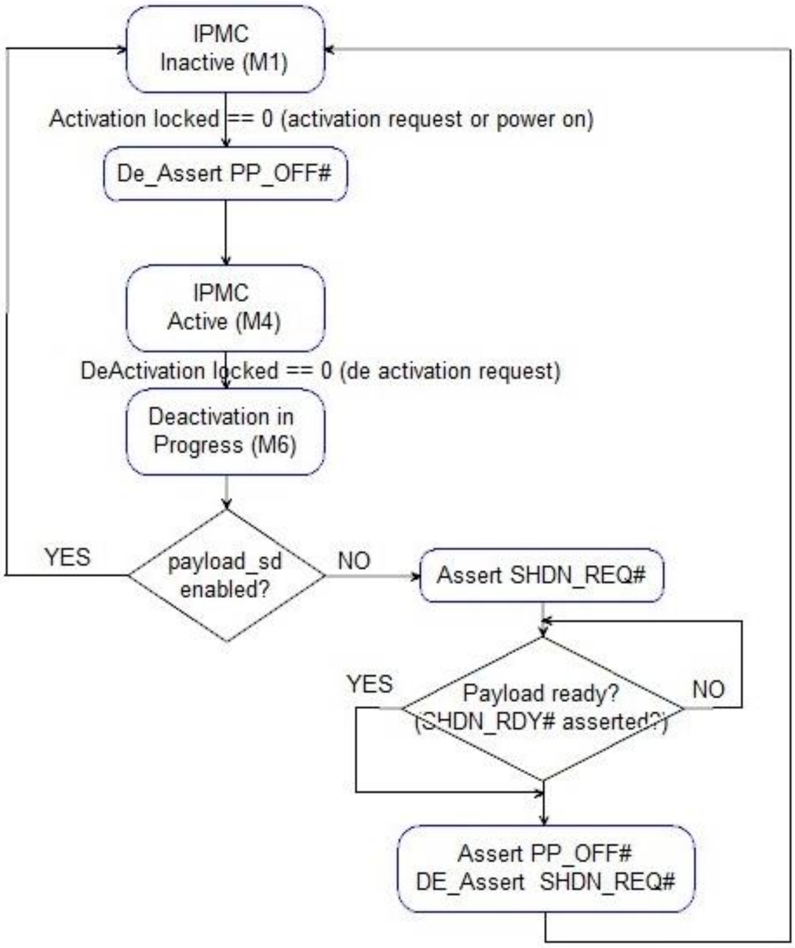


Illustration 19: IPMC Payload Shutdown Protocol

When a deactivation request is received, and the protocol is enabled the IPMC will move to M6 and signal to the payload that a shutdown has been requested: SHDN_REQ# will be asserted (the signal will become low). The IPMC will wait for the payload to finish the shutdown before advancing to the next state. When the payload has finished the shutdown, it will inform the IPMC by asserting the SHDN_RDY# signal. At this point the IPMC will assert the PP_OFF# signal to turn off the power to the payload and deassert the SHDN_REQ# signal as the shutdown process has been finished. Next the IPMC will advance to M1 as the deactivation has been accomplished.

7 Command Line Interface (CLI)

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

On Windows systems, we recommend the use of “Tera Term” or “Hyperterminal” as the terminal programs.

Terminal settings:

- 115200 bits per second
- data bits: 8
- parity: none
- stop bit: 1

For file transfer the CLI implements the xmodem protocol.

8 List of CLI commands

8.1 *bit command*

Syntax: bit
Function: Displays Power-On Built In Test (PBIT) or Continuous Built In Test (CBIT) results.

8.2 *channel command*

Syntax: channel
Function: Displays IPMC’s supported communication channels and their interfaces

8.3 *firewall command*

Syntax:
firewall [bridged] channel <*channel_no*> [netfn <*netfn_no*> [lun <*lun_no*>] [command <*cmd_no*> [oemgroup <*oem_iana*|*group_id*>] [enable|disable|subfn <*bitmask*>]]]
firewall bridged channel <*channel_no*> (policy | passtrhough) [<*value*>]

Function: Displays or sets firewall and bridged firewall configuration.

Firewall is used to enable or disable commands, or command sub-functions, to be executed by IPMC. The configuration is made for each channel and applies only to request messages received on that channel.
Bridged firewall applies to messages forwarded by IPMC from one channel to another, using Send Message command. The configuration is based on destination channel.
Command keywords:

Bridged – if bridged parameter is entered the command will get or change the bridged firewall configuration. If bridged parameter is missing, the command will get or set the firewall configuration.
channel – the message request source channel for firewall configuration or the message destination channel for bridged message
netfn – network function
lun – LUN number. If LUN parameter is not present it is assumed to be 0
command – command number
oemgroup – group number for netfn 0x2C or oem IANA private enterprise number for netfn 0x2E
enable – enables the specified command
disable – disables the specified command
subfn <bitmask> - stets the sub-function mask bits. Parameter bitmask could be one or two 32 bits numbers, representing the sub-function mask.

Command options:
firewall [bridged] channel <*channel_no*>
This command returns all netfn supported on a given channel

firewall [bridged] channel <*channel_no*> netfn <*netfn_no*>
This format of firewall command returns all commands supported on a given channel – netfn combination and the current status: enabled or disabled. Some enabled commands are not configurable and cannot be disabled.

firewall [bridged] channel <*channel_no*> netfn <*netfn_no*> [lun <*lun_no*>] command <cmd_no>
This syntax returns detailed information about a command: status, privilege level, command name and supported sub-functions details, if the command has sub-functions

firewall [bridged] channel <*channel_no*> netfn <*netfn_no*> [lun <*lun_no*>] command <cmd_no> enable
Enables the specified command is the command is supported and configurable

firewall [bridged] channel <*channel_no*> netfn <*netfn_no*> [lun <*lun_no*>] command <cmd_no> disable
Disables the specified command is the command is supported and configurable

firewall [bridged] channel <*channel_no*> netfn <*netfn_no*> [lun <*lun_no*>] command <*cmd_no*> subfn <*bitmask*>
set a command sub-function mask according to bitmask value. The bitmask could one 32 bits number or two 32 bits numbers, in case the commands support extended sub-functions (up to 64 subfunctions)

firewall bridged channel <*channel_no*> policy [<*value*>]

This command returns or sets (if value provided) the policy bits for bridged messages. The policy value is a 3 bits mask, where:
bit 2 allows, if set, unknown requests (commands which are not “known” by bridged firewall) to be forwarded to selected channel

firewall bridged channel <*channel_no*> passtrhough [<*value*>]
This command returns or sets (if value provided) the passthrough value for bridged messages. The passthrough value is a bit mask, where each bit represents a channel.
In case a bit is set, the commands originating for that channel are forwarded to destination channel <channel_no> directly, without applying the bridged firewall rules.

8.4 help command

Syntax: help
Function: Displays a list of the available commands.

8.5 info command

Syntax: info
Function: Displays IPMC’s IPMB address and boot count

8.6 ipmb command

Syntax: ipmb
Function: Displays the state of the IPMB A and IPMB B buses and number of detected errors.

8.7 logout command

Syntax: logout
Function: logs out current user and starts the login procedure

8.8 nvm command

Syntax:
nvm [destination (embedded) [(protect|unprotect) identification |configuration | log)]] |
[category (identification|configuration|log) [(protect|unprotect) (embedded)]]
Function: Get or set the local NVM write protect configuration which is used when NVMRO is not protecting the global memories.
The **nvm** command allows the memory to be protected based on destination: embedded, local, remote or by category: identification, configuration, log. The current IPMC implementation has only embedded memory, therefore only embedded keyword is shown.
Example:

```
%>nvm destination embedded
 Write protected categories: none
 Write unprotected categories: identification, configuration, logs

%>nvm category configuration protect embedded
 Done!
```

This command variant protects the configuration category located inside embedded memory.

8.9 payload_reset

Syntax: payload_reset [<*timeout_tens_of_ms*>]

Function: Asserts the payload reset signal, keeps it active for the time value entered as a parameter and then de-assert it.

8.10 payload_signals

Syntax: payload_signals

Function: Displays the status for the payload signals.

Example:

```
%>payload_signals
Payload Signals status:

 NVMRO: Deasserted / Pin: Deasserted
 GDiscrete1: Deasserted / Pin: Not Available
 SHDN_REQ#: Deasserted
 SHDN_RDY#: Deasserted
 PP_OFF#: Deasserted
 PowerGood: Asserted
 PP_RST#: Deasserted
```

8.11 reboot command

Syntax: reboot

Function: Restarts the IPMC

Example:

```
%>reboot
 System will restart! Please wait...
```

8.12 saveenv command

Syntax: saveenv

Function: Saves configuration parameters in the non-volatile memory, if the memory is not write-protected

8.13 sel command

Syntax: sel [(ageing en|di)] | [clr] | [print [startup | (<*start_index*> [<*count*>])]]

Function: Prints the local System Event Log (SEL), clears it or enable disable ageing

8.14 sensor command

Syntax: sensor

Function: Displays information for the installed set of sensors.

Example:

```
%>sensor
---------------------Sensor List----------------------------
```

```
--no--Name--------------Value--Unit---State-----------------
*   0  FRU State        M4: FRU Active
*   1  System IPMB      IPMB A: ok   , Enabled
                    IPMB B: ok   , Enabled
*   2  FRU Health       Healthy
*   3  FRU Voltage      Ok
*   4  FRU Temp         Ok
*   5  Payld Tst Res    Success
*   6  Payld Tst Status Done
*   7  Payload Mode     P2
```

8.15 settings command

Syntax: settings [tier [1|2]] | [payload_sd [en|di]] |
[default_act [0|1]] | [deact_ignored [0|1]]
[ipmbspeed [100|400]]

Function: displays or changes the current IPMC settings

tier : IPMC Tier Level. The IPMC could be configured to operate as either Tier1 or Tier2

payload_sd: if this process is enabled, the IPMC will negotiate the shutdown with the payload before turning off the power

default_act: non-volatile value for the Default Activation Locked bit. If default activation is disabled, the board will stay in M1 (payload power OFF) until it is enabled by Chassis Manager.

deact_ignored: non-volatile value for the Deactivation Ignored bit. If this bit is set a deactivation command is ignored.

ipmbspeed: The speed of the IPMC could be configured as either 100KHz or 400KHz

Example:

```
%>settings tier 1
Done!
%>settings default_act 1
Done!
```

8.16 uptime command

Syntax: uptime

Function: Displays the amount of time which has passed since the IPMC became operational.

Example:

```
%>uptime
 Uptime=0 days 03:05:12
```

8.17 user command

Syntax: user [<*id*> [(enable|disable) | (username <*new_name*>) | (password <new*_password*>)]]

Function: Displays information about supported users, change usernames and passwords

8.18 version command

Syntax: version

Function: Displays various information about the IPMC: firmware version, Hardware Id, Tier level

Example:

```
%>version
 IPMC VPX FW 2.3 V2
 Hardware Id : 4
 Tier 2
 Hardware Address: 0x44
 IPMB Address: 0x88
```

8.19 xmodem command

Syntax: xmodem fru | sdr

Function: Upload the FRU or SDR file to the IPMC using the xmodem protocol

Example:

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

9 Update Procedure

9.1 Updating the Firmware

The Firmware of the IPMC can be updated using the on-board bootloader.

For uploading a file the following steps are required :

- Connect to the CLI interface(**Terminal settings:** 115200 bits per second, data bits: 8, parity: none, stop bit: 1)
- Stop the bootloader by pressing 'x'
- Issue the **xmodem** firmware command

```
%> xmodem firmware
```

- Upload the **.firm* file using the terminal program
- After the file transfer is completed the firmware will be updated.

For boards running bootloader versions older than ***Rev 1.00 b 6*** the **.firm* file must be selected within 15 seconds from the moment xmodem firmware command is sent, otherwise the update process won’t start. From Rev 1.00 b 6 and beyond the time was increased to 1 minute.

For boards that came preloaded with firmware, in order to access the bootloader, the following steps are required:

- Login to gain access to all commands, by sending following commands:

```
%>logout
```

- Input login credentials

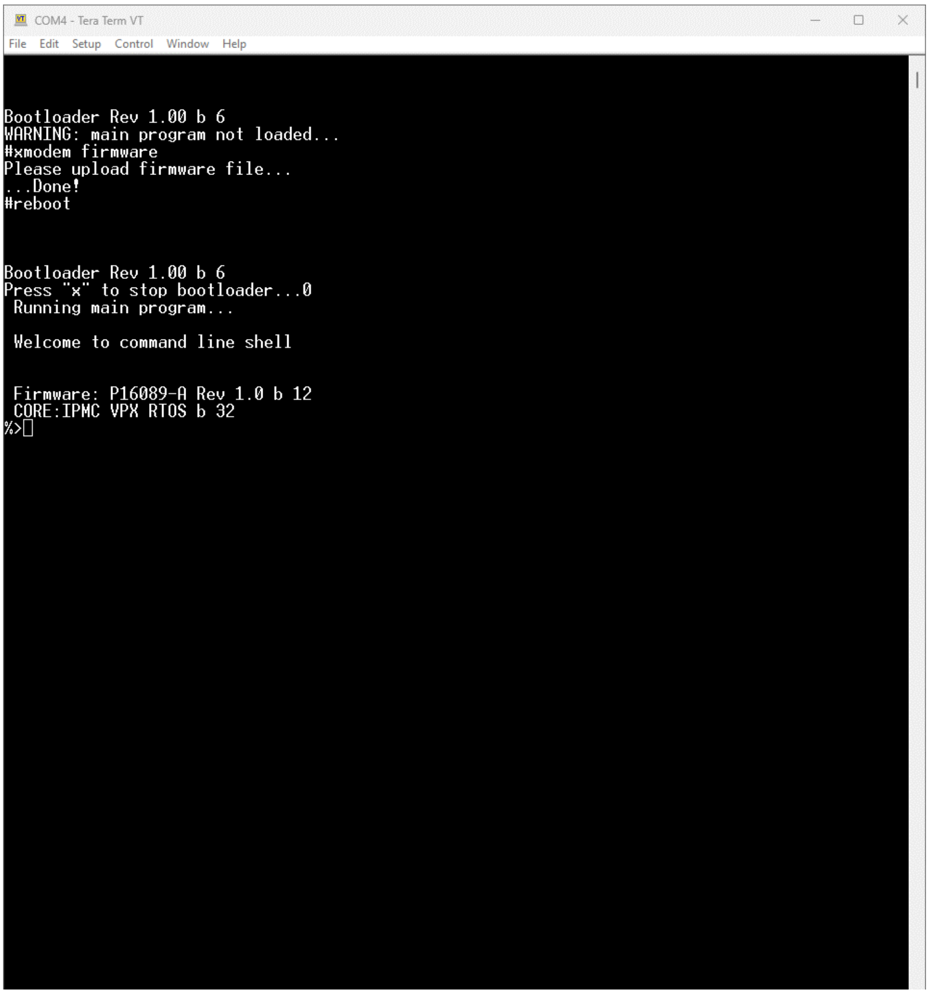
Default credentials (case sensitive):

Login:admin

Password:ADMIN

- Send “reboot -b” command.
- When the board reboots stop the bootloader by pressing 'x' before countdown ends.

For boards that came preloaded with firmware **Rev 1.0 b 12** or newer the “**xmodem firmware**” command cand be issued from the firmware CLI (login required); there is no need to upload the new **.firm* file via bootloader. After uploading image from main firmware, a reboot (send “**reboot**” command) is mandatory in order for the upgrade to complete.



```
Bootloader Rev 1.00 b 6
WARNING: main program not loaded...
#xmodem firmware
Please upload firmware file...
...Done!
#reboot


Bootloader Rev 1.00 b 6
Press "x" to stop bootloader...0
 Running main program...

 Welcome to command line shell


 Firmware: P16089-A Rev 1.0 b 12
 CORE:IPMC VPX RTOS b 32
%>
```

Figure 9: Firmware update example

9.2 Updating the FRU and SDR files

In order to configure the IPMC two files are required: the FRU and SDR file. Both can be easily created using the GUI software suites that accompany the IPMC: FRU File compiler and SDR File compiler.

Creating new files or modifying old ones is really straight forward due to the graphical interface. For more details on all the available options please refer to the respective software user manuals.

After the files are created they have to be uploaded using the CLI.

For uploading a file the following steps are required :

1. Connect to the CLI interface
2. Issue the **xmodem** command, using the correct parameter:

%> xmodem fru | sdr

3. Upload the file using the terminal program
4. After the file transfer is completed a confirmation message will be displayed. At this point the file has been saved and a reboot is required in order to activate the changes.

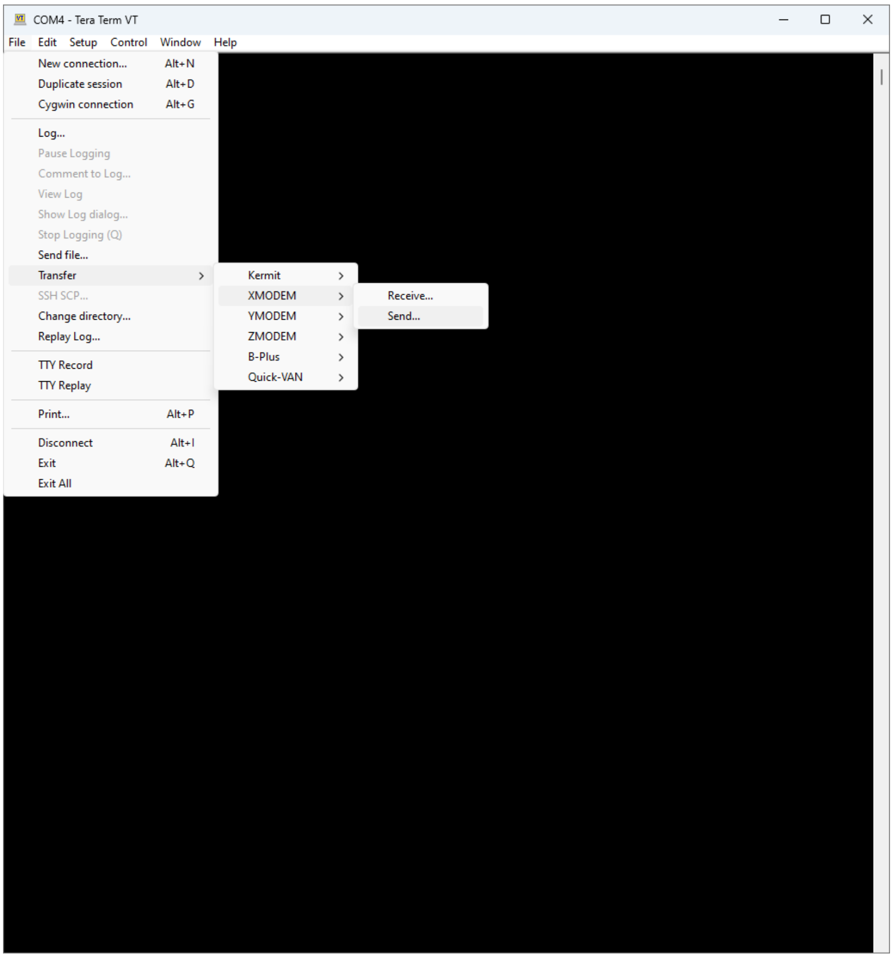


Figure 10: Tera Term Screen shot for sending a file using xmodem

10 Order code

SMW19A0V0 – VPX IPMC software programmed on NXP LPC55S28JBD64 microcontroller

SW18021 – VPX IPMC software “C” source code license