DMMC-STAMP

Command Line Reference

libdmmc V2.08

Contents

1 Introduction

The DESY MicroTCA® Management Controller System on Module (DMMC-STAMP) provides a full management solution to operate the targeted Advanced Mezzanine Card® (AMC) in a MicroTCA® based ecosystem. This guide provides a command line reference for commonly used operations when the DMMC-STAMP is present on an AMC within the target system. For more details on the DMMC-STAMP functionality please refer to the DMMC-STAMP User Manual.

2 ipmitool basics

`ipmitool` is a command line interface to the IPMI management protocol which is used for system remote accesses. When used within a MicroTCA ecosystem `ipmitool` communicates with the MCH by default. For reaching in-system components like AMCs additional parameters are needed (see below).

2.1 Double bridging

To make `ipmitool` communicate with a MMC on a AMC, directly a "double bridging" pattern has to be used:

- The communication between MCH and AMC MMC takes place over IPMB which means the MCH has to translate from LAN to IPMB.
- When translating from LAN to IPMB within the MCH the Shelf Manager and the Carrier Manager have to get bridged ("double bridge"):
  - from LAN (Shelf Manager) to IPMC (Carrier Manager)
  - from IPMC (Carrier manager) to IPMB (to the AMC MMC)

The Shelf Manager and Carrier Manager are logical entities that are usually integrated in a single MCH. How operators can tell `ipmitool` to realize the "double bridge" is described here:

- The first bridge needs to translate from Shelf- to Carrier Manager, meaning channel 0 (see MicroTCA.0 spec, REQ 3.463 & REQ 3.466) and address 0x82 (see REQ 3.194 & REQ 3.195). In `ipmitool` terms this is expressed with the arguments:

  `-B 0 -T 0x82`

- The second bridge targets the IPMB (channel 7) and the MMC IPMB address, or in `ipmitool` terms:

  `-b 7 -t <ipmb_addr>`

The full `ipmitool` invocation looks like this:

```
ipmitool -I lan -H <MCH_HOSTNAME> -A NONE -B 0 -b 7 -T 0x82 \
         -t <MMC_IPMB_ADDR> <command...>
```

For example, with a MCH at `mskmchhvf1.tech.lab`, a MMC at `0x7c` and sending the command `mc info` it is:

```
ipmitool -I lan -H mskmchhvf1.tech.lab -A NONE -B 0 -b 7 \
         -T 0x82 -t 0x7c mc info
```

2.2 Shell alias

By using a shell alias one can avoid to repeatedly typing the same options. Operators can put the following function in their `~/.bashrc` or `~/.zshrc`:

```
ipmbtool() {
    ipmitool -I lan -H $1 -A NONE -B 0 -b 7 -T 0x82 -t ${@:2}
}
```

Now the above mentioned example can be shortened to:

```
ipmbtool mskmchhvf1.tech.lab 0x7c mc info
```

3 IPMI sensors

The DMMC-STAMP exposes its own and additional AMC on-board sensors through the standard IPMI sensor interface. This facilitates reading of temperatures, voltages, power good signals etc. in a uniform way. IPMI sensors can also raise events; especially for overheating, which can trigger the MCH to increase the crate fan speed or shut down the whole AMC in the worst case.

3.1 Reading sensors by using MCH console

1. Open the MCH console (in this example we have a NAT MCH at `mskmchhvf1.tech.lab`):

```
$ telnet mskmchhvf1.tech.lab
Trying 192.168.1.209...
Connected to mskmchhvf1.tech.lab.
Escape character is '^]'.

 Welcome to N.A.T. MCH CM/ShM Firmware V2.23.2c Engineering (r20846M) (Sep 20
     ↪ 2022 – 17:36)


 Current open telnet sessions:
   192.168.1.92:36236 (this connection)

Type <?> to see a list of available commands.
nat>
```

2. Use `show_fru` to determine the FRU ID of the AMC in question:

```
nat> show_fru

FRU Information:
________________
 FRU  Device   State  Name
==========================================
  0   MCH       M4    NAT–MCH–CM
  3   mcmc1     M4    NAT–MCH–MCMC
  5   AMC1      M4    CCT AM G64/472
  6   AMC2      M1    DAMC–FMC1Z7IO
  7   AMC3      M4    DAMC–FMC2ZUP
  8   AMC4      M1    DAMC–FMC2ZUP
 40   CU1       M4    Schroff uTCA CU
 50   PM1       M4    NAT–PM–AC600
 60   Clock1    M4    MCH–Clock
 61   HubMod1   M4    MCH–PCIe
==========================================
nat>
```

Here we choose the DAMC-FMC2ZUP in AMC slot 3, with FRU ID 7.

3. Use `show_sensorinfo` to dump its sensors:

```
nat>show_sensorinfo 7
Sensor Information for FRU 7 / AMC3
==================================================================
  #   SDRType  Sensor Entity Inst  Value   State    Name
------------------------------------------------------------------
  -   MDevLoc         0xc1  0x63                     DAMC-FMC2ZUP
  0   Full     0xf2    0xc1  0x63  0x00               AMC Hot Swap
  1   Compact  0x0b    0xc1  0x63  0x00               0x00  801F12F0B063
  2   Full     Temp    0xc1  0x63  27.5 C     ok      STAMP Temp
  3   Full     Voltage 0xc1  0x63  3.392 V    ok      AMC MP 3V3
  4   Full     Voltage 0xc1  0x63  12.44 V    ok      AMC PP 12V
  5   Full     Current 0xc1  0x63  0.000 A    ok      I_RTM MP 3V3
  6   Full     Current 0xc1  0x63  0.00 A     ok      I_RTM PP 12V
  7   Compact  0x14    0xc1  0x63  0x01               0x00  CPLD Done
  8   Compact  0x14    0xc1  0x63  0x00               0x00  RTM MP 3V3 PG
  9   Compact  0x14    0xc1  0x63  0x00               0x00  RTM PP 12V PG
 10   Compact  0x14    0xc1  0x63  0x00               0x00  RTM Fault
 11   Compact  0x14    0xc1  0x63  0x01               0x00  PGood_A
 12   Compact  0x14    0xc1  0x63  0x01               0x00  PGood_B
 13   Compact  0x14    0xc1  0x63  0x01               0x00  FPGA1 Init
 14   Compact  0x14    0xc1  0x63  0x01               0x00  FPGA1 Done
 15   Compact  0x14    0xc1  0x63  0x01               0x00  FPGA2 Init
 16   Compact  0x14    0xc1  0x63  0x01               0x00  FPGA2 Done
 17   Full     Temp    0xc1  0x63  32.0 C     ok      Inlet Temp
 18   Full     Temp    0xc1  0x63  29.0 C     ok      Outlet Temp
 19   Full     Temp    0xc1  0x63  33.0 C     ok      LTM4630 Temp
 20   Full     Temp    0xc1  0x63  34.0 C     ok      LTM4650 Temp
 21   Full     Temp    0xc1  0x63  38.0 C     ok      LTM4633_F Temp
 22   Full     Temp    0xc1  0x63  39.0 C     ok      LTM4633_R Temp
 23   Full     Temp    0xc1  0x63  36.5 C     ok      ZUP IC Temp
 24   Full     Temp    0xc1  0x63  34.5 C     ok      S7 IC Temp
 25   Full     Current 0xc1  0x63  0.58 A     ok      IMON_AVTT
 26   Full     Current 0xc1  0x63  0.38 A     ok      IMON_AVTTY
 27   Full     Current 0xc1  0x63  0.496 A    ok      IMON_AVCC
 28   Full     Current 0xc1  0x63  0.224 A    ok      IMON_AVCCY
 29   Full     Voltage 0xc1  0x63  0.7168 V   ok      Vcore
 30   Full     Voltage 0xc1  0x63  1.8000 V   ok      VCC_Vadj
 31   Full     Voltage 0xc1  0x63  1.1904 V   ok      VCC_1V2
 32   Compact  0x14    0xc1  0x63  0x01               0x00  FMC-4SFP+ PG_M2C
 33   Compact  0xf0    0xc1  0x63  0x10               HS 007 AMC3
------------------------------------------------------------------
```

3.2 Reading sensors by using ipmitool

Use ipmitool's `sdr` command to retrieve sensor readings. Here we query the same board as in the example above. With IPMB address = 0x70 + slot_nr*2, the board in slot 3 can be reached at 0x76.

```
$ ipmbtool mskmchhvf1 0x76 sdr
AMC Hot Swap      | 0x00              | ok
801F12F0B063      | 0x00              | ok
STAMP Temp        | 35.50 degrees C   | ok
AMC MP 3V3        | 3.38 Volts        | ok
AMC PP 12V        | 12.44 Volts       | ok
I_RTM MP 3V3      | 0 Amps            | ok
I_RTM PP 12V      | 0 Amps            | ok
CPLD Done         | 0x01              | ok
RTM MP 3V3 PG     | 0x00              | ok
RTM PP 12V PG     | 0x00              | ok
RTM Fault         | 0x00              | ok
PGood_A           | 0x01              | ok
PGood_B           | 0x01              | ok
FPGA1 Init        | 0x01              | ok
FPGA1 Done        | 0x01              | ok
FPGA2 Init        | 0x01              | ok
FPGA2 Done        | 0x01              | ok
Inlet Temp        | 40 degrees C      | ok
Outlet Temp       | 40 degrees C      | ok
LTM4630 Temp      | 41 degrees C      | ok
LTM4650 Temp      | 43 degrees C      | ok
LTM4633_F Temp    | 47.50 degrees C   | ok
LTM4633_R Temp    | 48 degrees C      | ok
ZUP IC Temp       | 47 degrees C      | ok
S7 IC Temp        | 45.50 degrees C   | ok
IMON_AVTT         | 0.54 Amps         | ok
IMON_AVTTY        | 0.36 Amps         | ok
IMON_AVCC         | 0.38 Amps         | ok
IMON_AVCCY        | 0.22 Amps         | ok
Vcore             | 0.72 Volts        | ok
VCC_Vadj          | 1.80 Volts        | ok
VCC_1V2           | 1.20 Volts        | ok
FMC-4SFP+ PG_M2C  | 0x01              | ok
```

The `sensor` command will retrieve more detailed information, including the event thresholds of the sensors:

```
$ ipmbtool mskmchhvf1 0x76 sensor
AMC Hot Swap      | 0x0         | discrete    | 0x0000| na         | na         |...
801F12F0B063      | 0x0         | discrete    | 0x0000| na         | na         |...
STAMP Temp        | 35.500      | degrees C   | ok    | 0.000      | 3.000      |...
AMC MP 3V3        | 3.376       | Volts       | ok    | 2.800      | 2.968      |...
AMC PP 12V        | 12.440      | Volts       | ok    | 10.160     | 10.760     |...
I_RTM MP 3V3      | 0.000       | Amps        | ok    | 0.000      | 0.000      |...
...
```

4 MMC console

4.1 Local serial console

The DMMC-STAMP's debug USB connector exposes two virtual serial ports:

- primary: MMC console @ 115200 8N1
- secondary: FPGA/SoC console (either `FPGA1_RXD/TXD` on DMMC-STAMP or `FPGA2_RXD` ↪ `/TXD` - the multiplexer can be set with `fpu` command, see below)

```
$ picocom -b 115200 /dev/ttyUSB0
picocom v3.1

port is        : /dev/ttyUSB0
flowcontrol    : none
baudrate is    : 115200
parity is      : none
databits are   : 8
stopbits are   : 1
escape is      : C-a
local echo is  : no
noinit is      : no
noreset is     : no
hangup is      : no
nolock is      : no
send_cmd is    : sz -vv
receive_cmd is : rz -vv -E
imap is        :
omap is        :
emap is        : crcrlf,delbs,
logfile is     : none
initstring     : none
exit_after is  : not set
exit is        : no

Type [C-a] [C-h] to see available commands
Terminal ready
DAMC-FMC2ZUP@0x76 MMC>
```

4.2 Remote console (mmcterm)

When there is no USB connection to the debug port, the console can be opened remotely using `mmcterm`. The Python based tool uses "Serial over IPMB" which is a non-standard DESY protocol, based on custom IPMI commands (not to be confused with IPMI SOL / Serial over LAN).
`mmcterm` is available on GitHub and PyPI.

```
$ mmcterm --help
usage: mmcterm [-h] [-v] [-c CHANNEL] [-t INTERVAL] [-l] [-d] [-i] [-m
    ↪ MAX_PKT_SIZE] mch_addr mmc_addr

DESY MMC Serial over IPMB console

positional arguments:
  mch_addr              IP address or hostname of MCH
  mmc_addr              IPMB-L address of MMC

optional arguments:
  -h, --help            show this help message and exit
  -v, --version         show program's version number and exit
  -c CHANNEL, --channel CHANNEL
                        console channel (default 0)
  -t INTERVAL, --interval INTERVAL
                        polling interval in ms (default 10)
  -l, --list            list available channels
  -d, --debug           pyipmi debug mode
  -i, --ipmitool        make pyipmi use ipmitool instead of native rmcp
  -m MAX_PKT_SIZE, --max-pkt-size MAX_PKT_SIZE
                        max IPMB packet size to use (Higher numbers give better
                            ↪ performance, but can break depending on MCH model)
```

4.2.1 mmcterm channels

Use `-l` to query the available channels:

```
$ mmcterm mskmchhvf1.tech.lab 0x76 -l
channel 0: MMC Console
channel 1: ZUP Console
```

We see that the DAMC-FMC2ZUP MMC reports two channels: 0 for the MMC console and 1 for the console of the payload FPGA (Zynq Ultrascale+).
To open the MMC console:

```
$ mmcterm mskmchhvf1.tech.lab 0x76 -c 0
Press Ctrl-x to exit
DAMC-FMC2ZUP@0x76 MMC>
```

5 Basic MMC console control

5.1 `?`, `h`, `help` - Show command list

Shows all available console commands and their arguments.

5.2 `vb` - Get/set verbosity

The higher the verbosity level, the more log messages get printed on the console.

Name	Number	Comment
ERR	1	
WARN	2	
INFO	3	
VERB	4	
DBG	5	Also shows names of received / sent IPMI packets in realtime
IPMI_RAW	6	Also shows raw hex dump of IPMI traffic

Example: `vb 5` - set verbosity to `DBG`

5.3 `tm` - Get/set terminal mode

Name	Description
smart	Assume “smart” (VT100-compatible) terminal w/ color & line editing support
dumb	Assume “dumb” terminal (text only, no colors, no line editing)
auto	Try to auto-detect terminal type

Example: `tm auto` - set terminal mode to auto-detect

6 MMC diagnostic & housekeeping commands

6.1 `r`, `v`, `s` - Reset, Version, Status

Command	Description
r	Reset MMC
v	Show MMC firmware version, hardware revision & UID
s	Show MMC status (mode, handle, uptime, LEDs, sensors, power, . . .)

6.2 `fru` - Dump FRU information

Command	Argument	Description
`fru`		Dump all FRUs
`fru`	`0`	Dump MMC FRU
`fru`	`1`	Dump RTM FRU (if applicable)
`fru`	`2`	Dump FMC1 FRU (if applicable)
`fru`	`3`	Dump FMC2 FRU (if applicable)

Example: `fru 0` - dump MMC FRU

6.3 `i2cd` - Detect I2C peripherals

Command	Argument	Description
`i2cd`	Bus name	Detect I2C peripherals

Example: `i2cd sens` - detect all peripherals on the sensor bus

6.4 `i2cget`, `i2cset` - Get/set I2C registers

Command	Argument	Description
`i2cget`	Bus, addr, reg	Read I2C register(s)
`i2cset`	Bus, addr, reg, data	Write I2C register(s)

Example: `i2cget sens 51 0 10` - dump first 10 bytes of MMC EEPROM at `0x51` on the sensor bus

6.5 `eefd` - Set EEPROM factory defaults

Many commands (like `tm` or `vb`) will save configuration data to non-volatile storage. `eefd` will reset the whole DMMC-STAMP configuration to default settings.

7 HPM update

The DMMC-STAMP supports the PICMG HPM.1 standard to allow in-application updates of AMC components over IPMI.

7.1 HPM components

Following HPM components are available on a DMMC-STAMP based AMC board:

- 0: MMC firmware
- 1: MMC bootloader
- 2..n: Payload components, such as FPGA configuration flashes (application-specific)

The HPM file format (`.hpm`) wraps a raw update file (e.g. `.bit` or `.bin`) into a container with metadata (file IDs, checksums etc.) for safety. The `.hpm` file also encodes the IANA board ID and the component index (from the table above) to make sure the file is not programmed into a wrong board or into a wrong component. The most important properties of a HPM file are:

Name	Description
Manufacturer ID	IANA manufacturer ID (hex, 6 bytes)
Product ID	IANA product ID (hex, 4 bytes)
Component	Component ID (see table above)
Version	Major.minor version of update file
Aux. version	Auxiliary version information (hex, 4 bytes)

7.2 bin2hpm

`bin2hpm` is a tool to build a HPM image to be used for the in-application upgrade. It also supports RLE compression (useful for FPGA bitstreams). It is available on GitHub and PyPI.

- `pip3 install bin2hpm`

```
bin2hpm [-h] [--version] [-o OUTFILE] [-v FILE_VERSION] [-a AUXILLARY]
        [-c COMPONENT] [-d DEVICE] [-m MANUFACTURER] [-p PRODUCT] [-r]
        [-s DESCRIPTION] [-b | -n] infile
```

Most important options:
`-m`/`-p` manuf./prod. ID, `-c` component, `-v`/`-a` major/minor/aux. version (see table above)

7.3 Update of the DMMC-STAMP firmware

Show currently installed versions: `hpm check`

```
$ ipmbtool mskmchhvf1 0x76 hpm check

PICMG HPM.1 Upgrade Agent 1.0.9:

-------Target Information-------
Device Id          : 0x0
Device Revision    : 0x80
Product Id         : 0x200b
Manufacturer Id    : 0x053f (Unknown (0x53F))


---------------------------------------------------------------------------
|ID  | Name          |                     Versions                        |
|    |               |     Active       |     Backup       |     Deferred  |
---------------------------------------------------------------------------
|   0|FMC2ZUP-MMC   |   1.19 00000000 | ---.-- -------- | ---.-- -------- |
|   1|MMC_BOOTLDR   |   1.19 00000000 | ---.-- -------- | ---.-- -------- |
|   2|ZUP QSPI      |   0.00 00000000 | ---.-- -------- | ---.-- -------- |
|   3|ZUP QSPI2     |   0.00 00000000 | ---.-- -------- | ---.-- -------- |
|   4|S7 SPI        |   0.00 00000000 | ---.-- -------- | ---.-- -------- |
|   5|S7 SPI2       |   0.00 00000000 | ---.-- -------- | ---.-- -------- |
---------------------------------------------------------------------------
(*) Component requires Payload Cold Reset
```

Perform upgrade: `hpm upgrade <hpm_file>`

- The component (MMC, FPGA, ...) to upgrade is determined from the `component` parameter of the HPM file.
- The upgrade is only performed if the manufacturer / product ID matches and major/minor or aux. version is different.
- These checks can be overridden by using the `force` parameter: `hpm upgrade <hpm_file> force`

```
$ ipmbtool mskmchhvf1 0x76 hpm upgrade damc-fmc2zup_V2.00.hpm

PICMG HPM.1 Upgrade Agent 1.0.9:

Validating firmware image integrity...OK
Performing preparation stage...OK

Performing upgrade stage:

-------------------------------------------------------------------------------
|ID  | Name          |                     Versions                    | %  |
|    |               |     Active       |     Backup       |      File       |    |
|----|---------------|------------------|------------------|-----------------|----|
|   0|FMC2ZUP-MMC    |   1.19 00000000  | ---.-- --------  |   2.00 00000000 |100%|
|    |Upload Time: 02:05                | Image Size:  186580 bytes               |
-------------------------------------------------------------------------------
(*) Component requires Payload Cold Reset

Firmware upgrade procedure successful
```

7.4 Update of DMMC-STAMP internal components

These commands are relevant when bringing up a new board in post-production or when developing a new DMMC-STAMP board integration with the SDK.

Command	Argument	Description
`pc`		Toggle CPLD programming / JTAG forwarding
`cfu`		CPLD force update
`pmc`	`write`	Write power manager configuration
`pmc`	`verify`	Verify power manager configuration

7.5 Update of payload (FPGA flash memories)

- Use `bin2hpm` (see above) to build a HPM file from your update file
- `.bit` files (for Xilinx FPGAs) and raw binary files (e.g. for FSBLs) are supported
- FPGA image size can be significantly reduced with RLE compression enabled (`-r` option)
- warning: HPM update can take very long for large files (more than 10 minutes per MB)

8 Xmodem update (fallback option)

Update over Xmodem (USB debug port) can be used when HPM is not available for some reason.
Never try to download a `.hpm` file over Xmodem - use the raw binary file instead.

8.1 `xm` - Start Xmodem update

Command	Argument	Description
`xm`	`0`	Xmodem update of MMC
`xm`	`1`	Xmodem update of bootloader
`xm`	`2..n`	Xmodem update of payload components

9 Standard payload management commands

9.1 `pu`, `pd` - Payload power up / down

Command	Description
`pu`	Payload power up
`pd`	Payload power down

These commands can be used for remote-controlling the payload power without physical access to the AMC handle. For `pu` to work, the 12V payload power needs to be enabled. Use `fru_start` / `shutdown` commands on the MCH console to enable/disable 12V towards the AMC.

9.2 `ppf` - Payload power fail policy

When powering up the AMC payload, errors can occur (e.g. when the power manager fails to establish voltages, or configuration of clocks fails). It is possible to change the behavior in such cases, especially for board development and bring-up.

Command	Argument	Description
`ppf`	`stop`	In case of failure, stop immediately and go into error mode
`ppf`	`retry`	In case of failure, retry three times before going into error mode
`ppf`	`ignore`	Ignore any failure and move on to AMC power good mode

9.3 `sj` - JTAG multiplexing

The JTAG chain can be flexibly routed between different sources and targets.

```
sj [con|bp|raw] [fpga(1|2|12)|rtm|fmc(1|2)]
```

JTAG source	Description
`con`	Connector on PCB
`bp`	MicroTCA Backplane
`raw`	Raw EEPROM value (only for dev.)

JTAG target	Description
`fpga1`	Main FPGA
`fpga2`	Secondary FPGA
`fpga12`	Both FPGAs
`rtm`	RTM
`fmc1`	First FMC
`fmc2`	Second FMC

Example: `sj bp fpga1` - route JTAG from the MicroTCA backplane to the main FPGA/SoC.

9.4 `fd` - Flash detect

For board implementations that have SPI configuration flashes, the `fd` command can be used to verify a working SPI connection from the DMMC-STAMP to the flash chips. The command takes the number of the flash chip, starting with 0, as argument.

Command	Argument	Description
`fd`	`0..n`	Detect flash chip

9.5 `fpu` - FPGA UART select

For board implementations that use both `FPGA1_UART` and `FPGA2_UART`, the `fpu` command selects the index of the UART that is forwarded to the USB debug connector.

Command	Argument	Description
`fpu`	`1`	`FPGA1_UART` is forwarded to USB debug
`fpu`	`2`	`FPGA2_UART` is forwarded to USB debug

10 RTM management commands

10.1 `st` - Get/set RTM temp. sensor mask

RTMs can have up to four MAX6626 temperature sensors at different I2C addresses. The sensor mask is an OR combination of flags that determines which sensors are used.

Flag	Description
1	RTM Temp.1 at `0x48`
2	RTM Temp.2 at `0x49`

Flag	Description
4	RTM Temp.3 at `0x4a`
8	RTM Temp.4 at `0x4b`

Example: `st 3` - enable temperature sensors 1 and 2 at 0x48 and 0x49 (flags as bit array, i.e. 3=2|1)

10.2 `rte` - Get/set RTM e-keying policy

According to the MicroTCA 4.1 standard, RTMs and AMCs must have a “Zone 3 Compatibility Record” in their FRU. The MMC has to perform “e-keying” in the sense of matching the Zone 3 Compatibility Record between the AMC and RTM, and only power up the RTM if this e-keying succeeds. Since in the reality not all vendor RTMs implement the “Zone 3 Compatibility Record”, the MMC allows disabling the RTM e-keying.

Command	Argument	Description
`rte`	`enable`	Enable e-keying, power RTM only if Zone 3 record matches
`rte`	`override`	Disable e-keying, power RTM regardless of Zone 3 record

Example: `rte override` - Disable RTM e-keying

10.3 `rtp` - Get/set RTM Power Good polarity

The RTM Power Good signal (bit 4 on the RTM port expander) is declared active-low in the MicroTCA 4.1 draft, but active-high in the final MicroTCA 4.1 release. The MMC assumes active-high as per the released specification, however there are still some RTMs that implement it active-low.

Command	Argument	Description
`rtp`	`high`	Assume RTM PG active high polarity
`rtp`	`low`	Assume RTM PG active low polarity
`rtp`	`auto`	Use active low only if RTM matches against a list of known “legacy” boards

Example: `rtp high` - Assume active high polarity for RTM PG

10.4 `rto` - I_RTM PP 12V calibration

The DMMC-STAMP exposes the IPMI sensor `I_RTM PP 12V` measuring the current draw on the 12V rail towards the RTM. To account for tolerances in the parts of the measurement circuit, a calibration is conducted during post-production stage and offset/slope coefficients are saved in non-volatile storage.
For DMMC-STAMPs that did not go through this calibration step, it is possible to determine the offset coefficient later in application. For this to work, the AMC payload has to be powered up with no RTM mounted.

Command	Argument	Description
`rto`		Show I_RTM calibration coefficients
`rto`	`calibrate`	Conduct I_RTM offset calibration

11 FMC management commands

11.1 `fma` - Get/set FMC EEPROM address width

According to the VITA FMC standard, FMCs must keep their FRU information in a 2kbit (256 byte) EEPROM with an 8-bit address width. However, many FMCs use bigger EEPROMs with 16-bit address width instead. The MMC implements the `fma` command to support these "non-standard" FMC EEPROMs.

Command	Arg1	Arg2	Description
`fma`	FMC no.	`8`	Assume "standard" (8-bit address) FMC EEPROM
`fma`	FMC no.	`16`	Assume "non-standard" (16-bit address) FMC EEPROM
`fma`	FMC no.	`auto`	Auto-detect FMC EEPROM address width

Example: `fma 1 auto` - set FMC1 EEPROM address width to auto-detect

11.2 `fmv` - Get/set FMC V_{ADJ} voltage level

The allowed range of the V_{ADJ} voltage level for a FMC is usually provided in the FMC FRU. The MMC will try to find a suitable V_{ADJ} voltage that's in the voltage range of all mounted FMCs. Alternatively it can be set manually.

Command	Argument	Description
Command	0.95..1.9	Set voltage for V_{ADJ} manually
Command	`auto`	Determine V_{ADJ} level from FMC FRU

Example: `fmv 1.2` - set FMC V_{ADJ} to 1.2 volts

12 Board-specific payload management, example DAMC-FMC2ZUP

12.1 `bz` - Get/set ZUP boot mode

Command	Argument	Description
`bz`	`jtag`	Make ZUP boot from JTAG
`bz`	`qspi`	Make ZUP boot from primary QSPI flash
`bz`	`qspi2`	Make ZUP boot from secondary QSPI flash
`bz`	`sd`	Make ZUP boot from SD card
`bz`	`jtag`	Make ZUP boot from PJTAG

Command	Argument	Description
`bz`	`raw`	Set raw boot mode value

Example: `bz sd` - set ZUP boot mode to SD card
Note: `bz jtag` can be useful if one needs payload power active, but FPGA inactive.

12.2 `b7` - Get/set Spartan-7 boot mode

Command	Argument	Description
`b7`	`jtag`	Make S7 boot from JTAG
`b7`	`spi`	Make S7 boot from primary SPI flash
`b7`	`spi2`	Make S7 boot from secondary SPI flash

Example: `b7 spi` - set S7 boot mode to primary flash

12.3 `rz` - Re-configure ZUP

This command asserts `PS_POR` to trigger a reconfiguration of the ZUP.

12.4 `r7` - Re-configure Spartan-7

This command asserts `PROG_B` to trigger a reconfiguration of the Spartan-7.

12.5 `vc` - Set ZUP VCC_Core

VCC_Core can be set to a higher voltage if certain performance features of the ZUP are needed; else it can be set to lower voltage to reduce heat.

Command	Argument	Description
`vc`	`low`	Set ZUP VCC_Core to 0.72 volts
`vc`	`high`	Set ZUP VCC_Core to 0.85 volts

Example: `vc low` - set VCC_Core to 0.72 volts