# SUPERNOVA™ Firmware Reference Manual Rev 4.4

750 Naples Street, San Francisco, CA, 94112 | http://www.pumpkininc.com

Specifications subject to change without notice.

# TRADEMARKS

The following are Pumpkin trademarks. All other names are the property of their respective owners.

- Pumpkin™ and the Pumpkin logo
- Salvo™ and the Salvo logo
- CubeSat Kit™ and the CubeSat Kit logo

# DISCLAIMER

PUMPKIN RESERVES THE RIGHT TO MAKE ANY CHANGES WITHOUT FURTHER NOTICE TO ANY PRODUCTS HEREIN TO CORRECT ERRORS AND IMPROVE RELIABILITY, FUNCTION, APPEARANCE OR DESIGN. PUMPKIN DOES NOT ASSUME ANY LIABILITY ARISING OUT OF THE APPLICATION OR USE OF ANY PRODUCT OR CIRCUIT DESCRIBED HEREIN; NEITHER DOES IT CONVEY ANY LICENSE UNDER ITS PATENT RIGHTS, NOR THE RIGHTS OF OTHERS.

750 Naples Street
San Francisco, CA 94112 USA
tel: (415) 584-6360
fax: (415) 585-7948

http://www.pumpkinspace.com
info@pumpkinspace.com

http://cubesatkit.com
info@cubesatkit.com

## Overview

A SUPERNOVA™ nanosatellite incorporates a variety of modules, including:

- Payload Interface Module (PIM)
- Bus Interface Module (BIM)
- Motherboard Module (MBM)
- GPS Receiver Module (GPSRM)
- Electrical Power System Module (EPSM)

Battery Module (BM)

- Battery Switch Module (BSM)
- Solar Interface Module (SIM)
- Desktop Cubesat Power System (DCPS)
- Deployable Articulated Solar Array (DASA)

Each module uses a Supervisor Microcontroller Unit (SupMCU) to handle commands and telemetry, and to implement module-specific functions. A SupMCU's firmware is comprised of generic module functionality and module-specific functionality. Every module is an I2C slave device and is commanded via a human-readable readable command protocol over I2C; the module's SupMCU validates and interprets the commands and makes the desired telemetry available to SUPERNOVA's Command and Data Handling C&DH processor, an I2C master device.

## Bus Interface Module

The BIM SupMCU is in charge of interfacing to all components on the bus that are remote (not a part of standard CSK-compatible module). The current version allows for serial and power control of connected devices. The BIM also implements three serial UART interfaces on the main PCB.

In addition to this, BIM also runs the Analog to Digital Conversion (ADC) task required to read the temperature from the temperature sensors embedded on the solar panels if fit. It is also capable of running a task that is in charge of energizing the TiNi pin puller connected to J12 to deploy solar panels on a 6U or larger craft. See BIM data sheet for more information.

## GPS Receiver Module

The GPSRM provides communications interface to the GPS Novatel OEM615/719 receiver module. It controls its power supply and enables communication with the module by sending commands in predetermined formats specified in the Novatel OEM615/719 Reference Manual[1].
The GPSRM currently supports commands that enable data logging. GPSRM1 also runs Vinti7 and SGP4 Orbit propagators that allow an estimate of satellite orbit over a given time delta, position, velocity (for Vinti7) or given two line element (TLE) for SGP4. See GPSRM data sheet for more information.

[1] http://www.novatel.com/assets/Documents/Manuals/om-20000129.pdf

## Payload and Solar Interface Modules

The Payload Interface Module (PIM) supports one method of power switching and Ethernet for payloads. It is equipped with four voltage configurable power ports (determined at manufacture time), one 8-pin Ethernet port and three 4-pin Ethernet ports. A magnetometer is fitted for a MAI-400 ADCS if present on the bus. See the PIM data sheet for more information.

The Solar Interface Module (SIM) connects coarse sun sensors for the MAI-400 ADCS, power and temperature sensors for the solar panels along with firing the panel release modules for a MISC-3 CubeSat. See the SIM data sheet for more information.

## Desktop Cubesat Power System

The Desktop Cubesat Power System (DCPS) provides all standard power levels for a Cubesat from a single external DC supply. This module is used in place of an EPS for bench testing of a cubesat, since it emulates a similar SCPI command set. See the DCPS data sheet for more information.

## Electrical Power System Module

The Electrical Power System Module (EPSM) manages power distribution for high-power nano satellites. The EPSM provides 12V, 5V, 3.3V and AUX output rails. It also provides six solar panel input SAIs, and two battery input/output blocks. See EPSM data sheet for more information.

## Deployable Articulated Solar Array

The Deployable Articulated Solar Array (DASA) provides a platform to articulate up to six independent solar cell strings with 400 degrees of rotation per side. The DASA is equipped a TiNi pin puller to deploy the solar arrays, with a dedicated command set to enable, arm and fire (energize) the pin puller.

# SCPI Command Format

SUPERNOVA modules are commanded via a subset of the Standard Commands for Programmable Instruments (SCPI) protocol. In this document the UPPER CASE letters of a command are required and the lower case letters are optional. Possible options for a given slot in a command are separated with a '|' character. Below are samples of how commands are described in this document:

```
DEVice:TASK <Required param opt 1 | required param opt 2 | required param opt 3>,[Optional param 1 | optional param 2]
DEVice:SUBsystem:TASK <Required parameter>,[Optional parameter]
```

Each SCPI command must end with a <CR> (0x0A) character. Note that SCPI commands are human-readable; a protocol analyzer that is snooping the I2C bus will reveal easy-to-read commands being sent to the various SupMCUs in a SUPERNOVA system.

Commands for each module are broken out by module while commands that apply to all modules are broken out in the "SupMCU SCPI Commands" section.

## SupMCU SCPI Commands

These commands apply to every module in the Pumpkin SUPERNOVA ecosystem. These commands include:

- Resetting the Supervisor Microcontroller (SupMCU)
- Controlling the SupMCU indicator LED function
- Controlling SupMCU Self-test functions
- Turning on or off the SupMCU clock output (output only available on select modules).
- Manipulating SupMCU clock oscillator speed.
- Querying SupMCU Telemetry

| Commands | Explanation | Examples |
|---|---|---|
| SUPervisor:LED <OFF \| ON \| FLASH \| I2C> | Controls status LED behavior. The states are:<br>• OFF: The LED is turned off<br>• ON: The LED stays illuminated.<br>• FLASH: The default state of the LED, flashing at 2Hz<br>• I2C: LED illuminates on I2C activity for the SupMCU module.<br>Note that a module error *must be cleared* (via SUP:RES ERR) before the state of the LED can be changed | Turn off the status LED:<br>*SUP:LED OFF*<br>The status LED follows I$^2$C activity:<br>*SUP:LED I2C* |
| SUPervisor:RESet <NOW \| ERRor> | Resets the SupMCU or reset the SupMCU's error flag. Choices are:<br>• NOW: Reset the SupMCU module.<br>• ERRor: Clears any error on the SupMCU, also causing the statusto return to previous state before the erroneous command. | Reset SupMCU Module:<br>*SUP:RES NOW*<br>Reset the error status flag:<br>*SUP:RES ERR* |
| SUPervisor:CLOCk <OFF \| ON, <N>> | Sets the SupMCU's reference clock output off, or on with a divider applied.[2] The divider (N) is a power of 2, with a range of [0, 15], and is only specified when turning the clock output on, e.g., SUP:CLOC ON,4 | Turn on reference clock output with base clock divided by 2:<br>*SUP:CLOCK ON,1*<br>Turn off reference clock output:<br>*SUP:CLOCK OFF* |

[2] See the module's datasheet for the clock source and operating frequency available for use on the module's oscillator out function.

| Commands | Explanation | Examples |
|---|---|---|
| SUPervisor:SELFtest <STOP \| START \| REStart \| ENABLE,[N1,[N2,...[N6]]] \| DISABLE,[N1,[N2,...[N6]]]> | Controls SupMCU self tests[3], choices are listed below:START: Starts or resumes the enabled self-tests.<br>• STOP: Stops self-tests<br>• REStart: Restarts self-tests.<br>• ENABLE: Enables the given test numbers N1-N6.<br>• DISABLE: Disables the given test numbers N1-N6.<br>Note: As of document writing, *it has been observed that tests may report false errors. Their usage is not recommended at this time.*<br>Note: *Tests 1-6 are ENABLED by default.* | Enable Self-tests 1,3,5:<br>*SUP:SELF ENABLE,1,3,5*<br>Disable Self-tests 1,5:<br>*SUP:SELF DISABLE,1,5*<br>Start Self-test task:<br>*SUP:SELF START* |
| SUPervisor:I2C:RESet <NOW> | NOW: Resets the I$^{2}$C bus, useful if problems are encountered with I2C communications | Resetting I$^{2}$C task:<br>*SUP:I2C:RES NOW* |
| SUPervisor:DEBug <ENABle \| DISABle>, [debug bitfield] | Enables / disables debug messages via the user debug terminal. | Enable debug messages:<br>*SUP:DEB ENAB* |
| SUPervisor:TELemetry? <N>,[DATA \| ASCII \| NAME \| LENGTH] | Request telemetry for a specific field N in the given format. The formats are:<br>• DATA: Return telemetry in binary representation. *Not available over user debug terminal.*<br>• ASCII: Return telemetry in ASCII representation.<br>• NAME: The name of the telemetry item e.g. *SupMCU Uptime*<br>• LENGTH: Length (in bytes) of telemetry in its binary representation.<br>Note: I*f no format representation is specified, DATA is used as the default.*<br>*See SupMCU Telemetry section for more detail.* | Query SupMCU for its Firmware version in ASCII:<br>*SUP:TEL? 0,ASCII*<br>Query SupMCU for the name of the 3rd telemetry item:<br>*SUP:TEL? 2,NAME* |
| SUPervisor:OSCilator <INTernal>,<pre>,<div>,<post> | Configure the internal oscillator based on the external clock. A Prescaler, Divider and Postscaler satisfying the internal requirements of the processor are required. Clock is calculated by: (7.23 Mhz*div)/(pre*post) | Set clock OSC to 1.8075 Mhz:<br>*SUP:OSC INT,2,1,2* |

[3] Note there are the Self-test libraries provided by Microchip, of which only test subsets 1-6 are available. For more information visit https://www.microchip.com/SWLibraryWeb/product.aspx?product=16-bit%20CPU%20Self-test%20Library

| Commands | Explanation | Examples |
|---|---|---|
| SUPervisor:CALibration? LINEAR,<idx>,[ASCII \| NAME \| DATA] | Request calibration values for a specific calibration index \`idx\` in a given format. The formats are:<br>• DATA: Return calibration values in binary format. *Not available over user debug terminal.*<br>• ASCII: Return the calibration values in ASCII representation.<br>• NAME: The name of the calibration index. *e.g. CTMU*<br>Note: *The \`LINEAR\` argument is required. This is for forward compatibility with future calibration value types (e.g. quadratic).* | Query CTMU calibration values in ASCII:<br>SUP:CAL? LINEAR, 0, ASCII |
| SUPervisor:CALibration LINEAR,<idx>,<OFFSET \| SCALE_FACTOR>,<value> | Set calibration \`OFFSET\` or \`SCALE_FACTOR\` values for a specific calibration \`idx\` in the Supervisor MCU calibration values. The calibration values are used internally when calculating the telemetry values for the user, applying a linear correction factor (E.g. \`*final = scale_factor*initial + offset*\`<br>Possible value sets are:<br>• OFFSET: Set the offset of the calibration factor. <value> is a number in the range [-32768, 32767]<br>• SCALE_FACTOR: Set the scale factor of the calibration factor. <value> is a floating-point number from [-16.0, 16.0].<br>Note: *The \`LINEAR\` argument is required. This is for forward compatibility with future calibration value types (e.g. quadratic).*<br>Note: *These values directly impact the output of telemetry from the SupMCU modules.*<br>Note: *Calibration values must be saved via \`SUP:NVM\` commands.*<br>Note: *See SupMCU Calibration for more details.* | Set the SCALE_FACTOR to .975 and offset to -20 for the CTMU telemetry, save to NVM:<br>*SUP:CAL LINEAR,0,SCALE_FACTOR,0.975*<br>*SUP:CAL LINEAR,0,OFFSET,-20*<br>*SUP:NVM UNLOCK,12345*<br>*SUP:NVM WRITE,1* |
| SUPervisor:NVM <field>,<value><br><br>Fields: UNLOCK, WRITE, CRC, SERIAL, I2C, OSCTUN, RESET_TO, DEBUG | Controls access to and writes parameters to the non-volatile memory. Order of operations when changing NVM is: UNLOCK, <change value>, WRITE.<br>• UNLOCK: unlocks the non-volatile memory (must be done first). Value contain a key consisting of the device serial number + 12345<br>• WRITE: Writes all set NVM values to non-volatile memory, value = 1 | Change the unit's serial number to 123 (assuming serial number is currently 0):<br>*SUP:NVM UNLOCK,12345*<br>*SUP:NVM SERIAL,123*<br>*SUP:NVM WRITE,1* |

| Commands | Explanation | Examples |
|---|---|---|
| | • CRC: Sets the CRC value to value<br>• SERIAL: Sets the device serial number to value (decimal). Note: *Changing this value changes the key to unlock NVM memory on SupMCU Modules. Add 12345 to serial number for the new key.*<br>• I2C: Changes the 7 bit I2C address to value (hex or decimal)<br>• OSCTUN: Sets the oscillator tuning parameter to value.<br>• RESET_TO: Changes the time before -RESET is allowed to reset a SupMCU Module. Value is in seconds and is either 0 (-RESET **never** affects the module) or greater than or equal to 15 seconds. Note: *This is only used on BM2, EPSM, and DCPS currently.*<br>• DEBUG: Sets the debug mask to be set on boot. The current debug mask is also set to the mask value passed. | |

## BIM SCPI Commands

The BIM command set is responsible for:

- Controlling the TiNi Pin Puller for solar panel deployment.
- Switching the three UART ports on the top of the BIM.
- Switching the BIM Ethernet and EEProm daughter board (*if BIM is equipped with daughter board.*)
- Powering the IMU on the BIM (*if BIM is equipped with IMU.*)
- Powering the temperature sensors on the BIM.
- Querying BIM specific telemetry.
- Setting Non-volatile memory values specific to the BIM.

| Commands | Explanation | Examples |
|---|---|---|
| BIM:TINI <DISAble \| ENABle \| UNARm \| ARM \| FIRE, <N>> | Controls the TiNi Pin Puller. When energizing the pin puller, the order of operations is ENABLE → ARM → FIRE, N. The different parameter strings mean:<br>• DISAble: Disables the TiNi pin puller, effectively resetting the sequence.<br>• ENABle: Enables the TiNi pin puller, first step in panel deployment.<br>• ARM: Arms the pin puller, second step in panel deployment.<br>• UNARm: Disables the SupMCU task responsible for energizing the Pin Puller. *Does not disable the TiNi.*<br>• FIRE, Seconds: Energizes the Pin Puller, releasing the deployment mechanism of the solar panels. Will keep energized for N seconds where N is in the range [1, 30].<br>Note: *Any command done out of order will reset the sequence. E.g. if BIM:TINI ENAB → BIM:TINI FIRE,30 is done, then BIM:TINI ENAB must be sent again.* | Energize the TiNi pin puller for 25 seconds.<br>*BIM:TINI ENAB*<br>*BIM:TINI ARM*<br>*BIM:TINI FIRE,25*<br><br>Reset the TiNi pin puller sequence:<br>*BIM:TINI DISAble* |
| BIM:UART:POWer <N>,<OFF \| ON> | Turns power ON or OFF to a UART channel on the BIM. Specify channel number (N) 1, 2 or 3<br>E.g., BIM:UART:POW 1, ON. | Supply power BIM port #2<br>*BIM:UART:POW 2,ON*<br><br>Depower BIM port #3 |

| Commands | Explanation | Examples |
|---|---|---|
|  | Note: *The UART RX and TX are still electrically connected to the corresponding CSK UARTs when the port is OFF.* | *BIM:UART:POW 3,OFF* |
| BIM:ETH:POWer <OFF \| ON> | Controls Ethernet switch power *if the BIM has an Ethernet daughter board.* | Power the BIM Ethernet ON<br>*BIM:ETH:POW ON* |
| BIM:ETH:RESet <OFF \| ON> | Powers the RESET signal to the BIM Ethernet daughhterboard *if equipped.*<br>Note: *Does not power off RESET automatically.* | Turn off the RESET signal<br>*BIM:ETH:RES OFF* |
| BIM:ETH:MODE <N> | Put the Ethernet switch in the specified mode (N) 0-3 or OFF. E.g., BIM:ETH:MODE OFF. | Put BIM Ethernet in Mode 2<br>*BIM:ETH:MODE 2* |
| BIM:ETH:I2C <DISable \| ENABle> | Dis\|Enables the Ethernet $I^2C$ interface *if BIM is equipped with daughter board.* See data sheet for functionality of $I^2C$ on daughter board. | Disable $I^2C$ interface on BIM Daughter board.<br>*BIM:ETH:I2C DIS* |
| BIM:EEprom:I2C <DISable \| ENABle> | Dis\|Enables the EEPROM I2C interface *if BIM is equipped with daughter board.* See data sheet for functionality of EEProm. | Enable $I^2C$ link to EEProm on BIM daughter board<br>*BIM:EE:I2C ENABLE* |
| BIM:TEMP:POWer <OFF \| ON> | Turns power to BIM temperature sensors ON or OFF. Must be ON **before** querying temperature sensor telemetry. | Turn on BIM temperature sensors<br>*BIM:TEMP:POW ON* |
| BIM:IMU:POWer <OFF \| ON> | Turns power to IMU ON or OFF. Must be ON **before** querying IMU telemetry. | Turn IMU on.<br>*BIM:IMU:POW ON* |
| BIM:CALibration? LINEAR,<idx>,[ASCII \| NAME \| DATA] | Request calibration values for a specific calibration index `idx` in a given format. The formats are:<br>• DATA: Return calibration values in binary format. *Not available over user debug terminal.*<br>• ASCII: Return the calibration values in ASCII representation.<br>• NAME: The name of the calibration index. *e.g. CTMU*<br>Note: *The `LINEAR` argument is required. This is for forward compatibility with future calibration value types (e.g. quadratic).* | Query `temp sns. 1` calibration values in ASCII:<br>BIM:CAL? LINEAR, 0, ASCII |
| BIM:CALibration LINEAR,<idx>,<OFFSET \| SCALE_FACTOR>,<value> | Set calibration `OFFSET` or `SCALE_FACTOR` values for a specific calibration `idx` in the BIM calibration values. The calibration values are used internally when calculating the telemetry values for the user, applying a linear correction factor (E.g. `*final = scale_factor*initial + offset*` Possible value sets are: | Set the SCALE_FACTOR to .975 and offset to -20 for the `temp sns. 1` calibration value, save to NVM:<br>*BIM:CAL LINEAR,0,SCALE_FACTOR,0.975*<br>*BIM:CAL LINEAR,0,OFFSET,-20*<br>*SUP:NVM UNLOCK,12345* |

| Commands | Explanation | Examples |
|---|---|---|
| | - OFFSET: Set the offset of the calibration factor. <value> is a number in the range [-32768, 32767]<br>- SCALE_FACTOR: Set the scale factor of the calibration factor. <value> is a floating-point number from [-16.0, 16.0].<br>Note: *The `LINEAR` argument is required. This is for forward compatibility with future calibration value types (e.g. quadratic).*<br>Note: *These values directly impact the output of telemetry from the SupMCU modules.*<br>Note: *Calibration values must be saved via `SUP:NVM` commands.*<br>Note: *See BIM Calibration for more details.* | *SUP:NVM WRITE,1* |
| BIM:NVM <UNLOCK,<value> \| WRITE,<1>> | Controls access to and writes parameters to the non-volatile memory. Order of operations when changing NVM is: UNLOCK, <change value>, WRITE.<br>- UNLOCK: unlocks the non-volatile memory (must be done first). Value is a key consisting of: device serial number + 12345<br>- WRITE: Writes all set NVM values to non-volatile memory, value = 1.<br>Note: *The `UNLOCK` and `WRITE` subcommands provide the same functionality as the `SUP:NVM WRITE` and `SUP:NVM UNLOCK` subcommands.* | Unlock and write NVM memory (assuming serial number is 0.)<br>*BIM:NVM UNLOCK,12345*<br>*BIM:NVM WRITE,1* |
| BIM:WDT <FORce> | Forces the BIM WDT to cause a system reset.<br>Note: ***This command is currently not supported and won't function.*** | Force a bus reset via WDT on BIM:<br>*BIM:WDT FORCE* |
| BIM:EXTLED <R>,<G>,<B> | Sets the multicolor LED on the external interface board to a color defined by 8bit R,G and B values. | Set the BIM EXTLED to red<br>*BIM:EXTLED 255,0,0* |

## GPSRM SCPI Commands

The GPSRM Command set is responsible for:

- Controlling power to the OEM615/719.
- Controlling the RESET line to the OEM615/719.
- Propagating orbit via Vinti7/SGP4 orbit propagators.
- Setting UART communication channel for OEM615/719.
- Enabling or disabling UART communication for the OEM615/719.
- Enabling a subset of the logs available for the OEM615/719.
- Controlling power the BCT xACT ADCS

| Commands | Explanation | Examples |
|---|---|---|
| GPS:POWer <OFF \| ON> | Turns ON or OFF power to the OEM615 or OEM719 module. | Turn on power to the OEM719<br>*GPS:POW ON* |
| GPS:RESet <OFF \| ON> | Turns ON or OFF the RESET signal to the OEM615 or OEM719 module.<br>Note: *The reset signal is NOT turned off automatically, it must be turned off with a corresponding GPS:RES OFF command.* | Turn on the GPS reset signal<br>*GPS:RES ON* |
| GPS:PROPagate <VINTI7>, <t0 *in seconds*>,<t1 *in seconds*>, <Rx>, <Ry>, <Rz>, <Vx>, <Vy>, <Vz><br>**or**<br>GPS:PROPagate <SGP4>, <dt *in minutes*>,<TLE> | Propagate orbit using SGP4 (S) or Vinti7 (V) orbit propagators. The results of the orbit propagators can be seen via GPS:TEL? 2<br>Vinti7 works using:<br>• t0: An intial time in seconds (usually 0)<br>• t1: time of solution in seconds (time since t0).<br>• Rx, Ry, Rz: a position vector in kilometers Vx, Vy, Vz: velocity vector in kilometers<br>While SGP4 works using:<br>• dt: Time delta from time in TLE in minutes.<br>• TLE: Two-line element exactly 140 characters long. *The lines in a TLE are separated by space instead of newline.*<br>Note: *TLE's spacing is critical as the code parsing it expects elements to be at exact character locations.*<br>Note: *TLE's for current space vehicles can be found at* [*https://www.celestrak.com/NORAD/elements/*](https://www.celestrak.com/NORAD/elements/) | Propagate a spacecraft 120 minutes from now with a positional vector of (-18,982 Km, 25,047 Km, -173 Km) and velocity vector (2.96 Km/s, 0.33 Km/s 0.27 Km/s)<br>*GPS:PROP VINTI7, 0, 7200, -18982.0, -25047.0, -173.0, 2.96, 0.33, 0.27*<br><br>Propagate TELSTAR 19V orbit 30 minutes from TLE time with SGP4<br>*GPS:PROP SGP4,30,TELSTAR 19V 1 43562U 18059A 18228.55107419 -.00000304 00000-0 00000+0 0 9993 2 43562 0.0593 269.5551 0001312 309.6754 241.0290 1.00272513 473* |
| GPS:LED <SUP \| PASS \| GPS> | Controls the state of the SupMCU status light in several ways:<br>• SUP: Follows the state of the LED as given by the SUP:LED command<br>• PASS: The LED will remain ON if GPS passthrough is on, otherwise OFF. | Make the status light follow the power state of the OEM719<br>*GPS:LED GPS* |

| Commands | Explanation | Examples |
|---|---|---|
|  | • GPS: The LED will remain ON if the GPS is ON otherwise OFF. | Make the status light follow the pass through state of the OEM719<br>*GPS:LED PASS* |
| GPS:PASSthrough OFF \| ON | Controls the UART passthrough channel to the GPS receiver. For UART configuration, see the ASSY REV on the GPSRM data sheet for the particular module. | Turn OFF UART Passthrough<br>*GPS:PASS OFF* |
| GPS:LOG <OFF \| GGA \| GSV \| GSA \| RMC \| VTG \| ZDA \| BESTPOSA \| BESTPOSB \| BESTXYZA \| BESTXYZB>,[COMn] | Commands the OEM615/719 to log a subset of the available data from the receiver. An optional COMn (where n is 1 or 2) parameter can be given to have the GPS log to COM1 or COM2. If not present, COM1 is chosen. The log choices are:<br>• OFF/NONE: Turns off all logging from the OEM615/719 via NovaTel UNLOGALL command.<br>• GGA/GSV/GSA/RMC/VTG: Logs OEM615/719 data via "*LOG COMn GP<type> ONTIME 1*" NovaTel OEM command string (where type is log type given).<br>• ZDA/BESTPOSA/BESTPOSB/BESTXYZA/BESTXYZB: Logs OEM615/719 data via "*LOG COMn <type> ONTIME 1*" NovaTel OEM command string (where type is log type given).<br>Note: *More logs/command are available via the UART passthrough and commanding the OEM unit directly. For information about OEM commands see: https://docs.novatel.com/OEM7/Content/PDFs/OEM7_Commands_Logs_Manual.pdf* | Turn on logging BESTXYZA logs to COM2 (including powering GPS, enabling PASSthrough):<br>GPS:POW ON<br>GPS:PASS ON<br>*GPS:LOG BESTXYZA,COM2*<br>Turn on logging GPGGA string to COM1:<br>*GPS:LOG GGA* |
| GPS:BESTPOS <OFF \| BESTPOSA \| BESTXYZA> | Commands the OEM615/719 to log a subset of the available data from the receiver to COM2 port of the OEM615/719 unit. The log choices are:<br>• OFF: Turns off all logging from the OEM615/719 via NovaTel UNLOGALL command.<br>• BESTPOSA: Logs BESTPOS logs in ASCII format via "LOG COM2 BESTPOSA ONTIME 1" command string.<br>• BESTPOSB: Logs BESTPOS logs in binary format via "LOG COM2 BESTPOSB ONTIME 1" command string.<br>Note: ***This command is deprecated, use GPS:LOG instead.*** | Turn on BESTPOSA logging to COM2 port of OEM615/719:<br>*GPS:BESTPOS BESTPOSA* |
| GPS:TELemetry? <N>,[DATA \| ASCII \| NAME \| LENGTH] | Request telemetry for a specific field N in the given format. The formats are:<br>• DATA: Return telemetry in binary representation. *Not available over user debug terminal.* | Query the GPS Power status in binary representation:<br>*GPS:TEL? 0,DATA* |

| Commands | Explanation | Examples |
|---|---|---|
| | • ASCII: Return telemetry in ASCII representation.<br>• NAME: The name of the telemetry item e.g. *SupMCU Uptime*<br>• LENGTH: Length (in bytes) of telemetry in its binary representation.<br>Note: I*f no format representation is specified, DATA is used as the default.*<br>*See GPSRM Telemetry section for more detail.* | Query the name of the 2nd Telemetry item:<br>*GPS:TEL? 1,NAME* |
| GPS:CALibration? LINEAR,<idx>,[ASCII \| NAME \| DATA] | Request calibration values for a specific calibration index `idx` in a given format. The formats are:<br>• DATA: Return calibration values in binary format. *Not available over user debug terminal.*<br>• ASCII: Return the calibration values in ASCII representation.<br>• NAME: The name of the calibration index. *e.g. CTMU*<br>Note: *The `LINEAR` argument is required. This is for forward compatibility with future calibration value types (e.g. quadratic).* | Query `VCC_SYS` calibration values in ASCII:<br>BIM:CAL? LINEAR, 0, ASCII |
| GPS:CALibration LINEAR,<idx>,<OFFSET \| SCALE_FACTOR>,<value> | Set calibration `OFFSET` or `SCALE_FACTOR` values for a specific calibration `idx` in the GPSRM calibration values. The calibration values are used internally when calculating the telemetry values for the user, applying a linear correction factor (E.g. `*final = scale_factor*initial + offset*` Possible value sets are:<br>• OFFSET: Set the offset of the calibration factor. <value> is a number in the range [-32768, 32767]<br>• SCALE_FACTOR: Set the scale factor of the calibration factor. <value> is a floating-point number from [-16.0, 16.0].<br>Note: *The `LINEAR` argument is required. This is for forward compatibility with future calibration value types (e.g. quadratic).*<br>Note: *These values directly impact the output of telemetry from the SupMCU modules.*<br>Note: *Calibration values must be saved via `SUP:NVM` commands.*<br>Note: *See GPSRM Calibration for more details.* | Set the SCALE_FACTOR to .975 and offset to -20 for the `VCC_SYS` calibration value, save to NVM:<br>*GPS:CAL LINEAR,0,SCALE_FACTOR,0.975*<br>*GPS:CAL LINEAR,0,OFFSET,-20*<br>*SUP:NVM UNLOCK,12345*<br>*SUP:NVM WRITE,1* |

## PIM SCPI Commands

The PIM Command set is responsible for:

- Controlling the four assembly configurable power ports.
- Controlling the four port Ethernet switch on the PIM.

- Illuminating the Blue LED on the PIM.
- Querying the PIM's Telemetry.
- Setting Non-volatile memory values specific to the PIM.

| Commands | Explanation | Examples |
|---|---|---|
| PIM:PORT:POWer <OFF \| ON>,<N> | Power port N ON or OFF. N can be in the range [1, 4]. | Power Port #2 on<br>*PIM:PORT:POW ON,2* |
| PIM:TELemetry? <N>,[DATA \| ASCII \| NAME \| LENGTH] | Request telemetry for a specific field N in the given format. The formats are:<br>• DATA: Return telemetry in binary representation. *Not available over user debug terminal.*<br>• ASCII: Return telemetry in ASCII representation.<br>• NAME: The name of the telemetry item e.g. *SupMCU Uptime*<br>• LENGTH: Length (in bytes) of telemetry in its binary representation.<br>Note: I*f no format representation is specified, DATA is used as the default.*<br>*See PIM Telemetry section for more detail.* | Query PIM port status in ASCII format:<br>*PIM:TEL? 5,ASCII*<br>Query PIM channel voltages in binary format:<br>*PIM:TEL? 7*<br>Query telemetry item #4's length in bytes<br>*PIM:TEL? 3,LENGTH* |
| PIM:CALibration? LINEAR,<idx>,[ASCII \| NAME \| DATA] | Request calibration values for a specific calibration index `idx` in a given format. The formats are:<br>• DATA: Return calibration values in binary format. *Not available over user debug terminal.*<br>• ASCII: Return the calibration values in ASCII representation.<br>• NAME: The name of the calibration index. *e.g. CTMU*<br>Note: *The `LINEAR` argument is required. This is for forward compatibility with future calibration value types (e.g. quadratic).* | Query `SNS_CH1_I` calibration values in ASCII:<br>*PIM:CAL? LINEAR, 0, ASCII* |
| PIM:CALibration LINEAR,<idx>,<OFFSET \| SCALE_FACTOR>,<value> | Set calibration `OFFSET` or `SCALE_FACTOR` values for a specific calibration `idx` in the PIM calibration values. The calibration values are used internally when calculating the telemetry values for the user, applying a linear correction factor (E.g. `*final = scale_factor\*initial + offset*` Possible value sets | Set the SCALE_FACTOR to .975 and offset to -20 for the `SNS_CH1_I` calibration value, save to NVM:<br>*PIM:CAL LINEAR,0,SCALE_FACTOR,0.975*<br>*PIM:CAL LINEAR,0,OFFSET,-20* |

| Commands | Explanation | Examples |
|---|---|---|
| | are:<br>• OFFSET: Set the offset of the calibration factor. <value> is a number in the range [-32768, 32767]<br>• SCALE_FACTOR: Set the scale factor of the calibration factor. <value> is a floating-point number from [-16.0, 16.0].<br>Note: *The `LINEAR` argument is required. This is for forward compatibility with future calibration value types (e.g. quadratic).*<br>Note: *These values directly impact the output of telemetry from the SupMCU modules.*<br>Note: *Calibration values must be saved via `SUP:NVM` commands.*<br>Note: *See PIM Calibration for more details.* | *SUP:NVM UNLOCK,12345*<br>*SUP:NVM WRITE,1* |
| PIM:ETHernet ON \| OFF \| RESET \| UNRESET | Command the Ethernet switch to one of four different states:<br>• ON: The PIM Ethernet switch is ON<br>• OFF: The PIM Ethernet switch is OFF<br>• RESET: The PIM Ethernet switch is RESET.<br>• UNRESET: The PIM Ethernet switch RESET line is not driven high.<br>Note: *The RESET state is not turned off automatically, must be turned off with corresponding UNRESET command.* | Turn off the PIM's Ethernet:<br>*PIM:ETH OFF*<br>Reset the PIM's Ethernet switch:<br>*PIM:ETH RESET* |
| PIM:LED <BLUE \| RED>, <ON \| OFF> | Command the BLUE or RED LED to turn ON or OFF | Turn the RED LED ON:<br>*PIM:LED RED,ON* |

## SIM SCPI Commands

The SIM Command set is responsible for:

- Firing each of the four panel release modules (PRM's).

| Commands | Explanation | Examples |
|---|---|---|
| SIM:PRM <number>,<ENABle \| DISable \| ARM \| UNARm \| FIRE, <burn_time>> | Controls the state of the given PRM **number** where number is in the range [1,4]. To burn a PRM, the sequence of commands is ENABLE → ARM → FIRE. The different parameter strings are:<br>• ENABle: Enables the given PRM number, first stage in burning a PRM.<br>• DISable: Disables a given PRM number, effectively restarting the firing sequence.<br>• ARM: Arms the given PRM number, second stage in buring a PRM.<br>• UNARm: Disarms a given PRM number, this does **not** disable a PRM, it can be rearmed with a SIM:PRM <number>, ARM command.<br>• FIRE: Starts the OS task to burn the PRM for the given **burn_time** in seconds where **burn_time** is a value between [1, 1800].<br>Note: *Any SIM:PRM command done out of order (e.g. SIM:PRM 1,ENAB → SIM:PRM 1,FIRE,30) will reset the PRM back to DISABLED state. The sequence must restart with SIM:PRM <number>,ENABle.* | Burn PRM 1 & 2: for 30 seconds<br>*SIM:PRM 1,ENAB*<br>*SIM:PRM 1,ARM*<br>*SIM:PRM 1,FIRE,30*<br>*Wait 30 seconds...*<br>*SIM:PRM 2,ENAB*<br>*SIM:PRM 2.ARM*<br>*SIM:PRM 2,FIRE,30* |

## BSM SCPI Commands

The BSM Command set is responsible for:

- Powering each of the five BSM ports ON or OFF
- Querying telemetry for each of the five BSM ports.
- Setting Non-volatile memory specific to the BSM.

| Commands | Explanation | Examples |
|---|---|---|
| BSM:PORT:POWer <OFF \| ON>, <N><br>BSM:PORT:POWer <FLASH>,<N>,<period>,<duty> | Power the given BSM port N ON or OFF. N is a value between [1,5] corresponding to the BSM port.<br>If FLASH is given, it will cause the BIM port LED to flash every given ms (period) with the LED being on for given duty% of the period. Duty is constrained to the integer range [1,100] and period must be greater than 0 | Power on BSM ports 1 & 2:<br>*BSM:PORT:POW ON,1*<br>*BSM:PORT:POW ON,2*<br><br>Power off BSM Port 3:<br>*BSM:PORT:POW OFF,3* |
| BSM:TELemetry? <N>,[DATA \| ASCII \| NAME \| LENGTH] | Request telemetry for a specific field N in the given format. The formats are:<br>• DATA: Return telemetry in binary representation. *Not available over user debug terminal.*<br>• ASCII: Return telemetry in ASCII representation.<br>• NAME: The name of the telemetry item e.g. *SupMCU Uptime*<br>• LENGTH: Length (in bytes) of telemetry in its binary representation.<br>Note: I*f no format representation is specified, DATA is used as the default.*<br>*See BSM Telemetry section for more detail.* | Query BSM current telemetry in ASCII format:<br>*BSM:TEL? 0,ASCII*<br><br>Query BSM port status in binary representation:<br>*BSM:TEL? 5,DATA*<br><br>Query the name of the 2nd BSM telemetry item:<br>*BSM:TEL? 1,NAME* |

<table>
<tr><th>Commands</th><th>Explanation</th><th>Examples</th></tr>
<tr><td>BSM:NVM <field,<PORT>,<VALUE> | UNLOCK,<key> | WRITE,<1>><br><br>fields:<br>SHUNT, I_LIMIT</td><td>Controls access to and writes parameters to the non-volatile memory. Order of operations when changing NVM is: UNLOCK, <change value>, WRITE.<br>• UNLOCK: unlocks the non-volatile memory (must be done first). Value is a key consisting of:<br>device serial number + 12345<br>• WRITE: Writes all set NVM values to non-volatile memory, value = 1<br>• SHUNT, PORT, VALUE: Sets the current shunt resistor value for the PORT given with the integer VALUE in micro-ohms. <i>Default is 5000 micro-ohms.</i><br>• I_LIMIT, PORT, VALUE: Sets the Current limit for the given PORT with the given integer VALUE in mA, <i>Default is 2 Amps</i></td><td>Set a current limit for BSM Port #1 & #5 to 2.5A (assuming BSM Serial Number is 0):<br><i>BSM:NVM UNLOCK,12345</i><br><i>BSM:NVM I_LIMIT,1,2500</i><br><i>BSM:NVM I_LIMIT,5,2500</i><br><i>BSM:NVM WRITE,1</i><br><br>Set shunt for BSM port #3 to 6500 micro-ohms (assuming BSM serial number is 0):<br><i>BSM:NVM UNLOCK,12345</i><br><i>BSM:NVM SHUNT,3,6500</i><br><i>BSM:NVM WRITE,1</i></td></tr>
</table>

## BM2 SCPI Commands

Please refer to BM2 Manual for information on the available SCPI commands.

## EPSM/DCPS SCPI Commands

The EPSM/DCPS Command set is responsible for:

- Turning on / off 3.3v, 5v, 12v and AUX converters.
- Controlling charging voltage and current to batteries.
- Turning on/off battery converters.
- Turning on/off SAI1-6 converters.

When using the **DCPS** prefix SCPI commands with **DCPS** instead of **EPSM** (e.g. `EPSM:BUS 3V3,ON` equivalent on DCPS is `DCPS:BUS 3V3,ON`)
Note: *Commands are marked with which modules (DCPS/EPSM) support the command/subcommand.*

| Commands | Explanation | Examples |
|---|---|---|
| EPSM:BUS <3V3, 5V, 12V, AUX, VBATT>,<ON \| OFF> | Turns the given EPSM bus ON or OFF:<br>• 3V3, <ON \| OFF>: Turns the 3.3v bus ON or OFF.<br>• 5V, <ON \| OFF>: Turns the 5v bus ON or OFF.<br>• 12V, <ON \| OFF>: Turns the 12v bus ON or OFF.<br>• AUX, <ON \| OFF>: Turns the AUX bus ON or OFF. **EPSM Only.**<br>• VBATT, <ON \| OFF>: Turns the VBATT bus ON or OFF. **DCPS Only.** | Turn the 3.3v and 5v buses off:<br>*EPSM:BUS 3V3,OFF*<br>*EPSM:BUS 5V,OFF*<br><br>Turn the AUX bus on:<br>*EPSM:BUS AUX,ON* |
| EPSM:SAI <saiNum>, <ON \| OFF> | Turn input ON or OFF from one of the six SAIs on the EPSM. **EPSM Only.** | Turn off input from SAI6 and turn on input from SAI1:<br>*EPSM:SAI 6,OFF*<br>*EPSM:SAI 1,ON* |
| EPSM:BATtery <batNum>,<CHG_V>,<mV><br>EPSM:BATtery <batNum>,<CHG_I>,<mA><br>EPSM:BATtery <batNum>,<ON \| OFF> | Controls the BAT1 and BAT2 ports on the EPSM, turning them ON or OFF, or setting charging voltage/current limits on the ports.<br>• <batNum>,CHG_V,<mV>:<br>Set charging voltage to indicated battery port. batNum is 1 or 2. mV: voltage in mV between 6000 and 16800 mV.<br>• <batNum>,CHG_I,<mA>:<br>Set charging current limit to indicated battery port. batNum is 1 or 2. mA: charging current limit in mA between 0 and 12000 mA. | Turn off output and input from BATT1 port:<br>*EPSM:BAT 1,OFF*<br><br>Set BATT1's voltage limit to 8.4 V and current limit to 5 A:<br>*EPSM:BAT 1,CHG_V,8400*<br>*EPSM:BAT 1,CHG_I,5000* |

| Commands | Explanation | Examples |
|---|---|---|
| | • <batNum>, <ON \| OFF>: Turns ON or OFF the given battery port. batNum is either 1 or 2.<br>**EPSM Only.** | |
| EPSM:DEBug <ENable \| DISable>,<mask> | Enables or disables debug information being printed out over the UART Debug port on the EPSM. Mask is the bits of the debug information to enable:<br>• ENable, <mask>: Enables the bits represented by the mask value. Mask is a bit field and bits that are 1 will enable debug information.<br>• DISable, <mask>: Disables the bits represented by the mask value. Mask is a bit field and bits that are 1 will disable debug information.<br>There are 10 bits used for debug information on the EPSM:<br>• 0x0001: Print out raw bytes read from FPGA while commanding the EPSM.<br>• 0x0002: Show time between successive reads of FPGA ADCs.<br>• 0x0004: Show FPGA Frequency.<br>• 0x0008: Show UART receive buffers.<br>• 0x0010: Show FPGA Temperatures<br>• 0x0020: Show FPGA Block (SAI1-6, 12V, 5V, 3V3, AUX, BAT1 & 2) data.<br>• 0x0040: Show FPGA Voltage reference and dosimeter data.<br>• 0x0080: Show SupMCU ADC reads.<br>**No DCPS Debug flags are present, EPSM only.** | Enable all debug information on EPSM, except FPGA reads and ADC timing:<br>*EPSM:DEB EN,0x00FC*<br><br>Disable debug information on SAI converters:<br>*EPSM:DEB DIS,0x0020* |
| EPSM:TELemetry? <N>,[DATA \| ASCII \| NAME \| LENGTH] | Request telemetry for a specific field N in the given format. The formats are:<br>• DATA: Return telemetry in binary representation. *Not available over user debug terminal.*<br>• ASCII: Return telemetry in ASCII representation.<br>• NAME: The name of the telemetry item e.g. *SupMCU* | Query EPSM SAI1 Converter data in ASCII:<br>*EPSM:TEL? 0,ASCII*<br><br>Query EPSM 3.3V Converter in binary representation:<br>*EPSM:TEL? 6,DATA* |

| Commands | Explanation | Examples |
|---|---|---|
| | *Uptime*<br>- LENGTH: Length (in bytes) of telemetry in its binary representation.<br><br>Note: I*f no format representation is specified, DATA is used as the default.*<br>*See EPSM/DCPS Telemetry section for more detail.*<br>**EPSM Only** | Query the name of the 2$^{nd}$EPSM telemetry item:<br>*EPSM:TEL? 1,NAME* |
| EPSM:REad <addr>,<format> | Reads one or two bytes from a register addr on the FPGA, depending on the format specifier given. Format specifiers are:<br>- T – One byte, print out as int8_t.<br>- U – One byte, print out as uint8_t.<br>- X – One byte, print out hexadecimal value.<br>- S – Two bytes, print out as int16_t.<br>- D – Two bytes, print out as uint16_t.<br>- XX – Two bytes, print out as hexadecmial value.<br>Note the value is also stored as `Last register Read` in `EPS:TEL? 40`.<br>**EPSM Only** | Print out two bytes starting at 0x20fe in hexadecimal format<br>*EPSM:RE 0x20fe,XX* |
| EPSM:CALibration? LINEAR,<idx>,[ASCII \| NAME \| DATA] | Request calibration values for a specific calibration index `idx` in a given format. The formats are:<br>- DATA: Return calibration values in binary format. *Not available over user debug terminal.*<br>- ASCII: Return the calibration values in ASCII representation.<br>- NAME: The name of the calibration index. *e.g. CTMU*<br>Note: *The `LINEAR` argument is required. This is for forward compatibility with future calibration value types (e.g. quadratic).* | Query `SAI1_IO` calibration values in ASCII:<br>*EPSM:CAL? LINEAR, 0, ASCII* |
| EPSM:CALibration LINEAR,<idx>,<OFFSET \| SCALE_FACTOR>,<value> | Set calibration `OFFSET` or `SCALE_FACTOR` values for a specific calibration `idx` in the EPSM calibration values. The calibration values are used internally when calculating the telemetry values for the user, applying a linear correction factor (E.g. `*final =* | Set the SCALE_FACTOR to .975 and offset to -20 for the `SAI1_IO` calibration value, save to NVM:<br>*EPSM:CAL LINEAR,0,SCALE_FACTOR,0.975*<br>*EPSM:CAL LINEAR,0,OFFSET,-20* |

| Commands | Explanation | Examples |
|---|---|---|
|  | *scale_factor*initial + offset*` Possible value sets are:<br>- OFFSET: Set the offset of the calibration factor. <value> is a number in the range [-32768, 32767]<br>- SCALE_FACTOR: Set the scale factor of the calibration factor. <value> is a floating-point number from [-16.0, 16.0].<br>Note: *The `LINEAR` argument is required. This is for forward compatibility with future calibration value types (e.g. quadratic).*<br>Note: *These values directly impact the output of telemetry from the SupMCU modules.*<br>Note: *Calibration values must be saved via `SUP:NVM` commands.*<br>Note: *See EPSM Calibration for more details.*<br>Note: *The DCPS has a different set of calibration values. See DCPS Calibration values for more details.* | *SUP:NVM UNLOCK,12345*<br>*SUP:NVM WRITE,1* |

| Commands | Explanation | Examples |
|---|---|---|
| EPSM:NVM UNLOCK,<key><br>EPSM:NVM WRITE,1<br>EPSM:NVM <BAT1 \| BAT2>,<V_CHG_MAX \| I_CHG_LIMIT \| I_DSG_LIMIT>,<value><br>EPSM:NVM AUX,I_LIMIT,<mA><br>EPSM:NVM DEBUG,<mask> | Controls access to and writes parameters to the non-volatile memory. Order of operations when changing NVM is: UNLOCK, <change value>, WRITE.<br>• UNLOCK, key: unlocks the non-volatile memory (must be done first). key is a value consisting of: device serial number + 12345<br>• WRITE, value: Writes all set NVM values to non-volatile memory, value = 1.<br>• BAT1 \| BAT2, V_CHG_MAX \| I_CHG_LIMIT \| I_DSG_LIMIT, <value>: Sets the maximum charge voltage (V_CHG_MAX) or maximum charge current (I_CHG_LIMIT) or maximum discharge current (I_DSG_LIMIT) of BAT1 or BAT2 ports. **EPSM Only**<br>• AUX,I_LIMIT,mA: Sets the current limit on the AUX bus to mA. The upper limit of mA is 8000 mA (8A). **EPSM Only.**<br>• DEBUG, mask: Sets the debug mask to be set on boot. The current debug mask is also set to the mask value passed. *See EPS:DEBug command to see list of masks.* | Set a charging current limit for BAT1 to 5A, discharge limit for BAT1 to 6A and charging voltage for BAT1 to 16.8V (default):<br>*EPSM:NVM UNLOCK,12345*<br>*EPSM:NVM BAT1,I_CHG_LIMIT,5000*<br>*EPSM:NVM BAT1,I_DSG_LIMIT,6000*<br>*EPSM:NVM BAT1,V_CHG_MAX,16800*<br>*EPSM:NVM WRITE,1*<br>Set debug mask to show SAI state upon boot and now:<br>*EPSM:NVM UNLOCK,12345*<br>*EPSM:NVM DEBUG,0x0020*<br>*EPSM:NVM WRITE,1*<br>Set the current limit on the AUX bus to 4A:<br>*EPSM:NVM UNLOCK,12345*<br>*EPSM:NVM AUX,I_LIMIT,4000*<br>*EPSM:NVM WRITE,1* |

## AIM2 SCPI Commands

The AIM2 Command set is responsible for:

- Controlling power to the OEM615/719.
- Controlling the RESET line to the OEM615/719.
- Propagating orbit via Vinti7/SGP4 orbit propagators.
- Setting UART communication channel for OEM615/719.
- Enabling or disabling UART communication for the OEM615/719.
- Enabling a subset of the logs available for the OEM615/719.
- Controlling power to the MAI-400 ADCS
- Controlling the RESET line to the MAI-400 ADCS
- Setting UART communications channel for MAI-400 ADCS
- Enabling or disabling UART communications to/from MAI-400 ADCS
- Querying AIM-2 and GPSRM Telemetry.

Note: *The AIM-2 Module supports **ALL** of the **GPSRM** commands **except GPS:ADCS:POW.** GPSRM commands can be prefixed with **GPS:… or AIM:GPS:… (AIM:GPS:POW ON** is equal to **GPS:POW ON)***

Note: *See GPSRM SCPI Commands for GPSRM commands. All AIM-2 Specific commands are listed below.*

| Commands | Explanation | Examples |
|---|---|---|
| GPS:COMM <NONE \| UART0 \| UART1 \| UART2 \| UART3 \| SUP1 \| SUP2><br>*or*<br>AIM:GPS:COMM <NONE \| UART0 \| UART1 \| UART2 \| UART3 \| SUP1 \| SUP2> | Selects what CSK UART channel the OEM615/719 COM2 port should communicate on. UART choices are:<br>• NONE: Disables UART communication to OEM615/719 COM2 port.<br>• UART0-UART3: Sets UART communication for OEM615/719 COM2 port to CSK UART 0, 1, 2 or 3. Note: *See MBM2 Datasheet for mapping of CSK UART's to BBB UART's.*<br>• SUP1-SUP2: Sets UART communication for OEM615/719 to one of the two available internal SupMCU UART communication channels. Note: *Ports only accessible by SupMCU.*<br>Note: *Communication must still be passed through via a* | Set OEM615/719 to communicate on CSK UART0 (BBB UART0) and enable commucations:<br>GPS:POW ON<br>*GPS:COMM UART0*<br>*GPS:PASS ON*<br><br>Set OEM615/719 to not communicate on any UART:<br>*GPS:COMM NONE*<br>*GPS:PASS OFF* |

| Commands | Explanation | Examples |
|---|---|---|
| | ***GPS:PASS ON*** *command.* | |
| AIM:ADCS:COMM <NONE \| UART0 \| UART1 \| UART2 \| UART3 \| SUP1 \| SUP2> | Selects what CSK UART channel the MAI-400 should communicate on. UART choices are:<br>• NONE: Disables UART communication to MAI-400.<br>• UART0-UART3: Sets UART communication for MAI-400 to CSK UART 0, 1, 2 or 3. Note: *See MBM2 Datasheet for mapping of CSK UART's to BBB UART's.*<br>• SUP1-SUP2: Sets UART communication for MAI-400 to one of the two available internal SupMCU UART communication channels. Note: *Ports only accessible by SupMCU.*<br>Note: *Communication must still be passed through via a* ***AIM:ADCS:PASS ON*** *command.* | Set MAI-400 to communicate on CSK UART2 (BBB UART 2) and enable communications:<br>AIM:ADCS:POW ON<br>*AIM:ADCS:COMM UART2*<br>*AIM:ADCS:PASS ON*<br><br>Set MAI-400 to not communicate on any UART:<br>*AIM:ADCS:COMM NONE*<br>*AIM:ADCS:PASS OFF* |
| AIM:ADCS:PASS ON \| OFF | Turn the UART connection to the MAI-400 ADCS ON or OFF. | Turn on UART passthrough to MAI-400:<br>*AIM:ADCS:PASS ON* |
| AIM:ADCS:POWer ON \| OFF | Turns power to the MAI-400 ADCS ON or OFF | Turn off the MAI-400:<br>*AIM:ADCS:POW OFF* |
| AIM:WDT <ENABLE \| DISABLE \| FORCE \| PERIOD, <period>> | Controls a SupMCU task to kick the WDT on the AIM-2. The control states are:<br>• ENABLE: Starts SupMCU task to kick the WDT.<br>• DISABLE: Stops SupMCU task to kick the WDT.<br>• FORCE: Kicks the WDT on the AIM once.<br>• PERIOD, period: Sets the interval between kicks for the task to kick the WDT, where period is the miliseconds between WDT kicks. *Default peroid is 10000 ms.*<br>Note: ***This command is currently not supported, and is not recommended for use.*** | Set the WDT task to 100 ms and start kicking the WDT:<br>*AIM:WDT PERIOD, 100*<br>*AIM:WDT ENABLE* |
| AIM:ADCS:RESET ON \| OFF | Turns the RESET signal to the MAI-400 ADCS ON or OFF (assert or revoke).<br>Note: *The RESET signal must be turned off with a corresponding AIM:ADCS:RESET OFF command. It is not turned off automatically.v* | Turn on the RESET signal for the MAI-400:<br>*AIM:ADCS:RESET ON* |

<table>
<tr><th>Commands</th><th>Explanation</th><th>Examples</th></tr>
<tr>
<td><i>NVM control commands</i><br>AIM:NVM <UNLOCK, <key><br>| WRITE, <1>><br><br><i>NVM values for OEM615/719</i><br>AIM:NVM <GPS>,<COMM, <CHAN> | <POW>, <ON | OFF>><br><br><i>NVM values for MAI-400</i><br>AIM:NVM <ADCS>, <COMM, <CHAN><br>| <POW>, <ON | OFF><br>| <UART>, <ENABLE | DISABLE>></td>
<td>Controls access to and writes parameters to the non-volatile memory. Order of operations when changing NVM is: UNLOCK, <change value>, WRITE.<br>• UNLOCK: unlocks the non-volatile memory (must be done first). Value is a key consisting of:<br>device serial number + 12345<br>• WRITE: Writes all set NVM values to non-volatile memory, value = 1<br>• GPS,COMM,<CHAN>: Sets the UART communications channel the OEM615/719 is on upon startup of the AIM-2. CHAN can be in the set {NONE, UART0, UART1, UART2, UART3, SUP1, SUP2}. <i>Default is NONE.</i><br>• GPS,POW,<ON | OFF>: Sets if the OEM615/719 should power ON or OFF upon startup of the AIM-2. <i>Default is OFF.</i><br>• ADCS,COMM,<CHAN>: Sets the UART communications channel the MAI-400 is on upon startup of the AIM-2. CHAN can be in the set {NONE, UART0, UART1, UART2, UART3, SUP1, SUP2}. <i>Default is NONE.</i><br>• ADCS,UART, <ENABLE | DISABLE>: Sets if UART pass through for the MAI-400 should be ENABLEd or DISABLEd upon startup of the AIM-2. <i>Default is Disabled.</i><br>• ADCS,POW, <ON | OFF>: Sets if the MAI-400 should be ON or OFF upon startup of the AIM-2. <i>Default is OFF.</i><br><br>Note: <i>There is NO NVM value for setting startup state of OEM615/719 UART pass through, it must be set explicitly ON or OFF after start with GPS:PASS ON or GPS:PASS OFF</i></td>
<td>Set the OEM615/719 to communicate on UART1 and MAI-400 to communicate on UART2 upon startup of the AIM-2 (assuming serial number of AIM-2 is 100):<br><i>AIM:NVM UNLOCK,12445</i><br><i>AIM:NVM GPS,COMM,UART1</i><br><i>AIM:NVM ADCS,COMM,UART2</i><br><i>AIM:NVM WRITE,1</i><br><br>Set the OEM615/719 and MAI-400 to power ON upon startup of AIM-2 (assuming serial number of AIM-2 is 0):<br><i>AIM:NVM UNLOCK,12345</i><br><i>AIM:NVM GPS,POW,ON</i><br><i>AIM:NVM ADCS,POW,ON</i><br><i>AIM:NVM WRITE,1</i></td>
</tr>
</table>

| Commands | Explanation | Examples |
|---|---|---|
| *AIM:TELemetry? <N>,[DATA \| ASCII \| NAME \| LENGTH]* | Request telemetry for a specific field N in the given format. The formats are:<br>• DATA: Return telemetry in binary representation. *Not available over user debug terminal.*<br>• ASCII: Return telemetry in ASCII representation.<br>• NAME: The name of the telemetry item e.g. *SupMCU Uptime*<br>• LENGTH: Length (in bytes) of telemetry in its binary representation.<br>Note: I*f no format representation is specified, DATA is used as the default.*<br>*See AIM Telemetry section for more detail.* | Query OEM615/719 Power status in ASCII representation:<br>*AIM:TEL? 3,ASCII*<br>Query MAI-400 ADCS status in binary representation<br>*AIM:TEL? 9*<br>Query ADCS UART Status binary representation byte length:<br>*AIM:TEL? 7,LENGTH* |

## RHM SCPI Commands

The RHM Command set is responsible for:

- Controlling power to GlobalStar Simplex radio
- Controlling UART communication channel to/from GlobalStar Simplex radio.
- Controlling Digital Input (DIN) pins for GlobalStar Simplex radio.
- Enabling or disabling UART communications to/from GlobalStar Simplex radio.
- ***RHM1 Only –*** Controlling power to Astrodev Lithium radio.
- ***RHM1 Only –*** Controlling UART communication channel to/from Astrodev Lithium radio.
- ***RHM1 Only –*** Enabling or disabling UART communications to/from Astrodev Lithium radio.
- Querying RHM specific Telemetry.
- Setting RHM specific NVM values

| Commands | Explanation | Examples |
|---|---|---|
| RHM:GStar:COMM <NONE \| UART0 \| UART1 \| UART2 \| UART3 \| UART4 \| I2C> | Selects what CSK UART channel the GlobalStar Simplex radio should communicate on. UART choices are:<br>• NONE: Disables UART communication to GlobalStar Simplex.<br>• UART0-UART4: Sets UART communication for GlobalStar Simplex to CSK UART 0, 1, 2, 3 or 4.<br>Note: *See MBM2 Datasheet for mapping of CSK UART's to BBB UART's.*<br>Note: *CSK UART4 is a TX-only UART.*<br>• I2C: Sets UART communication for GlobalStar Simplex to one of the two available internal SupMCU UART communication channels.<br>Note: *Ports only accessible by SupMCU.*<br>Note: *Communication must still be passed through via a* ***RHM:GS:PASS ON*** *command.* | Set GlobalStar Simplex to communicate on CSK UART3 (BBB UART4) and powering it on:<br>*RHM:GS:COMM UART3*<br>*RHM:GS:POW ON*<br>*RHM:GS:PASS ON*<br><br>Turn off UART communication to GlobalStar Simplex:<br>*RHM:GS:COMM NONE*<br>*RHM:GS:PASS OFF* |
| RHM:GStar:POWer <ON \| OFF> | Turns the power supply to the GlobalStar radio ON or OFF | Turn on the GlobalStar Simplex:<br>*RHM:GS:POW ON* |
| RHM:GStar:PASSthrough <ON \| OFF> | Sets the Globalstar's UART passthrough ON or OFF | Turn off UART Passthrough for GlobalStar Simplex:<br>*RHM:GS:PASS OFF* |

| Commands | Explanation | Examples |
|---|---|---|
| RHM:GS:DIN <ON \| OFF>,<pin><br>RHM:GS:DIN <ENABLE \| DISABLE> | Controls the Digital input pins on the GlobalStar radio. Choices for controlling DIN state are:<br>• ON \| OFF, PIN: Turns the given digital input PIN ON or OFF. PIN is either 1 or 2.<br>• ENABLE \| DISABLE: Enables or disables the GlobalStar Simplex Digital Input. | Turn on Digital Input pin #2 and enable digital input:<br>*RHM:GS:DIN ON,2*<br>*RHM:GS:DIN ENABLE* |
| RHM:LIthium:COMM <NONE \| UART0 \| UART1 \| UART2 \| UART3 \| UART4 \| I2C> | **Only on RHM1, REJECTED command on RHM2.**<br>Selects what CSK UART channel the Lithium radio should communicate on. UART choices are:<br>• NONE: Disables UART communication to Lithium.<br>• UART0-UART4: Sets UART communication for Lithium to CSK UART 0, 1, 2, 3 or 4.<br>Note: *See MBM2 Datasheet for mapping of CSK UART's to BBB UART's.*<br>Note: *CSK UART4 is a TX-only UART.*<br>• I2C: Sets UART communication for Lithium to one of the two available internal SupMCU UART communication channels.<br>Note: *Ports only accessible by SupMCU.*<br>Note: *Communication must still be passed through via a* ***RHM:LI:PASS ON*** *command.* | Set UART Communication to Lithium radio to CSK UART1 (BBB UART1) then power and enable communications:<br>*RHM:LI:COMM UART1*<br>*RHM:LI:POW ON*<br>*RHM:LI:PASS ON*<br><br>Disable UART communication to Lithium radio and disable passthrough:<br>*RHM:LI:COMM NONE*<br>*RHM:LI:PASS OFF* |
| RHM:LIthium:CONFig <1 \| 2>, <ON \| OFF> | **Only on RHM1, REJECTED command on RHM2.**<br>Sets the configure pin number (1 or 2) of the Lithium Radio ON or OFF. | Turn off configure pin #1<br>*RHM:LI:CONF 1,OFF* |
| RHM:LIthium:MCU <ON \| OFF> | **Only on RHM1, REJECTED command on RHM2.**<br>Turn the Lithium's MCU power ON or OFF. | Turn on Lithium MCU<br>*RHM:LI:MCU ON* |
| RHM:LIthium:PA <ON \| OFF> | **Only on RHM1, REJECTED command on RHM2.**<br>Turn the Lithium's power amplifier (PA) power ON or OFF | Turn on Lithium Power-amplifier<br>*RHM:LI:PA ON* |
| RHM:LIthium:POWer <ON \| OFF> | **Only on RHM1, REJECTED command on RHM2.**<br>Turn the Lithium's MCU and PA power ON or OFF | Turn on Lithium MCU and PA.<br>*RHM:LI:POW ON* |
| RHM:WDT <ENABLE \| DISABLE \| FORCE \| PERIOD, <period>> | Controls a SupMCU task to kick the WDT on the RHM. The control states are:<br>• ENABLE: Starts SupMCU task to kick the WDT.<br>• DISABLE: Stops SupMCU task to kick the WDT. | Set the WDT task to 100 ms and start kicking the WDT:<br>RHM:*WDT PERIOD, 100*<br>RHM:*WDT ENABLE* |

| Commands | Explanation | Examples |
|---|---|---|
| | • FORCE: Kicks the WDT on the RHM once.<br>• PERIOD, period: Sets the interval between kicks for the task to kick the WDT, where period is the miliseconds between WDT kicks. *Default peroid is 10000 ms.*<br>Note: ***This command is currently not supported, and is not recommended for use.*** | |
| RHM:DEBug <ENable \| DISable>, mask | Enables or disables debug information being printed out over the UART Debug port on the RHM. Mask is the bits of the debug information to enable:<br>• ENable, <mask>: Enables the bits represented by the mask value. Mask is a bit field and bits that are 1 will enable debug information.<br>• DISable, <mask>: Disables the bits represented by the mask value. Mask is a bit field and bits that are 1 will disable debug information.<br>There are 4 bits used for debug information on the EPSM:<br>• 0x0001: Prints out configuration of GlobalStar Simplex Radio when it is powered on.<br>• 0x0002: Prints out raw bytes being sent to Simplex Radio from RHM:SEND command.<br>• 0x0004: Prints out raw bytes read from Simplex radio from RHM:SEND command.<br>• 0x0008: Prints out state of BUSY Line during RHM:SEND command. | |
| *NVM control commands*<br>RHM:NVM <UNLOCK, <key> \| WRITE, <1> \| DEBUG, <mask>><br><br>*NVM values for GlobalStar Simplex*<br>RHM:NVM <GS>,<COMM, <CHAN> \| <UART>, <ENABLE \| DISABLE> \| <POW>, <ON \| OFF>><br><br>*NVM values for Lithium* | Controls access to and writes parameters to the non-volatile memory. Order of operations when changing NVM is: UNLOCK, <change value>, WRITE.<br>• UNLOCK: unlocks the non-volatile memory (must be done first). Value is a key consisting of: device serial number + 12345<br>• WRITE: Writes all set NVM values to non-volatile memory, value = 1<br>• DEBUG,<mask>: Sets the debug flag to use at boot of the module. *See RHM:DEBug command for list of masks.* | Set GlobalStar Simplex to be powered ON, and enable UART communications on CSK UART2 (BBB UART2) on startup. Assuming serial number of 0:<br>*RHM:NVM UNLOCK,12345*<br>*RHM:NVM GS,COMM,UART2*<br>*RHM:NVM GS,UART,ENABLE*<br>*RHM:NVM GS,POW,ON*<br>*RHM:NVM WRITE,1* |

| Commands | Explanation | Examples |
|---|---|---|
| RHM:NVM <LI>, <COMM, <CHAN><br>\| <POW>, <ON \| OFF><br>\| <UART>, <ON \| OFF><br>\| <PAPOW>, <ON \| OFF><br>\| <MCUPOW>, <ON \| OFF>> | • GS,COMM,<CHAN>: Sets the UART communications channel the GlobalStar Simplex is on upon startup of the RHM. CHAN can be in the set {NONE, UART0, UART1, UART2, UART3, UART4, I2C}. *Default is NONE.*<br>• GS,UART,<ENABLE \| DISABLE>: Sets if UART passthrough should be enabled or disabled to the GlobalStar Simplex upon startup of the RHM. *Default is disabled.*<br>• GS,POW,<ON \| OFF>: Sets if the GlobalStar Simplex should power ON or OFF upon startup of the RHM. *Default is off*<br>• LI,COMM,<CHAN>: (**RHM1 Only**)Sets the UART communications channel the Lithium radio is on upon startup of the RHM. CHAN can be in the set {NONE, UART0, UART1, UART2, UART3, UART4, I2C}. *Default is NONE.*<br>• LI,UART, <ENABLE \| DISABLE>: (**RHM1 Only**) Sets if UART pass through for the Lithium should be ENABLEd or DISABLEd upon startup of the RHM. *Default is disabled.*<br>• LI,POW, <ON \| OFF>: (**RHM1 Only**) Sets if the Lithium radio should be ON or OFF upon startup of the RHM (Both PA and MCU). *Default is OFF.*<br>• LI,PAPOW, <ON \| OFF>: (**RHM1 Only**) Sets if the Lithium radio's power-amplifier should be ON or OFF upon startup of the RHM. *Default is OFF.*<br>• LI,MCUPOW,<ON \| OFF>: (**RHM1 Only**) Sets if the Lithium radio's microcontroller unit (MCU) should be ON or OFF upon startup of RHM. *Default is OFF.* | Set Lithium radio to be OFF, and enable UART Communications on CSK UART1 (BBB UART1) on startup. Assuming serial number of 0:<br>*RHM:NVM UNLOCK,12345*<br>*RHM:NVM LI,COMM,UART1*<br>*RHM:NVM LI,POW,OFF*<br>*RHM:NVM LI,UART,ENABLE*<br>*RHM:NVM WRITE,1* |
| RHM:GS:SEND <hexstr> | Sends a 'hex' string (eg. 50756D706B696E for Pumpkin) (up to a max of 70 hex characters) to the GlobalStar radio. These requirements must be met **before** using this command:<br>• The UART communication port must be set to **I2C.** (RHM:GS:COMM I2C).<br>• Radio must be **powered.** (RHM:GS:POW ON)<br>• UART pass through must be **ENABLED.** (RHM:GS:PASS | Send "Hello World" through the GS radio.<br>*RHM:GS:POW ON*<br>*RHM:GS:PASS ON*<br>*RHM:GS:COMM I2C*<br>*RHM:GS:SEND 48656c6c6f20576f726c64* |

| Commands | Explanation | Examples |
|---|---|---|
| | ON).<br>• GS Busy line must not be HIGH for more than 20 seconds AFTER the command is sent.<br>• GS must NOT be transmitting currently.<br>This command will correctly wrap the packet for the GS Radio so wrapping does not need to be included.<br>This will increment the \`GS Transmission Sent\` if the GS Radio sends an ACK packet back, otherwise \`GS Transmission Failed\` is incremented.<br>*Note: This command is only available on RHM firmware 1.2.5a+* | |
| RHM:GS:CONFigure <BEACONtime \| CONTACTbeacons \| INHIBITtimer>, value<br>or<br>RHM:GS:CONFigure <APPly> | Controls the GlobalStar Simplex configuration packet. This command will automatically configure the radio for receiving the configuration packet, parse the configuration reply, and check for out of bounds configuration settings. *Note: default values are restored upon RHM:GS:CONF APPly command, the configuration will need to be redone between each RHM:GS:CONF APPly command.*<br>• BEACONtime, value: The time between contact beacons in seconds, where value is the amount of seconds to wait between automated contact beacons. Range is [0, 14400]. *Note: Default value is 600 seconds.*<br>• CONTACTbeacons, value: The amount of first contact beacons to send upon initial power of the GS Simplex radio. Range is [0, 20]. *Note: Default value is 10 beacons.*<br>• INHIBITtimer, value: The amount of time when the GS Simplex is powered up before transmitting first contact beacons. Range is [0, 90]. *Note: Default value is 10 minutes.* | Set the Simplex to delay for 15 minutes before sending 5 first contact beacons with a contact beacon every 10 minutes:<br>*RHM:GS:CONF INHIBIT,15*<br>*RHM:GS:CONF BEACON,600*<br>*RHM:GS:CONF CONTACT,5*<br>*RHM:GS:CONF APP* |
| *RHM:TELemetry? <N>,[DATA \| ASCII \| NAME \| LENGTH]* | Request telemetry for a specific field N in the given format. The formats are:<br>• DATA: Return telemetry in binary representation. *Not available over user debug terminal.*<br>• ASCII: Return telemetry in ASCII representation. | Query RHM for GlobalStar UART Status in ASCII:<br>*RHM:TEL? 4,ASCII*<br>Query RHM for name of 4th telemetry item:<br>*RHM:TEL? 3,NAME* |

| Commands | Explanation | Examples |
|---|---|---|
| | • NAME: The name of the telemetry item e.g. *SupMCU Uptime*<br>• LENGTH: Length (in bytes) of telemetry in its binary representation.<br>Note: I*f no format representation is specified, DATA is used as the default.*<br>*See RHM Telemetry section for more detail.* | |

## DASA SCPI Commands

The DASA Command set is responsible for:

- Articulating the DASA array clockwise/counter-clockwise a set amount of steps.
- Setting the articulated solar array to an absolute position in steps.
- Homing the articulated solar array to its center position.
- Manipulating motor drive manually.
- Energizing the solar panel release mechanism to deploy the solar array.
- Switching the -1.8v power rail to extend the range of the temperature sensors.

| Commands | Explanation | Examples |
|---|---|---|
| DASA:DEBug <ENable \| DISable>,<mask> | Enables or disables debug messages to be output from the SupMCU Module. The following masks are supported on the DASA module:<br>• 0x0001: Control debug messages outputting the state of A+/- and B+/- drive circuity. These messages are displayed when the array is rotating via *DASA:STEP* command.<br>• 0x0002: Shows the current HES sensor readings during a *DASA:STEP HOME* move.<br>Note: *Debug messages are only displayed via the Debug Serial connection. They will not be able to be retrieved via I2C.* | Enable debug messages for state of the drive circuity:<br>*DASA:DEB EN,0x0001*<br>Disable debug messages for state of the drive circuity:<br>*DASA:DEB DIS,0x0001* |
| DASA:RELease <DISAble \| ENABle \| UNARm \| ARM \| FIRE, <N>> | Controls the solar panel release mechanism. When energizing the release mechanism, the order of operations is ENABLE → ARM → FIRE, N. The different parameter strings mean:<br>• DISAble: Disables the solar panel release mechanism, effectively resetting the sequence.<br>• ENABle: Enables the solar panel release mechanism, first step in panel deployment.<br>• ARM: Arms the solar panel release mechanism, second step in panel deployment.<br>• UNARm: Disables the SupMCU task responsible for energizing the Pin Puller. *Does not disable the solar panel release mechanism.* | Energize the solar panel release mechanism for 25 seconds.<br>*DASA:REL ENAB*<br>*DASA:REL ARM*<br>*DASA:REL FIRE,25*<br><br>Reset the solar panel release mechanism sequence:<br>*DASA:REL DISAble* |

| Commands | Explanation | Examples |
|---|---|---|
| | • FIRE, Seconds: Energizes the Pin Puller, releasing the deployment mechanism of the solar panels. Will keep energized for N seconds where N is in the range [1, 30].<br>Note: *Any command done out of order will reset the sequence. E.g. if DASA:REL ENAB → DASA:REL FIRE,30 is done, then DASA:REL ENAB must be sent again.* | |
| DASA:STEP <STOP><br>or<br>DASA:STEP <REL \| ABS \| HOME>,<HALF \| FULL>,<rate>,<steps> | Controls the position of the articulated solar array. The command *DASA:DRIVE ENABLE* must be sent before the array will articulate.<br>• STOP:<br>Stops movement of the articulated solar array, canceling any motor movement.<br>• REL, HALF \| FULL, rate, steps:<br>Drive the motor circuit to rotate the array relative to the current position. The steps argument can be negative, which will rotate the array counter-clockwise, while a positive value rotates the array clockwise. FULL \| HALF argument controls whether to drive motor circuit by full steps, or half stepper motor steps.<br>*Note: The command will stop movement if the drive circuit detects the rotation limit via hall effect sensor during rotation.*<br>*Note: Maximum rate is 25 Hz, if a greater rate is given, it is clamped to 25 Hz.*<br>• HOME, HALF \| FULL, rate, steps:<br>Homes the articulated array to its 12 o'clock position by driving the motor circuit to rotate clockwise until either detecting the home position via hall effect sensor, or reaching the maximum steps limit given by *steps*. The Homing algorithm will attempt to find home via a min/max reading on the Hall-effect sensor, if found, set the position telemetry to 0, and set the homed bit to 1. Otherwise the DASA will stop rotation and leave homed | Turn the DASA array 200 steps counter-clockwise at 10 Hz step rate using FULL steps:<br>*DASA:DRIVE ENABLE*<br>*DASA:STEP REL,FULL,10,200*<br><br>Stop movement of the DASA array:<br>*DASA:STEP STOP*<br><br>Set the DASA array to a position of 269, moving at 7 Hz, holding the position at a half step:<br>*DASA:DRIVE ENABLE*<br>*DASA:DRIVE HOLD*<br>*DASA:STEP ABS,HALF,7, 269*<br><br>Home the DASA array, allowing it to move a maximum of 2000 steps either direction at 15 Hz:<br>*DASA:DRIVE ENABLE*<br>*DASA:STEP HOME,FULL,15,2000* |

| Commands | Explanation | Examples |
|---|---|---|
| | as 0 and position as the current value. Rate is the frequency at which the pulse-width modulation is driven. FULL \| HALF argument controls whether to drive motor circuit by full steps, or half stepper motor steps.<br>*Note: The homed bit is **NOT** set if the DASA array **DOES NOT detect the homed position via hall effect sensor**. Its position is also not set to 0 if it does not detect the rotation limit via hall effect sensor at the limits.*<br>*Note: Maximum rate is 25 Hz, if a greater rate is given, it is clamped to 25 Hz.*<br>• ABS, HALF \| FULL, rate, position:<br>Sets the array to an absolute position, rotating CW or CCW based on shortest path to position. FULL \| HALF argument controls whether to drive motor circuit by full steps, or half stepper motor steps.<br>*Note: The command will stop movement if the drive circuit detects the rotation limit via hall effect sensor during rotation.*<br>*Note: Maximum rate is 25 Hz, if a greater rate is given, it is clamped to 25 Hz.* | |
| DASA:ROTate <STOP><br>or<br>DASA:ROTate <REL \| ABS \| HOME>,<HALF \| FULL>,<rate>,<angle> | Controls the angle of the articulated solar array. This command converts the angle given to a number of steps to rotate the stepper motor drive:<br>• STOP:<br>Stops movement of the articulated solar array, canceling any motor movement.<br>• REL, HALF \| FULL, rate, angle:<br>Drive the motor circuit to rotate the array relative to the current angle. The angle argument can be negative, which will rotate the array counter-clockwise, while a positive value rotates the array clockwise. FULL \| HALF argument controls whether to drive motor circuit by full steps, or half stepper motor steps. | Turn the DASA array 45.7 degrees counter-clockwise at 1 degree per second using FULL steps:<br>*DASA:DRIVE ENABLE*<br>*DASA:ROT REL,FULL,1,45.7*<br>Stop movement of the DASA array:<br>*DASA:ROT STOP*<br>Set the DASA array to an angle of 110 degrees, moving at 1.5 degrees per second, using half- |

| Commands | Explanation | Examples |
|---|---|---|
|  | *Note: The command will stop movement if the drive circuit detects the rotation limit via hall effect sensor during rotation.*<br>*Note: Maximum rate is 25 Hz, if a greater rate is given, it is clamped to 25 Hz.*<br>• HOME, HALF \| FULL, rate, max_angle:<br>Homes the articulated array to its 12 o'clock position by driving the motor circuit to rotate clockwise until either detecting the home position via hall effect sensor, or reaching the maximum angle given by max_angle. The Homing algorithm will attempt to find home via a min/max reading on the Hall-effect sensor, if found, set the position telemetry to 0, and set the homed bit to 1. Otherwise the DASA will stop rotation and leave homed as 0 and position as the current value. Rate is the frequency at which the pulse-width modulation is driven. FULL \| HALF argument controls whether to drive motor circuit by full steps, or half stepper motor steps.<br>*Note: The homed bit is* ***NOT*** *set if the DASA array* ***DOES NOT detect the homed position via hall effect sensor****. Its position is also not set to 0 if it does not detect the rotation limit via hall effect sensor at the limits.*<br>*Note: Maximum rate is 25 Hz, if a greater rate is given, it is clamped to 25 Hz.*<br>• ABS, HALF \| FULL, rate, position:<br>Sets the array to an absolute position, rotating CW or CCW based on shortest path to position. FULL \| HALF argument controls whether to drive motor circuit by full steps, or half stepper motor steps.<br>*Note: The command will stop movement if the drive circuit detects the rotation limit via hall effect sensor during rotation.*<br>*Note: Maximum rate is 25 Hz, if a greater rate is given, it* | steps:<br>*DASA:DRIVE ENABLE*<br>*DASA:ROT ABS,HALF,1.5, 110*<br>Home the DASA array, allowing it to move a maximum of 360 degrees in either direction at 2 degrees per second in FULL steps:<br>*DASA:DRIVE ENABLE*<br>*DASA:ROT HOME,FULL,2,360* |

| Commands | Explanation | Examples |
|---|---|---|
| | *is clamped to 25 Hz.* | |
| DASA:CALibration? LINEAR,<idx>,[ASCII \| NAME \| DATA] | Request calibration values for a specific calibration index `idx` in a given format. The formats are:<br>• DATA: Return calibration values in binary format. *Not available over user debug terminal.*<br>• ASCII: Return the calibration values in ASCII representation.<br>• NAME: The name of the calibration index. *e.g. CTMU*<br>Note: *The `LINEAR` argument is required. This is for forward compatibility with future calibration value types (e.g. quadratic).* | Query `temp. sensor 1` calibration values in ASCII:<br>*DASA:CAL? LINEAR, 0, ASCII* |
| DASA:CALibration LINEAR,<idx>,<OFFSET \| SCALE_FACTOR>,<value> | Set calibration `OFFSET` or `SCALE_FACTOR` values for a specific calibration `idx` in the DASA calibration values. The calibration values are used internally when calculating the telemetry values for the user, applying a linear correction factor (E.g. `*final = scale_factor*initial + offset*` Possible value sets are:<br>• OFFSET: Set the offset of the calibration factor. <value> is a number in the range [-32768, 32767]<br>• SCALE_FACTOR: Set the scale factor of the calibration factor. <value> is a floating-point number from [-16.0, 16.0].<br>Note: *The `LINEAR` argument is required. This is for forward compatibility with future calibration value types (e.g. quadratic).*<br>Note: *These values directly impact the output of telemetry from the SupMCU modules.*<br>Note: *Calibration values must be saved via `SUP:NVM` commands.*<br>Note: *See DASA Calibration for more details.* | Set the SCALE_FACTOR to .975 and offset to -20 for the `temp. sensor 1` calibration value, save to NVM:<br>*DASA:CAL LINEAR,0,SCALE_FACTOR,0.975*<br>*DASA:CAL LINEAR,0,OFFSET,-20*<br>*SUP:NVM UNLOCK,12345*<br>*SUP:NVM WRITE,1* |
| DASA:HES <ENABle \| DISAble> | Enables or disables the DASA hall-effect sensor used to home the articulated array.<br>• ENABle: Enables the HES. It's value can be read via *DASA:TEL? 6.*<br>• DISAble: Disables the HES. | Enable the hall-effect sensor:<br>*DASA:HES ENABLE*<br><br>Disable the hall-effect sensor:<br>*DASA:HES DISABLE* |

| Commands | Explanation | Examples |
|---|---|---|
| | Note: *Disabling the HES during a homing move WILL cause the home to fail. The HES is automatically enabled during a HOME move.* | |
| DASA:CTRL <OPEN \| CLOSE \| AOFF \| APOS \| ANEG \| BOFF \| BPOS \| BNEG \| SENSE> | Manually controls the Motor A and B drive circuits.[4] The drive circuit must be opened via DASA:CTRL OPEN **before** sending any DASA:CTRL commands. Note to drive the array CW, the sequence is A+ ON → B+ ON → A- ON → B- ON, and to drive the array CCW, the sequence is B+ ON → A+ ON → B- ON → A- ON.<br>• OPEN: Opens the drive circuit, first step before manually controlling the DASA motor.<br>• CLOSE: Closes the drive circuit, last step to stop movement of controlling the DASA motor.<br>• AOFF: Turns off the A+/- motor drive.<br>• BOFF: Turns off the B+/- motor drive.<br>• APOS: Turns on A+ motor drive.<br>• BPOS: Turns on B+ motor drive.<br>• ANEG: Turns on A- motor drive.<br>• BNEG: Turns on B- motor drive.<br>• SENSE: Outputs current driven through A and B motor drive circuits to debug UART.<br>*Note: It is <u>highly recommended</u> to use the DASA:STEP or DASA:ROTate commands to rotate the array rather than the DASA:CTRL command, as the DASA:CTRL command is for debugging purposes.* | Turn the DASA array one step clockwise (CW):<br>*DASA:CTRL OPEN*<br>*DASA:CTRL APOS*<br>*DASA:CTRL BPOS*<br>*DASA:CTRL ANEG*<br>*DASA:CTRL BNEG*<br>*DASA:CTRL AOFF*<br>*DASA:CTRL BOFF*<br>*DASA:CTRL CLOSE*<br>Turn the DASA array one step counter-clockwise (CCW):<br>*DASA:CTRL OPEN*<br>*DASA:CTRL BPOS*<br>*DASA:CTRL APOS*<br>*DASA:CTRL BNEG*<br>*DASA:CTRL ANEG*<br>*DASA:CTRL BOFF*<br>*DASA:CTRL AOFF*<br>*DASA:CTRL CLOSE*<br>Output current flowing through A and B drive circuits:<br>*DASA:CTRL SENSE* |
| DASA:NPR <ENable \| DISable> | Turns the -1.8v power rail ON or OFF to the temperature sensing circuitry to increase the temperature ranges on the sensors.<br>*Note: In order to read the temperature sensors on the DASA, the NPR must be enabled via **DASA:NPR ENable** command.* | Enable the NPR:<br>*DASA:NPR ENABLE*<br>Read Temperature sensor 2 via serial debug: |

[4] The Motor drive circuit can be damaged by external control on J5.2 by driving A+ and A- at the same time or B+ and B- at the same time.

| Commands | Explanation | Examples |
|---|---|---|
| | | *DASA:NPR ENABLE*<br>*DASA:TEL? 7,ASCII* |
| DASA:TELemetry? <N>, [DATA \| ASCII \| NAME \| LENGTH] | Request telemetry for a specific field N in the given format. The formats are:<br>• DATA: Return telemetry in binary representation. *Not available over user debug terminal.*<br>• ASCII: Return telemetry in ASCII representation.<br>• NAME: The name of the telemetry item e.g. *SupMCU Uptime*<br>• LENGTH: Length (in bytes) of telemetry in its binary representation.<br>Note: I*f no format representation is specified, DATA is used as the default.*<br>*See DASA Telemetry section for more detail.* | Query DASA for Drive Motor Status in ASCII format:<br>*DASA:TEL? 1,ASCII*<br>Query DASA for drive motor position in binary format<br>*DASA:TEL? 0*<br>Query DASA for name of third telemetry item<br>*DASA:TEL? 2,NAME* |

## DASA on STM Board

The DASA can be emulated on the Pumpkin SupMCU Test Module (STM), which provides the same I$^2$C/Debug UART interface, and command/telemetry set as a flight-model DASA module. However, since the STM board does not have all of the peripheral functions as the DASA FM module, the function is emulated as:

1. The STM is programmed to raise a stall condition when:
   a) Attempting to rotate **clockwise** past Motor Drive Step Position 1000
   b) Attempting to rotate **counter-clockwise** past Motor Drive Step Position -1000
   *This configuration allows for testing code-paths that check the stalled status bit in the telemetry "Drive Motor Status".*
2. The STM has the following configuration for Drive motor outputs:
   a) **A+** Drive is on **H3.26**
   b) **A-** Drive is on **H3.27**
   c) **B+** Drive is on **H3.2**
   d) **B-** Drive is on **H3.4**
   *This allows the user to break these pins out on the STM board and wire them to an external motor drive board to test DASA:STEP commands.*

In addition to emulated function, there are several LED status indicator lights (labeled LED2 – LED9) on the STM board, these LED's follow the states outlined:

## DASA STM LED Status Indicators

Purpose of LED2-LED9 on STM board with DASA firmware.

| LED Label | Purpose |
|---|---|
| LED2 | Illuminated = The DASA array is currently rotating clockwise.<br>Not Illuminated = The DASA array is not rotating OR rotating counter-clockwise. |
| LED3 | Illuminated = The DASA array is currently rotating counter-clockwise.<br>Not Illuminated = The DASA array is not rotating OR rotating clockwise. |
| LED4 | Illuminated = The DASA array has been homed.<br>Not Illuminated = The DASA array is not homed.<br>LED4 mirrors the state of Bit 2 in "Drive Motor Status" (DASA:TEL? 1) |
| LED5 | Not used. |
| LED6 | Illuminated = TiNi is energized.<br>Not Illuminated = TiNi is not energized.<br>LED6 mirrors the state of "TiNi Pin-Puller Status" (DASA:TEL? 2) |
| LED7 | Illuminated = Negative power rail is ON.<br>Not Illuminated = Negative power rail is OFF.<br>LED6 mirrors the state of "Negative Power Rail Status" (DASA:TEL? 3) |
| LED8 | Not used. |
| LED9 | Not used. |

## Telemetry

Each SupMCU contains a telemetry table data structure comprised of SupMCU parent fields and module specific fields. This data structure is updated when telemetry is requested. Each field of the table can be queried by sending the corresponding field index when sending a SCPI Telemetry command.

Sending/writing a telemetry request via I2C should be followed by an I2C read to get the queried data. Pumpkin SupMCU modules have 4 data parameters

- data: telemetry data
- name: name of field in a 32 byte ASCII string
- length: 2 byte unsigned integer value of data length for I2C read
- ascii: Telemetry data in ASCII format (max 128 bytes)

The full list of SupMCU and Module telemetry fields are in the following tables. Examples of how to correctly send and receive telemetry requests and responses are provided in the sections following the telemetry field tables. These sections detail the header lengths associated with telemetry requests, and data lengths.

**Format Specifiers**

A list of format specifiers and their corresponding types from *stdint.h* std library header.

| Format | C Standard Type |
|---|---|
| $u_8$ , $u_{16}$, $u_{32}$, $u_{64}$ | uint8_t, uint16_t, uint32_t, uint64_t |
| $i_8$,$i_{16}$,$i_{32}$,$i_{64}$ | int8_t, int16_t, int32_t, int64_t |
| $f_{32}$, $f_{64}$ | float, double |
| hex | uint8_t (*usually encoding a bit-field*) |
| string/Ascii | char* |

## SupMCU Telemetry

| Commands | Parameters | | |
|---|---|---|---|
| | Name | Length [bytes] | Format |
| SUP:TEL? 0, [param] | Firmware Version | 48 | string |
| SUP:TEL? 1, [param] | SCPI Commands Parsed | 8 | $u_{64}$ |
| SUP:TEL? 2, [param] | SCPI Errors (*SCPI commands rejected*) | 8 | $u_{64}$ |
| SUP:TEL? 4, [param] | SupMCU CPU Selftests | 22 | $u_{64}$, $u_{64}$, $i_{16}$, $i_{16}$, $i_{16}$ |
| SUP:TEL? 5, [param] | Elapsed time in seconds | 8 | $u_{64}$ |
| SUP:TEL? 6, [param] | Context Switching Occurred | 8 | $u_{64}$ |
| SUP:TEL? 7, [param] | Idling Hooks Remaining | 8 | $u_{64}$ |
| SUP:TEL? 8, [param] | MCU Load | 4 | $f_{32}$ |
| SUP:TEL? 9, [param] | Serial Number | 2 | $u_{16}$ |
| SUP:TEL? 10,[param] | $I^2C$ Address | 1 | $u_8$ |
| SUP:TEL? 11,[param] | Oscillator Tuning Value | 1 | $i_8$ |
| SUP:TEL? 12,[param] | NVM Write Cycles | 2 | $u_{16}$ |
| SUP:TEL? 13, [param] | PIC24 RCON reset cause | 2 | $u_{16}$ |
| SUP:TEL? 14, [param] | Number of Telemetry Items<br>[SupMCU, Module] | 4 | $u_{16}$, $u_{16}$ |
| SUP:TEL? 15, [param] | SupMCU (CTMU) Temperature [0.1K] | 2 | $u_{16}$ |

| SupMCU CPU Selftests | |
|---|---|
| Describes the results of the six self-test routines provided by Microchip[5].<br>Note: *All values are 0 until self-tests are ran via SUP:SELF command. See SupMCU Commands for more detail.* | |
| **Explanation** | **Bytes** |
| Number of passed tests ($u_{64}$) | 0-7 |
| Number of failed tests ($u_{64}$) | 8-15 |
| Last test number ($u_{16}$)<br>Last test number is an integer between [0,6] | 16-17 |
| Last failed test number ($u_{16}$)<br>Last failed test number is an integer between [0,6] | 18-19 |
| Last test's failed result ($u_{16}$)<br>The test result value for the last failed test. | 20-21 |

*NOTE:* SupMCU logs the events as they happen. Therefore the telemetry fields 2 and 4 will only be logged once a SCPI error has occurred/self testshave been started. A telemetry request sent before these events ever happen will not return any data.

[5] Note there are the Self-test libraries provided by Microchip, of which only test subsets 1-6 are available. For more information visit https://www.microchip.com/SWLibraryWeb/product.aspx?product=16-bit%20CPU%20Self-test%20Library

## BIM Telemetry

| Commands | Parameters | | |
|---|---|---|---|
| | Name | Length [bytes] | Format |
| BIM:TEL? 0, [param] | Temperature [0.1 K] | 12 | $u_{32}$,$u_{32}$,$u_{32}$,$u_{32}$,$u_{32}$,$u_{32}$ |
| BIM:TEL? 1, [param] | UART Status | 6 | $i_{16}$, $i_{16}$, $i_{16}$ |
| BIM:TEL? 2, [param] | IMU Data | 12 | $i_{16}$, $i_{16}$, $i_{16,}$ $i_{16}$, $i_{16}$, $i_{16}$ |
| BIM:TEL? 3, [param] | Tini pin-puller status | 1 | $u_8$ |

### Temperature [0.1 K]

The temperature readings in 0.1 kelvin from the 6 temperature sensors on the BIM.
NOTE: *The temperature sensors must be powered for valid readings. Make sure BIM:TEMP:POW ON command has been sent before reading sensor data.*

| Bytes | Explanation |
|---|---|
| 0-1 | Sensor reading from sensor #1 ($u_{16}$) |
| 2-3 | Sensor reading from sensor #2 ($u_{16}$) |
| 4-5 | Sensor reading from sensor #3 ($u_{16}$) |
| 6-7 | Sensor reading from sensor #4 ($u_{16}$) |
| 8-9 | Sensor reading from sensor #5 ($u_{16}$) |
| 10-11 | Sensor reading from sensor #6 ($u_{16}$) |

<table>
<tr><th colspan="2">TiNi Status</th></tr>
<tr><td colspan="2">The power state of the TiNi Pin puller</td></tr>
<tr><th>Bytes</th><th>Explanation</th></tr>
<tr><td>0-1</td><td>The power status of the TiNi Pin Puller ($u_8$).<br>Only indicates if the TiNi is energized.<br>1 = Energized.<br>0 = Not energized.</td></tr>
</table>

## GPSRM Telemetry

| Commands | Parameters | | |
|---|---|---|---|
| | Name | Length [bytes] | Format |
| GPS:TEL? 0, [param] | Power Status | 2 | hex |
| GPS:TEL? 1, [param] | NMEA String<br>**(not functional)**[6] | 512 | Ascii |
| GPS:TEL? 2, [param] | Orbit Propagator Results | 56 | $f_{64}$, $f_{64}$, $f_{64}$, $f_{64}$, $f_{64}$, $f_{64}$, $f_{64}$ |
| GPS:TEL? 3, [param] | Novatel OEM615 Power [mW] | 8 | $u_{16}$, $u_{16}$, $u_{16}$, $u_{16}$ |

| Power Status (Bit-field) | |
|---|---|
| State values pertaining to GPS power, reset, position valid, and pass through. | |
| **Bit** | **Explanation** |
| 0 | 1 = Power is applied to OEM615/719 |
| 1 | 1 = RESET signal is applied to OEM615/719 |
| 2 | 1 = Position Valid pin on OEM615/719 is HIGH (ON) |
| 3 | 1 = Pass through is enabled on GPSRM. |

[6] The NMEA string currently returns garbage data and is not representative of current NMEA output. Do not use until a firmware update fixes the issue.

## Orbit Propagator Results

Results from Vinti7 or SGP4 orbit propagators. See GPSRM SCPI Commands for usage. All zeros until propagator has ran. Note: *Invalid propagator input will set all values to 0 for propagator results.*

| Bytes | Explanation |
|---|---|
| 0-7 | Time t1 (Vinti7) or time_delta from initial time (TLE SGP4) ($f_{64}$) |
| 8-15 | X-position of propagated orbit ($f_{64}$) |
| 16-23 | Y-position of propagated orbit ($f_{64}$) |
| 24-31 | Z-position of propagated orbit ($f_{64}$) |
| 32-39 | X-velocity of propagated orbit ($f_{64}$) |
| 40-47 | Y-velocity of propagated orbit($f_{64}$) |
| 48-55 | Z-velocity of propagated orbit($f_{64}$) |

## Novatel OEM615/719 Power

Power draw readings of the OEM615/719 voltage supply rails. Values are measured in milliwatts.
Note: *See GPSRM data sheet for more information on voltage rails.*

| Bytes | Explanation |
|---|---|
| 0-1 | Power draw reading in milliwatts from VCC_SYS ($u_{16}$) |
| 2-3 | Power draw reading in milliwatt from +5V_SYS ($u_{16}$) |
| 4-5 | Power draw reading in milliwatts from 3V3_USB ($u_{16}$) |
| 6-7 | Power draw reading in milliwatts from +5V_USB ($u_{16}$) |

## AIM2 Telemetry

| Commands | Parameters | | |
|---|---|---|---|
| | Name | Length [bytes] | Format |
| AIM:TEL? 0, [param] | GPS Status | 2 | hex |
| AIM:TEL? 1, [param] | NMEA String<br>**(not functional)**[7] | 512 | Ascii |
| AIM:TEL? 2, [param] | Orbit Propagator | 56 | $f_{64}$, $f_{64}$, $f_{64}$, $f_{64}$, $f_{64}$, $f_{64}$, $f_{64}$ |
| AIM:TEL? 3, [param] | OEM615/719 Power | 8 | $u_{16}$, $u_{16}$, $u_{16}$, $u_{16}$ |
| AIM:TEL? 4, [param] | GPS UART Status | 1 | $u_{8}$ |
| AIM:TEL? 5, [param] | GPS Event Pin Status | 1 | hex |
| AIM:TEL? 6, [param] | GPS Power Supply Status | 1 | hex |
| AIM:TEL? 7, [param] | ADCS UART Status | 1 | hex |
| AIM:TEL? 8, [param] | FTDI Chip Status | 1 | hex |
| AIM:TEL? 9, [param] | ADCS Reset and Power Status | 1 | hex |
| AIM:TEL? 10, [param] | WDT Period | 4 | $u_{32}$ |

[7] The NMEA string telemetry item currently returns garbage data and is not representative of current NMEA output. Do not use until a firmware update fixes the issue.

## AIM Telemetry Breakdown

**GPS Status (Bit-field)**

State values pertaining to GPS power, reset, position valid, and pass through.

| Bit | Explanation |
|---|---|
| 0 | 1 = Power is applied to OEM615/719 |
| 1 | 1 = RESET signal is applied to OEM615/719 |
| 2 | 1 = Position Valid pin on OEM615/719 is HIGH (ON) |
| 3 | 1 = Pass through is enabled on GPSRM. |

## Orbit Propagator Results

Results from Vinti7 or SGP4 orbit propagators. See GPSRM SCPI Commands for usage. All zeros until propagator has ran. Note: *Invalid propagator input will set all values to 0 for propagator results.*

| Bytes | Explanation |
|---|---|
| 0-7 | Time t1 (Vinti7) or time_delta from initial time (TLE SGP4) ($f_{64}$) |
| 8-15 | X-position of propagated orbit ($f_{64}$) |
| 16-23 | Y-position of propagated orbit ($f_{64}$) |
| 24-31 | Z-position of propagated orbit ($f_{64}$) |
| 32-39 | X-velocity of propagated orbit ($f_{64}$) |
| 40-47 | Y-velocity of propagated orbit($f_{64}$) |
| 48-55 | Z-velocity of propagated orbit($f_{64}$) |

## OEM615/719 Power

Power draw readings of the OEM615/719 voltage supply rails. Values are measured in milliwatts.
Note: *See AIM-2 Data sheet for more information on voltage rails.*

| Bytes | Explanation |
|---|---|
| 0-1 | Power draw reading in milliwatts from VCC_SYS ($u_{16}$) |
| 2-3 | Power draw reading in milliwatts from +3V3_USB ($u_{16}$) |
| 4-5 | Not used ($u_{16}$) |
| 6-7 | Not used ($u_{16}$) |

## GPS UART Status (Bit-field)

The state of the OEM615/719 UART Communications port and if UART pass through is ON or OFF. Use *GPS:COMM* and *GPS:PASS* commands to change these states.

| Bit | Explanation |
|---|---|
| 0 - 2 | GPS UART Communications is on:<br>0 = NONE<br>1 = UART0<br>2 = UART1<br>3 = UART2<br>4 = UART3<br>6 = SUP1<br>7 = SUP2 |
| 3 - 6 | RESERVED 0 |
| 7 | 1 = Pass through is on. |

## GPS Event Status (Bit-field)

The state of the EVENT1 and EVENT2 pins on the OEM615/719

| Bit | Explanation |
|---|---|
| 0 | 1 = OEM615/719 EVENT1 is HIGH (ON) |
| 1 | 1 = OEM615/719 EVENT2 is HIGH (ON) |

| GPS Power Supply Status (Bit-field) | |
|---|---|
| Status of voltage rails that supply 3v3 volts to OEM615/719. Bits are HIGH (1) if **NO** current or faults are detected. | |
| **Bit** | **Explanation** |
| 0 | 0 = Current is flowing from VCC_SYS to OEM615/719 3v3 Rail. |
| 1 | 0 = Current is flowing from 3V3_USB to OEM615/719 3v3 Rail. |
| 2 | 0 = Over current/Thermal fault on supply from VCC_SYS. |
| 3 | 0 = Over current/Thermal fault on supply from 3V3_USB. |

| ADCS UART Status (Bit-field) | |
|---|---|
| The state of the UART communication to the ADCS Unit. Use *AIM:ADCS:COMM* and *AIM:ADCS:PASS* commands to manipulate the state of UART communications to MAI-400. | |
| **Bit** | **Explanation** |
| 0 - 2 | ADCS UART Communications is on:<br>0 = NONE<br>1 = UART0<br>2 = UART1<br>3 = UART2<br>4 = UART3<br>6 = SUP1<br>7 = SUP2 |
| 3 - 6 | RESERVED 0 |
| 7 | 1 = Passthrough is on. |

### AIM FTDI Status

The state of the FTDI chip used to update firmware on the MAI-400 unit.

| Bit | Explanation |
|---|---|
| 0 | 1 = FTDI power on |
| 1 | 1 = FTDI suspend enabled |
| 2 - 7 | RESERVED 0 |

### AIM ADCS Status

The state of the MAI-400 unit. Contains if the MAI-400 is ON and if the RESET line is ON.

| Bit | Explanation |
|---|---|
| 0 | 1 = MAI ADCS is on |
| 1 | 1 = MAI ADCS Reset is on |
| 2 - 7 | RESERVED 0 |

## EPSM/DCPS Telemetry

| Commands | Parameters | | |
|---|---|---|---|
| | Name | Length [bytes] | Format |
| EPSM:TEL? 0, [param] **EPSM Only** | SAI1 Converter Data | 13 | $u_{16}$, $u_{16}$, $i_{16}$, $i_{16}$, hex, $i_{32}$ |
| EPSM:TEL? 1, [param] **EPSM Only** | SAI2 Converter Data | 13 | $u_{16}$, $u_{16}$, $i_{16}$, $i_{16}$, hex, $i_{32}$ |
| EPSM:TEL? 2, [param] **EPSM Only** | SAI3 Converter Data | 13 | $u_{16}$, $u_{16}$, $i_{16}$, $i_{16}$, hex, $i_{32}$ |
| EPSM:TEL? 3, [param] **EPSM Only** | SAI4 Converter Data | 13 | $u_{16}$, $u_{16}$, $i_{16}$, $i_{16}$, hex, $i_{32}$ |
| EPSM:TEL? 4, [param] **EPSM Only** | SAI5 Converter Data | 13 | $u_{16}$, $u_{16}$, $i_{16}$, $i_{16}$, hex, $i_{32}$ |
| EPSM:TEL? 5, [param] **EPSM Only** | SAI6 Converter Data | 13 | $u_{16}$, $u_{16}$, $i_{16}$, $i_{16}$, hex, $i_{32}$ |
| EPSM:TEL? 6, [param] **EPSM Only** | BAT1 Converter Data | 13 | $u_{16}$, $u_{16}$, $i_{16}$, $i_{16}$, hex, $i_{32}$ |
| EPSM:TEL? 7, [param] **EPSM Only** | BAT2 Converter Data | 13 | $u_{16}$, $u_{16}$, $i_{16}$, $i_{16}$, hex, $i_{32}$ |
| EPSM:TEL? 8, [param] | 3.3V Converter Data | 13 | $u_{16}$, $u_{16}$, $i_{16}$, $i_{16}$, hex, $i_{32}$ |
| EPSM:TEL? 9, [param] | 5V Converter Data | 13 | $u_{16}$, $u_{16}$, $i_{16}$, $i_{16}$, hex, $i_{32}$ |
| EPSM:TEL? 10, [param] | 12V Converter Data | 13 | $u_{16}$, $u_{16}$, $i_{16}$, $i_{16}$, hex, $i_{32}$ |
| EPSM:TEL? 11, [param] | AUX Converter Data<br>(VBATT **on DCPS**) | 13 | $u_{16}$, $u_{16}$, $i_{16}$, $i_{16}$, hex, $i_{32}$ |
| EPSM:TEL? 12, [param] **EPSM Only** | VREF1 Data | 4 | $u_{16}$, $u_{16}$ |
| EPSM:TEL? 13, [param] **EPSM Only** | VREF2 Data | 4 | $u_{16}$, $u_{16}$ |
| EPSM:TEL? 14, [param] **EPSM Only** | VREF3 Data | 4 | $u_{16}$, $u_{16}$ |
| EPSM:TEL? 15, [param] **EPSM Only** | VREF4 Data | 4 | $u_{16}$, $u_{16}$ |
| EPSM:TEL? 16, [param] **EPSM Only** | VREF5 Data | 4 | $u_{16}$, $u_{16}$ |
| EPSM:TEL? 17, [param] **EPSM Only** | HSK -5V REF Data | 4 | $u_{16}$, $u_{16}$ |
| EPSM:TEL? 18, [param] **EPSM Only** | Core 1.5V REF Data | 4 | $u_{16}$, $u_{16}$ |
| EPSM:TEL? 19, [param] **EPSM Only** | HSK 3.3V REF Data | 4 | $u_{16}$, $u_{16}$ |
| EPSM:TEL? 20, [param] **EPSM Only** | HSK 5V REF Data | 4 | $u_{16}$, $u_{16}$ |
| EPSM:TEL? 21, [param] **EPSM Only** | HSK 10V REF Data | 4 | $u_{16}$, $u_{16}$ |
| EPSM:TEL? 22, [param] **EPSM Only** | RING1 REF Data | 4 | $u_{16}$, $u_{16}$ |

| | | | |
|---|---|---|---|
| EPSM:TEL? 23, [param] **EPSM Only** | RING2 REF Data | 4 | $u_{16}$, $u_{16}$ |
| EPSM:TEL? 24, [param] **EPSM Only** | RING3 REF Data | 4 | $u_{16}$, $u_{16}$ |
| EPSM:TEL? 25, [param] **EPSM Only** | SAI1 Temperature Data | 2 | $i_{16}$ |
| EPSM:TEL? 26, [param] **EPSM Only** | SAI2 Temperature Data | 2 | $i_{16}$ |
| EPSM:TEL? 27, [param] **EPSM Only** | SAI3 Temperature Data | 2 | $i_{16}$ |
| EPSM:TEL? 28, [param] **EPSM Only** | SAI4 Temperature Data | 2 | $i_{16}$ |
| EPSM:TEL? 29, [param] **EPSM Only** | SAI5A Temperature Data | 2 | $i_{16}$ |
| EPSM:TEL? 30, [param] **EPSM Only** | SAI5B Temperature Data | 2 | $i_{16}$ |
| EPSM:TEL? 31, [param] **EPSM Only** | SAI6A Temperature Data | 2 | $i_{16}$ |
| EPSM:TEL? 32, [param] **EPSM Only** | SAI6B Temperature Data | 2 | $i_{16}$ |
| EPSM:TEL? 33, [param] **EPSM Only** | SNS VUSB REF Data | 2 | $u_{16}$ |
| EPSM:TEL? 34, [param] **EPSM Only** | SNS -RESET Data | 2 | $u_{16}$ |
| EPSM:TEL? 35, [param] **EPSM Only** | SNS 4096mV REF Data | 2 | $u_{16}$ |
| EPSM:TEL? 36, [param] **EPSM Only** | *Reserved* | - | - |
| EPSM:TEL? 37, [param] **EPSM Only** | FPGA Version | 2 | hex, hex |

## EPSM/DCPS Telemetry Breakdown

<table>
<tr><th colspan="2">SAI1 - SAI6 Converter Data,</th></tr>
<tr><td colspan="2">The current voltage, voltage set point, current, current limit and status. <i>Note: The format is the same for SAI1 to SAI6 converters.</i></td></tr>
<tr><th>Bytes</th><th>Explanation</th></tr>
<tr><td>0-1</td><td>Voltage reading in mV on converter ($u_{16}$).</td></tr>
<tr><td>2-3</td><td>Voltage maximum set point in mV on converter ($u_{16}$).</td></tr>
<tr><td>4-5</td><td>Current reading in mA on converter ($i_{16}$).</td></tr>
<tr><td>6-7</td><td>Current limit in mA on converter ($i_{16}$).</td></tr>
<tr><td>8</td><td>Status of SAI. Possible values are:<br>• 0x00 – Enabled.<br>• 0x01 – Disabled.<br>• 0x02 – Current limit tripped, enabled.<br>• 0x03 – Current limit tripped, disabled.</td></tr>
<tr><td>9-13</td><td>Power in mW ($i_{32}$)</td></tr>
</table>

## 3.3V, 5V, 12V, AUX, BAT1, BAT2 Converter Data

The current voltage, voltage set point, current, current limit and status for converters. *Note: This telemetry format is the same for 3.3V, 5V, 12V, AUX, BAT1 and BAT2 converters.*

| Bytes | Explanation |
|---|---|
| 0-1 | Voltage reading in mV on converter ($u_{16}$). |
| 2-3 | Voltage maximum set point in mV on converter ($u_{16}$). |
| 4-5 | Current reading in mA on converter ($i_{16}$). |
| 6-7 | Current limit in mA on converter ($i_{16}$). |
| 8 | Status of voltage rail. Possible values are:<br>• 0x00 – Enabled.<br>• 0x01 – Disabled.<br>• 0x02 – Current limit tripped, enabled.<br>• 0x03 – Current limit tripped, disabled. |
| 9-13 | Power in mW ($i_{32}$) |

## FPGA Version

The firmware version on the EPSM FPGA.

| Bytes | Explanation |
|---|---|
| 0 | Lower byte of FPGA version. |
| 1 | Upper byte of FPGA version. |

## VREF1-5, HSK -5V/5V/3.3V/10V, Core 1.5V, RING1-3 Reference Data

The breakdown of different voltage references read from EPSM FPGA.

| Bytes | Explanation |
|---|---|
| 0-1 | Current voltage of reference. ($u_{16}$) |
| 2-3 | The setpoint of the voltage reference. ($u_{16}$). |

## SAI1-6B Temperature Data Breakdown

The breakdown of temperature sensor readout on SAI1-6B ports.

| Bytes | Explanation |
|---|---|
| 0-1 | The reading on temperature sensor ADC in [0.1 K] ($i_{16}$). |

## RHM Telemetry

| Commands | Parameters | | |
|---|---|---|---|
| | Name | Length [bytes] | Format |
| RHM:TEL? 0, [param] | Lithium Power Status | 1 | hex |
| RHM:TEL? 1, [param] | Lithium UART Status | 1 | $u_8$ |
| RHM:TEL? 2, [param] | Lithium Config Status | 1 | hex |
| RHM:TEL? 3, [param] | GlobalStar Power Status | 1 | hex |
| RHM:TEL? 4, [param] | GlobalStar UART Status | 1 | $u_8$ |
| RHM:TEL? 5, [param] | GS DIN Outputs | 1 | hex |
| RHM:TEL? 6, [param] | GS Busy | 1 | hex |
| RHM:TEL? 7, [param] | WDT Period | 4 | $u_{32}$ |
| RHM:TEL? 8, [param] | GS Transmission Status | 1 | hex |
| RHM:TEL? 9, [param] | GS Transmissions Sent | 4 | $u_{32}$ |
| RHM:TEL? 10, [param] | GS Transmissions Failed | 4 | $u_{32}$ |

## RHM Telemetry Breakdown

| RHM Lithium Power Status (Bit-field) | |
|---|---|
| State of Lithium Radio power amplifier (PA) and microcontroller unit (MCU) Power. Use *RHM:LI:PAPOW, RHM:LI:MCUPOW* and *RHM:LI:POW* commands to manipulate state. | |
| **Bit** | **Explanation** |
| 0 | 1 = 3v3 supply (MCU) is on, 0 otherwise |
| 1 | 1 = transmitter power supply (PA) is on, 0 otherwise |
| 2 - 7 | RESERVED 0 |

### RHM Lithium UART Status (Bit-field)

State of Lithium Radio UART communications. Represents current communication channel and if UART pass through is ON or OFF. Control via *RHM:LI:COMM* and *RHM:LI:PASS* commands.

| Bit | Explanation |
|---|---|
| 0 - 2 | 0 = UART0<br>1 = UART1<br>2 = UART2<br>3 = UART3<br>4 = UART4<br>5 = SupMCU UART1<br>6 = SupMCU UART2 |
| 3 - 6 | RESERVED 0 |
| 7 | 1 = UART pass through enabled, 0 otherwise |

### RHM Lithium Config Status (Bit-field)

State of the configure pins on the Lithium radio. Controlled via the *RHM:LI:CONF* command.

| Bit | Explanation |
|---|---|
| 0 | Status of CONFIG1 pin |
| 1 | Status of CONFIG2 pin |
| 2 - 7 | RESERVED 0 |

### RHM GlobalStar Power Status (Bit-field)

Reporting power status for the GlobalStar Simplex radio. Controlled via *RHM:GS:POW* command.

| Bit | Explanation |
|---|---|
| 0 | 1 = GlobalStar power is on, 0 otherwise |
| 1 - 7 | RESERVED 0 |

## RHM GlobalStar UART Status (Bit-field)

State of UART Communication and transmission status for GlobalStar Simplex radio. Reports current UART channel, pass through enabled, and transmission status. Use *RHM:GS:COMM, RHM:GS:PASS* to control UART state, and *RHM:GS:SEND* to control transmission state (bits 4-6).
Note: *Transmission status (bits 4-6) is ONLY controlled by RHM:GS:SEND command, it DOES NOT react to external devices (outside of RHM SupMCU) controlling the Simplex radio.*

| Bit | Explanation |
|---|---|
| 0 - 3 | 0 = NONE<br>1 = UART0<br>2 = UART1<br>3 = UART2<br>4 = UART3<br>5 = UART4<br>6 = SupMCU UART1<br>7 = SupMCU UART2 |
| 4 - 6 | 0 = No transmission requested<br>1 = Transmission Pending<br>2 = Waiting for a busy radio<br>3 = Transmitting<br>4 = The preceding transmission was successful<br>5 = The preceding transmission failed |
| 7 | 1 = UART Enabled, 0 otherwise |

| RHM GlobalStar DIN Outputs (Bit-field) | |
| --- | --- |
| The state of the Digital Input (DIN) pins on GlobalStar Simplex radio. Use *RHM:GS:DIN* to control state of pins. | |
| **Bit** | **Explanation** |
| 0 | State of GlobalStar DIN1 pin |
| 1 | State of GlobalStar DIN2 pin |
| 2 - 6 | RESERVED 0 |
| 7 | 1 = DIN pins on GlobalStar radio are disabled, 0 enabled |

| RHM GlobalStar Busy (Bit-field) | |
| --- | --- |
| State of BUSY line on GlobalStar Simplex radio. This is raised HIGH (ON) when a transmission starts, or during boot of the GlobalStar Simplex. See GlobalStar Simplex ICD for information on when BUSY line is HIGH. | |
| **Bit** | **Explanation** |
| 0 | 1 = Busy, 0 otherwise |
| 1 - 7 | RESERVED 0 |

| GS Transmission Status | |
| --- | --- |
| State of the GS transmission triggered by the `RHM:GS:SEND` SCPI command. | |
| **Value** | **Explanation** |
| 0 | No Transmission |
| 1 | Waiting for GlobalStar BUSY to be low |
| 2 | The GlobalStar transmission is ready |
| 3 | The beacon is being sent to the GlobalStar radio |
| 4 | Last transmission was successful (GS radio sent ACK) |
| 5 | Last transmission was unsuccessful (No reply/invalid reply/NACK) |

## PIM Telemetry

| Commands | Parameters | | |
| --- | --- | --- | --- |
| | Name | Length [bytes] | Format |

| | | | |
|---|---|---|---|
| PIM:TEL? 0, [param] | Channel Currents (mA) | 8 | $u_{16}$, $u_{16}$, $u_{16}$, $u_{16}$ |
| PIM:TEL? 1, [param] | Channel Shunt Resistor Values (μΩ) | 8 | $u_{16}$, $u_{16}$, $u_{16}$, $u_{16}$ |
| PIM:TEL? 2, [param] | Channel Current Limits (mA) | 8 | $u_{16}$, $u_{16}$, $u_{16}$, $u_{16}$ |
| PIM:TEL? 3, [param] | PIM Status Register | 1 | hex |
| PIM:TEL? 4, [param] | PIM Overcurrent event log (mA) | 8 | $u_{16}$, $u_{16}$, $u_{16}$, $u_{16}$ |
| PIM:TEL? 5, [param] | Channel Voltages (mV) | 8 | $u_{16}$, $u_{16}$, $u_{16}$, $u_{16}$ |

## PIM Telemetry Breakdown

### Channel Currents (mA)

Current readings from PIM Power ports #1-4 in miliamps (mA). If current readings are beyond ±5% error, make sure SNS_CH#_I calibration is set correctly for PIM module (controlled by *PIM:CAL* commands).

| Bytes | Explanation |
|---|---|
| 0-1 | Current reading from Port #1 ($u_{16}$) in mA |
| 2-3 | Current reading from Port #2 ($u_{16}$) in mA |
| 4-5 | Current reading from Port #3 ($u_{16}$) in mA |
| 6-7 | Current reading from Port #4 ($u_{16}$) in mA |

### Channel Shunt Resistor Values (μΩ)

Current shunt resistor values for PIM power ports in micro-ohms ( μΩ). Controlled by *PIM:NVM SHUNT.*

| Bytes | Explanation |
|---|---|
| 0-1 | Shunt resistor value for Port #1 ($u_{16}$) in μΩ |
| 2-3 | Shunt resistor value for Port #2 ($u_{16}$) in μΩ |
| 4-5 | Shunt resistor value for Port #3 ($u_{16}$) in μΩ |
| 6-7 | Shunt resistor value for Port #4 ($u_{16}$) in μΩ |

### Channel Current Limits (mA)

Current limits for PIM power ports in mA. Controlled by PIM:NVM I_LIMIT command.

| Bytes | Explanation |
|---|---|
| 0-1 | Current limit value for Port #1 ($u_{16}$) in mA |
| 2-3 | Current limit value for Port #2 ($u_{16}$) in mA |
| 4-5 | Current limit value for Port #3 ($u_{16}$) in mA |
| 6-7 | Current limit value for Port #4 ($u_{16}$) in mA |

## PIM Status Register (Bit-Field)

State of power to Ports #1-4, if Ethernet is ON, if PIM button is pressed, and Revision of PIM. Use *PIM:PORT:POW* to control port power, and *PIM:ETH:POW* to control Ethernet power.

| Bit | Explanation |
|---|---|
| 1 | PORT_1 Status: 1 = on |
| 2 | PORT_2 Status: 1 = on |
| 3 | PORT_3 Status: 1 = on |
| 4 | PORT_4 Status: 1 = on |
| 5 | Ethernet Power Status: 1 = on |
| 6 | Button Status: 1 = pressed |
| 7-8 | RevA PIM =0<br>RevB PIM =1<br>RevC PIM=2<br>RevD PIM=3 |

## PIM Overcurrent event log (mA)

Log of last overcurrent event in PIM. An over-current is generated when a device draws more current from a PIM port than the current limit. If no over-current event has occurred, then the value is 0. Restart the PIM device to reset the overcurrent log.

| Bytes | Explanation |
|---|---|
| 0-1 | Last over-current amount from Port #1 ($u_{16}$) in mA, otherwise 0 |
| 2-3 | Last over-current amount from Port #2 ($u_{16}$) in mA, otherwise 0 |
| 4-5 | Last over-current amount from Port #3 ($u_{16}$) in mA, otherwise 0 |
| 6-7 | Last over-current amount from Port #4 ($u_{16}$) in mA, otherwise 0 |

### Channel Voltages (mV)

Channel voltages for each PIM Power port. These are determined by the ASSY REV for the PIM module. See PIM Data sheet for more details.

| Bytes | Explanation |
|---|---|
| 0-1 | Voltage value for Port #1 ($u_{16}$) in mV |
| 2-3 | Voltage value for Port #2 ($u_{16}$) in mV |
| 4-5 | Voltage value for Port #3 ($u_{16}$) in mV |
| 6-7 | Voltage value for Port #4 ($u_{16}$) in mV |

## BSM Telemetry

| Commands | Parameters | | |
|---|---|---|---|
| | Name | Length [bytes] | Format |
| BSM:TEL? 0, [param] | Channel Currents (mA) | 10 | $u_{16}$, $u_{16}$, $u_{16}$, $u_{16}$, $u_{16}$ |
| BSM:TEL? 1, [param] | Channel Shunt Resistor Values (µΩ) | 10 | $u_{16}$, $u_{16}$, $u_{16}$, $u_{16}$, $u_{16}$ |
| BSM:TEL? 2, [param] | Channel Current Limits (mA) | 10 | $u_{16}$, $u_{16}$, $u_{16}$, $u_{16}$, $u_{16}$ |
| BSM:TEL? 3, [param] | Channel Current Offsets for linear fit | 20 | $f_{32}$, $f_{32}$, $f_{32}$, $f_{32}$, $f_{32}$ |
| BSM:TEL? 4, [param] | Channel Current Scaling Factors for linear fit | 20 | $f_{32}$, $f_{32}$, $f_{32}$, $f_{32}$, $f_{32}$ |
| BSM:TEL? 5, [param] | BSM Status Register | 1 | hex |
| BSM:TEL? 6, [param] | BSM Overcurrent event log (mA) | 10 | $u_{16}$, $u_{16}$, $u_{16}$, $u_{16}$, $u_{16}$ |
| BSM:TEL? 7, [param] | Extra Temperature Sensor 9 [0.1 deg K] | 2 | $u_{16}$ |
| BSM:TEL? 8, [param] | Extra Temperature Sensor 10 [0.1 deg K] | 2 | $u_{16}$ |
| BSM:TEL? 9, [param] | Extra Temperature Sensor 11 [0.1 deg K] | 2 | $u_{16}$ |
| BSM:TEL? 10, [param] | Extra Temperature Sensor 12 [0.1 deg K] | 2 | $u_{16}$ |
| BSM:TEL? 11, [param] | Extra Temperature Sensor Offsets for linear fit | 16 | $f_{32}$, $f_{32}$, $f_{32}$, $f_{32}$ |
| BSM:TEL? 12, [param] | Extra Temperature Sensor Scalar factors for linear fit | 16 | $f_{32}$, $f_{32}$, $f_{32}$, $f_{32}$ |

## BSM Telemetry Breakdown

### Channel Currents (mA)

Current readings from BSM Power ports #1-5 in miliamps (mA). If current readings are beyond ±5% error, make sure I_SCALE and I_OFFSET is set correctly for BSM module (controlled by BSM:*NVM I_SCALE/I_OFFSET* commands).

| Bytes | Explanation |
|---|---|
| 0-1 | Current reading from Port #1 ($u_{16}$) in mA |
| 2-3 | Current reading from Port #2 ($u_{16}$) in mA |
| 4-5 | Current reading from Port #3 ($u_{16}$) in mA |
| 6-7 | Current reading from Port #4 ($u_{16}$) in mA |
| 8-9 | Current reading from Port #5 ($u_{16}$) in mA |

### Channel Shunt Resistor Values (μΩ)

Current shunt resistor values for BSM power ports in micro-ohms ( μΩ). Controlled by BSM:*NVM SHUNT.*

| Bytes | Explanation |
|---|---|
| 0-1 | Shunt resistor value for Port #1 ($u_{16}$) in μΩ |
| 2-3 | Shunt resistor value for Port #2 ($u_{16}$) in μΩ |
| 4-5 | Shunt resistor value for Port #3 ($u_{16}$) in μΩ |
| 6-7 | Shunt resistor value for Port #4 ($u_{16}$) in μΩ |
| 8-9 | Shunt resistor value for Port #5 ($u_{16}$) in μΩ |

## Channel Current Limits (mA)

Current limits for BSM power ports in mA. Controlled by BSM:NVM I_LIMIT command.

| Bytes | Explanation |
|---|---|
| 0-1 | Current limit value for Port #1 ($u_{16}$) in mA |
| 2-3 | Current limit value for Port #2 ($u_{16}$) in mA |
| 4-5 | Current limit value for Port #3 ($u_{16}$) in mA |
| 6-7 | Current limit value for Port #4 ($u_{16}$) in mA |
| 8-9 | Current limit value for Port #5 ($u_{16}$) in mA |

## Channel Current Offsets for linear fit

Offsets used to linearly approximate the current draw from each BSM power port. Offset used in equation `current_amount=raw_current_reading*scalar_factor+offset`used calculate real current draw in mA.

| Bytes | Explanation |
|---|---|
| 0-3 | Offset used to calculate current reading for Port #1 ($f_{32}$) |
| 4-7 | Offset used to calculate current reading for Port #2 ($f_{32}$) |
| 8-11 | Offset used to calculate current reading for Port #3 ($f_{32}$) |
| 12-15 | Offset used to calculate current reading for Port #4 ($f_{32}$) |
| 16-19 | Offset used to calculate current reading for Port #5 ($f_{32}$) |

## Channel Current Scaling Factors for linear fit

Scalar factors used to linearly approximate the current draw from each BSM power port. Offset used in equation **`current_amount=raw_current_reading*scalar_factor+offset`**used calculate real current draw in mA.

| Bytes | Explanation |
|---|---|
| 0-3 | Scalar factor used to calculate current reading for Port #1 ($f_{32}$) |
| 4-7 | Scalar factor used to calculate current reading for Port #2 ($f_{32}$) |
| 8-11 | Scalar factor used to calculate current reading for Port #3 ($f_{32}$) |
| 12-15 | Scalar factor used to calculate current reading for Port #4 ($f_{32}$) |
| 16-19 | Scalar factor used to calculate current reading for Port #5 ($f_{32}$) |

## BSM Status Register (Bit-Field)

State of power to Ports #1-5, and Revision of BSM. Use BSM*:PORT:POW* to control port power.

| Bit | Explanation |
|---|---|
| 1 | PORT_1 Status: 1 = on |
| 2 | PORT_2 Status: 1 = on |
| 3 | PORT_3 Status: 1 = on |
| 4 | PORT_4 Status: 1 = on |
| 5 | PORT_5 Status: 1 = on |
| 6 | RESERVED 0 |
| 7-8 | RevA BSM =0<br>RevB BSM =1<br>RevC BSM=2<br>RevD BSM=3 |

## BSM Overcurrent event log (mA)

Log of last overcurrent event in BSM. An over-current is generated when a device draws more current from a BSM port than the current limit. If no over-current event has occurred, then the value is 0. Restart the BSM device to reset the overcurrent log.

| Bytes | Explanation |
|---|---|
| 0-1 | Last over-current amount from Port #1 ($u_{16}$) in mA, otherwise 0 |
| 2-3 | Last over-current amount from Port #2 ($u_{16}$) in mA, otherwise 0 |
| 4-5 | Last over-current amount from Port #3 ($u_{16}$) in mA, otherwise 0 |
| 6-7 | Last over-current amount from Port #4 ($u_{16}$) in mA, otherwise 0 |
| 8-9 | Last over-current amount from Port #5 ($u_{16}$) in mA, otherwise 0 |

| Extra Temperature Sensor Offsets for linear fit | |
|---|---|
| Offsets used to linearly approximate the temperature reading from the extra temperature sensors on the BSM. Offset used in equation **current_amount=raw_current_reading*scalar_factor+offset**used calculate temperature in 0.1 Kelvin | |
| **Bytes** | **Explanation** |
| 0-3 | Offset used to calculate temperature for sensor #1 ($f_{32}$) |
| 4-7 | Offset used to calculate temperature for sensor #2 ($f_{32}$) |
| 8-11 | Offset used to calculate temperature for sensor #3 ($f_{32}$) |
| 12-15 | Offset used to calculate temperature for sensor #4 ($f_{32}$) |

| Extra Temperature Sensor Scalar factors for linear fit | |
|---|---|
| Scalar factors used to linearly approximate the temperature reading from the extra temperature sensors on the BSM. Scalar factor used in equation **current_amount=raw_current_reading*scalar_factor+offset**used calculate temperature in 0.1 Kelvin | |
| **Bytes** | **Explanation** |
| 0-3 | Scalar factor used to calculate temperature for sensor #1 ($f_{32}$) |
| 4-7 | Scalar factor used to calculate temperature for sensor #2 ($f_{32}$) |
| 8-11 | Scalar factor used to calculate temperature for sensor #3 ($f_{32}$) |
| 12-15 | Scalar factor used to calculate temperature for sensor #4 ($f_{32}$) |

## BM2 Telemetry

Please refer to BM2 Manual for information on the BM2 telemetry.

## DASA Telemetry

| Commands | Parameters | | |
|---|---|---|---|
| | Name | Length [bytes] | Format |
| DASA:TEL? 0, [param] | Drive Motor Position | 4 | $i_{32}$ |
| DASA:TEL? 1, [param] | Drive Motor Angle | 4 | $f_{32}$ |
| DASA:TEL? 2, [param] | Drive Motor Status | 1 | hex |
| DASA:TEL? 3, [param] | TiNi Pin-Puller status | 1 | $u_8$ |
| DASA:TEL? 4, [param] | NPR Status [enabled, mV] | 1 | $u_8$, $u_{16}$ |
| DASA:TEL? 5, [param] | HES Status [enabled, mV] | 3 | hex, $u_{16}$ |
| DASA:TEL? 6, [param] | Temperature 1 [0.1 K] | 2 | $u_{16}$ |
| DASA:TEL? 7, [param] | Temperature 2 [0.1 K] | 2 | $u_{16}$ |
| DASA:TEL? 8, [param] | Temperature 3 [0.1 K] | 2 | $u_{16}$ |
| DASA:TEL? 9, [param] | Temperature 4 [0.1 K] | 2 | $u_{16}$ |

### DASA Drive Motor Status (Bit-field)

The current state of the articulated array. Contains direction of movement, if articulated array is moving, and if the articulated device has been homed.

| Bit | Explanation |
|---|---|
| 0 | 1 = DASA Array is moving, 0 = No movement |
| 1 | 1 = Rotating counter-clockwise, 0 = Rotating clockwise. |
| 2 | 1 = Articulated array has been homed, 0 = Articulated array is not homed. |
| 3 | Reserved. |
| 4 | 1 = HOLD the motor after stepping, 0 = RELEASE motor after step command. |
| 5-7 | Reserved. |

## DASA Panel Release Status

The current state of the solar panel release mechanism.

| Value | Explanation |
|---|---|
| 0 | The solar panel release mechanism is not energized. |
| 1 | The solar panel release mechanism **is** energized. |

## NPR Status [enabled, mV]

The current state of the negative power rail for the temperature sensors connected to the DASA module. Enable to increase range.

| Bytes | Explanation |
|---|---|
| 0 | 1 = NPR is enabled, 0 = NPR is disabled. ($u_8$) |
| 1-2 | The current reading on the NPR in mV. ($u_{16}$) |

## HES Status [enabled, mV].

The current state of the Hall-effect sensor used to home the DASA array. Controlled by *DASA:HES ENABle* and *DASA:HES DISAble.*
Note: The Hall-effect sensor will always be enabled during a *DASA:STEP/ROTATE HOME* command.

| Bytes | Explanation |
|---|---|
| 0 | 1 = HES is enabled, 0 = HES is disabled. |
| 1-2 | The measurement [in mV] of the Hall-effect sensor. This will be an invalid value if the Hall-effect sensor is disabled. ($u_{16}$) |

## Temperature 1-4 [0.1 K]

The reading of the temperature sensors 1-4 in 0.1 K.
Note: The telemetry will read \`0\` if the negative power rail to the temperature sensors are not enabled. Use *DASA:NPR ENable* to enable the negative power rail.

| Bytes | Explanation |
|---|---|
| 0-1 | The measurement [in 0.1 K] of the temperature sensor. ($u_{16}$) |

## Calibration

SupMCU modules provide a calibration system which allows the user to apply a linear slope and offset correction to select telemetry values and modules come pre-calibrated, where necessary (e.g. the PIM comes with factory calibration values installed for current measurement telemetry). Modules can be calibrated by the user for their specific application, or to refine the calibration values already entered. Each module has a set of SupMCU global calibration values, and module specific calibration values. These sets have their own version of the calibration commands.

The calibration values are represented in the telemetry format: scale factor ($f_{32}$), offset ($i_{16}$)

There are two calibration commands: Query and Set. In order to query a calibration value, use:

```
<SUP | MOD>:CAL? LINEAR,<idx>,[DATA | ASCII | NAME]
```

The calibration query response for DATA will only be sent over I2C, for the format and structure of the I2C response, see section Structure and Format of Telemetry and Calibration. **Data requests will not be sent over the Debug UART port, ASCII format must be used.**

The ASCII response will output the value of the given calibration index in ASCII string representation, which is sent over I2C and via UART Debug Port. A typical query will look like:

```
0)00:00:00:26.41 RHM SCPI|0x55> SUP:CAL? LINEAR,0,ASCII
0)00:00:00:39.52 RHM SCPI|0x55> [3:3954] 1.0000,0
```

The NAME response will output the name of the calibration index, up to a max of 32 ASCII characters. This data is sent over I2C and via UART Debug Port. A typical query will look like:

```
0)00:00:19:10.86 RHM (on STM) SCPI|0x55> SUP:CAL? LINEAR,0,NAME
0)00:00:19:12.21 RHM (on STM) SCPI|0x55> [1:115223] CTMU
```

Note, when the [DATA | ASCII | NAME] subcommand is omitted, DATA is selected as the default for the query command.

In order to set a calibration value, use:

<SUP | MOD>:CAL LINEAR,<idx>,<OFFSET, <integer> | SCALE_FACTOR, <floating-point>>

Note the SCALE_FACTOR and OFFSET values are set separately. Note that the SCALE_FACTOR entered won't always match the user input value, due to how the values are stored internally. An example to set the CTMU calibration factor to a scale factor of 0.98 and offset of -20 is:

```
0)00:00:37:29.75 RHM (on STM) SCPI|0x55> SUP:CAL LINEAR,0,SCALE_FACTOR,0.98
0) 00:00:44:33.72 supmcu_scpi_cmds: Accepted "SUP:CAL LINEAR,0,SCALE_FACTOR,0.98"
command.
0)00:00:47:00.46 RHM (on STM) SCPI|0x55> SUP:CAL LINEAR,0,OFFSET,-20
0)00:00:47:00.49 supmcu_scpi_cmds: Accepted "SUP:CAL LINEAR,0,OFFSET,-20" command.
0)00:00:47:10.23 RHM (on STM) SCPI|0x55> SUP:CAL? LINEAR,0,ASCII
[3:283025] 0.9800,-20
0)00:00:47:10.28 supmcu_scpi_cmds: Accepted "SUP:CAL? LINEAR,0,ASCII" command.
```

After setting calibration values, the NVM memory needs to be **unlocked and written** to in order for the calibration to be **persistent** on next restart of the module. Use the SUP:NVM command in the SupMCU SCPI command section in order to unlock and write NVM memory.

Below are the listing of calibration values possible as well as notes about them.

## SupMCU Calibration Values

| Index | Name | Note |
|---|---|---|
| 0 | CTMU | Applies calibration to the CTMU telemetry (SUP:TEL? 15) |

## BIM Calibration Values

| Index | Name | Note |
|---|---|---|
| 0 | Temp. sns 1 | Applies calibration to BIM temperature sensor telemetry (BIM:TEL? 0) |
| 1 | Temp. sns 2 | Applies calibration to BIM temperature sensor telemetry (BIM:TEL? 0) |
| 2 | Temp. sns 3 | Applies calibration to BIM temperature sensor telemetry (BIM:TEL? 0) |
| 3 | Temp. sns 4 | Applies calibration to BIM temperature sensor telemetry (BIM:TEL? 0) |
| 4 | Temp. sns 5 | Applies calibration to BIM temperature sensor telemetry (BIM:TEL? 0) |
| 5 | Temp. sns 6 | Applies calibration to BIM temperature sensor telemetry (BIM:TEL? 0) |

## PIM Calibration Values

| Index | Name | Note |
|---|---|---|
| 0 | SNS_CH1_I | Applies calibration to PIM current telemetry (PIM:TEL? 0) |
| 1 | SNS_CH1_V | Applies calibration to PIM voltage telemetry (PIM:TEL? 5) |
| 2 | SNS_CH2_I | Applies calibration to PIM current telemetry (PIM:TEL? 0) |
| 3 | SNS_CH2_V | Applies calibration to PIM voltage telemetry (PIM:TEL? 5) |
| 4 | SNS_CH3_I | Applies calibration to PIM current telemetry (PIM:TEL? 0) |
| 5 | SNS_CH3_V | Applies calibration to PIM voltage telemetry (PIM:TEL? 5) |
| 6 | SNS_CH4_I | Applies calibration to PIM current telemetry (PIM:TEL? 0) |
| 7 | SNS_CH4_V | Applies calibration to PIM voltage telemetry (PIM:TEL? 5) |

## AIM Calibration Values

| Index | Name | Note |
|---|---|---|
| 0 | VCC_SYS | Applies calibration to AIM OEM619/715 Power telemetry (AIM:TEL? 3) |
| 1 | 3V3_USB | Applies calibration to AIM OEM619/715 Power telemetry (AIM:TEL? 3) |

## GPSRM Calibration Values

| Index | Name | Note |
|---|---|---|
| 0 | VCC_SYS | Applies calibration to GPS OEM619/715 Power telemetry (GPS:TEL? 3) |
| 1 | 3V3_USB | Applies calibration to GPS OEM619/715 Power telemetry (GPS:TEL? 3) |
| 2 | P5V_SYS | Applies calibration to GPS OEM619/715 Power telemetry (GPS:TEL? 3) |
| 3 | P5V_USB | Applies calibration to GPS OEM619/715 Power telemetry (GPS:TEL? 3) |

## DASA Calibration Values

| Index | Name | Note |
|---|---|---|
| 0 | Temp. sensor 1 | Applies calibration to DASA Temp. 1 telemetry (DASA:TEL? 6) |
| 1 | Temp. sensor 2 | Applies calibration to DASA Temp. 2 telemetry (DASA:TEL? 7) |
| 2 | Temp. sensor 3 | Applies calibration to DASA Temp. 3 telemetry (DASA:TEL? 8) |
| 3 | Temp. sensor 4 | Applies calibration to DASA Temp. 4 telemetry (DASA:TEL? 9) |
| 4 | release current | Applies current calibration to DASA release telemetry (DASA:TEL? 3) |
| 5 | release voltage | Applies voltage calibration to DASA release telemetry (DASA:TEL? 3) |
| 6 | drive B sns | Applies calibration to DASA B-drive current telemetry (DASA:TEL? 2) |
| 7 | drive A sns | Applies calibration to DASA A-drive current telemetry (DASA:TEL? 2) |
| 8 | NPR | Applies calibration to DASA NPR temperature reference telemetry (DASA:TEL? 4). This will also affect the temperature readings as well. |
| 9 | +5V_SYS | Not used. |
| 10 | Hall-effect sensor | Applies calibration to HES sensor reading telemetry (DASA:TEL? 5). If set incorrectly, DASA homing will fail. |

## EPSM Calibration Values

| Index | Name | Note |
|---|---|---|
| 0 | SAI1_IO | Applies calibration to EPSM SAI1 temp. telemetry (EPSM:TEL? 25) |

| | | |
|---|---|---|
| 1 | SAI2_IO | Applies calibration to EPSM SAI2 temp. telemetry (EPSM:TEL? 26) |
| 2 | SAI3_IO | Applies calibration to EPSM SAI3 temp. telemetry (EPSM:TEL? 27) |
| 3 | SAI4_IO | Applies calibration to EPSM SAI4 temp. telemetry (EPSM:TEL? 28) |
| 4 | SAI5A_IO | Applies calibration to EPSM SAI5A temp. telemetry (EPSM:TEL? 29) |
| 5 | SAI5B_IO | Applies calibration to EPSM SAI5B temp. telemetry (EPSM:TEL? 30) |
| 6 | SAI6A_IO | Applies calibration to EPSM SAI6A temp. telemetry (EPSM:TEL? 31) |
| 7 | SAI6B_IO | Applies calibration to EPSM SAI6B temp. telemetry (EPSM:TEL? 32) |
| 8 | SNS_VUSB | Applies calibration to EPSM SNS VUSB telemetry (EPSM:TEL? 33) |
| 9 | SNS_nRESET | Applies calibration to EPSM SNS nRESET telemetry (EPSM:TEL? 34) |
| 10 | SNS_VREF_4096MV | Applies calibration to EPSM VREF 4096 telemetry (EPSM:TEL? 35) |
| 11 | V_BAT2 | Applies voltage calibration to EPSM BAT2 telemetry (EPSM:TEL? 7) |
| 12 | I_BAT2 | Applies current calibration to EPSM BAT2 telemetry (EPSM:TEL? 7) |
| 13 | V_SAI1 | Applies voltage calibration to EPSM SAI1 telemetry (EPSM:TEL? 0) |
| 14 | I_SAI1 | Applies current calibration to EPSM SAI1 telemetry (EPSM:TEL? 0) |
| 15 | V_SAI2 | Applies voltage calibration to EPSM SAI2 telemetry (EPSM:TEL? 1) |
| 16 | I_SAI2 | Applies current calibration to EPSM SAI2 telemetry (EPSM:TEL? 1) |
| 17 | V_RING1 | Applies calibration to EPSM VRING1 telemetry (EPSM:TEL? 22) |
| 18 | V_REF1 | Applies calibration to EPSM VREF1 telemetry (EPSM:TEL? 12) |
| 19 | V_BAT1 | Applies voltage calibration to EPSM BAT1 telemetry (EPSM:TEL? 6) |
| 20 | I_BAT1 | Applies current calibration to EPSM BAT1 telemetry (EPSM:TEL? 6) |
| 21 | V_SAI5 | Applies voltage calibration to EPSM SAI5 telemetry (EPSM:TEL? 4) |
| 22 | I_SAI5 | Applies current calibration to EPSM SAI5 telemetry (EPSM:TEL? 4) |
| 23 | V_SAI6 | Applies voltage calibration to EPSM SAI6 telemetry (EPSM:TEL? 5) |
| 24 | I_SAI6 | Applies current calibration to EPSM SAI6 telemetry (EPSM:TEL? 5) |
| 25 | V_RING2 | Applies calibration to EPSM VRING1 telemetry (EPSM:TEL? 23) |
| 26 | V_REF2 | Applies calibration to EPSM VREF2 telemetry (EPSM:TEL? 13) |

| | | |
|---|---|---|
| 27 | V_AUX | Applies voltage calibration to EPSM AUX telemetry (EPSM:TEL? 11) |
| 28 | I_AUX | Applies current calibration to EPSM AUX telemetry (EPSM:TEL? 11) |
| 29 | V_SAI3 | Applies voltage calibration to EPSM SAI3 telemetry (EPSM:TEL? 2) |
| 30 | I_SAI3 | Applies current calibration to EPSM SAI3 telemetry (EPSM:TEL? 2) |
| 31 | V_SAI4 | Applies voltage calibration to EPSM SAI4 telemetry (EPSM:TEL? 3) |
| 32 | I_SAI4 | Applies current calibration to EPSM SAI4 telemetry (EPSM:TEL? 3) |
| 33 | V_RING3 | Applies calibration to EPSM VRING3 telemetry (EPSM:TEL? 24) |
| 34 | V_REF3 | Applies calibration to EPSM VREF3 telemetry (EPSM:TEL? 14) |
| 35 | V_12V | Applies voltage calibration to EPSM 12V telemetry (EPSM:TEL? 10) |
| 36 | I_12V | Applies current calibration to EPSM 12V telemetry (EPSM:TEL? 10) |
| 37 | V_3V3 | Applies voltage calibration to EPSM 3V3 telemetry (EPSM:TEL? 8) |
| 38 | I_3V3 | Applies current calibration to EPSM 3V3 telemetry (EPSM:TEL? 8) |
| 39 | V_5V | Applies voltage calibration to EPSM 5V telemetry (EPSM:TEL? 9) |
| 40 | I_5V | Applies current calibration to EPSM 5V telemetry (EPSM:TEL? 9) |
| 41 | T_SB | Used internally. Edit only when directed by Pumpkin |
| 42 | V_REF4 | Applies calibration to EPSM VREF4 telemetry (EPSM:TEL? 15) |
| 43 | V_HSK_10V | Applies calibration to EPSM HSK 10V ref telemetry (EPSM:TEL? 21) |
| 44 | V_RAD | Applies calibration to EPSM dosimeter telemetry (EPSM:TEL? 41) |
| 45 | V_HSK_3V3 | Applies calibration to EPSM HSK 3V3 ref telemetry (EPSM:TEL? 8) |
| 46 | V_CORE_1V5 | Applies calibration to EPSM CORE 1V5 ref telemetry (EPSM:TEL? 18) |
| 47 | T_FPGA | Used internally. Edit only when directed by Pumpkin |
| 48 | V_HSK_5V | Applies calibration to EPSM HSK 5V ref telemetry (EPSM:TEL? 20) |
| 49 | V_HSK_N5V | Applies calibration to EPSM HSK N5V ref telemetry (EPSM:TEL? 17) |
| 50 | V_REF5 | Applies calibration to EPSM VREF3 telemetry (EPSM:TEL? 16) |

## DCPS Calibration Values

| Index | Name | Note |
|---|---|---|

| | | |
|---|---|---|
| 0 | I_SNS_12V | Applies current calibration to DCPS 12V telemetry (DCPS:TEL? 9) |
| 1 | I_SNS_3V3 | Applies current calibration to DCPS 3V3 telemetry (DCPS:TEL? 8) |
| 2 | I_SNS_VBATT | Applies current calibration to DCPS VBATT telemetry (DCPS:TEL? 11) |
| 3 | CADEX | Not used. |
| 4 | I_SNS_5V | Applies current calibration to DCPS 5V telemetry (DCPS:TEL? 9) |

## Structure and Format of Telemetry and Calibration

There are four different types of telemetry requests that can be made for a SupMCU Module: DATA, ASCII, LENGTH, and NAME. Each type returns a response with a five-byte header followed by a variable length payload, in little endian byte order (**LSB**). The header contains a 1-byte unsigned integer data ready flag, proceeded by a 4-byte unsigned integer timestamp. After the header is the payload, whose size in bytes and format depends on the field/query type.

| Telemetry/Calibration response for $I^2C$ | | | | | | |
|---|---|---|---|---|---|---|
| **Byte Position** | **0** | **1** | **2** | **3** | **4** | **5..N** |
| Field | Data Ready ($u_8$) | Timestamp ($u_{32}$) | | | | Telemetry Data |
| Purpose | 0 = Data invalid.<br>1 = Telemetry data valid.<br>3 = Calibration data valid. | OS Ticks since the boot of the SupMCU Module. | | | | Variable sized determined by field/query type. |

Note if the Data Ready flag **is 0**, then all response data should be **discarded.** Data not being ready is a symptom of querying a SupMCU Module **too fast**. Allow ~75-100 ms between $I^2C$ write and $I^2C$ read, as the SupMCU module needs time to gather and format the data.

If more $I^2C$ data is read from the SupMCU module than the length of the telemetry response, each extra byte read is 0x00.

If querying telemetry from the serial Debug Port on the SupMCU Module, the output is in ASCII plain-text and is formatted as follows:

**[<Ready Flag>:<Timestamp>] <ASCII Formatted Data>**

Note that a DATA type query **cannot** query telemetry data in its binary format (e.g. SUP:TEL? 1,DATA, SUP:CAL? 0,DATA), and will return the string “*CLI does not support TEL?/CAL? n,data commands!*”.

## Writing Data Queries

Querying telemetry or calibration data is a two step process requiring two independent I2C transactions (with their own start and stop conditions, repeated start conditions will not be accepted). The first step/transaction is to write your telemetry or calibration query to the slave module and the second step/transaction is to read an appropriate number of bytes from the slave module. This process is as follows:

Telemetry/calibration requests consist of the ASCII string representing the telemetry request (e.g. BIM:TEL? 1,DATA) followed by a termination byte(s):

1. 0x0A (**\n**) for requests over $I^2C$
2. 0x0A (**\n**) or 0x0D (**\r**) or 0x0D0A(**\r\n**) for requests over serial debug port.

When commanding via the serial debug port make sure the serial console being used appends the **\n** or **\r** or **\r\n** as some consoles don’t append it automatically. An example of a telemetry request for the BIM’s UART status (BIM:TEL? 1,DATA)

| BIM:TEL? 1,DATA over $I^2C$ | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Byte Position | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| ASCII | B | I | M | : | T | E | L | ? | | 1 | , | D | A | T | A | \n |
| HEX | 42 | 49 | 4d | 3a | 54 | 45 | 4c | 3f | 20 | 31 | 2c | 44 | 41 | 54 | 41 | 0a |

The response to the query will contain 10 bytes:

| BIM:TEL? 1,DATA Response | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Byte Position | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Field | Data Ready | Timestamp | | | | Telemetry Data (BIM UART Status) | | | | | |
| HEX | 01 | 00 | 00 | 20 | 00 | 00 | 01 | 00 | 01 | 00 | 01 |

The equivalent ASCII text response on the serial Debug Port is: **[1:8192] 1,1,1** asserting that all three BIM UART ports are powered.

## Telemetry LENGTH and Telemetry/Calibration NAME responses

A Telemetry/calibration request for the byte length of a telemetry item is denoted as: <MOD>:TEL? <Index>,LENGTH where <MOD> is the name of the module (e.g. SUP, BIM, PIM, GPS, AIM, …) and <Index> is the index of the telemetry item (e.g. SupMCU Uptime index is 5). The format of the LENGTH response is:

| TEL? LENGTH Response I$^2$C | | | | | | | |
|---|---|---|---|---|---|---|---|
| Byte Position | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| Field | Data Ready | Timestamp | | | | Telemetry Length in bytes (u$_{16}$) | |

The equivalent output via Serial Debug Port is **[<Data Ready>:<Timestamp>] <Telemetry Length in bytes>**(e.g. an example SUP:TEL? 5,LENGTH query returns **[1:100] 8**). **Note the calibration SCPI query commands don't include length subcommands, since it is fixed to 6 bytes always (f$_{32}$,i$_{16}$ format).**

The request for the name of the given telemetry/calibration item is denoted similarly:

<MOD>:<TEL? | CAL?> <Index>,NAME. The format of the NAME response is:

| TEL?/CAL? NAME Response I$^2$C | | | | | | |
|---|---|---|---|---|---|---|
| Byte Position | 0 | 1 | 2 | 3 | 4 | 5…36 |
| Field | Data Ready | Timestamp | | | | Telemetry name (string [32 characters]) |

The equivalent output via Serial Debug Port is **[<Data Ready>:<Timestamp>] <Telemetry name>** (e.g. an example SUP:TEL? 5,NAME query returns **[1:100] Elapsed time in seconds**

## Telemetry ASCII responses

Telemetry requests in ASCII format (<MOD>:TEL? <Index>,ASCII) are formatted using standard C language format strings (e.g. sprintf format strings). For a given telemetry format, it is converted to its equivalent format specification given the following table:

**Format Specifiers**

List of SupMCU Telemetry format specifier to backing C language format specifier

| Format | C Format specifier |
|---|---|
| $u_{8}$ | %u |
| $u_{16}$ | %u |
| $u_{32}$ | %lu |
| $u_{64}$ | %llu |
| $i_{8}$ | %u *if value is positive otherwise* -%u *after val*-1* |
| $i_{16}$ | %d |
| $i_{32}$ | %ld |
| $i_{64}$ | %lld |
| $f_{32}$ | %.4f |
| $f_{64}$ | %.4lf |
| hex | 0x%02X |
| string/Ascii | %s |

The ASCII data response is encoded in the following string format

**[<Data Ready>:<Timestamp>] <Data>**

**Data Ready** and **Timestamp** are same fields used in all other responses. Telemetry data is formatted after converting the telemetry format specifier to the C format specifier and using sprintf to output a string representing the data. This output string is at **max 256 characters**, and can be read over I$^2$C or is output to serial console connected via SupMCU Debug Port.

If a telemetry format specification lists multiple elements (e.g. BSM:TEL? 6 lists $u_{16}$, $u_{16}$, $u_{16}$, $u_{16}$, $u_{16}$ ) the individual elements contained in the format specification are separated by comma. A format specifier with three $u_{16}$ is translated to: **%u,%u,%u.**

The ASCII output for BIM UART Status when all three BIM UART ports are on (and OS ticks = 100, Data Ready = 1) is: **[1:100] 1,1,1**

## Supported SCPI Standards

Pumpkin SupMCU modules only support a subset of the SCPI v1999.0 standard. To enable output over the Debug UART for these standard SCPI commands, the SupMCU Debug command must be sent:

```
SUP:DEB EN,0x00000001
```

The list of enabled SCPI standard commands is:

- *CLS
  - Clear status command. Clears the SCPI error queue. Any existing errors in the SCPI error queue will be cleared.
- *ESE <register>
  - Standard Event Status Enable Command. This is not used in SupMCU architecture.
- *ESE?
  - Stand Event Status Enable Query. This is not used in SupMCU architecture, returns 0.
- *ESR?
  - This is set to 32 if a SCPI command has failed to be processed, 0 otherwise.
- *IDN?
  - Identification Query. This is not used in SupMCU architecture.
- *OPC
  - Operation Complete Command. This is not used in SupMCU architecture.
- *OPC?
  - Operation Complete Query. This is not used in SupMCU architecture.
- *RST
  - Reset Command. This is not used in SupMCU architecture. Use *SUP:RES NOW* instead to reset the module.
- *SRE <register>
  - Service Request Enable Command. This is not used in SupMCU architecture.
- *SRC?
  - Service Request Enable Query. This is not used in SupMCU architecture.
- *STB?
  - Read Status Byte Query. This is not used in SupMCU architecture.
- *TST?
  - Self-Test Query. This is not used in SupMCU architecture.
- *WAI
  - Wait-to-Continue Command. This is not used in SupMCU architecture.
- SYSTem:ERRor? or SYSTem:ERRor:NEXT?
  - System Error Queue Query. Prints the reason for the last failed SCPI command. Use *CLS to clear the System Error Queue.
- SYSTem:VERSion?
  - System Version Query. Prints out “1999”, stating the SCPI standard the SupMCU architecture is based off of.