PSU_610, PSU_610_0001, PSU_610_4WS Communications Manual

Version 1.7.4

The PSU_610 is a programmable DC constant current lamp power supply, with a software constant voltage mode suitable for the control of driver-free sources. It can be controlled via USB or front panel, and is capable of sourcing up to 10.4A DC current at a voltage of up to 26V.

Quick Start

To begin controlling the PSU_610 via USB first turn the current control knob to PC. Once in PC mode it is ready to be controlled via its SCPI interface (see Physical Interface).
Here are some example commands

- Set the output current to 5.0A and enable the output
  - :SOUR:CURR 5.0
  - :OUTP 1
- Disable the output, set the output current to 10.0A and enable the output
  - :OUTP 0
  - :SOUR:CURR 10.0
  - :OUTP 1
- Read the sense voltage and current
  - :VOLT? returns 10.0
  - :CURR? returns 5.0
  - :IV? returns 5.0,10.0

Communication Protocol

Many of Bentham's instruments, this one included, expose a SCPI command interface. SCPI is chosen as it is a well established standard that provides a good compromise between capability and ease-of-use for the system user or programmer. We provide a short introduction to SCPI below, but much more information is available online, for example:

- Standard Commands for Programmable Instruments (SCPI) Wikipedia page
- SCPI Specification
- Agilent/Keysight SCPI Learning Centre

Physical Interface

The SCPI specification does not specify a particular physical interface (e.g. RS232, GPIB, USB). The majority of Bentham's instruments use a USB connection configured as an HID device: this is readily supported by all operating systems and suitable libraries are available for all major programming and scripting languages. It does however put an upper limit of 64 bytes per line sent to the instrument, although, in practice, this is rarely a problem as SCPI commands are typically around 8-30 bytes long. More care has to be taken if you are sending multiple commands on a single line (which is possible with SCPI, see below).

To issue a command to an instrument:

1. encode the command as an ASCII sequence of bytes;
2. ensure the string is terminated with either a null or newline byte (nothing after this termination will be read)
3. transmit the text as a HID message to the instrument (depending on the library/ language used Windows hosts may require a leading null byte).

Some (but not all) SCPI commands return one or more values (whether they do is explicitly detailed in the command descriptions in this document). When data is returned, it is in ASCII format within a 64-byte HID buffer.

SCPI Basics

SCPI commands are text commands and are designed to be human-readable. Some examples of SCPI commands are:

1. :SOUR:CURR 5.0 - sets a power source to generate 5.0A DC
2. :SOUR:VOLT 16.5 - sets a power source to generate 16.5V DC
3. :SOUR:VOLT? - gets the current voltage setting from a power source
4. :MONO:WAVE 654.0 - set a monochromator to 654 nm
5. :MONO:WAVE? - get the current wavelength from a monochromator
6. :MONO:SLIT:BAND 2,3.2 - sets slit 2 of a monochromator to 3.2 nm bandwidth
7. *IDN? - get system identification

These examples also show some other key points about SCPI commands:

- Commands that end with a ? are 'query commands' and will return one or more (comma-separated) values;
- Commands that take a value have a space separating the command from the value;
- Some commands take multiple values, which should be comma-separated;
- There are a number of SCPI standard commands that are supported by all SCPI compliant instruments, these start with an asterisk (*).

Parameters & Data Types

Where a SCPI command takes parameters these are well-defined in both their meaning and type. Each command described in this document lists the required parameters, along with their type. Common types are:

- boolean - this can either be sent as OFF or ON, or numerically with 0 for false and 1 for true;
- 32-bit integer - integer value as decimal ASCII text;
- text - an ASCII string, which should be quoted in double quotation marks (");
- IEEE-754 32-bit float - a floating point value as decimal ASCII text.

Returned values

Some SCPI commands return no data, and in this case no buffer is returned by the instrument to the controlling device. These commands are usually the commands that set a variable.

Other SCPI commands return one or more values. These are all returned on a single line, are comma-separated (with no spaces) and terminated with a null byte. Each command description below lists the returned values and their type. Boolean parameters are returned as 0 or 1, and text is contained within double quotation marks, ".

Command abbreviation

If you look through the command descriptions in this document you will see that each command contains a mixture of upper and lower case characters and some contain sections between square brackets ([and]). In fact, the lower case characters and any section between a [] pair are optional. This allows the user to choose between human-readability and command length. Here we show two commands to illustrate, showing the command description followed by some valid abbreviations in use.

- :MONOchromator[:WAVElength][:SET] <wavelength>
  - :MONOchromator:WAVElength:SET 800
  - :MONO:WAVE:SET 800
  - :MONO:WAVElength 800
  - :MONO:WAVE 800
  - :MONO 800
- [:MEASure]:CURRent?
  - :MEASure:CURRent?
  - :MEAS:CURR?
  - :CURR?

Multiple commands per line

A single line of text sent to the instrument can contain multiple SCPI commands. Each command just needs to be separated with a ; (semicolon). For example:

1. :SOUR:CURR 5.0; :OUTP ON - sets a power source to generate 5.0A DC and switches it on
2. :MONO:GRAT 1; :MONO:WAVE 654.0; :MONO:MOVE - set a monochromator to grating 1 at 654 nm

Hierarchical Commands

SCPI commands are hierarchical in structure. Similar commands are grouped in the tree like structure, just like files in a filesystem. Whereas a filesystem uses a slash character to separate the layers, SCPI uses a : (colon).

A leading : indicates the root of the command structure, and a command that starts with a : is fully-qualified, just like an absolute file path.

Commands without a leading : inherit their position from the previous command and hence the fully-qualified name does not need to be sent every time. However, we would recommend that this feature is not widely used when programming for the automated use of the interface as it can lead to difficulty porting to different instruments. The one case where it may be considered is when putting multiple commands on a line, which leads to our recommendation that all lines sent to a SCPI device start with a fully-qualified command.

Synchronisation

As with all machine-to-machine interfaces, some care has to be taken to ensure that the controlling device and instrument remain synchronised or that the controlling device is not sending the instrument commands faster than it can cope with them. We make these recommendations to avoid these issues:

- Do not send more than two consecutive lines of SCPI commands where none of the commands return a value.
- When a SCPI command returns a value wait for it before continuing.
- It is good practice to check the instruments has no pending errors by using the SYST:ERR:COUNT?

command after setting some values.
- A very robust way to ensure synchronisation is to set a variable and then query it on the same line.

Command Index

*CLS
*IDN? ⇒ "<manufacturer>", "<model>", "<serial_number>", "<revision>"
*RCL "profilename"
[:DIAGnostic]:ECHO[:TEXT]? "<echo_to_device>" ⇒ "<echo_from_device>"
:DISPlay:ACTive:BRIGhtness <brightness>
:DISPlay:ACTive:BRIGhtness? ⇒ <brightness>
:DISPlay[:DIMmed]:BRIGhtness <brightness>
:DISPlay[:DIMmed]:BRIGhtness? ⇒ <brightness>
:DISPlay[:DIMmed]:DELAY <time>
:DISPlay[:DIMmed]:DELAY? ⇒ <time>
:DISPlay[:ENABle] <state>
:DISPlay[:ENABle]? ⇒ <state>
:FETCh:BURNtime? ⇒ <burntime>
[:MEASure]:CURRent? ⇒ <measured_current>
[:MEASure]:IV? ⇒ <measured_current>, <measured_voltage>
[:MEASure]:POWer? ⇒ <measured_power>
[:MEASure]:POWer:STDev? ⇒ <power_std>
[:MEASure]:RESistance? ⇒ <measured_resistance>
[:MEASure]:VOLTage? ⇒ <measured_voltage>
[:OUTPut]:ATTarget? ⇒ <state>
:OUTPut:MODE:CURRent
:OUTPut:MODE:CURRent? ⇒ <current_mode_state>
:OUTPut:MODE:VOLTage
:OUTPut:MODE:VOLTage? ⇒ <voltage_mode_state>
[:OUTPut]:SAM <state>
[:OUTPut]:SAM? ⇒ <state>
:OUTPut[:STATe] <target_state>
:OUTPut[:STATe]? ⇒ <output_state>
[:PARAmeter]:WIRE:RESistance <wire_resistance>
[:PARAmeter]:WIRE:RESistance? ⇒ <wire_resistance>
:RESEt:BURNtime
:SOURce:CURRent <target_current>
:SOURce:CURRent? ⇒ <target_current>
:SOURce:VOLTage <target_voltage>
:SOURce:VOLTage? ⇒ <target_voltage>
:SYSTem:ERRor:COUNt? ⇒ <number_of_errors>
:SYSTem:ERRor[:NEXT]? ⇒ <error_code>, "<error_msg>"
SYSTEM:REBOOT
:SYSTem:VERSion[:APP]? ⇒ "<emben_version>"
:SYSTem:VERSion:EMBEN? ⇒ "<emben_version>"

Command Syntax

*CLS

This command clears the error buffer.

See Also SYSTem:ERRor[:NEXT]?

*IDN? ⇒ "<manufacturer>", "<model>", "<serial_number>", "<revision>"

This command returns meta data information from the device.

Return	Type	Description
"<manufacturer>"	text	"Bentham Instruments Ltd."
"<model>"	text	The device's model designation.
"<serial_number>"	text	The device's serial number.
"<revision>"	text	The device's revision number.

*RCL "profilename"

This command loads a particular configuration profile and restarts the device.

[:DIAGnostic]:ECHO[:TEXT]? "<echo_to_device>" ⇒ "<echo_from_device>"

This command is used to send a string to the device and have the device send it back as a communication diagnostic.

Parameter	Type	Description
"<echo_to_device>"	text	The text to be returned (in quotation marks).

Return	Type	Description
"<echo_from_device>"	text	The returned text.

Examples :ECHO? "hello!" returns "hello!"

:DISPlay:ACTive:BRIGhtness <brightness>

This command sets the brightness of the illuminated, active display. The brightness parameter has a range of 0 to 1. It is worth noting that a brightness of zero does not disable the screen, this can be achieved using the :DISPlay[:ENABle] command.

Parameter	Type	Description
<brightness>	IEEE-754 32-bit float	The brightness of the active display (where 0 ≤ brightness ≤ 1).

Examples
:DISP:ACT:BRIG 0.5 sets the brightness to 50%
:DISP:ACT:BRIG 1.0 sets the brightness to its maximum.

See Also
:DISPlay[:ENABle]
:DISPlay[:DIMmed]:BRIGhtness
:DISPlay:ACTive:BRIGhtness?

:DISPlay:ACTive:BRIGhtness? ⇒ <brightness>

This command returns the brightness of the illuminated, active display. The brightness parameter has a range of 0 to 1. It is worth noting that a brightness of zero does not disable the screen, this can be achieved using the :DISPlay[:ENABle] command.

Return	Type	Description
<brightness>	IEEE-754 32-bit float	The brightness of the active display (where 0 ≤ brightness ≤ 1).

See Also
:DISPlay[:DIMmed]:BRIGhtness?
:DISPlay:ACTive:BRIGhtness

:DISPlay[:DIMmed]:BRIGhtness <brightness>

This command sets the brightness of the dimmed display. The brightness parameter has a range of 0 to 1. It is worth noting that a brightness of zero does not disable the screen, this can be achieved using the :DISPlay[:ENABle] command.

Parameter	Type	Description
<brightness>	IEEE-754 32-bit float	The brightness of the dimmed display (where 0 ≤ brightness ≤ 1).

Examples
:DISP:BRIG 0.5 sets the brightness to 50%
:DISP:DIMMED:BRIG 1.0 sets the brightness to its maximum.

See Also
:DISPlay[:ENABle]
:DISPlay:ACTive:BRIGhtness
:DISPlay[:DIMmed]:BRIGhtness?

:DISPlay[:DIMmed]:BRIGhtness? ⇒ <brightness>

This command returns the brightness of the dimmed display. The brightness parameter has a range of 0 to 1. It is worth noting that a brightness of zero does not disable the screen, this can be achieved using the :DISPlay[:ENABle] command.

Return	Type	Description
<brightness>	IEEE-754 32-bit float	The brightness of the dimmed display (where 0 ≤ brightness ≤ 1).

See Also
:DISPlay[:DIMmed]:BRIGhtness
:DISPlay[:ENABle]
:DISPlay:ACTive:BRIGhtness?

:DISPlay[:DIMmed]:DELAY <time>

This command sets the time the display stays at its full brightness after waking.

Parameter	Type	Description
<time>	IEEE-488.2 Number, optional suffix	Time for which the display stays at its full brightness before it dims again. In seconds, or units can be specified.

Examples
:DISP:DELAY 500ms sets the delay to 500ms
:DISP:DELAY 0.5 sets the delay to 500ms

See Also
:DISPlay[:DIMmed]:BRIGhtness
:DISPlay:ACTive:BRIGhtness
:DISPlay[:DIMmed]:DELAY?

:DISPlay[:DIMmed]:DELAY? ⇒ <time>

This command returns the time the display stays at its full brightness after waking.

Return	Type	Description
<time>	IEEE-754 32-bit float	Time for which the display stays at its full brightness before it dims again, in seconds

See Also
:DISPlay[:DIMmed]:DELAY
:DISPlay[:DIMmed]:BRIGhtness
:DISPlay:ACTive:BRIGhtness

:DISPlay[:ENABle] <state>

This command enables or disables the illuminated display.

Parameter	Type	Description
<state>	boolean (0 or 1)	The target state for the display.

Examples :DISP 0 disables the display.
:DISP 1 enables the display.

See Also :DISPlay[:ENABle]?

:DISPlay[:ENABle]? ⇒ <state>

This command returns the enabled state of the display.

Return	Type	Description
<state>	boolean (0 or 1)	Whether or not the display is enabled

See Also :DISPlay[:ENABle]

:FETCh:BURNtime? ⇒ <burntime>

This command returns the cumulative time in hours for which the output has been enabled since the last burntime reset.

Return	Type	Description
<burntime>	IEEE-754 32-bit float	The cumulative time in hours for which the output has been enabled since the last burntime reset.

See Also :RESEt:BURNtime

[:MEASure]:CURRent? ⇒ <measured_current>

This command returns the measured output current in Amps. This current may differ from the target current if the power supply is in the process of ramping the output to achieve the set target.

Return	Type	Description
<measured_current>	IEEE-754 32-bit float	The actual output current in Amps.

Examples CURRENT? will return 3.004 if the output current was 3.004A.

See Also [:MEASure]:VOLTage?
[:MEASure]:IV?

[:MEASure]:IV? ⇒ <measured_current>, <measured_voltage>

This command returns both the measured output current in Amps and the measured output voltage in Volts. Calling this single compound command is equivalent to calling both the [:MEASure]:CURRent? and [:MEASure]:VOLTage? commands.

Return	Type	Description
<measured_current>	IEEE-754 32-bit float	The measured output current in Amps.
<measured_voltage>	IEEE-754 32-bit float	The output voltage in Volts measured across the sense pins (PSU_610_4WS model) or corrected for the series resistance of the supply wires (PSU_610 model).

Examples :IV? will return 8.251,12 if 8.251A is flowing through the circuit resulting in a potential difference of 12V across ⊕ and ⊖ pins (or the sense pins if available).

See Also [:MEASure]:CURRent?
[:MEASure]:VOLTage?

[:MEASure]:POWer? ⇒ <measured_power>

This command returns the measured output power in Watts. It is calculated using $P = I \times V$
where
V is returned by [:MEASure]:VOLTage?
I is returned by [:MEASure]:CURRent?.

Return	Type	Description
<measured_power>	IEEE-754 32-bit float	The measured output power in Watts.

Examples :POW? will return 150.0 if 10.0A is flowing through the circuit resulting in a potential difference of 15.0V across ⊕ and ⊖ pins (or the sense pins if available).

See Also [:MEASure]:CURRent?
[:MEASure]:VOLTage?

[:MEASure]:POWer:STDev? ⇒ <power_std>

This command returns the standard deviation of the measured power taken over the last 10 internal samples. This command will result in a SCPI execution error if used when the output is off as no samples are available.

Return	Type	Description
<power_std>	IEEE-754 64-bit float	the standard deviation of measured power, Watts.

See Also [:MEASure]:POWer?

[:MEASure]:RESistance? ⇒ <measured_resistance>

This command returns the measured resistance of the load in Ohms. It is calculated using $R = V / I$ where
V is returned by [:MEASure]:VOLTage?
I is returned by [:MEASure]:CURRent?.

Return	Type	Description
<measured_resistance>	IEEE-754 32-bit float	The measured load resistance in Ohms.

Examples :RES? will return 1.0 if 5.0A is flowing through the circuit resulting in a potential difference of 5.0V across ⊕ and ⊖ pins (or the sense pins if available).

See Also [:MEASure]:CURRent?
[:MEASure]:VOLTage?

[:MEASure]:VOLTage? ⇒ <measured_voltage>

This command returns the output voltage in Volts measured across the sense pins (PSU_610_4WS model) or corrected for the series resistance of the supply wires (PSU_610 model).

Return	Type	Description
<measured_voltage>	IEEE-754 32-bit float	The output voltage in Volts measured across the sense pins (PSU_610_4WS model) or corrected for the series resistance of the supply wires (PSU_610 model).

Examples :VOLT? will return 11.513 if a potential difference of 11.513V is measured across the ⊕ and ⊖ pins (or the sense pins if available) and the wire resistance parameter is zero.
:VOLT? will return 11.6 if a raw potential difference of 12V is measured across the ⊕ and ⊖ pins (or the sense pins if available), 4A of current is flowing, and the wire resistance parameter is 0.1Ω (*11.6V = 12V - 4A*0.1Ω*).

See Also [:MEASure]:CURRent?
[:MEASure]:IV?
[:PARAmeter]:WIRE:RESistance?
[:PARAmeter]:WIRE:RESistance

[:OUTPut]:ATTarget? ⇒ <state>

This command returns the at-target state of the PSU_610 indicating whether or not the system is at the target current. This is also indicated by the output LED, red equivalent to at-target being false, green equivalent to at-target being true.

Return	Type	Description
<state>	boolean (0 or 1)	whether or not the system is at its intended target

:OUTPut:MODE:CURRent

This command activates the constant current mode. It does not change the target current. Therefore if the software constant voltage mode was previously enabled then the target current will remain at the current last dynamically set by the voltage mode to maintain the target voltage.

Examples OUTP:MODE:CURRENT enables constant current mode.

See Also :OUTPut:MODE:CURRent?
:OUTPut:MODE:VOLTage
:SOURce:CURRent

:OUTPut:MODE:CURRent? ⇒ <current_mode_state>

This command returns the enabled state of the constant current mode.

Return	Type	Description
<current_mode_state>	boolean (0 or 1)	The enabled state of the current mode.

Examples if OUTP:MODE:CURR? returned 1 the PSU_610 output is in constant current mode.

See Also :OUTPut:MODE:CURRent
:OUTPut:MODE:VOLTage?
:SOURce:CURRent

:OUTPut:MODE:VOLTage

This command activates the software constant voltage mode. When the output is enabled, the PSU_610 dynamically adjusts the current target in order to reach the voltage target.

Examples OUTP:MODE:VOLT enables software constant voltage mode.

See Also :OUTPut:MODE:VOLTage?
:OUTPut:MODE:CURRent
:SOURce:VOLTage

:OUTPut:MODE:VOLTage? ⇒ <voltage_mode_state>

This command returns the enabled state of the software constant voltage mode.

Return	Type	Description
<voltage_mode_state>	boolean (0 or 1)	The enabled state of the software constant voltage mode.

Examples if OUTP:MODE:VOLT? returned 1 the PSU_610 output is in software constant voltage mode.

See Also :OUTPut:MODE:CURRent?
:OUTPut:MODE:VOLTage
:SOURce:VOLTage

[:OUTPut]:SAM <state>

This command sets the SAM output (24V across the dedicated port) of the PSU_610_0001. Not available on all models.
If the SAM is not present then an execution error (code 200) is pushed to the error stack.

Parameter	Type	Description
<state>	boolean (0 or 1)	The target SAM state, ON/1 = energised, OFF/0 = relaxed

See Also [:OUTPut]:SAM?

[:OUTPut]:SAM? ⇒ <state>

This command reads the SAM output state (24V across the dedicated port) of the PSU_610_0001. Not available on all models.
If the SAM is not present then an execution error (code 200) is pushed to the error stack. In this case the following error message is returned instead of the specified return values.
Error: SAM not found

Return	Type	Description
<state>	boolean (0 or 1)	The current SAM state, ON/1 = energised, OFF/0 = relaxed

See Also [:OUTPut]:SAM

:OUTPut[:STATe] <target_state>

This command enables or disables the output current of the PSU_610. When enabled the last set target is used, this could either be a current or a voltage depending on the output mode. When disabled the current is ramped down to 0A and then the output disabled.

Parameter	Type	Description
<target_state>	boolean (0 or 1)	The target state for the PSU_610 output.

Examples :OUTP ON enables the output and ramps the current to the target.
:OUTP OFF ramps current down to 0A, then disables the output.

See Also :OUTPut[:STATe]?

:OUTPut[:STATe]? ⇒ <output_state>

This command returns the PSU_610's output state.

Return	Type	Description
<output_state>	boolean (0 or 1)	The PSU_610's output state.

Examples if OUTP? returned 1 the PSU_610 output is enabled.

See Also :OUTPut[:STATe]

[:PARAmeter]:WIRE:RESistance <wire_resistance>

This command sets the value of the wire resistance assumption in Ohms. Once set a resistor of this value will assumed to be in series with the load when measured voltage calculations are made.

Parameter	Type	Description
<wire_resistance>	IEEE-754 32-bit float	The wire resistance in Ohms to be stored in persistent memory.

Examples After WIRE:RES 0.1 the calculation of all voltages will take into account a 0.1Ω in series resistance.

See Also [:PARAmeter]:WIRE:RESistance?

[:PARAmeter]:WIRE:RESistance? ⇒ <wire_resistance>

This command returns the value of the wire resistance in Ohms. It is assumed to be in series with the output load and used to correct the measured output voltage returned by the commands like [:MEASure]:VOLTage?. The factory default value for this parameter is zero.

Return	Type	Description
<wire_resistance>	IEEE-754 32-bit float	The wire resistance in Ohms.

Examples WIRE:RES? returns 0.1 if the wire resistance parameter has been previously set to 0.1Ω.

See Also [:PARAmeter]:WIRE:RESistance

:RESEt:BURNtime

This command resets to zero the persistent parameter tracking the cumulative time the output has been enabled since the last burntime reset.

See Also :FETCh:BURNtime?

:SOURce:CURRent <target_current>

This command attempts to set the target output current in Amps. The command is only successful if the target current is greater than zero and less than or equal to the the maximum allowed current of 10.4A. This command will result in a SCPI execution error if used when in software constant voltage mode.

Parameter	Type	Description
<target_current>	IEEE-754 32-bit float	The target current in Amps.

Examples :SOUR:CURR 8.25 sets the current target to 8.25A.

See Also
:SOURce:VOLTage
:SOURce:CURRent?
:OUTPut[:STATe]
:OUTPut:MODE:VOLTage
:OUTPut:MODE:CURRent

:SOURce:CURRent? ⇒ <target_current>

This command returns the target output current in Amps.

Return	Type	Description
<target_current>	IEEE-754 32-bit float	The target current in Amps.

Examples :SOUR:CURR? will return 8.25 if the current target was set to 8.25A (e.g. using the command :SOUR:CURR 8.25)

See Also :SOURce:CURRent

:SOURce:VOLTage <target_voltage>

This command sets the target output voltage for the software constant voltage mode.

Parameter	Type	Description
<target_voltage>	IEEE-754 32-bit float	The target output voltage in Volts.

Examples :SOUR:VOLT 12.5 sets the target output voltage to 12.5V.

See Also
:SOURce:CURRent
:SOURce:VOLTage?
:OUTPut[:STATe]
:OUTPut:MODE:VOLTage
:OUTPut:MODE:CURRent

:SOURce:VOLTage? ⇒ <target_voltage>

This command returns the target output voltage of the software constant-voltage mode.

Return	Type	Description
<target_voltage>	IEEE-754 32-bit float	the target output voltage in Volts.

Examples :SOUR:VOLT? will return 12.5 if the voltage target was previously set to 12.5V.

See Also :SOURce:VOLTage
:SOURce:CURRent?

:SYSTem:ERRor:COUNt? ⇒ <number_of_errors>

This command returns the number of errors currently in the error queue.

Return	Type	Description
<number_of_errors>	32-bit integer	The number of errors on the error queue.

See Also :SYSTEM:ERRor[:NEXT]?

:SYSTem:ERRor[:NEXT]? ⇒ <error_code>, "<error_msg>"

This command pops the next error from the start of the error queue and returns information about the error. If there was no error then error code 0 and information "No Error" are returned.

Return	Type	Description
<error_code>	32-bit integer	A numeric value identifying the error.
"<error_msg>"	text	A string of characters describing the error.

Examples BAD:COMMAND followed by :SYST:ERR? results in -113,"Undefined header"
a subsequent :SYST:ERR? results in 0,"No error"

See Also :SYSTem:ERRor:COUNt?

SYSTEM:REBOOT

This command restarts the control electronics of the device. This command can not be used in conjunction with other SCPI commands as it is processed separately. It is not in the SCPI command hierarchy and does not use a leading colon (:) but is inherently fully qualified. Unsaved changes will be lost.

:SYSTem:VERSion[:APP]? ⇒ "<emben_version>"

This command is used to query the firmware application version, which identifies the revision of the high level functionality of the firmware application for this specific product. This consists of a standard 3-part version followed by an optional label.

Return	Type	Description
"<emben_version>"	text	The emben version as text.

Examples SYST:VERS? returns "1.2.3-15127"

See Also :SYSTem:VERSion:EMBEN?

:SYSTem:VERSion:EMBEN? ⇒ "<emben_version>"

This command is used to query the emben firmware library version, which identifies the revision of lower-level code shared across many different products.

Return	Type	Description
"<emben_version>"	text	The emben version as text.

Examples SYST:VERS:EMBEN? returns "1.0.0"

See Also :SYSTem:VERSion[:APP]?