SOURCE MEASURE UNIT–X200

PROGRAMMING DOCUMENATION

Manual version: 2.4.C
Product code: P2005A
Product Version: 2.0
Firmware Version: 2.4

Ossila.com

Contents

The Ossila Source Measure Unit can be controlled directly over USB or Ethernet using the commands in this document. These can be sent as strings, enabling it to be controlled using a large variety of programming languages, including Python, MATLAB, LabVIEW, Java, and C/C++. We have a Python package that can be downloaded from the Python Package Index, for more details, see **Section 0**.

1. Argument and Response Formats

Any response that returns after sending a command is expected to return the newline character (\n) after the data has finished transferring. Similarly, commands that do not return any data will also not return the newline character.

(I) Integer

An integer is any whole (non-decimal) number. Where integer types are accepted or returned, there is usually a wrapping that applies. This means that if number 6 is sent in set range command, then range 1 will be set as there are only 5 ranges (6 modulo 5 = 1).

(II) Float

Floats are floating point numbers (decimals). For the commands in this document, these are 8-byte floating point numbers, sometimes called double or long floats. When returned, if the precision is too low to accommodate the number, they will be returned in scientific notation (for example, 0.0000123 will return as 1.23e-5).

(III) Boolean

Booleans are logic variables, i.e., True and False, or 1 and 0.

(IV) String

A string is a list of characters. These characters can be letters, numbers, or punctuation. When a string is specified as an input, it should be a series of characters that end with a space character. When a string is specified as a return format, any string of characters can be returned.

(V) Array

Arrays are represented using MATLAB's array syntax. This uses square brackets, with each entry separated by a semicolon, e.g. [21;1;7;3.4].

(VI) Matrix

Matrices are denoted using standard MATLAB matrix syntax. Entries are separated by a semicolon and the properties of an entry are separated by commas. Unless otherwise stated when a matrix is returned it will be a matrix of floats.

(VII)None

When None is mentioned in the documentation as a function return, that simply means that the function will not return any data. This includes a newline character, as mentioned above.

(VIII) IP Address

The IP address type is a standard IPv4 address (integer.integer.integer.integer), where each integer is between 0 and 255. IPv6 addresses are not supported, so if you are using an IPv6 network ensure that IPv4-only devices can connect.

(IX) MAC Address

The MAC address type is the standard hexadecimal MAC address. This consists of six pairs of hexadecimal characters separated by colons (AA:BB:CC:DD:EE:FF).

(X) Version

The version type is the way of describing the version number of a module. It uses the semantic versioning syntax to enable peace of mind when deciding whether to upgrade. The full description of semantic versioning can be found at http://semver.org.

2. Compliance

The Source Measure Unit has been designed with an internal compliance check feature. This feature is linked with the operation mode (unsafe) flag, where it can be set via *`smu# set unsafe true`* command (documented in **Section 4**). In unsafe mode all the voltage and current compliance checks are ignored. In this state, current measurements above the limit of the range will saturate the measurement, and both the device under test and the Source Measure Unit will have no protection from excessive currents, potentially resulting in damage to either.

Warning: Any damage to the Source Measure Unit resulting from use of unsafe mode is not covered by the Warranty.

2.1 Single Point Measurements

The compliance check during a single point measurement is performed whenever a new voltage is set on a SMU channel or run the measure command. If the resulting voltage or current equals or exceeds the maximum current of the selected range or the user-defined current or voltage limits, then the SMU output voltage is set to 0 V, the error LED is turned on, and an empty matrix is returned. This resets automatically after setting a new voltage if the new voltage and current are within the compliance limits.

2.2 Multiple Point Measurements

The compliance check in multiple point measurement commands such as `sweep` is performed point by point. For most commands, if compliance is reached at any point during the measurement, the SMU output voltage is set to 0 V, the error LED is turned on, and command terminates immediately, returning the data which has been measured up to that point. For example, if compliance occurs on the first point, then the returned measurement will an empty matrix.

Multiple point measurement commands that end with an `f` will not terminate the command when compliance is reached, instead changing the output of the SMU based upon the selected SMU's output mode and continuing onto the next point. More details of these commands can be found in **Section 4**.

3. CLOI

CLOI is the Command Language for Ossila Instruments. It is the core that runs the Ossila Source Measure Unit. The main module provides the interface to all the other instruments/channels in the system – including the SMUs, the Vsenses, and the shutter/trigger.

All functions that are directed towards CLOI *have* the prefix `cloi`, which has been included in the functions list. The supported functions of the unit are listed below, with descriptions on the intended usages of each one.

3.1 Properties

precision: integer	

3.2 Commands

Returns	Command
array	cloi devices
integer	cloi get precision
string	cloi hello
	cloi set precision *integer*
string	cloi speedtest
version	cloi version

3.3 Property Documentation

(I) precision: integer

This property holds the number digits for float data output, including the decimal point (for example a precision of 7 will return data in the format 10.1234).

The power ON value is 5.

Access functions:

integer	cloi get precision
	cloi set precision *integer*

3.4 Command Documentation

cloi hello

Arguments

None

Expected response format – *String*

Returns a set String, which is `Hello World\n`.

cloi version

Arguments

None

Expected response format – *Version*

Returns the version of the CLOI module. This is different from the top-level function version, which will return the overall version of the hardware and firmware of the unit.

cloi devices

Arguments

None

Expected response format – *[string;….;string] – Array*

Returns an array of the registered devices' names.

cloi speedtest

Arguments

Integer - None

Expected response format – *String*

Returns a dummy string '123456789' repeated 20480 times (204800 characters), the time required to transfer this data over the link in seconds, and the transfer rate (speed) in bits per second.

4. SMU

The SMU module is the driver for both source measure unit (SMU) channels and is activated by the either the prefix *`smu1`* or *`smu2`*. The supported commands are listed and detailed below using the *`smu1`* prefix. The same commands can be used for the second SMU channel by changing the prefix to *`smu2`*.

4.1 Properties

delay: float	**limitv:** float
enabled: Boolean	**limitv_max:** float
error: Boolean	**limitv_min:** float
filter: integer	**offset:** float
hiz: Boolean	**osr:** integer
limiti: float	**range:** integer
limiti_max: float	**unsafe:** Boolean
limiti_min: float	**voltage:** float

4.2 Commands

Returns	Command
	smu1 clear error
float	smu1 get delay
Boolean	smu1 get enabled
Boolean	smu1 get error
integer	smu1 get filter
Boolean	smu1 get hiz
float	smu1 get limiti
float	smu1 get limiti_max
float	smu1 get limiti_min
float	smu1 get limitv
float	smu1 get limitv_max
float	smu1 get limitv_min
float	smu1 get offset
integer	smu1 get osr

integer	smu1 get range
Boolean	smu1 get unsafe
float	smu1 get voltage
matrix	smu1 measure
matrix	smu1 measure *integer*
array	smu1 measurei
array	smu1 measurei *integer*
array	smu1 measurev
array	smu1 measurev *integer*
matrix	smu1 oneshot *float*
	smu1 set delay *float*
	smu1 set enabled *Boolean*
	smu1 set filter *integer*
	smu1 set hiz *Boolean*
	smu1 set limiti *float*
	smu1 set limiti_max *float*
	smu1 set limiti_min *float*
	smu1 set limitv *float*
	smu1 set limitv_max *float*
	smu1 set limitv_min *float*
	smu1 set offset *float*
	smu1 set osr *integer*
	smu1 set range *integer*
	smu1 set unsafe *Boolean*
	smu1 set voltage *float*
matrix	smu1 sweep *float float float integer*
matrix	smu1 sweep *float float float integer* d
matrix	smu1 sweep *float float float integer integer* f
	smu1 sweepv *float float float integer*
	smu1 sweepv *float float float integer* d

4.3 Property Documentation

(I) delay: float

This property holds the delay in microseconds (μs) between the SMU setting a voltage and measuring a data point. The power ON value is 1000 μs.

Access functions:

float	smu1 get delay
	smu1 set delay *float*

(II) enabled: Boolean

This property holds whether the SMU is turned on or off. The power ON value is False.

Access functions:

Boolean	smu1 get enabled
	smu1 set enabled *Boolean*

(III) error: Boolean

This property holds the state of the compliance flag for the SMU. If either the voltage or current limits are reached this becomes True. See **Section 2** for more details on compliance.

Access functions:

	smu1 clear error
Boolean	smu1 get error

(IV) filter: integer

This property holds the number of measurements to average for each data point. Note that this is in addition to the OSR. The power ON value is 1.

Access functions:

integer	smu1 get filter
	smu1 set filter *integer*

(V) hiz: Boolean

This property holds whether output of the SMU is in hi-impedance mode or not. The power ON value is False.

Access functions:

Boolean	smu1 get hiz
	smu1 set hiz *Boolean*

(VI) limiti: float

This property holds the current limit for the SMU in amps. It applies to both positive and negative currents. See **Section 2** for more details on compliance. The power ON value is |0.225| A.

Access functions:

float	smu1 get limiti
	smu1 set limiti *float*

(VII) limiti_max: float

This property holds the upper current limit for the SMU in amps. Overwrites and is overwritten by **limiti**. See **Section 2** for more details on compliance. The power ON value is +0.225 A.

Access functions:

float	smu1 get limiti_max
	smu1 set limiti_max *float*

(VIII) limiti_min: float

This property holds the lower current limit for the SMU in amps. Overwrites and is overwritten by **limiti**. See **Section 2** for more details on compliance. The power ON value is -0.225 A.

Access functions:

float	smu1 get limiti_min
	smu1 set limiti_min *float*

(IX) limitv: float

This property holds the voltage limit for the SMU in volts. It applies to both positive and negative voltages. See **Section 2** for more details on compliance. The power ON value is |10.5| V.

Access functions:

float	smu1 get limitv
	smu1 set limitv *float*

(X) limitv_max: float

This property holds the upper voltage limit for the SMU in volts. Overwrites and is overwritten by **limitv**. See **Section 2** for more details on compliance. The power ON value is +10.5 V.

Access functions:

float	smu1 get limitv_max
	smu1 set limitv_max *float*

(XI) limitv_min: float

This property holds the lower voltage limit for the SMU in volts. Overwrites and is overwritten by **limitv**. See **Section 2** for more details on compliance. The power ON value is -10.5 V.

Access functions:

float	smu1 get limitv_min
	smu1 set limitv_min *float*

(XII) offset: float

This property holds the offset that is applied to current measurements (in amps) before data points are returned. The power ON value is 0.

Access functions:

float	smu1 get offset
	smu1 set offset *float*

(XIII) osr: integer

This property holds the Oversample Rate (OSR) index of the SMU. OSR can take values 0 – 19 with wrapping supported (i.e., 22 will set the OSR to 2). This determines the number of samples that are taken per data point, or the sampling speed. More details can be found in **Section 6.2** of the Ossila Source Measure Unit User Manual. The power ON value is 5.

Access functions:

integer	smu1 get osr
	smu1 set osr *integer*

(XIV) range: integer

This property holds the current range of the SMU. This determines the maximum measurable currents, and measurement accuracy and resolution. More details can be found in **Section 6.2** of the Ossila Source Measure Unit User Manual. The power ON value is 1.

Access functions:

integer	smu1 get range
	smu1 set range *integer*

(XV) unsafe: Boolean

This property holds whether the SMU is in unsafe mode or not. In unsafe mode, internal compliance checks are not performed. See **Section 2** for more details on compliance. The power ON value is False.

Boolean	smu1 get unsafe
	smu1 set unsafe *Boolean*

(XVI) voltage: float

This property holds the output voltage of the SMU in volts.

Access functions:

float	smu1 get voltage
	smu1 set voltage *float*

4.4 Command Documentation

smu1 measure
Arguments
Integer – number of points to measure
Expected response format – *[float,float] – [voltage,current] – Matrix*
Measures the voltage and current and returns them in a 1×2 matrix.
smu1 measure *integer*
Arguments
Integer – number of points to measure
Expected response format – *[float,float;…;float,float] – [voltage,current;…] – Matrix*
Measures the voltage and current the specified number of times and returns them in an Nx2 matrix.
smu1 measurei
Arguments
None
Expected response format – *[float] – Array*
Measures the current and returns it.

smu1 measurei *integer*

Arguments

Integer – number of points to measure

Expected response format – *[float;float;...;float;float] – Array*

Measures the current the specified number of times and returns them in an array.

smu1 measurev

Arguments

None

Expected response format – *[float] – Array*

Measures the voltage and returns it.

smu1 measurev *integer*

Arguments

Integer – number of points to measure

Expected response format – *[float;float;...;float;float] – Array*

Measures the voltage the specified number of times and returns them in an array.

smu1 oneshot *float*

Arguments

Float – voltage to set (V)

Expected response format – *[float,float] – [voltage,current] – Matrix*

Sets the output voltage of the SMU, then measures the voltage and current and returns them in a 1×2 matrix.

smu1 sweep *float float float integer*

Arguments

Float – start voltage (V)

Float – voltage increment (V, positive values only)

Float – end voltage (V)

Integer – delay (ms)

Expected response format – *[float,float;...;float,float] – [voltage,current;...] – Matrix*

Sweeps the output voltage from the start voltage to the end voltage inclusive at the given increments then sets the output voltage to 0 V. Delay sets the delay in milliseconds between setting a voltage and measuring a data point. Returns a matrix containing the measured voltage and current at each point. Sending any other command will stop the measurement immediately and returns the measured points.

smu1 sweep *float float float integer* d

Arguments

Float – start voltage (V)

Float – voltage increment (V, positive values only)

Float – end voltage (V)

Integer – delay (ms)

Expected response format – *[float,float;...;float,float] – [voltage,current;...] – Matrix*

Sweeps the output voltage from the start voltage to the end voltage inclusive at the given increments. Afterwards the device sweeps in reverse (hysteresis) then sets the output voltage to 0 V. Delay sets the delay in milliseconds between setting a voltage and measuring a data point. Returns both sets as a single matrix containing the voltage and current at each point. Sending any other command will stop the measurement immediately and returns the measured points.

smu1 sweep *float float float integer integer* f

Arguments

Float – start voltage (V)

Float – voltage increment (V, positive values only)

Float – end voltage (V)

Integer – delay (ms)

Integer – SMU output mode

Expected response format – *[float,float;...;float,float] – [voltage,current;...] – Matrix*

Sweeps the output voltage from the start voltage to the end voltage inclusive at the given increments then sets the output voltage to 0 V. Delay sets the delay in milliseconds between setting a voltage and measuring a data point. Does not stop if compliance is reached, instead changing SMU output based upon the output mode value: 0 sets the output voltage to 0 V, 1 switches output off, 2 sets the output to floating. Returns a matrix containing the voltage and current at each point. Sending any other command will stop the measurement immediately and returns the measured points.

smu1 sweepv *float float float integer*

Arguments

Float – start voltage (V)

Float – voltage increment (V, positive values only)

Float – end voltage (V)

Integer – delay (ms)

Expected response format – *None*

Sweeps the output voltage from the start voltage to the end voltage inclusive at the given increments then sets the output voltage to 0 V. Delay sets the delay in milliseconds between each voltage step. Does not perform any measurements and does not return any data.

smu1 sweepv *float float float integer* d
Arguments
Float – start voltage (V)
Float – voltage increment (step size) (V) +Ve values only
Float – end voltage (V)
Integer – delay (ms)
Expected response format – *None*
Sweeps the output voltage from the start voltage to the end voltage inclusive at the given increments. Afterwards the device sweeps in reverse (hysteresis) then sets the output voltage to 0 V. Delay sets the delay in milliseconds between each voltage step. Does not perform any measurements and does not return any data. Sending any other command will stop the measurement immediately and returns the measured points.

5. Vsense

The Vsense module is the driver for both voltmeter (Vsense) channels onboard. It is activated by either the prefix *vsense1* or *vsense2*. The supported commands are listed and detailed below using the *vsense1* prefix. The same commands can be used for the second Vsense channel *vsense2* by changing the prefix to match.

5.1 Properties

enabled: Boolean	**osr:** integer

5.2 Commands

Returns	Command
Boolean	vsense1 get enabled
integer	vsense1 get osr
	vsense1 measure
	vsense1 measure *integer*
	vsense1 set enabled *Boolean*
	vsense1 set osr *integer*

5.3 Property Documentation

(I) enabled: Boolean

This property holds whether the Vsense is turned on or off. The power ON value is False.

Access functions:

Boolean	vsense1 get enabled
	vsense1 set enabled *Boolean*

(II) osr: integer

This property holds the Oversample Rate (OSR) index of the Vsense. OSR can take values 0 – 19 with wrapping supported (i.e., 22 will set the OSR to 2). This determines the number of samples that are taken per data point, or the sampling speed. More details can be found in **Section 6.2** of the Ossila Source Measure Unit User Manual. The power ON value is 5.

Access functions:

integer	vsense1 get osr
	vsense1 set osr *integer*

5.4 Command Documentation

vsense1 measure
Arguments
None
Expected response format – *[float] – Array*
Measures the voltage and returns it.
vsense1 measure *integer*
Arguments
Integer – number of data points to measure
Expected response format – *[float;float;.....;float] – Array*
Performs the specified number of voltage measurements and returns them in an array.

6. Shutter/Trigger

The Shutter/Trigger is connected to the same BNC can be used either as an input or an output. It is therefore not possible to use the shutter and the trigger functionality at the same time.

6.1 Properties

shutter: Boolean	

6.2 Commands

Returns	Command
Boolean	get shutter
	set shutter *Boolean*
	trigger wait

6.3 Property Documentation

(I) Shutter: Boolean

This property holds whether the shutter is on or off. When on the shutter outputs 5 V. The power ON value is False.

Access functions:

Boolean	get shutter
	set shutter *Boolean*

6.4 Command Documentation

trigger wait

Arguments

None

Expected response format – *None*

This command accepts no arguments and returns no value.

When running this command, the device will be blocked, not allowing any further commands to run until the trigger line changes, either from **low** to **high** or **high** to **low**. This allows the user to send another command that will be run whenever the trigger line changes.

7. Ethernet

The Ethernet connector has some basic CLOI support to enable access to the most used functionality, such as resending a DHCP request. Ethernet commands use the prefix *`eth0`*.

7.1 Properties

dhcp: Boolean	**network:** array

7.2 Commands

Returns	Command
	eth0 clear network
	eth0 connect
Boolean	eth0 get dhcp
array	eth0 get network
	eth0 ip
	eth0 load network
MAC	eth0 mac
	eth0 save network
	eth0 set dhcp *Boolean*
	eth0 set network *ip-address ip-address ip-address ip-address*

7.3 Property Documentation

(I) dhcp: Boolean

This property holds whether the Source Measure Unit will use DHCP when connecting to a network. The default power ON property is True.

Access functions:

Boolean	eth0 get dhcp
	eth0 set dhcp *Boolean*

(II) network: array

This property holds an array of 4 IP addresses used to set/connect to static networks when DHCP is disabled. The IP addresses are in the order: Source Measure Unit IP address, subnet mask, default gateway, DNS server. They are returned in an array and are set and returned in the same order. The default power ON value is [255.255.255.255;255.255.255.255;255.255.255.255;0.0.0.0] or [192.168.0.200;255.255.255.0;0.0.0.0;8.8.8.8] if DHCP is disabled.

Access functions:

array	eth0 get network
	eth0 set network *ip-address ip-address ip-address ip-address*

7.4 Command Documentation

eth0 clear network
Arguments
None
Expected response format – *Boolean*
Resets the static network settings to the default values.
eth0 connect
Arguments
None
Expected response format – *None*
This command will start a DHCP request to request a new IP address. The function returns no value as the request is run through an internal command, however, an easy way to check if this function succeeded is to check for an IP address.
eth0 ip
Arguments
None
Expected response format – *IP_address*
Returns the assigned IP address.
If the ethernet is disconnected, or if the board has not been assigned an IP address, the command will return either 0.0.0.0 or 255.255.255.255. Both are invalid addresses and as such indicate failure to connect.
eth0 load network
Arguments
None
Expected response format – *Boolean*
Loads the saved static network settings.

eth0 mac
Arguments
None
Expected response format – *MAC address AA:BB:CC:DD:EE:FF*
Returns the MAC address of the board.
eth0 save network
Arguments
None
Expected response format – *Boolean*
Saves the DHCP flag and static network settings set using the ***`set network`*** command so that they are retained when the unit is switched off and loaded after being switched on.

8. I/O Interface

The Ossila Source Measure Unit provides several I/O functionalities through the external I/O sockets on the rear of the unit. These sockets provide digital/analogue pins, serial bus, i2c bus, reset, 3.3 V, and GND. Due to the intersections between the controller used in the Source Measure Unit and the Arduino community, these functions allow compatibility with some basic Arduino functions.

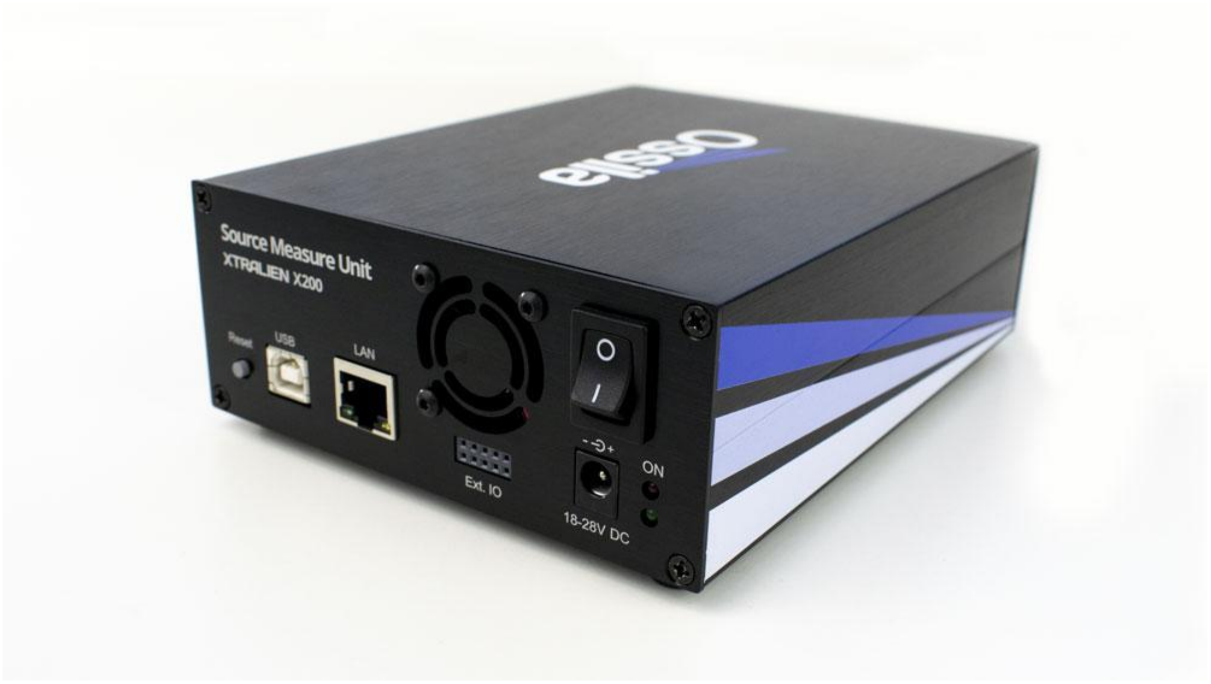

Figure 8.1. Back panel of the Source Measure Unit.

A labelled schematic of the socket is shown below.

Table 8.1. External I/O socket schematic.

SDA / D20	SCL / D21	3.3 V	GND	D54 / A0
Tx / D1	Rx / D0	3.3 V	GND	RTS

Caution

Caution: these lines are connected directly to the system controller and are rated for 3.3 V only, with a source/sink current of 3 ~ 5 mA. Misuse of this interface may damage the unit and void its warranty.

8.1 Commands

Returns	Commands
float	io analog read *integer*
	io analog write *integer float*
Boolean	io digital read *integer*
	io digital write *integer Boolean*
	io exserial available
	io exserial end
Boolean	io exserial find *string*
Boolean	io exserial flush
	io exserial print *string*
Boolean	io exserial println *string*
string	io exserial read *integer*
string	io exserial readbuf
	io exserial start *integer*
	io exserial write *integer*
	io i2c begin
	io i2c clock integer
integer	io i2c read *integer integer integer*
integer	io i2c readd *integer integer integer*
integer	io i2c readw *integer integer integer*
	io i2c write *integer integer integer*
	io i2c writed *integer integer integer*
	io i2c writew *integer integer integer*
	io set *string integer*

8.2 Command Documentation

io analog read *integer*
Arguments
Integer – IO pin number
Expected response format – *Float*
Returns the current analogue value of the given IO pin.
io analog write *integer float*
Arguments
Integer – IO pin number
Float – pin value (*0 to 3.3V*)
Expected response format – *None*
Sets the given analogue value to given IO pin.
io digital read *integer*
Arguments
Integer – IO pin number
Expected response format – *Boolean*
Returns the state of the given IO pin.
io digital write *integer Boolean*
Arguments
Integer – IO pin number
Boolean – pin value
Expected response format – *None*
Sets the given value to given IO pin.
io exserial available
Arguments
None
Expected response format – *integer*
Returns the number of bytes that are available to read in the buffer.
io exserial end
Arguments
None
Expected response format – *Boolean*
Ends and closes the external serial port, returns True on success.

io exserial find *string*

Arguments

String – data string to find

Expected response format – *Boolean*

Returns True if the given string data found in the external serial port buffer, False if timeout.

io exserial flush

Arguments

None

Expected response format – *Boolean*

Waits for the transmission of outgoing external serial data to complete, returns True on success.

io exserial print *string*

Arguments

String – data string

Expected response format – *None*

Prints the given string data to the external serial port.

io exserial println *string*

Arguments

String – data string

Expected response format – *Boolean*

Prints the given string data to the external serial port with new line feed.

io exserial read *integer*

Arguments

Integer – No of bytes to read

Expected response format – *String*

Returns string data of given No of bytes from the external serial port buffer.

io exserial readbuf

Arguments

None

Expected response format – *String*

Returns string data of all data in the external serial port buffer.

io exserial start *integer*

Arguments

Integer – baud rate

Expected response format – *None*

Starts the external serial port.

io exserial write *integer*

Arguments

Integer – 8-bit data value

Expected response format – *None*

Sends the given **unsigned 8-bit integer** data to the external serial port.

io i2c read *integer integer integer*

Arguments

Integer – I2C device address

Integer – I2C register address

Integer – No. of byte to be read

Expected response format – *Integer*

Returns the given bytes value in **unsigned 8-bit integer** format.

io i2c readd *integer integer integer*

Arguments

Integer – I2C device address

Integer – I2C register address

Integer – No. of floats to be read

Expected response format – *float*

Returns the given floats value in **float** format.

io i2c readw *integer integer integer*

Arguments

Integer – I2C device address

Integer – I2C register address

Integer – No. of words to be read

Expected response format – *Integer*

Returns the given words value in **unsigned 16-bit integer** format.

io i2c write *integer integer integer*

Arguments

Integer – I2C device address

Integer – I2C register address

Integer – I2C register value

Expected response format – *None*

io i2c writed *integer integer float*

Arguments

Integer – I2C device address

Integer – I2C register address

float – I2C register value

Expected response format – *None*

io i2c writew *integer integer integer*

Arguments

Integer – I2C device address

Integer – I2C register address

Integer – I2C register value ***unsigned 16-bit integer*** *format.*

Expected response format – *None*

io set *string integer*

Arguments

String – IO pin mode (input, output, pullup)

Integer – IO pin number

Expected response format – *None*

Sets the given mode for the given IO pin. Pullup mode reflects input with internal pullup.

9. Top Level Commands

The top-level commands featured here do not belong to any specific module, and so have been attached to the top level (board level).

9.1 Properties

dark mode: Boolean	**fan mode:** integer

9.2 Commands

Returns	Commands
string	board no
Boolean	get dark mode
integer	get fan mode
string	product id
	reset
string	serial
	set dark mode *Boolean*
	set fan mode *integer*
float	temp read
matrix	version

9.3 Property Documentation

(I) dark mode: Boolean

This property holds whether the SMU indicator LEDs switch on when the SMU is on. When False, the SMU lights will not turn on. The power ON value is True.

Access functions:

Boolean	get dark mode
	set dark mode *Boolean*

(II) fan mode: integer

This property holds the mode of operation for the on-board fan. 0 turns the fan off, 1 turns the fan on, and 2 or greater sets the fan to automatically turn on and off based on the temperature of the unit. The power ON value is 2.

Access functions:

integer	get fan mode
	set fan mode *integer*

9.4 Command Documentation

board no
Arguments
None
Expected response format – *String*
Returns the number of the board (xxx).
product id
Arguments
None
Expected response format – *String*
Returns the product ID of the board.
reset
Arguments
None
Expected response format – *None*
Resets the unit.
Warning: this will reset everything to their power on values.
serial
Arguments
None
Expected response format – *String*
Returns the serial number of the board in six hexadecimal bytes.
temp read
Arguments
None
Expected response format – *Float*
Returns the temperature measurement of the onboard sensor in °C.

version

Arguments

None

Expected response format – *[version,version] – [hardware,firmware] – Matrix*

Returns the hardware version and the firmware version of the unit.

10. Python Package: xtralien

For controlling the Source Measure Unit using Python we have a package that can be downloaded and installed from the Python Package Index (https://pypi.org/project/xtralien/) via pip by using:

```
pip install xtralien
```

With this package, every command in this document can be used via dot notation. As an example, to perform a ***oneshot*** measurement with SMU 1, you can use:

```python
measurement = xtralien.Device(address).smu1.oneshot(set_voltage)
```

The package converts array and matrix returns into NumPy arrays, and IP addresses, MAC addresses, and versions into strings.

Please note, to control the unit via USB you will also need to download and install the pySerial package (https://pypi.org/project/pyserial/).

10.1 The Device Class

xtralien.Device(address: str = None, port: int = None, serial_timeout: float = 0.1)

The ***Device*** class creates and holds the Source Measure Unit object. When creating a ***Device*** object, you give it the COM (USB) or IP (Ethernet) address of Source Measure Unit you want to connect to. For COM addresses you can optionally supply a timeout for the connection using the keyword argument ***serial_timeout***, which is 0.1 s by default. For IP addresses you will also need to supply a port number using the ***port*** keyword argument. The port is typically 8888.

```python
device = xtralien.Device(‘COM5’)
device = xtralien.Device(‘192.168.0.20’, port=8888)
```

You can then use any of the commands in this document with dot notation:

```python
measurement = device.smu1.oneshot(set_voltage)
```

For any command that does not return a value, ***response=0*** must be included in the arguments, or your code will hang. For example, when enabling SMU 2, you would use:

```
device.smu2.set.enabled(True, response=0)
```

When you are finished using the Source Measure Unit, you should close the connection using the close function:

```
device.close()
```

If you do not do this, you will need to reset the unit before you can connect to it again. Alternatively, you can use the `with` built-in statement to connect to the device, which will automatically handle closing the connection when you are finished.

10.2 Example Script

Below is a brief example showing how a simple current-voltage measurement using SMU 1, sweeping from 0 to 10 V in steps of 1 V, can be performed using the Xtralien Python package:

```
import xtralien


### 1. Connect to the Source Measure Unit using USB ###
with xtralien.Device('COM5') as SMU:

    ### 2. Turn on SMU 1 ###
    SMU.smu1.set.enabled(True, response=0)

    ### 3. Loop through the voltages to measure ###
    for set_voltage in range(0, 11):

        ### 4. Set voltage, measure voltage and current ###
        voltage, current = SMU.smu1.oneshot(set_voltage)[0]

        ### 5. Print measured voltage and current ###
        print(f'V: {voltage} V; I: {current} A')

    ### 6. Reset output voltage and turn off SMU 1 ###
    SMU.smu1.set.voltage(0, response=0)
    SMU.smu1.set.enabled(False, response=0)

```