1. Purpose

This document presents the Application Program Interface (API) for controlling the Vescent Gen2 Oscillator board.

1. Scope

This document describes the method of communication with the Gen2 Oscillator board using the USB, DB9 serial, or I2C interface. It provides a descritpion of each command in the API

1. USB Interface

Communication with the Gen2 Oscillator board can be done using the USB interface using ASCII based serial commands issued from a terminal program such as Putty or Tera Term. Commands sent this way use a text command followed by space delimited arguments.

1. DB9 Serial Interface

Communication with the Gen2 Oscillator board can be done using the 9 pin DB9 connection to a PC serial port interface using ASCII based serial commands issued from a terminal program such as Putty or Tera Term. Commands sent this way use a text command followed by space delimited arguments.

1. USB / DB9 Command Format

Commands use the following format:
[command name] [argument] [argument] [argument].

1. I2C interface

Communication with the Gen2 Oscillator board can be done using a 2 wire I2C interface. The Gen2 Oscillator board firmware contains metadata which can be extracted to expose all available commands by using a sequence of device enumeration commands described in this document.
The Gen2 Oscillator board is compatable with I2C Fast Mode 400 kbit/s speed and uses 7 bit addressing with the address of TBD.
Little Endian (least significant byte first) data format is used for multi-byte data types.

1. I2C Address

The default I2C address for the Gen2 Oscillator board is **0.**This address can be set using the **I2CADDR** command described at the end of this document. This command automatically saves the address to the board’s EEPROM so it will persist through power cycles.

1. I2C Command Format

Commands use the following format:

Byte 0 = Address

Byte 1 = Command Index Number

Bytes 2 – 7 = Argument Bytes

Bytes 2 through 7 are used for parameters in the order defined for the command. Unsigned 16 bit integer and Floating point values must be in Little Endian format (Least Significant Byte First)

1. I2C Command Metadata

The first two available API commands can be used to extract command metadata for all available commands.
**ENUMDEV** (USB or DB9 Serial Format)
or
I2C format:

Byte 0 = Address

Byte 1 = 0

The return value from this command contains 8 bytes. Bytes 0 and 1 are the important ones:

Byte 0 = device type (Gen2 Oscillator is type 18)

Byte 1 = number of commands

Call the next command twice for each of the number of commands returned by the ENUMDEV command.
**ENUMCMD** Command: (Called twice for each **command index** in number of commands)
First Iteration:
**ENUMCMD 0** (USB or DB9 Serial Format)
or
I2C format:

Byte 0 = Address

Byte 1 = 1

Byte 2 = command index

Byte 3 = **0**
The return value from the first iteration is:
Byte 0 = command index

Byte 1 = number of parameter bytes the command expects

Byte 2 = The types of parameters expected (See Parameter Types section below)
Byte 3 = The data type of the return value.

Second iteration

**ENUMCMD 1** (USB or DB9 Serial Format)
or
I2C format:

Byte 0 = Address

Byte 1 = 1

Byte 2 = command index

Byte 3 = **­1**

The return value from the second iteration is:
The ASCII command string (up to 8 bytes)

1. Parameter types

Byte 3 of the return from the first iteration defines the data type of any parameters used by the command. Two bits of the byte are used to identify the data type in each position of the command.
The parameters are defined by their position using these two bit values as follows:

#define CMD\_ARG0\_UINT8 (CMD\_UINT8 << 6)

#define CMD\_ARG0\_UINT16 (CMD\_UINT16 << 6)

#define CMD\_ARG0\_INT16 (CMD\_INT16 << 6)

#define CMD\_ARG0\_FLOAT (CMD\_FLOAT << 6)

#define CMD\_ARG1\_UINT8 (CMD\_UINT8 << 4)

#define CMD\_ARG1\_UINT16 (CMD\_UINT16 << 4)

#define CMD\_ARG1\_INT16 (CMD\_INT16 << 4)

#define CMD\_ARG1\_FLOAT (CMD\_FLOAT << 4)

#define CMD\_ARG2\_UINT8 (CMD\_UINT8 << 2)

#define CMD\_ARG2\_UINT16 (CMD\_UINT16 << 2)

#define CMD\_ARG2\_INT16 (CMD\_INT16 << 2)

#define CMD\_ARG2\_FLOAT (CMD\_FLOAT << 2)

#define CMD\_ARG3\_UINT8 CMD\_UINT8

#define CMD\_ARG3\_UINT16 CMD\_UINT16

#define CMD\_ARG3\_INT16 CMD\_INT16

#define CMD\_ARG3\_FLOAT CMD\_FLOAT

Where the bits are set according to these **basic types**:
#define CMD\_UINT8 0x00

#define CMD\_UINT16 0x01

#define CMD\_INT16 0x02

#define CMD\_FLOAT 0x03

#define CMD\_RAW 0x04 (string of up to 8 bytes)

#define CMD\_STATUS 0x05

#define CMD\_ASCII 0x06

#define CMD\_TEST 0x07

#define CMD\_UINT32 0x08

#define CMD\_NO\_ARGS 0xFF

#define CMD\_NONE 0xFF

Note **Basic types** define the return type

1. Description of Available Commands

The number in **[ ]** is the command index. This number is used for I2C commands only. The ASCII command is used for USB or DB9 Serial commands.
**NOTE:** Some commands which appear in the ENUMDEV / \_ENUMCMD enumeration sequence are not listed here. The omitted commands are either non-functional in this application or are used for factory test sequences.

|  |  |
| --- | --- |
| [0] EnumDev | Enumerate Device. Returns the device ID, firmware revision, device capabilities, command set revision, and number of supported commands. |
| *Parameters*:None | *Example:* ENUMDEV  0 1 1 1 25 |
| [1] \_ENUMCMD | Reads metadata for all supported commands. Two calls are needed for each command. |
| *Parameters*: [uint8] command index [uint8] 0 for first call 1 for second call | *Example:* \_ENUMCMD [command index] 0 byte 1 = Command Index byte 2 = Number of Argument bytes byte 3 = Encoded Argument Types byte 4 = Return Type  \_ENUMCMD [command index] 1 up to 8 bytes ASCII Command String |
| [2] Reset | Resets the ICE board to initial power on conditions. |
| *Parameters*:None | *Example:* Reset  |
| [4] \_FACTORY | Tells ICE2 board to restore factory default settingsNOTE: There is no return value from this function.Power Cycle the device to complete the restoration. |
| *Parameters*: [uint8] any unsigned byte value | *Example:* FACTORY 1   |
| [5] Status | Queries ICE board for its current status.  |
| *Parameters*:None  | *Example:* Status  Busy |
| [6] Abort | Tells ICE board to abort the long-running command it is currently executing. |
| *Parameters*:None  | *Example:* Abort  |
| [10] Save | Saves the board’s current settings into EEPROM. Unsaves changes will be lost when the board is powered off. Returns 0 = SUCCESS |
| *Parameters*:None  | *Example:* Save 0 |
| [13] Version | Returns firmware version of ICE2 board. |
| *Parameters*:None | *Example:* Version  (Returns 8 bytes. Bytes 5 and 6 are the major and minor firmware versions) |

# FL Controller

|  |  |
| --- | --- |
| [16] CONTROL? | Returns the enumerated operating mode for Channel.Where:Channel 0 (Temperature Channel)0 = Constant Current Mode OFF1 = Temperature Control Mode OFF2 = Constant Current Mode ON3 = Temperature Control Mode ON Channel 1 (High Voltage Channel)128 = Gain = 1 V/V; Range = +/- 10V OFF129 = Gain = 20 V/V; Range = 0-200 OFF130 = Gain = 1 V/V; Range = +/- 10V ON131 = Gain = 20 V/V; Range = 0-200 ON**NOTE: Channel 1 (HV channel adds 128 to the return value to distinguish it from the Temperature channel.)** |
| *Parameters*: [uint8] CHANNEL 0 = Temp Chan 1 = HV Chan | *Example:* CONTROL? 1  128 |

|  |  |
| --- | --- |
| [17] CONTROL | Sets the enumerated operating mode for Channel.Where:Channel 0 (Temperature Channel)0 = Constant Current Mode OFF1 = Temperature Control Mode OFF2 = Constant Current Mode ON3 = Temperature Control Mode ON Channel 1 (High Voltage Channel)0 = Gain = 1 V/V; Range = +/- 10V OFF1 = Gain = 20 V/V; Range = 0-200 OFF2 = Gain = 1 V/V; Range = +/- 10V ON3 = Gain = 20 V/V; Range = 0-200 ONReturns CONTROL? |
| *Parameters*: [uint8] CHANNEL [uint8] MODE | *Example:* CONTROL 1 2 130 |
| [18] ERROR? | Reads the Error codes for ChannelReturn Values:49152 = No Errors49408 = Power Limit Exceeded |
| *Parameters*: [uint8] Channel | *Example:* Error? 1 49152 |
| [19] ERROR | Clears an Error code for ChannelNote: use this command to clear an error code obtained from the ERROR? command. Values to clear error codes:256 Clears error code 49408Returns the Error code resulting from clearing the error |
| *Parameters*: [uint8] Channel [uint16] Code | *Example:* Error 1 256 49152 |
| [20] LIMITS? | Returns the Minimum or Maximum Voltage Limit for the parameter.Where:0 = Minimum HV Voltage1 = Maximum HV Voltage |
| *Parameters*: [uint8] Limit Index | *Example:* Limits? 0  0.0 |

# Temperature Channel Specific (Channel 0)

|  |  |
| --- | --- |
| [24] TEMPSET? | Returns the temperature setpoint for the temperature channel (channel 0) |
| *Parameters*: [uint8] CHANNEL 0 = Temp Chan 1 = INVALID | *Example:* Tempset? 0  25.0 |
| [25] TEMPSET | Sets the temperature setpoint for the temperature channel (channel 0) |
| *Parameters*: [uint8] CHANNEL 0 = Temp Chan 1 = INVALID [Float] Temperature | *Example:* Tempset 0 24.0 24.0 |
| [26] Bipolar? | Returns whether the temperature loop for channel CHANNEL is bipolar. Return is a STATUS Enumeration where:4 = ON5 = OFF |
| *Parameters*: [uint8] CHANNEL (MUST BE 0) | *Example:* Bipolar? 0  4 |
| [27] Bipolar | Sets the temperature loop for channel CHANNEL bipolar (heating only) on or off. Turn Bipolar off when driving resistive heaters. Returns output from Bipolar? |
| *Parameters*: [uint8] Channel (MUST BE 0) [uint8]1 = true0 = false | *Example:* Bipolar 0 1 4 |

|  |  |
| --- | --- |
| [28] Temp? | Returns the current measured temperature for CHANNEL in degrees Celcius. |
| *Parameters*: [uint8] CHANNEL(Must be 0) | *Example:* Temp? 0  24.21 |
| [29] TERROR? | Returns the temperature error (Tsetpoint – Tactual) in degrees Celcius. |
| *Parameters*: [uint8] CHANNEL(Must be 0) | *Example:* TError? 0  .0024 |
| [30] CURRENT? | Returns the current flowing through TEC (or resistive heater) in Amps. |
| *Parameters*: [uint8] CHANNEL(Must be 0) | *Example:* Current? 0  .654 |
| [31] TEMPMIN? | Returns the minimum temperature for CHANNEL in degrees Celcius. |
| *Parameters*: [uint8] CHANNEL(Must be 0) | *Example:* TEMPMIN? 0 -5.000000 |
| [32] TEMPMIN | Sets the minimum temperature CHANNEL to MINTEMP degrees Celcius. Returns TempMin? Note: You cannot set MINTEMP greater than the temperature setpoint. Attempting to do so will not change the minumum temperature. |
| *Parameters*: [uint8] CHANNEL(Must be 0) [Float] MINTEMP | *Example:* TEMPMIN 0 -5 -5.000000 |
| [33] TEMPMAX? | Returns the maximum temperature for CHANNEL in degrees Celcius.  |
| *Parameters*: [uint8] CHANNEL(Must be 0) | *Example:* TEMPMAX? 0 55.000000 |

|  |  |
| --- | --- |
| [34] TEMPMAX | Sets the maximum temperature CHANNEL to MAXTEMP. Returns TempMax?. Note: You cannot set MAXTEMP less than the temperature setpoint. Attempting to do so will not change the maximum temperature. |
| *Parameters*: [uint8] CHANNEL(Must be 0) [uint8] State  | *Example:* TEMPMAX 0 24.5 24.500000 |
| [35] MAXCURR? | Returns the maximum current limit for CHANNEL in Amps. (Applies only to temperature channel 0) |
| *Parameters*: [uint8] CHANNEL (Must be 0)  | *Example:* Maxcurr? 0 2.000 |
| [36] MAXCURR | Sets the current limit for CHANNEL in Amps. Returns the result of MAXCURR? in Amps(Applies only to temperature channel 0) |
| *Parameters*: [uint8] CHANNEL (Must be 0) [Float] Current  | *Example:* Maxcurr 0 3.5 3.5000 |
| [37] PGAIN? | Returns the proportional gain for CHANNEL 0 (Temperature Channel) |
| *Parameters*: [uint8] CHANNEL (Must be 0)  | *Example:* Pgain? 0 1.800 |
| [38] PGAIN | Sets the proportional gain for CHANNEL 0 (Temperature Channel)Returns the result of PGAIN? |
| *Parameters*: [uint8] CHANNEL (Must be 0) [Float] gain  | *Example:* Pgain 0 1.8 1.800 |
| [39] INTEG? | Returns the integral time constant for CHANNEL 0 (Temperature Channel) |
| *Parameters*: [uint8] CHANNEL (Must be 0)  | *Example:* Integ? 0 0.825 |
| [40] INTEG | Sets the integral time constant for CHANNEL 0 (Temperature Channel) Returns the result of INTEG?  |
| *Parameters*: [uint8] CHANNEL (Must be 0) [float] Integral time constant | *Example:* Integ 0 0.8 0.800 |
| [41] DERIV? | Returns the derivative time constant for CHANNEL 0 (Temperature Channel) |
| *Parameters*: [uint8] CHANNEL (Must be 0)  | *Example:* Deriv? 0 0.200 |
| [42] DERIV | Sets the derivative time constant for CHANNEL 0 (Temperature Channel). Returns the result of DERIV? |
| *Parameters*: [uint8] CHANNEL (Must be 0) [Float] derivative time constant  | *Example:* Deriv 0 0.2 0.200 |
| [43] SLEW? | Returns the slew rate for CHANNEL 0 (Temperature Channel) in degrees C per minute |
| *Parameters*: [uint8] CHANNEL (Must be 0)  | *Example:* Slew? 0 1.5 |
| [44] SLEW | Sets the slew rate for CHANNEL 0 (Temperature Channel) in degrees C per minute. Returns the result of SLEW? |
| *Parameters*: [uint8] Channel [Float] slew rate  | *Example:* Slew 0 1.5 1.5 |
| [45] PGAINEN? | Returns the enabled/disabled status of the proportional gain factor for CHANNEL 0 (Temperature Channel)Return is a STATUS Enumeration where:4 = ON5 = OFF |
| *Parameters*: [uint8] Channel (Must be 0)  | *Example:* Pgainen? 0 4 |
| [46] PGAINEN | Sets the enabled/disabled status of the proportional gain factor for CHANNEL 0 (Temperature Channel) Returns the result of PGAINEN? |
| *Parameters*: [uint8] Channel (Must be 0) [uint8] State 0 = Disabled 1 = Enabled  | *Example:* Pgainen 0 1 4 |
| [47] INTEGEN? | Returns the enabled/disabled status of the integral time constant factor for CHANNEL 0 (Temperature Channel)Return is a STATUS Enumeration where:4 = ON5 = OFF |
| *Parameters*: [uint8] Channel (Must be 0)  | *Example:* Integen? 0 5 |
| [48] INTEGEN | Sets the enabled/disabled status of the integral time constant factor for CHANNEL 0 (Temperature Channel) Returns the result of INTEGEN? |
| *Parameters*: [uint8] Channel (Must be 0) [uint8] State 0 = Disabled 1 = Enabled  | *Example:* Integen 0 0 5 |
| [49] DERIVEN? | Returns the enabled/disabled status of the derivative time constant factor for CHANNEL 0 (Temperature Channel)Return is a STATUS Enumeration where:4 = ON5 = OFF |
| *Parameters*: [uint8] Channel (Must be 0)  | *Example:* Deriven? 0 5 |
| [50] DERIVEN | Sets the enabled/disabled status of the derivative time constant factor for CHANNEL 0 (Temperature Channel) Returns the result of DERIVEN? |
| *Parameters*: [uint8] Channel (Must be 0) [uint8] State 0 = Disabled 1 = Enabled  | *Example:* Deriven 0 0 5 |
| [51] SLEWEN? | Returns the enabled/disabled status of the slew rate limiter for CHANNEL 0 (Temperature Channel)Return is a STATUS Enumeration where:4 = ON5 = OFF |
| *Parameters*: [uint8] Channel (Must be 0)  | *Example:* Slewen? 0 5 |
| [52] SLEWEN | Sets the enabled/disabled status of the slew rate limiter for CHANNEL 0 (Temperature Channel). Returns the result of SLEWEN?  |
| *Parameters*: [uint8] Channel (Must be 0) [uint8] State 0 = Disabled 1 = Enabled  | *Example:* Slewen 0 0 5 |
| [53] POWER? | Returns the power output for CHANNEL (Temperature Channel) in watts |
| *Parameters*: [uint8] Channel (Must be 0)  | *Example:* Power? 0 8.3 |
| [54] PERIOD? | Returns the sample interval for CHANNEL 0 (Temperature Channel) in ms (Default is 10 ms) |
| *Parameters*: [uint8] Channel (Must be 0)  | *Example:* PERIOD? 0 10 |
| [55] PERIOD | Sets the sample interval for CHANNEL 0 (Temperature Channel) in msNOTE: The minimum sample period is 10 msReturns PERIOD? |
| *Parameters*: [uint8] Channel (Must be 0) [uint16] Period  | *Example:* Period 0 10 10 |
| [56] POL? | Returns the output polarity for CHANNEL 0 (Temperature Channel). On indicates factory default Negative polarity, Off indicates alternative Positive polarityReturn is a STATUS Enumeration where:4 = ON5 = OFF |
| *Parameters*: [uint8] Channel (Must be 0) | *Example:* POL? 0 4 |
| [57] POLARITY | Sets the output polarity for CHANNEL 0 (Temperature Channel). State = 1 sets factory default Negative polarity, state = 0 indicates alternative Positive polarityReturns POL? |
| *Parameters*: [uint8] Channel (Must be 0) [uint8] State 0 = Disabled 1 = Enabled  | *Example:* POLARITY 0 0 5 |
| [58] BETA? | Returns the Betatherm Beta thermistor coefficient for CHANNEL 0 (Temperature Channel) |
| *Parameters*: [uint8] Channel (Must be 0)  | *Example:* BETA? 0 3450.00000 |
| [59] BETA | Sets the Betatherm Beta thermistor coefficient for CHANNEL 0 (Temperature Channel). Returns BETA?NOTE: This command triggers a recalculation of Steinhart-Hart A, B, and C coefficients.  |
| *Parameters*: [uint8] Channel (Must be 0) [Float] value  | *Example:* BETA 0 3450 3450.00000 |
| [60] REFTEMP? | Returns the Betatherm Reference temperature C for CHANNEL 0 (Temperature Channel) |
| *Parameters*: [uint8] Channel (Must be 0)  | *Example:* REFTEMP? 0 25.0 |
| [61] REFTEMP | Sets the Betatherm Reference temperature C for CHANNEL 0 (Temperature Channel). Returns REFTEMP?NOTE: This command triggers a recalculation of Steinhart-Hart A, B, and C coefficients. |
| *Parameters*: [uint8] Channel (Must be 0) [Float] value  | *Example:* REFTEMP 0 25.0 25.0 |
| [62] REFRES? | Returns the Betatherm Resistance at the reference temperature C for CHANNEL 0 (Temperature Channel) |
| *Parameters*: [uint8] Channel (Must be 0)  | *Example:* REFRES? 0 10000.0 |
| [63] REFRES | Sets the Betatherm Resistance at the reference temperature C for CHANNEL 0 (Temperature Channel).Returns REFRES?NOTE: This command triggers a recalculation of Steinhart-Hart A, B, and C coefficients. |
| *Parameters*: [uint8] Channel (Must be 0) [Float] value  | *Example:* REFRES? 0 10000.0 10000.0 |
| [64] TCOEFA? | Returns the Steinhart Hart A coefficient for CHANNEL 0 (Temperature Channel) |
| *Parameters*: [uint8] Channel (Must be 0) | *Example:* TCOEFA? 0 2.108508173 |
| [65] TCOEFA | Sets the Steinhart Hart A coefficient for CHANNEL 0 (Temperature Channel)Returns TCOEFA? |
| *Parameters*: [uint8] Channel (Must be 0) [Float] value  | *Example:* TCOEFA 0 2.108508173 2.108508173 |
| [66] TCOEFB? | Returns the Steinhart Hart B coefficient for CHANNEL 0 (Temperature Channel) |
| *Parameters*:[uint8] CHANNEL(Must be 0) | *Example:* TCOEFB? 0 0.797204727 |
| [67] TCOEFB | Sets the Steinhart Hart B coefficient for CHANNEL 0 (Temperature Channel)Returns TCOEFB?NOTE: This command will recalculate the Betatherm Beta value. |
| *Parameters*:[uint8] CHANNEL(Must be 0)[FLOAT] value | *Example:* TCOEFB 0 0.797204727 0.797204727 |
| [68] TCOEFC? | Returns the Steinhart Hart C coefficient for CHANNEL 0 (Temperature Channel) |
| *Parameters*:[uint8] CHANNEL | *Example:* TCOEFC? 0 6.535076315 |
| [69] TCOEFC | Sets the Steinhart Hart C coefficient for CHANNEL 0 (Temperature Channel)Returns TCOEFC? |
| *Parameters*:[uint8] CHANNEL(Must be 0)[FLOAT] value | *Example:* TCOEFC 1 6.535076315 6.535076315 |
| [70] MAXPWR? | Returns the power limit in Watts for CHANNEL 0 (Temperature Channel) |
| *Parameters*: [uint8] Channel (Must be 0)  | *Example:* MaxPwr? 0 7.0 |
| [71] MAXPWR | Sets the power limit in Watts for CHANNEL 0 (Temperature Channel)Returns MAXPWR?NOTE: Maximum Power for a channel is limited by the available power for the system. |
| *Parameters*: [uint8] Channel (Must be 0) [Float] watts  | *Example:* MaxPwr 0 7.0 7.0 |
| [72] CVOLT? | Returns the absolute voltage [V] across the load for CHANNEL 0 (Tempearture Channel) |
| *Parameters*:[uint8] CHANNEL | *Example:* Cvolt? 0 2.438 |
| [73] TWARN? | Returns the CHANNEL 0 (Temperature Channel) temperature control warning range [mK] |
| *Parameters*: [uint8] Channel (Must be 0)  | *Example:* Twarn? 0 1.000 |
| [74] TWARN | Sets the CHANNEL 0 (Temperature Channel) temperature control warning range [mK]Returns TWARN? |
| *Parameters*: [uint8] Channel (Must be 0) [Float] temperature  | *Example:* Twarn 0 2.0 2.000 |
| [75] CURRSET? | Returns the CHANNEL 0 (Temperature Channel) manual mode current setting [A] |
| *Parameters*: [uint8] Channel (Must be 0)  | *Example:* Currset 0 0.300 |
| [76] CURRSET | Sets the CHANNEL 0 (Temperature Channel) manual mode current setting [A] |
| *Parameters*: [uint8] Channel (Must be 0)  [Float] Current  | *Example:* Currset 0 0.200 0.200 |
| [77] AVLPWR? | Returns the available power [W] for CHANNEL 0 (Temperature Channel) |
| *Parameters*: None  | *Example:* Avlpwr? 3.5 |
| [78] TTLPWR? | Returns the total power [W] for CHANNEL 0 (Temperature Channel) |
| *Parameters*: None  | *Example:* TTLPWR? 5.0 |
| [79] TEMPLUT | Recalculates the temperature linearization table for CHANNEL 0 (Temperature Channel). This command should be run after changing the temperature setpoint.This command does not return a value. |
| *Parameters*: [uint8] Channel (Must be 0)  | *Example:* Templut 0 |
| [80] LDIMPD? | Returns the load impedance for CHANNEL 0 (Temperature Channel) |
| *Parameters*: [uint8] Channel (Must be 0)  | *Example:* LDIMPD? 0 6.300 |
| [81] SFTYTMT? | Returns the number of seconds to delay shutdown if CHANNEL 0 (Temperature Channel)’s temperature goes above or below the critical levels.  |
| *Parameters*: [uint8] Channel (Must be 0)  | *Example:* SFTYTMT? 0 30.00 |
| [82] SFTYTMT | Sets the number of seconds to delay shutdown if CHANNEL 0 (Temperature Channel)’s temperature goes above or below the critical levels.Returns SFTYTMT? |
| *Parameters*: [uint8] Channel (Must be 0) [Float] time  | *Example:* SFTYTMT 0 10 10.0000 |
| [83] LASTI? | Returns the most recent output current for CHANNEL 0 (Temperature Channel) |
| *Parameters*: [uint8] Channel (Must be 0)  | *Example:* LASTI? 0 2.3000 |
| [84] LASTV? | Returns the most recent output voltage for CHANNEL 0 (Temperature Channel) |
| *Parameters*: [uint8] Channel Must be 0)  | *Example:* Lastv? 0 3.800 |
| [85] MLDCTHR | Sets the Mode Lock DC Threshold [V] for CHANNEL 0 (Temperature Channel) Valid values are 0 – 8 VReturns MLDCTHR? |
| *Parameters*: [Float] voltage  | *Example:* MLDCTHR 0 3.0 3.0 |
| [86] MLDCTHR? | Returns the Mode Lock DC Threshold [V] for CHANNEL 0 (Temperature Channel) |
| *Parameters*: None  | *Example:* MLDCTHR? 3.0 |
| [87] MLRMTHR | Sets the Mode Lock RMS Threshold for CHANNEL 0 (Temperature Channel) Valid values are >= 0.0Returns MLRMTHR? |
| *Parameters*: [Float] RMS threhold  | *Example:* MLRMTHR 2.0 2.0 |
| [88] MLRMTHR? | Returns the Mode Lock RMS Threshold for CHANNEL 0 (Temperature Channel) |
| *Parameters*: None  | *Example:* MLRMTHR?  2.0 |
| [89] MLSMPLM | Sets the number of samples to use for mode lock calculation for Channel 0 (Temperature Channel)Maximum 250Returns MLSMPLM? |
| *Parameters*: [uint8] samples  | *Example:* MLSMPLM 100 100 |
| [90] MLSMPLM? | Returns the number of samples to use for mode lock calculation for Channel 0 (Temperature Channel) |
| *Parameters*: None  | *Example:* MLSMPLM?  100 |
| [91] MODELK? | Returns the Mode Lock State for CHANNEL 0 (Temperature Channel)Where:4 = ON5 = OFF |
| *Parameters*:  None  | *Example:* ModeLK?  4 |
| [92] MLMEAN? | Returns the Mode Lock Mean for CHANNEL 0 (Temperature Channel) |
| *Parameters*:  None  | *Example:* MLMEAN?  1.05 |
| [93] MLVAR? | Returns the Mode Lock Variance for CHANNEL 0 (Temperature Channel) |
| *Parameters*:  None  | *Example:* MLVAR?  0.0003 |
| [94] MLSTDDV? | Returns the Mode Lock Standard Deviation for CHANNEL 0 (Temperature Channel) |
| *Parameters*:  None  | *Example:* MLSTDDV? 0.233 |

# High Voltage Commands (Channel 1)

|  |  |
| --- | --- |
| [95] DCBIASV? | Returns DC Bias Voltage [V] for CHANNEL 1 (HV Channel) |
| *Parameters*: [uint8] CHANNEL (Must be 1) | *Example:* DCBIASV? 1  75.300 |
| [96] DCBIASV | Sets DC Bias Voltage [V] for CHANNEL 1 (HV Channel)Returns DCBIASV? |
| *Parameters*: [uint8] CHANNEL (Must be 1) [Float] voltage | *Example:* DCBIASV 1 80.0 80.0 |
| [97] VLIM? | Reads the Voltage Limit [V] for CHANNEL 1 (HV Channel) |
| *Parameters*: [uint8] Channel (Must be 1) | *Example:* VLIM? 1 180.00 |
| [98] VLIM | Sets the Voltage Limit [V] for CHANNEL 1 (HV Channel)Returns VLIM? |
| *Parameters*: [uint8] Channel [Float] voltage | *Example:* VLIM 1 170.0 170.000 |
| [99] OUTVOLT? | Returns Output Voltage [V] for CHANNEL 1 (HV Channel) |
| *Parameters*: [uint8] CHANNEL (Must be 1) | *Example:* OUTVOLT? 1 125.44 |
| [100] HWTEMP? | Returns Hardware Temperature [C] for CHANNEL 1 (HV Channel) |
| *Parameters*: [uint8] Channel  | *Example:* HWTEMP? 1 43.200 |

# I2C Configuration Commands

|  |  |
| --- | --- |
| [101] I2CADDR? | Returns I2C address |
| *Parameters*: None  | *Example:* I2CADDR? 28 |
| [102] I2CADDR | Sets I2C address (Range 0 – 127)Returns I2CADDR? |
| *Parameters*: [uint8] ADDRESS (Must be 0-127) | *Example:* I2CADDR 28 28 |