1. Purpose

This document presents the Application Program Interface (API) for controlling the Vescent Gen2 Laser Driver (LD) boards.

1. Scope

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

1. I2C interface

Communication with the Gen2 Laser Driver boards is achieved using a 2 wire I2C interface. The Gen2 Laser Driver 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 Laser Driver is compatable with I2C Fast Mode 400 kbit/s speed and uses 7 bit addressing with the address determined by the board’s installation position on the pump laser board..  
Little Endian (least significant byte first) data format is used for multi-byte data types.

1. I2C Address

The I2C address for the Gen2 Laser Driver board is determined by it’s position on the pump laser board**.** The address is fixed and cannot be set in the firmware.

1. I2C Command Format

Commands are between 2 and 8 bytes use the following format:

Byte 0 = I2C 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 API commands can be used to extract command metadata for all Laser Driver board commands.   
The command Metadata for each command contains the following information:

1. ASCII command string – Up to 8 ASCII characters which can be used to allow communication through an optional Serial to i2c translation program which can be written to facilitate control through a terminal program such as PuTTY or Tera Term.
2. Command Index - 1 byte integer value used for identifying the command
3. Number of argument bytes – 1 byte integer value
4. Parameter type byte – 1 byte integer value defined in the next section
5. Return type – 1 byte integer value defined in the next section

Extracting the metadata is a two step process which requires the use of the first two API commands.

1. The first Command obtains the device type and the number of commands available in the API  
   I2C format: (“ENUMDEV” is the ASCII identifier 0 is the I2C command index)  
     
   Byte 0 = I2C 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 Laser Driver is type **15**)  
   Byte 1 = number of commands
2. The second command is called twice iteratively for each of the number of commands returned by the ENUMDEV command.  
     
   **First Call:**  
   I2C format: (“\_ENUMCMD” is the ASCII identifier 1 is the I2C command index)  
   Byte 0 = I2C Address  
   Byte 1 = 1 (“ENUMCMD” command index)  
   Byte 2 = command index for the metadata requested  
   Byte 3 = **0** (Defines which return information to return for the command)  
   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 Call:**   
I2C format: (“\_ENUMCMD” is the ASCII identifier 1 is the I2C command index)  
Byte 0 = Address  
Byte 1 = 1 (“ENUMCMD” command index)  
Byte 2 = command index for the metadata requested  
Byte 3 = **­1 (**Defines which return information to return for the command)  
  
The return value from the second iteration is:  
The ASCII command string (up to 8 bytes)

1. I2C Parameter types

Byte 3 of the return from the first iteration of the ENUMCMD command 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.   
These definitions are or’d together to allow up to 4 parameters to be defined within the same byte.  
I.e. A command that uses a one byte channel parameter followed by a 4 byte floating point parameter will use (CMD\_ARG0\_UINT8 | CMD\_ARG1\_FLOAT) for the byte 3 parameter type definition.  
The parameters are defined by their position using the two bit values shown here:

#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 as the second byte (after the address byte) for I2C commands. The ASCII command is provided within the metadata and can be used if a command table is built to translate serial commands to I2C.  
Command Parameters must follow the address and command index bytes in the order shown. Multiple byte parameters. i.e. uint16 and floating point numbers must be loaded in little endian format (least significant byte first)  
**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 settings NOTE: 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) | |

# Laser Driver Controller

|  |  |
| --- | --- |
| [16] CONTROL? | Returns the enumerated operating mode for Channel. Where: Channel 0 (Temperature Channel) 0 = Constant Current Mode OFF 1 = Temperature Control Mode OFF  2 = Constant Current Mode ON 3 = Temperature Control Mode ON  Channel 1 (Current Control Channel) 128 = Constant Current Mode OFF 129 = Constant Power Servo Mode OFF  130 = Constant Current Mode ON 131 = Constant Power Servo Mode ON **NOTE: Channel 1 (Current Control 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 OFF 1 = Temperature Control Mode OFF  2 = Constant Current Mode ON 3 = Temperature Control Mode ON  Channel 1 (Current Control Channel) 0 = Constant Current Mode OFF 1 = Constant Power Servo Mode OFF  2 = Constant Current Mode ON 3 = Constant Power Servo Mode ON Returns CONTROL? |
| *Parameters*:  [uint8] CHANNEL  [uint8] MODE | *Example:*  CONTROL 1 2  130 |
| [18] ERROR? | Reads the Error codes for Channel. Error Codes are bit positions in a 16 bit unsigned integer value (little endian format) The most significant 2 bits of the return value are 1s. Error values are coded as follows: ERROR\_NONE = 0xC000  ERROR\_OPEN\_CIRCUIT = 0xC001  ERROR\_OVER\_TEMP\_HARDWARE = 0xC020  ERROR\_OVER\_TEMP\_AMBIENT = 0xC040  ERROR\_INTERLOCK = 0xC080  ERROR\_POWER\_LIMIT\_EXCEEDED = 0xC100  ERROR\_LASER\_TEMP\_BOUNDS\_EXCEEDED = 0xC400 |
| *Parameters*:  [uint8] Channel | *Example:*  Error? 1  49152 |
| [19] ERROR | Clears an Error code for Channel Note: use this command to clear an error code obtained from the ERROR? command.  Values to clear error codes:  256 Clears error code 49408 Returns the Error code resulting from clearing the error |
| *Parameters*:  [uint8] Channel  [uint16] Code | *Example:*  Error 1 256  49152 |

# Temperature Channel Specific (Channel 0)

|  |  |  |  |
| --- | --- | --- | --- |
| [28] TEMPSET? | | Returns the temperature setpoint for the temperature channel (channel 0) | |
| *Parameters*:  [uint8] CHANNEL  0 = Temp Chan  1 = INVALID | | *Example:*  Tempset? 0  25.0 | |
| [29] 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 | |
| [30] Bipolar? | | Returns whether the temperature loop for channel CHANNEL is bipolar.  Return is a STATUS Enumeration where: 4 = ON 5 = OFF | |
| *Parameters*:  [uint8] CHANNEL (MUST BE 0) | | *Example:*  Bipolar? 0  4 | |
| [31] 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 = true  0 = false | *Example:*  Bipolar 0 1  4 | | |

|  |  |  |  |
| --- | --- | --- | --- |
| [32] Temp? | Returns the current measured temperature for CHANNEL in degrees Celcius. | | |
| *Parameters*:  [uint8] CHANNEL  (Must be 0) | *Example:*  Temp? 0  24.21 | | |
| [33] TERROR? | | Returns the temperature error (Tsetpoint – Tactual) in millikelvin. | |
| *Parameters*:  [uint8] CHANNEL  (Must be 0) | | *Example:*  TError? 0  .0024 | |
| [34] TCURR? | | Returns the current flowing through TEC (or resistive heater) in Amps. | |
| *Parameters*:  [uint8] CHANNEL  (Must be 0) | | *Example:*  TCURR? 0  .654 | |
| [35] TEMPMIN? | | Returns the minimum temperature for CHANNEL in degrees Celcius. | |
| *Parameters*:  [uint8] CHANNEL  (Must be 0) | | *Example:*  TEMPMIN? 0  -5.000000 | |
| [36] 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 |
| [37] TEMPMAX? | | | Returns the maximum temperature for CHANNEL in degrees Celcius. |
| *Parameters*:  [uint8] CHANNEL (Must be 0) | | | *Example:*  TEMPMAX? 0  55.000000 |

|  |  |  |
| --- | --- | --- |
| [38] 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 | |
| [39] TC\_ILIM? | Returns the maximum current limit for Temperature channel in Amps.  (Applies only to temperature channel 0) | |
| *Parameters*:  [uint8] CHANNEL  (Must be 0) | *Example:*  TC\_ILIM? 0  2.000 | |
| [40] TC\_ILIM | Sets the current limit for Temperature channel in Amps. Returns the result of TC\_ILIM? in Amps (Applies only to temperature channel 0) | |
| *Parameters*:  [uint8] CHANNEL  (Must be 0)  [Float] Current | *Example:*  TC\_ILIM 0 3.5  3.5000 | |
| [41] PGAIN? | Returns the proportional gain for CHANNEL 0 (Temperature Channel) | |
| *Parameters*:  [uint8] CHANNEL  (Must be 0) | *Example:*  Pgain? 0  1.800 | |
| [42] 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 | |
| [43] INTEG? | Returns the integral time constant for CHANNEL 0 (Temperature Channel) | |
| *Parameters*:  [uint8] CHANNEL  (Must be 0) | *Example:*  Integ? 0  0.825 | |
| [44] 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 | |
| [45] DERIV? | Returns the derivative time constant for CHANNEL 0 (Temperature Channel) | |
| *Parameters*:  [uint8] CHANNEL  (Must be 0) | *Example:*  Deriv? 0  0.200 | |
| [46] 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 | |
| [47] 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 | |
| [48] 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 | |
| [49] PGAINEN? | Returns the enabled/disabled status of the proportional gain factor for CHANNEL 0 (Temperature Channel) Return is a STATUS Enumeration where: 4 = ON 5 = OFF | |
| *Parameters*:  [uint8] Channel  (Must be 0) | *Example:*  Pgainen? 0  4 | |
| [50] 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 | |
| [51] INTEGEN? | Returns the enabled/disabled status of the integral time constant factor for CHANNEL 0 (Temperature Channel) Return is a STATUS Enumeration where: 4 = ON 5 = OFF | |
| *Parameters*:  [uint8] Channel  (Must be 0) | *Example:*  Integen? 0  5 | |
| [52] 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 | |
| [53] DERIVEN? | Returns the enabled/disabled status of the derivative time constant factor for CHANNEL 0 (Temperature Channel) Return is a STATUS Enumeration where: 4 = ON 5 = OFF | |
| *Parameters*:  [uint8] Channel  (Must be 0) | *Example:*  Deriven? 0  5 | |
| [54] 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 | |
| [55] SLEWEN? | Returns the enabled/disabled status of the slew rate limiter for CHANNEL 0 (Temperature Channel) Return is a STATUS Enumeration where: 4 = ON 5 = OFF | |
| *Parameters*:  [uint8] Channel  (Must be 0) | *Example:*  Slewen? 0  5 | |
| [56] 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 | |
| [57] POWER? | Returns the power output for CHANNEL (Temperature Channel) in watts | |
| *Parameters*:  [uint8] Channel  (Must be 0) | *Example:*  Power? 0  8.3 | |
| [58] 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 | |
| [59] PERIOD | Sets the sample interval for CHANNEL 0 (Temperature Channel) in ms NOTE: The minimum sample period is 10 ms Returns PERIOD? | |
| *Parameters*:  [uint8] Channel  (Must be 0)  [uint16] Period | *Example:*  Period 0 10  10 | |
| [60] POLTC? | Returns the output polarity for CHANNEL 0 (Temperature Channel). On indicates factory default Negative polarity, Off indicates alternative Positive polarity Return is a STATUS Enumeration where: 4 = ON 5 = OFF | |
| *Parameters*:  [uint8] Channel  (Must be 0) | *Example:*  POLTC? 0  4 | |
| [61] POLTC | Sets the output polarity for CHANNEL 0 (Temperature Channel).  State = 1 sets factory default Negative polarity, state = 0 indicates alternative Positive polarity Returns POLTC? | |
| *Parameters*:  [uint8] Channel  (Must be 0)  [uint8] State  0 = Disabled  1 = Enabled | *Example:*  POLTC 0 0  5 | |
| [62] BETA? | Returns the Betatherm Beta thermistor coefficient for CHANNEL 0 (Temperature Channel) | |
| *Parameters*:  [uint8] Channel  (Must be 0) | *Example:*  BETA? 0  3450.00000 | |
| [63] 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 | |
| [64] REFTEMP? | Returns the Betatherm Reference temperature C for CHANNEL 0 (Temperature Channel) | |
| *Parameters*:  [uint8] Channel  (Must be 0) | *Example:*  REFTEMP? 0  25.0 | |
| [65] 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 | |
| [66] 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 | |
| [67] 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 | |
| [68] TCOEFA? | Returns the Steinhart Hart A coefficient for CHANNEL 0 (Temperature Channel) | |
| *Parameters*:  [uint8] Channel  (Must be 0) | *Example:*  TCOEFA? 0  2.108508173 | |
| [69] 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 | |
| [70] TCOEFB? | Returns the Steinhart Hart B coefficient for CHANNEL 0 (Temperature Channel) | |
| *Parameters*:  [uint8] CHANNEL (Must be 0) | *Example:*  TCOEFB? 0  0.797204727 | |
| [71] 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 | |
| [72] TCOEFC? | Returns the Steinhart Hart C coefficient for CHANNEL 0 (Temperature Channel) | |
| *Parameters*:  [uint8] CHANNEL | *Example:*  TCOEFC? 0  6.535076315 | |
| [73] 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 | |
| [74] MAXPWR? | Returns the power limit in Watts for CHANNEL 0 (Temperature Channel) | |
| *Parameters*:  [uint8] Channel  (Must be 0) | *Example:*  MaxPwr? 0  7.0 | |
| [75] 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 | |
| [76] CVOLTTC? | | Returns the absolute voltage [V] across the load for CHANNEL 0 (Tempearture Channel) |
| *Parameters*:  [uint8] CHANNEL | | *Example:*  Cvolttc? 0  2.438 |
| [77] TWARN? | Returns the CHANNEL 0 (Temperature Channel) temperature control warning range [mK] | |
| *Parameters*:  [uint8] Channel  (Must be 0) | *Example:*  Twarn? 0  1.000 | |
| [78] 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 | |
| [79] TCURSET? | Returns the CHANNEL 0 (Temperature Channel) manual mode current setting [A] | |
| *Parameters*:  [uint8] Channel  (Must be 0) | *Example:*  TCurset 0  0.300 | |
| [80] TCURSET | Sets the CHANNEL 0 (Temperature Channel) manual mode current setting [A] | |
| *Parameters*:  [uint8] Channel  (Must be 0)  [Float] Current | *Example:*  TCurset 0 0.200  0.200 | |
| [88] AVLPWR? | Returns the available power [W] for CHANNEL 0 (Temperature Channel) | |
| *Parameters*:  None | *Example:*  Avlpwr?  3.5 | |
| [89] TTLPWR? | Returns the total power [W] for CHANNEL 0 (Temperature Channel) | |
| *Parameters*:  None | *Example:*  TTLPWR?  5.0 | |
| [90] 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 | |
| [91] LDIMPD? | Returns the load impedance for CHANNEL 0 (Temperature Channel) | |
| *Parameters*:  [uint8] Channel  (Must be 0) | *Example:*  LDIMPD? 0  6.300 | |
| [92] 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 | |
| [93] 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 | |
| [94] LASTITC? | Returns the most recent output current for CHANNEL 0 (Temperature Channel) | |
| *Parameters*:  [uint8] Channel  (Must be 0) | *Example:*  LASTITC? 0  2.3000 | |
| [95] LASTVTC? | Returns the most recent output voltage for CHANNEL 0 (Temperature Channel) | |
| *Parameters*:  [uint8] Channel  (Must be 0) | *Example:*  LASTVTC? 0  3.800 | |
| [96] MLDCTHR | Sets the Mode Lock DC Threshold [V] for CHANNEL 0 (Temperature Channel) Valid values are 0 – 8 V Returns MLDCTHR? | |
| *Parameters*:  [Float] voltage | *Example:*  MLDCTHR 3.0  3.0 | |
| [97] MLDCTHR? | Returns the Mode Lock DC Threshold [V] for CHANNEL 0 (Temperature Channel) | |
| *Parameters*:  None | *Example:*  MLDCTHR?  3.0 | |
| [98] MLRMTHR | Sets the Mode Lock RMS Threshold for CHANNEL 0 (Temperature Channel) Valid values are >= 0.0 Returns MLRMTHR? | |
| *Parameters*:  [Float] RMS threhold | *Example:*  MLRMTHR 2.0  2.0 | |
| [99] MLRMTHR? | Returns the Mode Lock RMS Threshold for CHANNEL 0 (Temperature Channel) | |
| *Parameters*:  None | *Example:*  MLRMTHR?  2.0 | |
| [100] MLSMPLM | Sets the number of samples to use for mode lock calculation for Channel 0 (Temperature Channel) Maximum 250 Returns MLSMPLM? | |
| *Parameters*:  [uint8] samples | *Example:*  MLSMPLM 100  100 | |
| [101] MLSMPLM? | Returns the number of samples to use for mode lock calculation for Channel 0 (Temperature Channel) | |
| *Parameters*:  None | *Example:*  MLSMPLM?  100 | |
| [102] MODELK? | Returns the Mode Lock State for CHANNEL 0 (Temperature Channel) Where: 4 = ON 5 = OFF | |
| *Parameters*:  None | *Example:*  ModeLK?  4 | |
| [103] MLMEAN? | Returns the Mode Lock Mean for CHANNEL 0 (Temperature Channel) | |
| *Parameters*:  None | *Example:*  MLMEAN?  1.05 | |
| [104] MLVAR? | Returns the Mode Lock Variance for CHANNEL 0 (Temperature Channel) | |
| *Parameters*:  None | *Example:*  MLVAR?  0.0003 | |
| [105] MLSTDDV? | Returns the Mode Lock Standard Deviation for CHANNEL 0 (Temperature Channel) | |
| *Parameters*:  None | *Example:*  MLSTDDV?  0.233 | |

# Current Controller Commands (Channel 1)

|  |  |
| --- | --- |
| [106] CCURSET? | Returns Current Setpoint [Amps] for CHANNEL 1 (Current Control Channel) |
| *Parameters*:  [uint8] CHANNEL  (Must be 1) | *Example:*  CCURSET? 1  0.0205 |
| [107] CCURSET | Sets Current Setpoint [Amps] for CHANNEL 1 (Current Control Channel) Returns CCURSET? |
| *Parameters*:  [uint8] CHANNEL  (Must be 1)  [Float] current | *Example:*  CCURSET 1 0.180  0.180 |
| [108] CMAXCUR? | Reads the Current Limit [Amps] for CHANNEL 1 (Current Control Channel) |
| *Parameters*:  [uint8] Channel  (Must be 1) | *Example:*  CMAXCUR? 1  0.180 |
| [109] CMAXCUR | Sets the Current Limit [Amps] for CHANNEL 1 (Current Control Channel) Returns CMAXCUR? |
| *Parameters*:  [uint8] Channel  (Must be 1)  [Float] current | *Example:*  CMAXCUR 1 0.170  0.170 |
| [110] CCURR? | Returns Output Current [Amps] for CHANNEL 1 (Current Control Channel) |
| *Parameters*:  [uint8] CHANNEL  (Must be 1) | *Example:*  CCURR? 1  0.12544 |
| [111] PWRSET? | Returns Power Setpoint [mW] for CHANNEL 1 (Current Control Channel) |
| *Parameters*:  [uint8] Channel  (Must be 1) | *Example:*  PWRSET? 1  43.200 |
| [112] PWRSET | Sets Power Setpoint [mW] for CHANNEL 1 (Current Control Channel) Returns PWRSET? |
| *Parameters*:  [uint8] Channel  (Must be 1)  [Float] Power | *Example:*  PWRSET 1 50.0  50.00 |
| [113] MAXPWR? | Reads the Power Limit [mW] for CHANNEL 1 (Current Control Channel) |
| *Parameters*:  [uint8] Channel  (Must be 1) | *Example:*  MAXPWR? 1  180.00 |
| [114] MAXPWR | Sets the Power Limit [mW] for CHANNEL 1 (Current Control Channel) Returns MAXPWR? |
| *Parameters*:  [uint8] Channel  (Must be 1)  [Float] Power | *Example:*  MAXPWR 1 170.0  170.000 |
| [115] POWER? | Returns Output Power [mW] for CHANNEL 1 (Current Control Channel) |
| *Parameters*:  [uint8] CHANNEL  (Must be 1) | *Example:*  POWER? 1  125.44 |
| [116] CVOLTCC? | Returns Compliance Voltage [V] for CHANNEL 1 (Current Control Channel) |
| *Parameters*:  [uint8] Channel  (Must be 1) | *Example:*  CVOLTCC? 1  11.200 |
| [117] GAIN? | Returns the Constant Power Mode Gain [dB] for CHANNEL 1 (Current Control Channel) |
| *Parameters*:  [uint8] Channel  (Must be 1) | *Example:*  GAIN? 1  -3.2 |
| [118] GAIN | Sets the Constant Power Mode Gain [dB] for CHANNEL 1 (Current Control Channel) Returns GAIN? |
| *Parameters*:  [uint8] Channel  (Must be 1)  [float] Gain | *Example:*  GAIN? 1 5.2  5.2 |
| [119] RESPVTY? | Returns the Detector Responsivity for the Constant Power Mode [A/W] for CHANNEL 1 (Current Control Channel) |
| *Parameters*:  [uint8] Channel  (Must be 1) | *Example:*  RESPVTY? 1  2.2 |
| [120] RESPVTY | Sets the Detector Responsivity for the Constant Power Mode [A/W] for CHANNEL 1 (Current Control Channel) Returns RESPVTY? |
| *Parameters*:  [uint8] Channel  (Must be 1)  [float] Responsivity | *Example:*  RESPVTY 1 3.6  3.6 |
| [121] POLCC? | Returns the Polarity of the Constant Power Mode for CHANNEL 1 (Current Control Channel)  Returns 4 = On  Returns 5 = Off |
| *Parameters*:  [uint8] Channel  (Must be 1) | *Example:*  POLCC? 1  4 |
| [122] POLCC | Sets the Polarity of the Constant Power Mode for CHANNEL 1 (Current Control Channel)  Returns POLCC? |
| *Parameters*:  [uint8] Channel  (Must be 1)  [uint8] state 0 or 1 | *Example:*  POLCC 1 0  5 |
| [123] VTOP? | Returns the Top Voltage [V] for CHANNEL 1 (Current Control Channel) |
| *Parameters*:  [uint8] Channel  (Must be 1) | *Example:*  VTOP? 1   2.256 |
| [124] VTOP | Sets the Top Voltage [V] for CHANNEL 1 (Current Control Channel) Returns VTOP? |
| *Parameters*:  [uint8] Channel  (Must be 1)  [float] Voltage | *Example:*  VTOP 1 3.6  3.6 |
| [125] INTERLK? | Returns the status of the Interlock Circuit for CHANNEL 1 (Current Control Channel) Returns 4 = Interlock circuit is closed  Returns 5 = Interlock circuit is open |
| *Parameters*:  [uint8] Channel  (Must be 1) | *Example:*  INTERLK? 1   4 |
| [126] ATEMP? | Returns the Ambient Temperature [C] inside the enclosure for CHANNEL 1 (Current Control Channel |
| *Parameters*:  [uint8] Channel  (Must be 1) | *Example:*  ATEMP? 1   38.65 |
| [127] HWTEMP? | Returns the Hardware Temperature for CHANNEL (Current Control Channel) circuits |
| *Parameters*:  [uint8] Channel  (Must be 1) | *Example:*  HWTEMP? 1   46.2 |
| [128] CURROFST | Sets the Current Calibration offset [Amps] for CHANNEL (Current Control Channel)  Returns the Current Argument |
| *Parameters*:  [uint8] Channel  (Must be 1)  [float] offset | *Example:*  CURROFST 1 0.025  0.025 |
| [133] MODCURR? | Reads the Modulation Current [mA] for CHANNEL (Current Control Channel) |
| *Parameters*:  [uint8] Channel  (Must be 1) | *Example:*  MODCURR? 1   2.653 |
| [135] \_LTMAX | Sets the Maximum Operating Temperature [C] for the Laser Driver Current. Laser current shuts down if this boundary is exceeded. Temperature control will remain active if this limit is exceeded subject to the limit set using the TEMPMAX command. Returns \_LTMAX? |
| *Parameters*:  [uint8] Channel  (any integer) [float]temperature | *Example:*  \_LTMAX 1 50.0  50.0 |
| [136] \_LTMAX? | Reads the Maximum Temperature [C] at which the Laser Driver Current remains active for CHANNEL (Current Control Channel) |
| *Parameters*:  [uint8] Channel  (any integer) | *Example:*  \_LTMAX? 1   45.0 |
| [137] \_LTMIN | Sets the Minimum Operating Temperature [C] for the Laser Driver Current. Laser current shuts down if this boundary is exceeded. Temperature control will remain active if this limit is exceeded subject to the limit set using the TEMPMIN command. Returns \_LTMIN? |
| *Parameters*:  [uint8] Channel  (any integer) [float]temperature | *Example:*  \_LTMIN 1 -5.0  -5.0 |
| [138] \_LTMIN? | Reads the Minimum Temperature [C] at which the Laser Driver Current remains active for CHANNEL (Current Control Channel) |
| *Parameters*:  [uint8] Channel  (any integer) | *Example:*  \_LTMIN? 1   -5.0 |