

ILT1000 API

International Light technologies has created a simple text-based API for communication between the ILT1000 and user applications, or a basic terminal server. The API is documented below. Please Note; you can also obtain a sample of the more commonly used API commands by type help in the CLI menu.

While we have provided you with the full version of the API, most customers will only use a small sub-set of the API's vast capabilities. In many cases, customers will need only to use our preconfigured Datalog, Trend and Bar programs and may not find a need to understand the programming capabilities of CLI and the API commands as shown below.

Documentation Convention

The ILT1000 API table below contains two columns. In the first column is the API/command name. In the second column is the API/command's definition. The definition contains several elements and convention as follows.

- Actual ILT1000 command and responses will use the `Courier New` font.
- Command parameters are identified in [brackets], within the "Syntax" description. The brackets themselves are not used in the command syntax, as is illustrated by the "Example command" listed for any command with parameters.
- The "Normal return value(s)" and "Error return value(s)" information is pertinent to the echooff mode of operation (see `echoon` and `echooff` commands in the table below). When using echoon mode, return values are self-explanatory.
- An N/A for an "Error return value(s)" indicates that errors are not returned and not expected for this command.
- "Example return" is listed, on some commands, to provide an example of the output. In this field, any text following a dash ("-"), as well as the dash itself, are not part of the actual output and only listed to help describe the output.
- All standard commands are listed alphabetically (followed by programming commands which have an * before the name.)
- The optional "Persist through power-cycle" field will indicate whether or not the configuration is stored in flash memory and "sticks" through power cycles.
- Note that all commands are case sensitive, and always all-lowercase.

Product Generations

The ILT1000 API supports First Generation (Gen1) and Second Generation (Gen2) models. The Gen2 models add additional automatic-gain-control stages, a 4-20 mA output and programmable feedback resistors. As a result some API's will have both Gen1 and Gen2 behavior, while others are specific to Gen2 devices. This is explained, as appropriate, in the API definitions below.

echooff	<p>Syntax: echooff</p> <p>What it does: This commands enters a mode where only values are sent back in response to a command. This is the default mode on power up of the device, and the mode used for programming by an application.</p> <p>Normal return value: 0 on success</p> <p>Error return value(s): N/A</p> <p>Persist through power-cycle: No</p>
echoon	<p>Syntax: echoon</p> <p>What is does: This command enters a verbose mode whereby contextual help is echoed back for each command completion. The mode is useful when interacting with the ILT1000 from a terminal server or from the ILT1000-CLI program.</p> <p>Normal return value: 0 on success</p> <p>Error return value(s): N/A</p> <p>Persist through power-cycle: No</p>
eraselogdata	<p>Syntax: eraselogdata</p> <p>What it does: This command erases any log data that has been stored in the ILT1000 flash memory. See also startlogdata and stoplogdata.</p> <p>Normal return value(s): 0 on success</p> <p>Error return value(s):</p> <ul style="list-style-type: none"> -500 if logging is currently active (must use stoplogdata first) -501 error erasing the data from flash <p>Persist through power-cycle: Yes</p>
get100perc	<p>Syntax: get100perc</p> <p>What it does: This command returns the sensor voltage at the time the 100% value was set with set100perc.</p> <p>Normal return value(s): Voltage for 100% transmission, in microvolts</p> <p>Error return value(s): -500 if the 100% value was never set</p>
getambienttemp (Gen2 only)	<p>Syntax: getambienttemp</p> <p>What it does: This command returns the ambient temperature as sensed by the device, times 100. This differs from gettemp which returns the internal temperature of the microcontroller.</p> <p>Normal return value(s): Temperature in degrees F (x100).</p> <p>Error return value(s): N/A</p>

getauxserialno	<p>Syntax: getauxserialno</p> <p>What it does: This command returns the auxiliary serial number of the ILT1000 device. The serial number is stored in one-time-programmable memory at the time of manufacture.</p> <p>Normal return value(s): Device serial number</p> <p>Error return value(s): -500 device serial number has not been set</p> <p>Example return: sn17839-0001</p>
getcurrent	<p>Syntax: getcurrent</p> <p>What it does: This command returns the sensor current in picoamps.</p> <p>Normal return value(s): Current in picoamps</p> <p>Error return value(s): -500 if there is a voltage saturation, indicating the reading must be discarded</p>
getdarkmode	<p>Syntax: getdarkmode</p> <p>What it does: This command returns the dark mode currently in use by the device. The dark mode can be NO DARK (there is no consideration for photodiode dark current), FACTORY DARK (the dark current set at the factory), or USER DARK (the dark current set with setuserdark).</p> <p>Return value(s):</p> <ul style="list-style-type: none"> 0 = NO DARK 1 = FACTORY SET 2 = USER SET <p>Error return value(s): N/A</p>
getdatetime (Gen2 only)	<p>Syntax: getdatetime</p> <p>What it does: This command returns the real time clock's date and time in formats as shown below. This format includes a typical date and time format (mm/dd/yyyy hh:mm:ss), followed by the Epoch time (seconds since 1970). Note that the device is programmed to UTC/GMT time and, as a result, might return a time that does not match the local timezone. Modern computer programming languages will correctly convert the Epoch time data to local time when saving the timestamped log data. This is the case with the Data Logger software.</p> <p>Return value(s): Real time clock time as shown below</p> <p>Error return value(s): N/A</p> <p>Example return: 12/05/2013 19:02:05 1386270125</p>
getfactorydark	<p>Syntax: getfactorydark</p> <p>What it does: This command returns the dark voltage at the various transimpedance amplifier gain stages, as set at the factory.</p> <p>Normal return value(s): The x1 dark voltage in microvolts, followed by a space, followed by the x101 dark voltage in microvolts</p> <p>Error return value(s): -500 if the factory dark voltage has not been set</p> <p>Example return:12756 9234</p>

getfwversion	<p>Syntax: <code>getfwversion</code></p> <p>What it does: This command returns the firmware version running on the ILT1000 device.</p> <p>Normal return value(s): Device firmware version</p> <p>Error return value(s): N/A</p> <p>Example return: <code>1.3.0.5</code></p>
getgeneration	<p>Syntax: <code>getgeneration</code></p> <p>What it does: This command returns the generation of the ILT1000 device.</p> <p>Normal return value(s):</p> <ul style="list-style-type: none"> -999 for first generation products that did not have this command defined at time of release 1 for first generation products 2 for second generation products (programmable Rf, realtime clock, ambient temp sensor) <p>Error return value(s): N/A</p> <p>Example return: <code>2</code></p>
getirradiance	<p>Syntax: <code>getirradiance</code></p> <p>What it does: This command returns the irradiance value in user-defined units. The ILT1000 determines irradiance based on the following priority list:</p> <ol style="list-style-type: none"> 1. If a calibration factor is in use (see <code>usecalfactor</code>), this current-to-irradiance multiplier value is used. This is the most common method to derive irradiance. 2. If no calibration factor is in use, or the calibration factor has its multiplier value set to 65535 (0xFFFF), the custom calibration table set with either <code>setsimpleirrcal</code> or multiple calls to <code>setirrdatapoint</code> is used. <p>Normal return value(s): Irradiance value in user-defined units, x 1000</p> <p>Error return value(s):</p> <ul style="list-style-type: none"> -500 if no irradiance calibration data has been set -501 if the detector current is out of range for the current-to-irradiance curve (case #2 above) -502 if there is a voltage saturation, indicating the reading must be discarded <p>Example return (for an irradiance value of 73.798): <code>73798</code></p>

getlogdata	<p>Syntax: getlogdata</p> <p>What it does: This command returns the log data stored in flash memory. This command can be run during an active logging session (and datalogging will continue) or after a session is stopped with stoplogdata. The command will first output three values (total number of values, an integer indicating what information was logged, the logging period in seconds), followed by lines of comma-delimited data with date-time-stamp (in seconds since 1970 or “Unix epoch time”) followed by all recorded values for that date-time-stamp.</p> <p>Normal return value(s): Total number of Date-Time-Stamp + Value pairs logged Recorded Value Indicator bitmask as follows: 1=Optical Density (x100) 2=Percent Transmission (x10) 4=Detector Current (picoamps) 8=Detector Voltage (microvolts) 16=ILT1000 temperature (degrees F) 32=Calibrated Irradiance (see getirradiance) Logging Period (in Seconds) Seconds Since 1970, value #1, value #2, value #n Seconds Since 1970, value #1, value #2, value #n Seconds Since 1970, value #1, value #2, value #n</p> <p>Error return value(s): -500 if no log data present</p> <p>Notes on returned data values:</p> <ul style="list-style-type: none"> • If the 100% is not set, both Optical Density and Percent Transmission will return 0. • Log data does not support negative Optical Density values • Calibrated Irradiance will return 0 if there is no calibration data <p>Example return (Notes after the ‘-’ are for documentation purposes and not returned):</p> <pre> 5 – total time-stamp + value pairs 4 – detector current, in picoamps, is what was logged 60 – period in seconds (sample every minute) 1378738200, 159564 – 09 Sep 2013 14:50:00 GMT, 159.564 nanoamps 1378738260, 134657 – 09 Sep 2013 14:51:00 GMT, 134.657 nanoamps 1378738320, 145671 – 09 Sep 2013 14:52:00 GMT, 145.671 nanoamps 1378738380, 174801 – 09 Sep 2013 14:53:00 GMT, 174.801 nanoamps 1378738440, 163714 – 09 Sep 2013 14:54:00 GMT, 163.714 nanoamps </pre>
getmodelname	<p>Syntax: getmodelname</p> <p>What it does: This command returns the model name of the ILT1000 device.</p> <p>Normal return value(s): Device model name</p> <p>Error return value(s): N/A</p> <p>Example return: ILT1000</p>

getod	<p>Syntax: <code>getod</code></p> <p>What it does: This command returns the optical density, multiplied by 100. Because optical density is a relative measurement, this command requires that the 100% (0.00 OD) setting is established with <code>set100perc</code>.</p> <p>Normal return value(s): Optical density x 100</p> <p>Error return value(s): -500 if the 100% value has not been previously set with <code>set100perc</code></p> <p>Example return (for an optical density of 1.07): 107</p>
getserialnumber	<p>Syntax: <code>getserialnumber</code></p> <p>What it does: This command returns the serial number of the ILT1000 device. The serial number is stored in one-time-programmable memory at the time of manufacture.</p> <p>Normal return value(s): Device serial number</p> <p>Error return value(s): -500 device serial number has not been set</p> <p>Example return: ILT1000#12345</p>
gettemp	<p>Syntax: <code>gettemp</code></p> <p>What it does: This command returns the internal temperature of the ILT1000 microcontroller. Note that, while the detector temperature can be loosely inferred from this, this is not equivalent to the detector temperature. The return value is in degrees F, with any units conversion performed by the application software.</p> <p>Normal return value(s): ILT1000 microcontroller temperature, in degrees F</p> <p>Error return value(s): N/A</p> <p>Example return: 107</p>
gettrans	<p>Syntax: <code>gettrans</code></p> <p>What it does: This command returns the percent transmission, multiplied by 10. Because percent transmission is a relative measurement, this command requires that the 100% (0.00 OD) setting is established with <code>set100perc</code>.</p> <p>Normal return value(s): Percent Transmission x 10</p> <p>Error return value(s): -500 if the 100% value has not been previously set with <code>set100perc</code></p> <p>Example return (for a percent transmission of 67.3): 673</p>
getuserdark	<p>Syntax: <code>getuserdark</code></p> <p>What it does: This command returns the dark voltage at the various transimpedance amplifier gain stages, as set by the user with <code>setuserdark</code>.</p> <p>Normal return value(s): The x1 dark voltage in microvolts, followed by a space, followed by the x101 dark voltage in microvolts</p> <p>Error return value(s): -500 if the factory dark voltage has not been set</p> <p>Example return: 13014 9832</p>

getvoltage	<p>Syntax: getvoltage</p> <p>What it does: This command returns the voltage output of the transimpedance amplifier, after it is passed through the automatic-gain-control circuit.</p> <p>Normal return value(s): Detector voltage, in microvolts</p> <p>Error return value(s): N/A</p> <p>Example return (for 2.415896 volts): 2415896</p>
set100perc	<p>Syntax: set100perc</p> <p>What it does: This command reads the detectors voltage level (and by inference its current and irradiance level) and sets this as the 100% value to use in Optical Density and %Transmission calculations.</p> <p>Normal return value(s): The 100% voltage value, in microvolts</p> <p>Error return value(s): 1 if the value is too low to use as a 100% reference voltage, currently set at < 0.020 V 2 if the value is too high to use as a 100% reference voltage, currently set at > 3.200 V</p> <p>Persist through power-cycle: No</p> <p>Example return (1.421... volt 100% or “full scale”): 1421045</p>
setautaveraging	<p>Syntax: setautaveraging</p> <p>What it does: This command sets the amount of averaging performed by the ILT1000 to be set automatically based on the transimpedance amplifier voltage as follows:</p> <p style="padding-left: 40px;">For firmware levels < 2.0.0.0 Voltage > 30mv, averaging equivalent of setlowaveraging 3mv < Voltage < 30mv, averaging equivalent to setmedaveraging Voltage < 3mv, averaging equivalent to sethiaveraging</p> <p style="padding-left: 40px;">For firmware levels > 2.0.0.0 Current > 10uA, averaging equivalent of setlowaveraging 0.32uA < Current < 10uA, averaging equivalent to setmedaveraging Current < 0.32uA, averaging equivalent to sethiaveraging</p> <p style="padding-left: 40px;">This is the default setting for averaging upon power-up.</p> <p>Normal return value(s): 0 on success</p> <p>Error return value(s): N/A</p> <p>Persist through power-cycle: No</p>

<pre>setcurrentloop (Gen2 only, firmware 2.0.0.5 and later)</pre>	<p>Syntax: <code>setcurrentloop log</code> <code>setcurrentloop midpoint</code> <code>setcurrentloop [min picoamps] [max picoamps]</code> <code>setcurrentloop [0-24]</code></p> <p>What it does: This command controls the 4-20mA current loop output of devices that support such a current loop. NOTE: for the current loop to behave properly, the current loop needs to be powered before the ILT1000.</p> <p><code>setcurrentloop log</code> sets the device to output a logarithmic scale current that is in relation to the current sensed by the detector. The transfer function is:</p> $4\text{-}20\text{mA current} = (\text{LOG}_{10}(\text{detector current}) + 8) * 3 + 5$ $\text{Detector Current} = 10^{((4\text{-}20\text{mA Current} - 5) / 3 - 8)}$ <p>If the detector current is below 10 nA, the 4-20mA current is set to 4mA. As a result, measurements fluctuating around 10nA will result in the 4-20mA output bouncing between 4mA and 5mA. If a calibration factor is set, the 4-20mA output will have a similar relationship to irradiance, but multiplied by the calibration factor.</p> <p><code>setcurrentloop midpoint</code> sets the device to output a linear scale where:</p> <p>4mA = 0 detector current 12mA = the detector current when this command was set 20mA = double the detector current when this command was set</p> <p><code>setcurrentloop [min picoamps] [max picoamps]</code> sets the device to output a linear scale where:</p> <p>4mA = min picoamps 20mA = max picoamps</p> <p><code>setcurrentloop [0-24]</code> sets the device to manually output the current indicated, in milliamps.</p> <p>Normal return value(s): 0 on success</p> <p>Error return value(s):</p> <ul style="list-style-type: none"> -500 if command not supported, i.e. on Gen1 devices -501 if missing fields -502 if there is a bad current loop value (applies to <code>setcurrentloop [0-24]</code>) <p>Persist through power-cycle: Yes for <code>setcurrentloop log</code>, <code>setcurrentloop midpoint</code>, and <code>setcurrentloop [min picoamps] [max picoamps]</code>.</p> <p>No for <code>setcurrentloop [0-24]</code>, which is intended as more of a test function.</p>
---	--

setdatetime (Gen2 only)	<p>Syntax: setdatetime 12/05/2013 19:02:05</p> <p>What it does: This command sets the real time clock's date and time. It accepts either of the following formats:</p> <p style="text-align: center;">mm/dd/yyyy hh:mm:ss mm/dd/yy hh:mm:ss</p> <p>The device is designed such that this date/time setting is UTC/GMT time. The device stores all date/timestamps in Epoch time format, which is later read out and converted to local time by an application. This is the case with the Data Logger software.</p> <p>Normal return value(s): 0 on success</p> <p>Error return value(s): -500 if missing fields -501 if parameters are out of range</p> <p>Persist through power-cycle: Yes, assuming coin/cell battery is in place with adequate charge</p>
sethiaveraging	<p>Syntax: sethiaveraging</p> <p>What it does: This command sets the averaging to be performed by the ILT1000 to high, resulting in approximately 1 sample every 2 seconds. This mode provides the highest amount of over-sampling, or noise reduction. This mode is typically used for either detecting very low light levels or for monitoring noisy light sources.</p> <p>Normal return value(s): 0 on success</p> <p>Error return value(s): N/A</p> <p>Persist through power-cycle: No</p>
setlowaveraging	<p>Syntax: setlowaveraging</p> <p>What it does: This command sets the averaging to be performed by the ILT1000 to low, resulting in approximately 5 samples per second.</p> <p>Normal return value(s): 0 on success</p> <p>Error return value(s): N/A</p> <p>Persist through power-cycle: No</p>
setmedaveraging	<p>Syntax: setmedaveraging</p> <p>What it does: This command sets the averaging to be performed by the ILT1000 to medium, resulting in approximately 2 samples per second.</p> <p>Normal return value(s): 0 on success</p> <p>Error return value(s): N/A</p> <p>Persist through power-cycle: No</p>

<code>setsimpleirrcal</code>	<p>Syntax: <code>setsimpleirrcal</code> [current-to-irradiance multiplier, x 1,000,000]</p> <p>What it does: This command sets a simple multiplier to be used to convert detector current to irradiance. In the absence of factory-defined calibration factors, it is the easiest method to establish a user-defined irradiance value from detector current. The command automatically populates a straight-line (constant) current-to-irradiance relationship using the more advanced <code>setirrrdatapoint</code> command. As a result, the <code>storeirrrdata</code> command is required to persist this between power cycles.</p> <p>Normal return value(s): 0 on success</p> <p>Error return value(s): -500 if missing fields</p> <p>Persist through power-cycle: No. See <code>storeirrrdata</code> to persist between power cycles.</p> <p>Example command (to set a factor of 0.250 to apply to current to derive irradiance): <code>setsimpleirrcal 250000</code></p>
<code>setuserdark</code>	<p>Syntax: <code>setuserdark</code></p> <p>What it does: This command captures the detector voltage signal, at all automatic-gain-control stages, and stores the values in device flash memory. When used in conjunction with <code>useuserdark</code>, all subsequent readings will have this value removed automatically. Starting in firmware version 2.0.0.5, if the mode is already set to <code>useuserdark</code> then the new values will automatically be applied.</p> <p>Normal return value(s): User dark value, in microvolts, for all gain stages (separated by a space)</p> <p>Error return value(s): -500 if error setting dark value</p> <p>Persist through power-cycle: Yes</p> <p>Example return: 11289 10728</p>

startlogdata	<p>Syntax: startlogdata [variable bitmask] [logging period / 10] [seconds since 1970]</p> <p>What it does: This command defines the logging parameters, and immediately starts logging. This command cannot be run when either a logging session is already active or a session has been stopped (with stoplogdata), but the log data has not yet been erased with eraselogdata. The command takes three parameters, with a space between parameters, as follows:</p> <p>[variable bit mask]</p> <ul style="list-style-type: none"> 1=Optical Density (x100) 2=Percent Transmission (x10) 4=Detector Current (picoamps) 8=Detector Voltage (microvolts) 16=ILT1000 temperature (degrees F) 32=Calibrated Irradiance (see getirradiance) 128=Use Real Time Timestamps (as opposed to relative time stamps, Gen2 only) <p>[logging period / 10] – <i>for firmware versions 2.0.0.1 and earlier</i></p> <p>A 1, for example, would indicate a 10 second delay between logging; a value of 360 would indicate a 3600 second delay, or 1 hour, between log entries. The maximum value is 8640, which is equivalent to 86400 seconds or 1 day. Any values above this will results in a 1 day logging period.</p> <p>[logging period] – <i>for firmware versions 2.0.0.2 and later</i></p> <p>A 1, for example, would indicate a 1 second delay between logging; a value of 3600 would indicate a 3600 second delay, or 1 hour, between log entries. The maximum value is 86400, which is equivalent to 86400 seconds or 1 day. Any values above this will results in a 1 day logging period.</p> <p>[seconds since 1970]</p> <p>This is “Unix epoch time”, with most application coding environments having a mechanism to convert a date-time structure to epoch time. This is ignored, and should be set to 0, when using real-time timestamps.</p> <p>Normal return value(s): 0 on success</p> <p>Error return value(s):</p> <ul style="list-style-type: none"> -500 missing parameters -501 session already started. Must use stoplogdata and eraselogdata to start new. -502 errors with the variable bit mask <p>Persist through power-cycle: Yes.</p> <p>Common usage is to use startlogdata, disconnect the ILT1000 from the computer, connect the ILT1000 to a battery or AC-power in the lab or field, and the device will continue logging on power up using time-stamps that are relative to the start time. In this scenario care must be taken to ensure any power down time is negligible to the desired time-stamp accuracy. Gen2 devices have the option of using a battery-back real-time clock (see 128 bitmask value above).</p> <p>Example command (to log detector current and ILT1000 temperature, every minute (Firmware 2.0.0.2), starting at 09 Sep 2013 14:50:00 GMT. Note bitmask of 20 is 4/current + 16/temperature):</p> <pre>startlogdata 20 60 1378738200</pre> <p>Example command (same as above, but adding 128 to the bit mask for use of a real time clock on a Gen2 device):</p> <pre>startlogdata 148 6 0</pre>
--------------	--

stoplogdata	<p>Syntax: stoplogdata</p> <p>What it does: This command stops a log session that was started with startlogdata. Note that, because date-time-stamping is relative to the start time set with startlogdata, logging cannot be restarted after a stoplogdata. Instead, log data must first be erased using eraselogdata.</p> <p>Normal return value(s): 0 on success</p> <p>Error return value(s): -500 if no active logging session to stop</p>
usecalfactor	<p>Syntax: usecalfactor [calibration number, 0-20]</p> <p>What it does: This command selects a particular calibration factor (see setcalfactor), which in turn will define the detector current-to-irradiance multiplier as well as the detector saturation current. A calibration number of 0 is a special case. Using 0 results in no longer using any calibration factors. When this is done, getirradiance will only return a value if setsimpleirrca or setirrdatapoint has been used.</p> <p>Normal return value(s): 0 on success</p> <p>Error return value(s): -500 if missing fields -501 if factor number is outside the 0-20 range -502 if factor not defined</p> <p>Persist through power-cycle: Yes (firmware Version 2.0.0.0 and later), No otherwise</p> <p>Example command: usecalfactor 5</p>
usefactorydark	<p>Syntax: usefactorydark</p> <p>What it does: This command will cause all subsequent readings to remove the factory dark value before presenting any voltage, current, etc. readings. This is the default dark setting upon power up.</p> <p>Normal return value(s): 0 on success</p> <p>Error return value(s): -500 if there has been no factory dark value set</p> <p>Persist through power-cycle: No</p>
usefeedbackres (Gen2 only)	<p>Syntax: usefeedbackres [resistor selection 0-4]</p> <p>What it does: For Gen2 devices, which have 3 available feedback resistors, this command selects the resistor to be used as follows:</p> <ul style="list-style-type: none"> 0: have the device automatically select a resistor based on light-level 1: use feedback resistor #1, usually the lowest value resistor 2: use feedback resistor #2, usually the middle value resistor 3: use feedback resistor #3, usually the highest value resistor <p>Normal return value(s): 0 on success</p> <p>Error return value(s): -500 if missing parameters -501 if the device is not a Gen2 device -502 if the resistor selection value out of range -503 resistor selected, but error saving the change to flash memory</p> <p>Persist through power-cycle: Yes</p>

usenodark	<p>Syntax: usenodark</p> <p>What it does: This command will eliminate any dark current consideration.</p> <p>Normal return value(s): 0 on success</p> <p>Error return value(s): N/A</p> <p>Persist through power-cycle: No</p>
useuserdark	<p>Syntax: useuserdark</p> <p>What it does: This command will cause all subsequent readings to remove the user dark value (see <code>setuserdark</code>) before presenting any voltage, current, etc. readings.</p> <p>Normal return value(s): 0 on success</p> <p>Error return value(s): -500 if there has been no user dark value set</p> <p>Persist through power-cycle: No</p>

*erasecalfactor	<p>Syntax: <code>erasecalfactor</code> [calibration number, 1-20]</p> <p>What it does: This command erases the calibration data associated with the calibration factor. This includes the calibration factor description, the current-to-irradiance multiplier, and the saturation current.</p> <p>Normal return value(s): 0 on success</p> <p>Error return value(s):</p> <ul style="list-style-type: none"> -500 if missing fields -501 if bad factor number -502 if error erasing flash <p>Persist through power-cycle: Yes</p> <p>Example command: <code>erasecalfactor 5</code></p>
*getcalfactor	<p>Syntax: <code>getcalfactor</code> [optional, calibration factor, 1-20]</p> <p>What it does: Without the optional parameter, this command returns the calibration factor currently in use. With the optional parameter, this command returns the calibration factor details for a particular calibration factor. This includes the following information, separated by space: calibration factor description, current-to-irradiance multiplier(x1000), and saturation current in microamps. A typical use case is to use the command without the optional parameter to determine which calibration factor is in use, followed by issuing the command with the optional calibration factor to determine the details of the calibration factor definition.</p> <p>Normal return value(s): Without the optional command line parameter:</p> <ul style="list-style-type: none"> 0 if no calibration factor in use 1-20 indicating which calibration factor is in use <p>With the optional command line parameter: [calibration factor description] [current-to-irradiance multiplier(x1000)] [saturation current]</p> <p>Error return value(s):</p> <ul style="list-style-type: none"> -501 if calibration factor is out of the 1-20 range -502 if the calibration factor is not defined <p>Example command: <code>getcalfactor 1</code></p> <p>Example return: <code>calfact1 500 50</code></p>
*getclockfreq	<p>Syntax: <code>getclockfreq</code></p> <p>What it does: This command returns the clock frequency, in MHz, as defined by <code>setclockfreq</code>. If <code>setclockfreq</code> has not yet been called, the default clock frequency is returned. See <code>setclockfreq</code> for more detail.</p> <p>Normal return value(s): Clock frequency, in MHz, times 100</p> <p>Error return value(s): N/A</p> <p>Example return (for 80.41 MHz frequency): 8041</p>

*getfeedbackres	<p>Syntax: <code>getfeedbackres</code></p> <p>What it does: This command returns the value of the feedback resistor, in kilo-Ohms x 10.</p> <p>Normal return value(s): Transimpedance amplifier feedback resistance in kOhms x 10</p> <p>Error return value(s): N/A</p> <p>Example return (for a 3K Ohm feedback resistor): 30</p>
*setcalfactor	<p>Syntax: <code>setcalfactor</code> [calibration factor, 1-20] [desc.] [multiplier x 1000] [saturation current uA]</p> <p>What it does: This command defines the particular calibration factor details. Details include the calibration factor description (up to 100 characters), the current-to-irradiance multiplier(x1000), and saturation current in microamps.</p> <p style="padding-left: 40px;">If the multiplier value is 0xFFFF (65535) the calibration table as established by <code>setirrdatapoint</code> will be used to convert detector current to irradiance.</p> <p>Normal return value(s): 0 on success</p> <p>Error return value(s):</p> <ul style="list-style-type: none"> -500 if missing fields -501 if calibration factor is out of the 1-20 range -502 if multiplier < 0 -503 error saving to flash <p>Persist through power-cycle: Yes</p> <p>Example command ("calfact1" description, 0.500 current-to-irradiance factor, 50 uA saturation current): <code>setcalfactor 1 calfact1 500 50</code></p>
*setclockfreq	<p>Syntax: <code>setclockfreq</code></p> <p>What it does: This command starts a procedure that records an accurate measurement of the ILT1000 microcontroller clock frequency. While the clock frequency is nominally 80.00 Mhz, in reality it is slightly different from this value. Use <code>setclockfreq</code> to calibrate the time-stamping routines. This command cannot be run during an active logging session because it must take control of the timer interrupt, which is active during logging.</p> <p>Procedure: This is the only ILT1000 command that is more involved than a simple command/response sequence. If the device is being run in the <code>echoonmode</code>, from a terminal server or ILT1000-CLI, the device will prompt the user through the proper procedure. If the command is being driven by an application, in <code>echooff</code> mode, the procedure is as follows :</p> <ol style="list-style-type: none"> 1. send the <code>setclockfreq</code> command 2. send the single character "A" 3. wait exactly 60 seconds 4. send the single character "B" <p>Normal return value(s): 0 on success</p> <p>Error return value(s):</p> <ul style="list-style-type: none"> -500 if a logging session is active -501 if there is an error with the procedure -502 if the clock frequency measured is off by more than 1MHz from the 80MHz nominal <p>Persist through power-cycle: Yes</p>

<p>*setsamplecount (firmware 2.0.0.6 and later)</p>	<p>Syntax: <code>setsamplecount [1 – 200]</code></p> <p>What it does: This command sets the number of samples taken by the Analog-to-Digital converter when reading the voltage from the transimpedance amplifier. This is the immediate averaging done on the input signal, before additional averaging is done by the <code>set*averaging</code> commands. Typically a value of 200 is used (the default) for standard operation, or a value of 1 is used for high-speed sampling (up to 100 samples/sec). WARNING: modifying this value will impact the communication between the ILT1000 and any device talking to it over the USB port. Talk to the manufacturer for details.</p> <p>Normal return value(s): 0 on success</p> <p>Error return value(s):</p> <ul style="list-style-type: none"> -500 if missing fields -501 if value is out of range <p>Persist through power-cycle: Yes</p> <p>Example command (for sampling at 100/sec): <code>setsamplecount 1</code></p> <p>Example command (for default sampling rate): <code>setsamplecount 200</code></p>
---	---