LightEngineAPI

Lumencor Inc
Version 1.0.8
Mon Jan 7 2019

File Documentation

LightEngineAPI.h File Reference

Macros

- #define **LUM_API**
- #define **LUM_API_VERSION** "1.0.8"
- #define **LUM_PART_NO** "55-10185-Rev-A"
- #define **LUM_MAX_MESSAGE_LENGTH** 1024
- #define **LUM_DEFAULT_TCP_PORT** 8095
- #define **LUM_LEGACY_BAUD_RATE** 9600
- #define **LUM_STANDARD_BAUD_RATE** 115200
- #define **LUM_OK** 0
- #define **LUM_API_CALL_FAILED** 11001
- #define **LUM_ENGINE_NOT_CONNECTED** 11002
- #define **LUM_COMMAND_TIMED_OUT** 11003
- #define **LUM_CONNECTION_FAILURE** 11004
- #define **LUM_STRINGCOPY_ERROR** 11005
- #define **LUM_CALL_NOT_SUPPORTED** 11006
- #define **LUM_LEGACY_MODEL_ERR** 11007
- #define **LUM_INVALID_HANDLE** 11008
- #define **lum_true** 1
- #define **lum_false** 0

Typedefs

- typedef int **lum_bool**

Enumerations

- enum **LEGACY_MODEL** { **NONE** = 0, **AURA2**, **LIDA**, **MIRA**, **RETRA**, **SOLASE**, **SOLA**, **SPECTRA7**, **SPECTRAX**, **LUMA** }

Functions

DLL Initialization*Create and destroy light engine instances*

- LUM_API **lum_createLightEngine** (void **handle)
- LUM_API **lum_createLegacyLightEngine** (void **handle, LEGACY_MODEL legacyModel)
- LUM_API **lum_createLegacyLightEngineByName** (void **handle, const char *legacyName)
- LUM_API **lum_deleteLightEngine** (void *handle)
- LUM_API **lum_getAPIVersion** (char *versionTxt, int maxLength)

Error Handling*Obtain specific error information: code and description*

- LUM_API **lum_getLastErrorText** (void *handle, char *errMessage, int maxLength)
- LUM_API **lum_getLastErrorCode** (void *handle, int *code)
- LUM_API **lum_resetError** (void *handle)

Connection Management*Connect light engine instance to specific hardware unit. Only one unit can be connected to one API instance.*

- LUM_API **lum_connectCOM** (void *handle, const char *port, unsigned int baud)
- LUM_API **lum_connectTCP** (void *handle, const char *ip, unsigned short port)
- LUM_API **lum_disconnect** (void *handle)
- LUM_API **lum_shutDown** (void *handle)
- LUM_API **lum_restart** (void *handle)
- LUM_API **lum_getConnected** (void *handle, lum_bool *connected)

- LUM_API **lum_getLegacy** (void *handle, lum_bool *legacy)
- Information and StatusLUM_API **lum_getModel** (void *handle, char *modelTxt, int length)
- LUM_API **lum_getSerialNumber** (void *handle, char *serialNumberTxt, int length)
- LUM_API **lum_getVersion** (void *handle, char *versionTxt, int length)
- LUM_API **lum_getIP** (void *handle, char *ipTxt, int length)
- LUM_API **lum_setIP** (void *handle, const char *ipTxt)
- LUM_API **lum_getTemperature** (void *handle, double *tempC)
- LUM_API **lum_getStatusCode** (void *handle, int *statusCode)

Light Output Control*On/Off switching and intensity commands*

- LUM_API **lum_getMaximumIntensity** (void *handle, int *maxint)
- LUM_API **lum_getNumberOfChannels** (void *handle, int *numChannels)
- LUM_API **lum_getChannelName** (void *handle, int channelIndex, char *name, int length)
- LUM_API **lum_getChannelIndex** (void *handle, const char *name, int *channelIndex)
- LUM_API **lum_setChannel** (void *handle, int channelIndex, lum_bool state)
- LUM_API **lum_getChannel** (void *handle, int channelIndex, lum_bool *state)
- LUM_API **lum_setMultipleChannels** (void *handle, lum_bool *stateArray, int numChannels)
- LUM_API **lum_getMultipleChannels** (void *handle, lum_bool *stateArray, int numChannels)
- LUM_API **lum_setIntensity** (void *handle, int channelIndex, int intensity)
- LUM_API **lum_getIntensity** (void *handle, int channelIndex, int *intensity)
- LUM_API **lum_setMultipleIntensities** (void *handle, int *intensityArray, int numChannels)
- LUM_API **lum_getMultipleIntensities** (void *handle, int *intensityArray, int numChannels)

Power control*power measurements and control*

- LUM_API **lum_getPowerLock** (void *handle, lum_bool *enabled)
- LUM_API **lum_setPowerLock** (void *handle, lum_bool enable)
- LUM_API **lum_getChannelPowerCount** (void *handle, int channelIndex, int *power)
- LUM_API **lum_getChannelPowerMW** (void *handle, int channelIndex, double *power)
- LUM_API **lum_setPowerReference** (void *handle, int *referenceArray, int numChannels)
- LUM_API **lum_getPowerReference** (void *handle, int **referenceArray, int *numChannels)
- LUM_API **lum_getSupplyPowerW** (void *handle, double *power)
- TTL Input ControlLUM_API **lum_setTTLEnable** (void *handle, lum_bool state)
- LUM_API **lum_getTTLEnable** (void *handle, lum_bool *state)
- LUM_API **lum_setTTLPolarity** (void *handle, lum_bool positive)
- LUM_API **lum_getTTLPolarity** (void *handle, lum_bool *positive)

Raw Command Interface*Execute arbitrary light engine commands*

- LUM_API **lum_executeCommand** (void *handle, const char *command, char *response, int length)
- AutomationLUM_API **lum_executeScript** (void *handle, const char *script)

Function Documentation

LUM_API lum_connectCOM (void * *handle*, const char * *port*, unsigned int *baud*)

Connects to Light Engine hardware through serial port. Won't work if the baud rate does not match the setting on the hardware. See header file for default baud rate.

Only one hardware unit can be connected to one instance, regardless of the communication protocol. This command will implicitly disconnect any previously established link.

Returns:

LUM_API

Parameters:

handle	

port	- COM port name
baud	- baud rate

LUM_API lum_connectTCP (void *   *handle*, const char *   *ip*, unsigned short   *port*)

Connects to Light Engine hardware through a TCP port. Unless configured otherwise LE listens on port LUM_DEFAULT_TCP_PORT.

Only one hardware unit can be connected to one instance, regardless of the communication protocol. This command will implicitly disconnect any previously established link. With TCP communication multiple instances can be connected to the same LE unit (IP address).

Returns:

LUM_API

Parameters:

handle	
ip	- IP address of the LE unit
port	- TCP port

LUM_API lum_createLegacyLightEngine (void **   *handle*, LEGACY_MODEL *legacyModel*)

Creates Legacy light engine instance. Legacy instances use obsolete wire protocol supported by old models. Some API calls may fail because legacy interface does not support them.

Returns:

LUM_API

Parameters:

handle	- handle to newly created legacy light engine instance
legacyModel	- legacy model enum

LUM_API lum_createLightEngine (void **   *handle*)

Creates Light Engine instance. Multiple LE instances can be created and all have to be deleted before exiting the program.

Returns:

LUM_API

Parameters:

handle	- handle to newly created Light Engine instance

LUM_API lum_deleteLightEngine (void *   *handle*)

Destroys Light Engine instance. Failing to destroy light engine instance after it goes out of scope will create a memory leak.

Returns:

LUM_API

Parameters:

handle	

LUM_API lum_disconnect (void *   *handle*)

Disconnects from hardware, regardless of the connection type.

Returns:

LUM_API

Parameters:

handle	

LUM_API lum_executeCommand (void * *handle*, const char * *command*, char * *response*, int *length*)

Execute generic light engine command string.

Returns:

LUM_API

Parameters:

handle	
command	- a string containing command, see command reference for formatting
response	- raw response as returned from light engine, if no response, error is is returned. Response is truncated to length.
length	- length for the response buffer

LUM_API lum_executeScript (void * *handle*, const char * *script*)

Execute "Chai" script stored in a file. Errors and messages are printed on the standard output. All calls in LumencorAPI.h are accessible within the script. Function names are the same, except without "lum_" prefix.

For syntax and documentation see: `http://chaiscript.com/` This is experimental, unsupported feature.

Returns:

LUM_API

Parameters:

handle	- light engine handle
script	- script file name

LUM_API lum_getAPIVersion (char * *versionTxt*, int *maxLength*)

Retrieves version of this DLL. API instance is not required. Last error code is not set in case of error in this call.

Returns:

LUM_API - LUM_STRINGCOPY_ERROR if buffer too small, or any other issue with string handling

Parameters:

versionTxt	- version string
maxLength	- length of the text buffer

LUM_API lum_getChannel (void * *handle*, int *channelIndex*, lum_bool * *state*)

Retrieves channel state ON (true, or 1) or OFF (false, or 0).

Returns:

LUM_API

Parameters:

handle	
channelIndex	- channel index
state	- integer 0 or 1, use lum_bool macro

LUM_API lum_getChannelIndex (void * *handle*, const char * *name*, int * *channelIndex*)

Retrieves channel index for a given channel name. The call will fail if name does not exist on the connected light engine.

Returns:

LUM_API

Parameters:

handle	
name	- channel name to look up
channelIndex	- index matching given name

LUM_API lum_getChannelName (void * *handle*, int *channelIndex*, char * *name*, int *length*)

Retrieves channel name (usually a color) corresponding to a given index.

Returns:

LUM_API

Parameters:

handle	
channelIndex	
name	- channel name (color)
length	- maximum length for the name buffer

LUM_API lum_getChannelPowerCount (void * *handle*, int *channelIndex*, int * *power*)

Returns raw power for a given channel as measured by the internal spectrometer This value is direct, uncalibrated reading from the sensor.

Returns:

LUM_API

Parameters:

handle	
channelIndex	- index of the channel to measure power from
power	- sensor reading

LUM_API lum_getChannelPowerMW (void * *handle*, int *channelIndex*, double * *power*)

Returns output light power in mW.

Returns:

LUM_API

Parameters:

handle	
channelIndex	- light channel
power	- light power in mW

LUM_API lum_getConnected (void * *handle*, lum_bool * *connected*)

Finds out whether we are connected or not.

Returns:

LUM_API

Parameters:

handle	
connected	- true if connected

LUM_API lum_getIntensity (void * *handle*, int *channelIndex*, int * *intensity*)

Retrieves intensity for the specified channel.

Returns:

LUM_API

Parameters:

handle	
channelIndex	- channel index
intensity	- intensity setting

LUM_API lum_getIP (void * *handle*, char * *ipTxt*, int *length*)

Retrieves current IP address.

Returns:

LUM_API

Parameters:

handle	
ipTxt	- string containing the address
length	- max length of the ip buffer

LUM_API lum_getLastErrorCode (void * *handle*, int * *code*)

Retrieves the last error code. A valid API instance (handle) is required for retrieving error info.

Returns:

LUM_API

Parameters:

handle	
code	- error code

LUM_API lum_getLastErrorText (void * *handle*, char * *errMessage*, int *maxLength*)

Retrieves last error text message. A valid API instance (handle) is required for retrieving error info.

Returns:

LUM_API

Parameters:

handle	
errMessage	- error message, truncated to the buffer length
maxLength	- length of the buffer to receive the message

LUM_API lum_getMaximumIntensity (void * *handle*, int * *maxint*)

Retrieve maximum intensity setting for light channels.

Returns:

LUM_API

Parameters:

handle	
maxint	- maximum intensity, depends on the model, version and legacy mode

LUM_API lum_getModel (void * *handle*, char * *modelTxt*, int *length*)

Retrieves model of the Light Engine.

Returns:

LUM_API

Parameters:

handle	

modelTxt	- model, truncated to buffer length
length	- max buffer length

LUM_API lum_getMultipleChannels (void * *handle*, lum_bool * *stateArray*, int *numChannels*)

Retrieve on/off states for all available channels. Caller is expected to allocate array and manage array lifetime.

Returns:

LUM_API

Parameters:

handle	
stateArray	- pre-allocated array of state variables
numChannels	- actual number of channels, must be equal to the size of the array

LUM_API lum_getMultipleIntensities (void * *handle*, int * *intensityArray*, int *numChannels*)

Retrieve current intensities for all channels with a single command. Assuming that the caller allocates the array and manages the array lifetime.

Returns:

LUM_API

Parameters:

handle	
array	- pre-allocated array of integers representing intensities
numChannels	- size of the intensity array, must be equal to the number of channels

LUM_API lum_getNumberOfChannels (void * *handle*, int * *numChannels*)

Retrieves the number of available channels.

Returns:

LUM_API

Parameters:

handle	
numChannels	- number of channels

LUM_API lum_getPowerLock (void * *handle*, lum_bool * *enabled*)

Returns the state of the "power lock" feature, representing the PID power control/ If it is enabled it means that PID is active and attempting to keep power at the levels specified by the "reference" (see setPowerReference() command).

Returns:

LUM_API

Parameters:

handle	
enabled	- integer 0 or 1, use lum_bool macro

LUM_API lum_getPowerReference (void * *handle*, int ** *referenceArray*, int * *numChannels*)

Return power references for all channels, when PID control is engaged. References are raw sensor values. Negative value means that channel is not under PID control.

Returns:

LUM_API

Parameters:

handle	
referenceArray	- references for all channels (negative value means not controlled)
numChannels	- number of channels

LUM_API lum_getSerialNumber (void * *handle*, char * *serialNumberTxt*, int *length*)

Retrieves serial number.

Returns:

LUM_API

Parameters:

handle	
serialNumberTxt	- serial number, truncated to length
length	- length of the serial buffer

LUM_API lum_getStatusCode (void * *handle*, int * *statusCode*)

Retrieves the engine status code. See Command Reference for information on specific codes.

Returns:

LUM_API

Parameters:

handle	
statusCode	- status code

LUM_API lum_getSupplyPowerW (void * *handle*, double * *power*)

Return power in Watts, drawn by the light engine power supply.

Returns:

LUM_API

Parameters:

handle	
power	- power draw by the supply in Watts

LUM_API lum_getTemperature (void * *handle*, double * *tempC*)

Retrieves light engine temperature in degrees Celsius.

Returns:

LUM_API

Parameters:

handle	
tempC	

LUM_API lum_getTTLEnable (void * *handle*, lum_bool * *state*)

Get TTL enable state: enabled or disabled.

Returns:

LUM_API

Parameters:

handle	
state	- true if enabled

LUM_API lum_getTTLPolarity (void * *handle*, lum_bool * *positive*)

Returns current TTL polarity setting.

Returns:

LUM_API

Parameters:

handle	
positive	- true if polarity is positive

LUM_API lum_getVersion (void * *handle*, char * *versionTxt*, int *length*)

Retrieves version of the LE firmware.

Returns:

LUM_API

Parameters:

handle	
versionTxt	- version, truncated to length
length	- text buffer length

LUM_API lum_resetError (void * *handle*)

Clears last error by setting "OK" state.

Returns:

LUM_API

Parameters:

handle	

LUM_API lum_restart (void * *handle*)

Restarts light engine. During restart procedure (20 seconds or less) Light Engine will be unavailable. After light engine restars, the existing connection will become invalid.

Returns:

LUM_API

Parameters:

handle	

LUM_API lum_setChannel (void * *handle*, int *channelIndex*, lum_bool *state*)

Turns channel ON (true or 1) or OFF (false or 0).

Returns:

LUM_API

Parameters:

handle	
channelIndex	- channel index
state	- integer 1 for ON, 0 for OFF, use lum_bool

LUM_API lum_setIntensity (void * *handle*, int *channelIndex*, int *intensity*)

Sets light intensity for the specified channel.

Returns:

LUM_API

Parameters:

handle	
channelIndex	- channel index
intensity	- intensity

LUM_API lum_setIP (void * *handle*, const char * *ipTxt*)

Sets IP address for the light engine. This command will cause automatic reboot and the light engine will be temporarily unavailable. Disconnect and re-connect is probably needed afterwards, if COM port is used.

Returns:

LUM_API

Parameters:

handle	
ipTxt	- IP address formatted as string, e.g. "102.168.3.9"

LUM_API lum_setMultipleChannels (void * *handle*, lum_bool * *stateArray*, int *numChannels*)

Sets state of multiple channels in a single command.

Returns:

LUM_API

Parameters:

handle	
stateArray	- array of desired channel states
numChannels	- number of elements in the array

LUM_API lum_setMultipleIntensities (void * *handle*, int * *array*, int *numChannels*)

Set multiple channel intensities with a single command.

Returns:

LUM_API

Parameters:

handle	
array	- array of integers representing intensities
numChannels	- number of elements in the array

LUM_API lum_setPowerLock (void * *handle*, lum_bool *enable*)

Activates or de-activates the PID control feature. If "locked" PID will attempt to keep power equal to the reference established by the setPowerReference command.

Returns:

LUM_API

Parameters:

handle	
enable	- integer 0 or 1, use lum_bool macro

LUM_API lum_setPowerReference (void * *handle*, int * *referenceArray*, int *numChannels*)

Sets power references for PID control for all channels. Values are target raw sensor readings. Negative value means that channel is not under PID control.

Returns:

LUM_API

Parameters:

handle	
referenceArray	- array of channel power references (negative means "no reference")
numChannels	- number of channels

LUM_API lum_setTTLEnable (void * *handle*, lum_bool *state*)

Enable or disable TTL inputs on the light engine. On power up inputs are disabled and must be enabled explicitly before use.

Returns:

LUM_API

Parameters:

handle	
state	- true enables, false disables TTL

LUM_API lum_setTTLPolarity (void * *handle*, lum_bool *positive*)

Determines TTL input polarity, whether TTL high means ON (positive logic), or TTL high means OFF (negative logic).

Returns:

LUM_API

Parameters:

handle	
positive	- true for positive convention, false for negative

LUM_API lum_shutDown (void * *handle*)

Shuts down the Light Engine hardware. After this command the unit powers down and becomes unavailable. Can be restarted only manually with a power switch.

Returns:

LUM_API

Parameters:

handle	