

# LumaUSB DLL X/Y/Z Stage Control API

## Introduction

This help file describes the API (application programming interface) for the 'LumaUSB.DLL' for interfacing with the Lumascope x/y/z stage. In order to interact with the stage, the Trinamic virtual COM USB driver needs to be installed. Currently (May 2015), this driver is unsigned and thus for Windows 8.0 and 8.1 you will need to follow a procedure for installing unsigned drivers. The LumaUSB DLL is written with C# but the user can interface to the DLL with other language.

## Example Application, 'StageController.exe', to Help Get Started

The C# project provides a good example how to interface to LumaUSB.DLL. You can examine the code to see how the application interacts with the DLL. A compiled version of 'StageController.exe' is included in the development kit, which you can immediately run and see work. Ensure that 'StageController.exe' and 'LumaUSB.dll' are in the same folder before running 'StageController.exe'. We recommend making sure that you can run 'StageController.exe' successfully before attempting your application development. Also, 'StageController.exe' provides a sanity-check when developing, showing that interface to the x/y/z stage is working.

## Initialization of the 'StageController' Object

The C# project provides a good example how to interface to the stage control of LumaUSB.DLL. You can examine the code to see how the application interacts with the DLL. All the functionality that you will need to interact with the LumaScope is contained in the 'StageController' class in the DLL. In your application, you will need to create an instance of it. Here is how you do it in C#:

```
EtalumaStage.StageController stageController = new EtalumaStage.StageController();
```

## Note on Initialization

## API Listing

Below follows documented API calls. All the calls are members of the 'StageController' class. Note that your application code must use the namespace, "using EtalumaStage".

### EtalumaStage.StageController.Z\_MAX\_TRAVEL\_MILLIMETERS

---- This is the maximum that the Z-axis is able to travel to get to the limits ----- Z cannot be less than zero. This is the maximum "up" position. Derived empirically for the January 2014 stage version.

### EtalumaStage.StageController.#ctor

CTOR.

### EtalumaStage.StageController.abortStageMovement

Calling this stops all axes moving and initialization (if called during initialization).

### EtalumaStage.StageController.OneStepInitialization(System.String@)

The client may call this function to start the initialization of the stage. The client must poll 'OneStepInitializationComplete()' to determine when the initialization is complete. If the client subsequently calls this function after the stage as been completely initialized, the stage will not reinitialize.

**Return:** True if successfully started.

### **EtalumaStage.StageController.OneStepInitializationComplete**

Poll this function to determine when the initialization is complete.

**Return:** True if initialization is complete.

### **EtalumaStage.StageController.OpenCommLink(System.String@)**

Opens the serial-over-USB communication link to the Trinamic board.

**Return:** True if this function was successfully able to open the COM Port.

### **EtalumaStage.StageController.IsCommLinkOpen**

Indicates if the serial-over-USB link is open.

**Return:** True if the serial-over-USB link is open.

### **EtalumaStage.StageController.CloseCommLink**

Closes the serial-over-USB link.

### **EtalumaStage.StageController.InitializeXAxisToReferencePoint**

Moves the x-axis to the electro-mechanical reference points of the stage. The electro-mechanical reference points are magnetic sensors where the Trinamic board considers the zero reference. Keep in mind that the "home" point from the user perspective is the upper-left corner of a microplate, which is different.

### **EtalumaStage.StageController.StopXAxisReferenceSearch**

Call this to stop the x-axis is moving during the movement to the reference position. Use only during reference searching!

**Note:** A reference search is used when initializing the stage so it can find the "zero" position. On the physical stage, ther reference points are usually magnetic sensors.

### **EtalumaStage.StageController.xAxisReferenceSearchMoving**

Call this to see if the x-axis is moving during a reference search. Use only during reference searching!

**Return:** True if the x-axis is moving.

**Note:** A reference search is used when initializing the stage so it can find the "zero" position. On the physical stage, ther reference points are usually magnetic sensors.

### **EtalumaStage.StageController.InitializeYAxisToReferencePoint**

Moves the y-axis to the electro-mechanical reference points of the stage. The electro-mechanical reference points are magnetic sensors where the Trinamic board considers the zero reference. Keep in mind that the "home" point from the user perspective is the upper-left corner of a microplate, which is different.

### **EtalumaStage.StageController.StopYAxisReferenceSearch**

Call this to stop the y-axis is moving during the movement to the reference position. Use only during reference searching!

**Note: A reference search is used when initializing the stage so it can find the "zero" position. On the physical stage, ther reference points are usually magnetic sensors.**

### **EtalumaStage.StageController.yAxisReferenceSearchMoving**

Call this to see if the y-axis is moving during a reference search. Use only during reference searching!

**Return:** True if the y-axis is moving.

**Note: A reference search is used when initializing the stage so it can find the "zero" position. On the physical stage, ther reference points are usually magnetic sensors.**

### **EtalumaStage.StageController.InitializeZAxisToReferencePoint**

Moves the z-axis to the electro-mechanical reference points of the stage. The electro-mechanical reference points are magnetic sensors where the Trinamic board considers the zero reference. Keep in mind that the zero-zero point from the user perspective is the upper-left corner of a microplate, which is different.

### **EtalumaStage.StageController.StopZAxisReferenceSearch**

Call this to stop the z-axis is moving during the movement to the reference position. Use only during reference searching!

**Note: A reference search is used when initializing the stage so it can find the "zero" position. On the physical stage, ther reference points are usually magnetic sensors.**

### **EtalumaStage.StageController.zAxisReferenceSearchMoving**

Call this to see if the z-axis is moving during a reference search. Use only during reference searching!

**Return:** True if the z-axis is moving.

**Note: A reference search is used when initializing the stage so it can find the "zero" position. On the physical stage, ther reference points are usually magnetic sensors.**

### **EtalumaStage.StageController.SetYAxisMaximumSpeed(System.Int32)**

Sets the maximum speed of the y-axis stage.

**Param** "speed": Units of speed, see p. 70 of 'TMCM-6110\_TMCL\_firmware\_manual.pdf' for a details.

**Return:** True is the speed is successfully set.

### **EtalumaStage.StageController.SetYAxisMaximumAcceleration(System.Int32)**

Sets the maximum acceleration of the y-axis stage.

**Param** "acceleration": Units of acceleration, see p. 70 of 'TMCM-6110\_TMCL\_firmware\_manual.pdf' for a details.

**Return:** True is the speed is successfully set.

### **EtalumaStage.StageController.SetXAxisMaximumSpeed(System.Int32)**

Sets the maximum speed of the x-axis stage.

**Param** "speed": Units of speed, see p. 70 of 'TMCM-6110\_TMCL\_firmware\_manual.pdf' for a details.

**Return:** True is the speed is successfully set.

**EtalumaStage.StageController.SetXAxisMaximumAcceleration(System.Int32)**

Sets the maximum acceleration of the x-axis stage.

**Param** "acceleration"

Units of acceleration, see p. 70 of 'TMCM-6110\_TMCL\_firmware\_manual.pdf' for a details.

**Return:** True if the speed is successfully set.

**EtalumaStage.StageController.GetAbsoluteYPositionMillimeters(System.Single@)**

Gets the "absolute" position of the y-axis stage in millimeters but relative to the home position.

**Param** "yAbsolutePositionMillimeters": This gets set to the absolute position of the y-axis stage during a successful call to this function.

**Return:** True if the position was successfully read.

**EtalumaStage.StageController.GetAbsoluteXPositionMillimeters(System.Single@)**

Gets the "absolute" position of the x-axis stage in millimeters but relative to the home position.

**Param** "xAbsolutePositionMillimeters": This gets set to the absolute position of the x-axis stage during a successful call to this function.

**Return:** True if the position was successfully read.

**EtalumaStage.StageController.GetAbsoluteZPositionMillimeters(System.Single@)**

Gets the "absolute" position of the z-axis stage in millimeters but relative to the home position.

**Param** "zAbsolutePositionMillimeters": This gets set to the absolute position of the z-axis stage during a successful call to this function.

**Return:** True if the position was successfully read.

**EtalumaStage.StageController.GetAbsoluteYPositionMicroSteps(System.Int32@)**

Gets the absolute position of the y-axis stage in microsteps.

**Param** "yAbsolutePositionMicroSteps": This gets set to the absolute position of the y-axis during a successful call to this function.

**Return:** True if the position was successfully read.

**EtalumaStage.StageController.GetAbsoluteXPositionMicroSteps(System.Int32@)**

Gets the absolute position of the x-axis stage in microsteps.

**Param** "xAbsolutePositionMicroSteps": This gets set to the absolute position of the x-axis during a successful call to this function.

**Return:** True if the position was successfully read.

**EtalumaStage.StageController.GetAbsoluteZPositionMicroSteps(System.Int32@)**

Gets the absolute position of the z-axis stage in microsteps.

**Param** "zAbsolutePositionMicroSteps": This gets set to the absolute position of the z-axis during a successful call to this function.

**Return:** True if the position was successfully read.

**EtalumaStage.StageController.IsXStageMoving**

Indicates if the x-axis if the stage is moving. Use only during non-reference searching!

**Return:** True if the x-axis is moving.

**Note:** A reference search is used when initializing the stage so it can find the "zero" position. On the physical stage, the reference points are usually magnetic sensors.

### **EtalumaStage.StageController.IsYStageMoving**

Indicates if the y-axis if the stage is moving. Use only during non-reference searching!

**Return:** True if the y-axis is moving.

**Note:** A reference search is used when initializing the stage so it can find the "zero" position. On the physical stage, the reference points are usually magnetic sensors.

### **EtalumaStage.StageController.IsZStageMoving**

Indicates if the z-axis if the stage is moving. Use only during non-reference searching!

**Return:** True if the z-axis is moving.

**Note:** A reference search is used when initializing the stage so it can find the "zero" position. On the physical stage, the reference points are usually magnetic sensors.

### **EtalumaStage.StageController.StopXMotor**

This stops the x motor. Use only during non-reference searching!

**Return:** True if the x motor was successfully commanded to stop.

**Note:** A reference search is used when initializing the stage so it can find the "zero" position. On the physical stage, ther reference points are usually magnetic sensors.

### **EtalumaStage.StageController.StopYMotor**

This stops the y motor. Use only during non-reference searching!

**Return:** True if the y motor was successfully commanded to stop.

**Note:** A reference search is used when initializing the stage so it can find the "zero" position. On the physical stage, ther reference points are usually magnetic sensors.

### **EtalumaStage.StageController.StopZMotor**

This stops the z motor. Use only during non-reference searching!

**Return:** True if the z motor was successfully commanded to stop.

**Note:** A reference search is used when initializing the stage so it can find the "zero" position. On the physical stage, ther reference points are usually magnetic sensors.

### **EtalumaStage.StageController.GetXAbsoluteRangeMicrometers(System.Single@,System.Single@)**

This function returns the range of travel possible for the x-axis of the stage.

**Param** "xMinimumMillimeters": Lower limit of travel in millimeters.

**Param** "xMaximumMillimeters": Upper limit of travel in millimeters.

**EtalumaStage.StageController.GetYAbsoluteRangeMicrometers(System.Single@,System.Single@)**

This function returns the range of travel possible for the y-axis of the stage.

**Param** "yMinimumMillimeters": Lower limit of travel in millimeters.

**Param** "yMaximumMillimeters": Upper limit of travel in millimeters.

**EtalumaStage.StageController.ConvertMicroStepsToMicrometers(System.Int32)**

Converts the parameter (microsteps) to micrometers.

**Param** "microSteps": The microsteps value to convert.

**Return**: The value in micrometers.

**EtalumaStage.StageController.ConvertMicrometersToMicroSteps(System.Int32)**

Converts the parameter (micrometers) to microsteps.

**Param** "microMeters": The micro-meters value to convert.

**Return**: The value in microsteps.

**EtalumaStage.StageController.ConvertZMicroStepsToMicrometers(System.Int32)**

Converts the parameter (microsteps) to micrometers for the z-axis.

**Param** "microSteps": The microsteps value to convert.

**Return**: The value in micrometers.

**EtalumaStage.StageController.ConvertZMicrometersToMicroSteps(System.Int32)**

Converts the parameter (micrometers) to microsteps.

**Param** "microMeters": The micro-meters value to convert.

**Return**: The value in microsteps.

**EtalumaStage.StageController.MoveXStageToAbsolutePositionMillimeters(System.Single)**

Moves the x-axis to the specified absolute position (in millimeters).

The zero position is home. **Param** "milliMeters": The desired absolute position of the x-axis, in millimeters.

**Return**: True if the x axis was successfully commanded to move.

**EtalumaStage.StageController.MoveYStageToAbsolutePositionMillimeters(System.Single)**

Moves the y-axis to the specified absolute position (in millimeters). The zero position is home.

**Param** "milliMeters": The desired absolute position of the y-axis, in millimeters.

**Return**: True if the y axis was successfully commanded to move.

**EtalumaStage.StageController.MoveZStageToAbsolutePositionMillimeters(System.Single)**

Moves the z-axis to the specified absolute position (in millimeters).

**Param** "milliMeters": The desired absolute position of the z-axis, in millimeters.

**Return**: True if the z axis was successfully commanded to move.

**EtalumaStage.StageController.MoveXStageRelativeToMicroplateHomeMillimeters(System.Single)**

Moves the x-axis to the specified position relative to the upper-left corner of the microplate (in millimeters). The zero

position is the upper-left corner of the microplate.

**Param** "milliMeters": The desired relative position of the x-axis, in millimeters. This value must be positive.

**Return:** True if the x-axis was successfully commanded to move.

### **EtalumaStage.StageController.MoveYStageRelativeToMicroplateHomeMillimeters(System.Single)**

Moves the y-axis to the specified position relative to the upper-left corner of the microplate (in millimeters). The zero position is the upper-left corner of the microplate.

**Param** "milliMeters": The desired relative position of the y-axis, in millimeters. This value must be negative.

**Return:** True if the y-axis was successfully commanded to move.

### **EtalumaStage.StageController.MoveXStageToAbsolutePositionMicroSteps(System.Int32)**

Moves the x-stage to the specified absolute position (in microsteps).

**Param** "microSteps": The desired absolute position of the x-stage, in microsteps.

**Return:** True if the x axis was successfully commanded to move.

### **EtalumaStage.StageController.MoveYStageToAbsolutePositionMicroSteps(System.Int32)**

Moves the y-stage to the specified absolute position (in microsteps).

**Param** "microSteps": The desired absolute position of the y-stage, in microsteps.

**Return:** True if the y axis was successfully commanded to move.

### **EtalumaStage.StageController.MoveZStageToAbsolutePositionMicroSteps(System.Int32)**

Moves the z-stage to the specified absolute position (in microsteps).

**Param** "microSteps": The desired absolute position of the z-stage, in microsteps.

**Return:** True if the z axis was successfully commanded to move.

### **EtalumaStage.StageController.MoveXStageToRelativePositionMicroSteps(System.Int32)**

Moves the x-stage to the specified relative position (in microsteps).

**Param** "microSteps": The desired relative position of the x-stage, in microsteps.

**Return:** True if the x axis was successfully commanded to move.

### **EtalumaStage.StageController.MoveYStageToRelativePositionMicroSteps(System.Int32)**

Moves the y-stage to the specified relative position (in microsteps).

**Param** "microSteps": The desired relative position of the y-stage, in microsteps.

**Return:** True if the y axis was successfully commanded to move.

### **EtalumaStage.StageController.MoveZStageToRelativePositionMicroSteps(System.Int32)**

Moves the z-stage to the specified relative position (in microsteps).

**Param** "microSteps": The desired relative position of the z-stage, in microsteps.

**Return:** True if the z axis was successfully commanded to move.

### **EtalumaStage.StageController.MoveXStageToRelativePositionMilliMeters(System.Single)**

Moves the x-axis to the specified relative position (in millimeters).

**Param** "milliMeters": The desired relative position of the axis, in millimeters.

**Return:** True if the axis was successfully commanded to move.

### **EtalumaStage.StageController.MoveYStageToRelativePositionMilliMeters(System.Single)**

Moves the y-axis to the specified relative position (in millimeters).

**Param** "milliMeters": The desired relative position of the axis, in millimeters.

**Return:** True if the axis was successfully commanded to move.

### **EtalumaStage.StageController.MoveZStageToRelativePositionMilliMeters(System.Single)**

Moves the z-axis to the specified relative position (in millimeters).

**Param** "milliMeters": The desired relative position of the axis, in millimeters.

**Return:** True if the axis was successfully commanded to move.

### **EtalumaStage.StageController.ReachedAxisPosition(System.Byte)**

Indicates if a specified axis has reached the previously-commanded position.

**Param** "motor": Specifies the axis, X\_MOTOR, Y\_MOTOR or Z\_MOTOR.

**Return:** True if the specified axis has reached the position.

### **EtalumaStage.StageController.SendCmd**

**(System.Byte,System.Byte,System.Byte,System.Byte,System.Int32,EtalumaStage.TmclReplyMessage@)**

Send a binary TMCL command. This functions blocks for the 'REPLY\_MESSAGE\_TIMEOUT\_MS' timeout waiting for the reply message from the board. The function will return 'false' if no reply message is received within the timeout period. Do not send the next command before you have received the reply! (p. 16 of 'TMCM-6110\_TMCL\_firmware\_manual.pdf') e.g. SendCmd(ComHandle, 1, TMCL\_MVP, MVP\_ABS, 1, 50000); will be MVP ABS, 1, 50000 for a module with address 1  
Parameters: Handle: Handle of the serial port (returned by OpenRS232).

**Param** "address": address of the module (factory default is 1).

**Param** "command": the TMCL command (see the constants at the beginning of this file)

**Param** "type": the "Type" parameter of the TMCL command (set to 0 if unused)

**Param** "motor": the motor number (set to 0 if unused)

**Param** "value": the "Value" parameter (depending on the command, set to 0 if unused)

**Param** "tmclReplyMessage": this gets set if the function is successful with response parameters

**Return:** 'true' if the reply message is received within the timeout period, 'false' if not.

### **EtalumaStage.StageController.xyDefaultMaximumPositioningSpeed**

The default (initial) speed to position the x and y axes.

### **EtalumaStage.StageController.xyDefaultMaximumAcceleration**

The default (initial) acceleration to position the x and y axes.

### **EtalumaStage.StageController.xyLowerMaximumPositioningVelocity**

The lower limit of the maximum velocity used when executing a ramp to a position for the x and y axes.

### **EtalumaStage.StageController.xyUpperMaximumPositioningVelocity**

The upper limit of the maximum velocity used when executing a ramp to a position for the x and y axes.



**EtalumaStage.StageController.xyLowerMaximumAcceleration**

The lower limit of the maximum acceleration used to accelerate or decelerate the motor for the x and y axes.

**EtalumaStage.StageController.xyUpperMaximumAcceleration**

The upper limit of the maximum acceleration used accelerate or decelerate the motor for the x and y axes.

**EtalumaStage.StageController.xMaximumAxisTravelMicrosteps**

This specifies the maximum distance that the x-axis can move, in microsteps.

**EtalumaStage.StageController.yMaximumAxisTravelMicrosteps**

This specifies the maximum distance that the x-axis can move, in microsteps.

**EtalumaStage.StageController.zMaximumAxisTravelMicrosteps**

This specifies the maximum distance that the x-axis can move, in microsteps.

LumaUSB v14.10.1