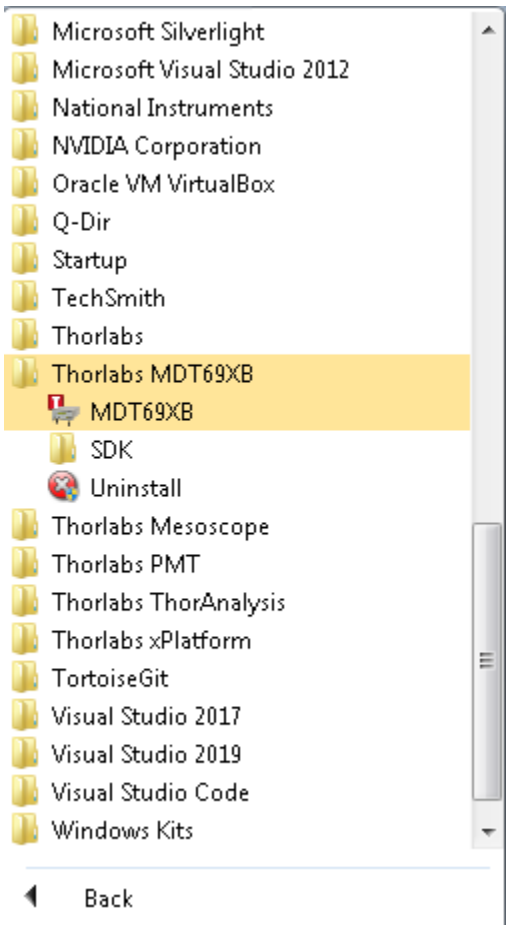
MDT69XB

SDK Manual

Software Development

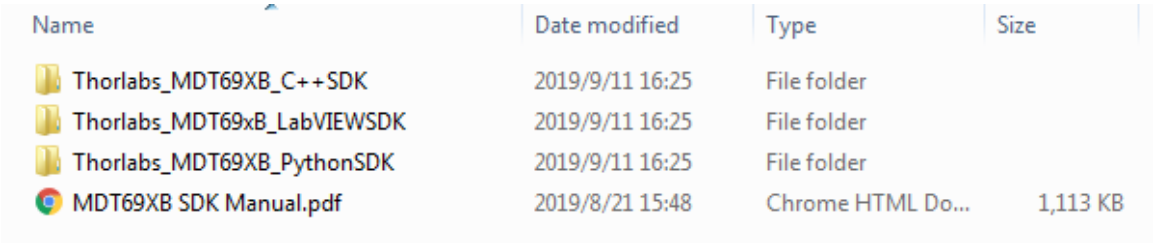
User can start software development in C/C++ develop environment, Python, LabVIEW etc.

The software development interface can be found in the start menu.



or by clicking *Help* in software menu.

In this directory, you will find two folders and the support files for software development, as shown below.



Software Development (C/C++)

User can start software development with MDT_COMMAND_LIB_win32.dll in C/C++ development environment which can be found in Thorlabs_MDT69XB_C++SDK. The corresponding header file is CmdLibrary.h.

Copy MDT_COMMAND_LIB_win32.dll to your program folder, and make sure the library file and exe file are in the same folder.

Commands and Queries

The following list shows all available commands and queries, and summarizes their functions for the MDT693B:

Command	C++ function	Description
Product Information	GetId	Returns the product header and firmware version
Restore Default Settings	RestoreDefaultSettings	Restores default factory settings
Limit Switch Setting	GetLimitVoltage	Returns output voltage limit setting.
Get Display Intensity	GetDisplayIntensity	Returns the display intensity
Set Display Intensity	SetDisplayIntensity	Set the Display Intensity (0-15)
Set All Voltages	SetAllVoltage	Sets all outputs to desired voltage
Get Master Scan Enable	GetMasterScanEnable	Returns the state of the Master Scan enable
Set Master Scan Enable	SetMasterScanEnable	Sets Master Scan mode (1 = enable; 0 = disable)
Get Master Scan Voltage	GetMasterScanVoltage	Reads and Returns the master scan voltage
Set Master Scan Voltage	SetMasterScanVoltage	Sets a master scan voltage that adds to the x, y, and z axis voltages. (Sets master scan DAC)
Read X-Axis Voltage	GetXAxisVoltage	Reads and returns the X axis output voltage.
Set X-Axis Voltage	SetXAxisVoltage	Set the output voltage for the X axis.
Read Y-Axis Voltage	GetYAxisVoltage	Reads and returns the Y axis output voltage.
Set Y-Axis Voltage	SetYAxisVoltage	Set the output voltage for the Y axis.
Read Z-Axis Voltage	GetZAxisVoltage	Reads and returns the Z axis output voltage.
Set Z-Axis Voltage	SetZAxisVoltage	Set the output voltage for the Z axis.
Read Min. X-Axis Voltage	GetXAxisMinVoltage	Reads the minimum output voltage limit for X axis.
Set Min. X-Axis Voltage	SetXAxisMinVoltage	Sets the minimum output voltage limit for X axis.
Read Min. Y-Axis Voltage	GetYAxisMinVoltage	Reads the minimum output voltage limit for Y axis.
Set Min. Y-Axis Voltage	SetYAxisMinVoltage	Sets the minimum output voltage limit for Y axis.
Read Min. Z-Axis Voltage	GetZAxisMinVoltage	Reads the minimum output voltage limit for Z axis.
Set Min. Z-Axis Voltage	SetZAxisMinVoltage	Sets the minimum output voltage limit for Z axis.
Read Max. X-Axis Voltage	GetXAxisMaxVoltage	Reads and returns the maximum output voltage

		limit for X axis.
Set Max. X-Axis Voltage	SetXAxisMaxVoltage	Sets the maximum output voltage limit for X axis.
Read Max. Y-Axis Voltage	GetYAxisMaxVoltage	Reads and returns the maximum output voltage limit for Y axis.
Set Max. Y-Axis Voltage	SetYAxisMaxVoltage	Sets the maximum output voltage limit for Y axis.
Read Max. Z-Axis Voltage	GetZAxisMaxVoltage	Reads and returns the maximum output voltage limit for Z axis.
Set Max. Z-Axis Voltage	SetZAxisMaxVoltage	Sets the maximum output voltage limit for Z axis.
Get voltage adjustment resolution	GetVoltageAdjustment Resolution	Reads the current step resolution.
Set voltage adjustment resolution	SetVoltageAdjustment Resolution	Sets the step resolution when using up/down arrow keys (n = 1 to 1000)
Get Serial Number	GetSerialNumber	Returns the serial number.
Read X,Y,Z axis voltages	GetXYZAxisVoltage	Reads and return the X,Y,Z axis output voltages.
Set X,Y,Z axis voltages	SetXYZAxisVoltage	Set the X, Y, Z axis output voltages.

The following list shows all available commands and queries, and summarizes their functions for the MDT694B:

Command	C++ function	Description
Product Information	GetId	Returns the product header and firmware version
Restore Default Settings	RestoreDefaultSettings	Restores default factory settings
Limit Switch Setting	GetLimitVoltage	Returns output voltage limit setting.
Get Display Intensity	GetDisplayIntensity	Returns the display intensity
Set Display Intensity	SetDisplayIntensity	Set the Display Intensity (0-15)
Read X-Axis Voltage	GetXAxisVoltage	Reads and returns the X axis output voltage.
Set X-Axis Voltage	SetXAxisVoltage	Set the output voltage for the X axis.
Read Min. X-Axis Voltage	GetXAxisMinVoltage	Reads the minimum output voltage limit for X axis.
Set Min. X-Axis Voltage	SetXAxisMinVoltage	Sets the minimum output voltage limit for X axis.
Read Max. X-Axis Voltage	GetXAxisMaxVoltage	Reads and returns the maximum output voltage limit for X axis.
Set Max. X-Axis Voltage	SetXAxisMaxVoltage	Sets the maximum output voltage limit for X axis.
Get voltage adjustment resolution	GetVoltageAdjustment Resolution	Reads the current step resolution.
Set voltage adjustment resolution	SetVoltageAdjustment Resolution	Sets the step resolution when using up/down arrow keys (n = 1 to 1000)
Get Serial Number	GetSerialNumber	Returns the serial number.

CmdLibrary.h File Reference

Defines

- #define **COMMANDLIB_API**

Functions

- **COMMANDLIB_API** int **List** (unsigned char *serialNo)
- **COMMANDLIB_API** int **Open** (unsigned char *serialNo, int nBaud, int timeout)
- **COMMANDLIB_API** int **IsOpen** (unsigned char *serialNo)
- **COMMANDLIB_API** int **Close** (int hdl)
- **COMMANDLIB_API** int **Read** (int hdl, unsigned char *b, int limit)
- **COMMANDLIB_API** int **Write** (int hdl, unsigned char *b, int size)
- **COMMANDLIB_API** int **Set** (int hdl, unsigned char *c, int size)
- **COMMANDLIB_API** int **Get** (int hdl, unsigned char *c, char *d)
- **COMMANDLIB_API** int **SetTimeout** (int hdl, int time)
- **COMMANDLIB_API** int **GetId** (int hdl, unsigned char *id)
- **COMMANDLIB_API** int **GetSerialNumber** (int hdl, unsigned char *sn)
- **COMMANDLIB_API** int **RestoreDefaultSettings** (int hdl)
- **COMMANDLIB_API** int **GetLimitVoltage** (int hdl, double *voltage)
- **COMMANDLIB_API** int **GetDisplayIntensity** (int hdl, int *intensity)
- **COMMANDLIB_API** int **SetDisplayIntensity** (int hdl, int intensity)
- **COMMANDLIB_API** int **SetAllVoltage** (int hdl, double voltage)
- **COMMANDLIB_API** int **GetMasterScanEnable** (int hdl, int *state)
- **COMMANDLIB_API** int **SetMasterScanEnable** (int hdl, int state)
- **COMMANDLIB_API** int **GetMasterScanVoltage** (int hdl, double *voltage)
- **COMMANDLIB_API** int **SetMasterScanVoltage** (int hdl, double voltage)
- **COMMANDLIB_API** int **GetXAxisVoltage** (int hdl, double *voltage)
- **COMMANDLIB_API** int **SetXAxisVoltage** (int hdl, double voltage)
- **COMMANDLIB_API** int **GetYAxisVoltage** (int hdl, double *voltage)
- **COMMANDLIB_API** int **SetYAxisVoltage** (int hdl, double voltage)
- **COMMANDLIB_API** int **GetZAxisVoltage** (int hdl, double *voltage)
- **COMMANDLIB_API** int **SetZAxisVoltage** (int hdl, double voltage)
- **COMMANDLIB_API** int **GetXAxisMinVoltage** (int hdl, double *voltage)
- **COMMANDLIB_API** int **SetXAxisMinVoltage** (int hdl, double voltage)
- **COMMANDLIB_API** int **GetYAxisMinVoltage** (int hdl, double *voltage)
- **COMMANDLIB_API** int **SetYAxisMinVoltage** (int hdl, double voltage)
- **COMMANDLIB_API** int **GetZAxisMinVoltage** (int hdl, double *voltage)
- **COMMANDLIB_API** int **SetZAxisMinVoltage** (int hdl, double voltage)
- **COMMANDLIB_API** int **GetXAxisMaxVoltage** (int hdl, double *voltage)
- **COMMANDLIB_API** int **SetXAxisMaxVoltage** (int hdl, double voltage)
- **COMMANDLIB_API** int **GetYAxisMaxVoltage** (int hdl, double *voltage)
- **COMMANDLIB_API** int **SetYAxisMaxVoltage** (int hdl, double voltage)
- **COMMANDLIB_API** int **GetZAxisMaxVoltage** (int hdl, double *voltage)
- **COMMANDLIB_API** int **SetZAxisMaxVoltage** (int hdl, double voltage)
- **COMMANDLIB_API** int **GetVoltageAdjustmentResolution** (int hdl, int *step)
- **COMMANDLIB_API** int **SetVoltageAdjustmentResolution** (int hdl, int step)
- **COMMANDLIB_API** int **GetXYZAxisVoltage** (int hdl, double *xVoltage, double *yVoltage, double *zVoltage)
- **COMMANDLIB_API** int **SetXYZAxisVoltage** (int hdl, double xVoltage, double yVoltage, double zVoltage)

Variables

- param *< returns > negative **number**
  Purge the RX and TX buffer on port.

- param *< returns > negative int **flag**

Define Documentation

#define COMMANDLIB_API

Function Documentation

COMMANDLIB_API int List (unsigned char * *serialNo*)

list all the possible port on this computer.

Parameters:

serialNo	port list returned string include serial number and device descriptor, separated by comma

Returns:

non-negative number: number of device in the list; negative number: failed.

COMMANDLIB_API int Open (unsigned char * *serialNo*, int *nBaud*, int *timeout*)

open port function.

Parameters:

serialNo	serial number of the device to be opened, use GetPorts function to get exist list first.
nBaud	bit per second of port
timeout	set timeout value in (s)

Returns:

non-negative number: hdl number returned Successfully; negative number: failed.

COMMANDLIB_API int IsOpen (unsigned char * *serialNo*)

check opened status of port

Parameters:

serialNo	serial number of the device to be checked.

Returns:

0: port is not opened; 1: port is opened.

COMMANDLIB_API int Close (int *hdl*)

close current opened port

Parameters:

hdl	handle of port.

Returns:

0: Success; negative number: failed.

COMMANDLIB_API int Read (int *hdl*, unsigned char * *b*, int *limit*)

Read string from device through opened port.

make sure the port was opened Successful before call this function.

Parameters:

hdl	handle of port.
b	returned string buffer
limit	

ABS(limit): max length value of b buffer.

SIGN(limit) == 1 : wait RX event until time out value expired;

SIGN(limit) == -1: INFINITE wait event untill RX has data;

Returns:

non-negative number: size of actual read data in byte; negative number: failed.

COMMANDLIB_API int Write (int *hdl*, unsigned char * *b*, int *size*)

Write string to device through opened port.

make sure the port was opened Successful before call this function.

Parameters:

hdl	handle of port.
b	input string
size	size of string to be written.

Returns:

non-negative number: number of bytes written; negative number: failed.

COMMANDLIB_API int Set (int *hdl*, unsigned char * *c*, int *size*)

set command to device according to protocol in manual.

make sure the port was opened Successful before call this function.

make sure this is the correct device by checking the ID string before call this function.

Parameters:

hdl	handle of port.
c	input command string
size	lenth of input command string (<255)

Returns:

0: Success; negative number: failed.
0xEA: CMD_NOT_DEFINED;
0xEB: time out;
0xED: invalid string buffer;

COMMANDLIB_API int Get (int *hdl*, unsigned char * *c*, char * *d*)

set command to device according to protocol in manual and get the return string.
make sure the port was opened Successful before call this function.
make sure this is the correct device by checking the ID string before call this function.

Parameters:

hdl	handle of port.
c	input command string (<255)
d	output string (<255)

Returns:

0: Success; negative number: failed.
0xEA: CMD_NOT_DEFINED;
0xEB: time out;
0xED: invalid string buffer;

COMMANDLIB_API int SetTimeout (int *hdl*, int *time*)

set time out value for read or write process.

Parameters:

hdl	handle of port.
time	time out value

Returns:

0: Success; negative number: failed.

COMMANDLIB_API int GetId (int *hdl*, unsigned char * *id*)

Get the product header and firmware version
make sure the port was opened Successful before call this function.

make sure this is the correct device by checking the ID string before call this function.

Parameters:

hdl	handle of port.
id	model number and firmware version

Returns:

0: Success; negative number: failed.

COMMANDLIB_API int GetSerialNumber (int *hdl*, unsigned char * *sn*)

get the serial number of device.

make sure the port was opened Successful before call this function.

make sure this is the correct device by checking the ID string before call this function.

Parameters:

hdl	handle of port.
sn	serial number of device.

Returns:

0: Success; negative number: failed.

COMMANDLIB_API int RestoreDefaultSettings (int *hdl*)

Restores default factory settings.

make sure the port was opened Successful before call this function.

make sure this is the correct device by checking the ID string before call this function.

Parameters:

hdl	handle of port.

Returns:

0: Success; negative number: failed.

COMMANDLIB_API int GetLimitVoltage (int *hdl*, double * *voltage*)

Get output voltage limit setting.

make sure the port was opened Successful before call this function.

make sure this is the correct device by checking the ID string before call this function.

Parameters:

hdl	handle of port.
voltage	output voltage limit setting.

Returns:

0: Success; negative number: failed.

COMMANDLIB_API int GetDisplayIntensity (int *hdl*, int * *intensity*)

Get the display intensity.

make sure the port was opened Successful before call this function.

make sure this is the correct device by checking the ID string before call this function.

Parameters:

hdl	handle of port.
intensity	intensity of display pannel.

Returns:

0: Success; negative number: failed.

COMMANDLIB_API int SetDisplayIntensity (int *hdl*, int *intensity*)

Set the Display Intensity (0-15).

make sure the port was opened Successful before call this function.

make sure this is the correct device by checking the ID string before call this function.

Parameters:

hdl	handle of port.
intensity	target intensity range:(1~15)

Returns:

0: Success; negative number: failed.

COMMANDLIB_API int SetAllVoltage (int *hdl*, double *voltage*)

Set all outputs to desired voltage.

make sure the port was opened Successful before call this function.

make sure this is the correct device by checking the ID string before call this function.

Parameters:

hdl	handle of port.
voltage	target voltage range:(0 ~ limit voltage)

Returns:

0: Success; negative number: failed.

COMMANDLIB_API int GetMasterScanEnable (int *hdl*, int * *state*)

Get the state of the Master Scan enable.

make sure the port was opened Successful before call this function.

make sure this is the correct device by checking the ID string before call this function.

Parameters:

hdl	handle of port.

state	current master scan state.(1-enable,0-disable)

Returns:

0: Success; negative number: failed.

COMMANDLIB_API int SetMasterScanEnable (int *hdl*, int *state*)

Set Master Scan mode.

make sure the port was opened Successful before call this function.

make sure this is the correct device by checking the ID string before call this function.

Parameters:

hdl	handle of port.
state	target state of master scan:(1-enable,0-disable)

Returns:

0: Success; negative number: failed.

COMMANDLIB_API int GetMasterScanVoltage (int *hdl*, double * *voltage*)

Get the master scan voltage.

make sure the port was opened Successful before call this function.

make sure this is the correct device by checking the ID string before call this function.

Parameters:

hdl	handle of port.
voltage	current master scan voltage

Returns:

0: Success; negative number: failed.

COMMANDLIB_API int SetMasterScanVoltage (int *hdl*, double *voltage*)

Set a master scan voltage that adds to the x, y, and z axis voltages.

make sure the port was opened Successful before call this function.

make sure this is the correct device by checking the ID string before call this function.

Parameters:

hdl	handle of port.
voltage	target voltage range:(0 ~ limit voltage)

Returns:

0: Success; negative number: failed.

COMMANDLIB_API int GetXAxisVoltage (int *hdl*, double * *voltage*)

Get the X axis output voltage.

make sure the port was opened Successful before call this function.

make sure this is the correct device by checking the ID string before call this function.

Parameters:

hdl	handle of port.
voltage	current x-axis output voltage.

Returns:

0: Success; negative number: failed.

COMMANDLIB_API int SetXAxisVoltage (int *hdl*, double *voltage*)

Set the output voltage for the X axis.

make sure the port was opened Successful before call this function.

make sure this is the correct device by checking the ID string before call this function.

Parameters:

hdl	handle of port.
voltage	target voltage range:(0 ~ limit voltage)

Returns:

0: Success; negative number: failed.

COMMANDLIB_API int GetYAxisVoltage (int *hdl*, double * *voltage*)

Get the Y axis output voltage.

make sure the port was opened Successful before call this function.

make sure this is the correct device by checking the ID string before call this function.

Parameters:

hdl	handle of port.
voltage	current y-axis output voltage.

Returns:

0: Success; negative number: failed.

COMMANDLIB_API int SetYAxisVoltage (int *hdl*, double *voltage*)

Set the output voltage for the Y axis.

make sure the port was opened Successful before call this function.

make sure this is the correct device by checking the ID string before call this function.

Parameters:

hdl	handle of port.

voltage	target voltage range:(0 ~ limit voltage)

Returns:

0: Success; negative number: failed.

COMMANDLIB_API int GetZAxisVoltage (int *hdl*, double * *voltage*)

Get the Z axis output voltage.

make sure the port was opened Successful before call this function.

make sure this is the correct device by checking the ID string before call this function.

Parameters:

hdl	handle of port.
voltage	current z-axis output voltage.

Returns:

0: Success; negative number: failed.

COMMANDLIB_API int SetZAxisVoltage (int *hdl*, double *voltage*)

Set the output voltage for the Z axis.

make sure the port was opened Successful before call this function.

make sure this is the correct device by checking the ID string before call this function.

Parameters:

hdl	handle of port.
voltage	target voltage range:(0 ~ limit voltage)

Returns:

0: Success; negative number: failed.

COMMANDLIB_API int GetXAxisMinVoltage (int *hdl*, double * *voltage*)

Get the minimum output voltage limit for X axis.

make sure the port was opened Successful before call this function.

make sure this is the correct device by checking the ID string before call this function.

Parameters:

hdl	handle of port.
voltage	current x-axis min output voltage.

Returns:

0: Success; negative number: failed.

COMMANDLIB_API int SetXAxisMinVoltage (int *hdl*, double *voltage*)

Set the minimum output voltage limit for X axis.

make sure the port was opened Successful before call this function.

make sure this is the correct device by checking the ID string before call this function.

Parameters:

hdl	handle of port.
voltage	target voltage range:(0 ~ limit voltage)

Returns:

0: Success; negative number: failed.

COMMANDLIB_API int GetYAxisMinVoltage (int *hdl*, double * *voltage*)

Get the minimum output voltage limit for Y axis.

make sure the port was opened Successful before call this function.

make sure this is the correct device by checking the ID string before call this function.

Parameters:

hdl	handle of port.
voltage	current y-axis min output voltage.

Returns:

0: Success; negative number: failed.

COMMANDLIB_API int SetYAxisMinVoltage (int *hdl*, double *voltage*)

Set the minimum output voltage limit for Y axis.

make sure the port was opened Successful before call this function.

make sure this is the correct device by checking the ID string before call this function.

Parameters:

hdl	handle of port.
voltage	target voltage range:(0 ~ limit voltage)

Returns:

0: Success; negative number: failed.

COMMANDLIB_API int GetZAxisMinVoltage (int *hdl*, double * *voltage*)

Get the minimum output voltage limit for Z axis.

make sure the port was opened Successful before call this function.

make sure this is the correct device by checking the ID string before call this function.

Parameters:

hdl	handle of port.

voltage	current min z-axis output voltage.

Returns:

0: Success; negative number: failed.

COMMANDLIB_API int SetZAxisMinVoltage (int *hdl*, double *voltage*)

Set the minimum output voltage limit for Z axis.
make sure the port was opened Successful before call this function.
make sure this is the correct device by checking the ID string before call this function.

Parameters:

hdl	handle of port.
voltage	target voltage range:(0 ~ limit voltage)

Returns:

0: Success; negative number: failed.

COMMANDLIB_API int GetXAxisMaxVoltage (int *hdl*, double * *voltage*)

Get the maximum output voltage limit for X axis.
make sure the port was opened Successful before call this function.
make sure this is the correct device by checking the ID string before call this function.

Parameters:

hdl	handle of port.
voltage	current x-axis max output voltage.

Returns:

0: Success; negative number: failed.

COMMANDLIB_API int SetXAxisMaxVoltage (int *hdl*, double *voltage*)

Set the maximum output voltage limit for X axis.
make sure the port was opened Successful before call this function.
make sure this is the correct device by checking the ID string before call this function.

Parameters:

hdl	handle of port.
voltage	target voltage range:(0 ~ limit voltage)

Returns:

0: Success; negative number: failed.

COMMANDLIB_API int GetYAxisMaxVoltage (int *hdl*, double * *voltage*)

Get the maximum output voltage limit for Y axis.

make sure the port was opened Successful before call this function.

make sure this is the correct device by checking the ID string before call this function.

Parameters:

hdl	handle of port.
voltage	current y-axis max output voltage.

Returns:

0: Success; negative number: failed.

COMMANDLIB_API int SetYAxisMaxVoltage (int *hdl*, double *voltage*)

Set the maximum output voltage limit for Y axis.

make sure the port was opened Successful before call this function.

make sure this is the correct device by checking the ID string before call this function.

Parameters:

hdl	handle of port.
voltage	target voltage range:(0 ~ limit voltage)

Returns:

0: Success; negative number: failed.

COMMANDLIB_API int GetZAxisMaxVoltage (int *hdl*, double * *voltage*)

Get the maximum output voltage limit for Z axis.

make sure the port was opened Successful before call this function.

make sure this is the correct device by checking the ID string before call this function.

Parameters:

hdl	handle of port.
voltage	current max z-axis output voltage.

Returns:

0: Success; negative number: failed.

COMMANDLIB_API int SetZAxisMaxVoltage (int *hdl*, double *voltage*)

Set the maximum output voltage limit for Z axis.

make sure the port was opened Successful before call this function.

make sure this is the correct device by checking the ID string before call this function.

Parameters:

hdl	handle of port.

voltage	target voltage range:(0 ~ limit voltage)

Returns:

0: Success; negative number: failed.

COMMANDLIB_API int GetVoltageAdjustmentResolution (int *hdl*, int * *step*)

Get the current step resolution.

make sure the port was opened Successful before call this function.

make sure this is the correct device by checking the ID string before call this function.

Parameters:

hdl	handle of port.
step	current step solution.

Returns:

0: Success; negative number: failed.

COMMANDLIB_API int SetVoltageAdjustmentResolution (int *hdl*, int *step*)

Set the step resolution when using up/down arrow keys.

make sure the port was opened Successful before call this function.

make sure this is the correct device by checking the ID string before call this function.

Parameters:

hdl	handle of port.
step	target step range:(1 ~ 1000)

Returns:

0: Success; negative number: failed.

COMMANDLIB_API int GetXYZAxisVoltage (int *hdl*, double * *xVoltage*, double * *yVoltage*, double * *zVoltage*)

Get the x,y,z axis output voltages.

make sure the port was opened Successful before call this function.

make sure this is the correct device by checking the ID string before call this function.

Parameters:

hdl	handle of port.
xVoltage	current x axis output voltage.
yVoltage	current y axis output voltage.
zVoltage	current z axis output voltage.

Returns:

0: Success; negative number: failed.

COMMANDLIB_API int SetXYZAxisVoltage (int *hdl*, double *xVoltage*, double *yVoltage*, double *zVoltage*)

Set the x,y,z axis output voltages.
make sure the port was opened Successful before call this function.
make sure this is the correct device by checking the ID string before call this function.

Parameters:

hdl	handle of port.
xVoltage	target x axis voltage range:(0 ~ limit voltage)
yVoltage	target y axis voltage range:(0 ~ limit voltage)
zVoltage	target z axis voltage range:(0 ~ limit voltage)

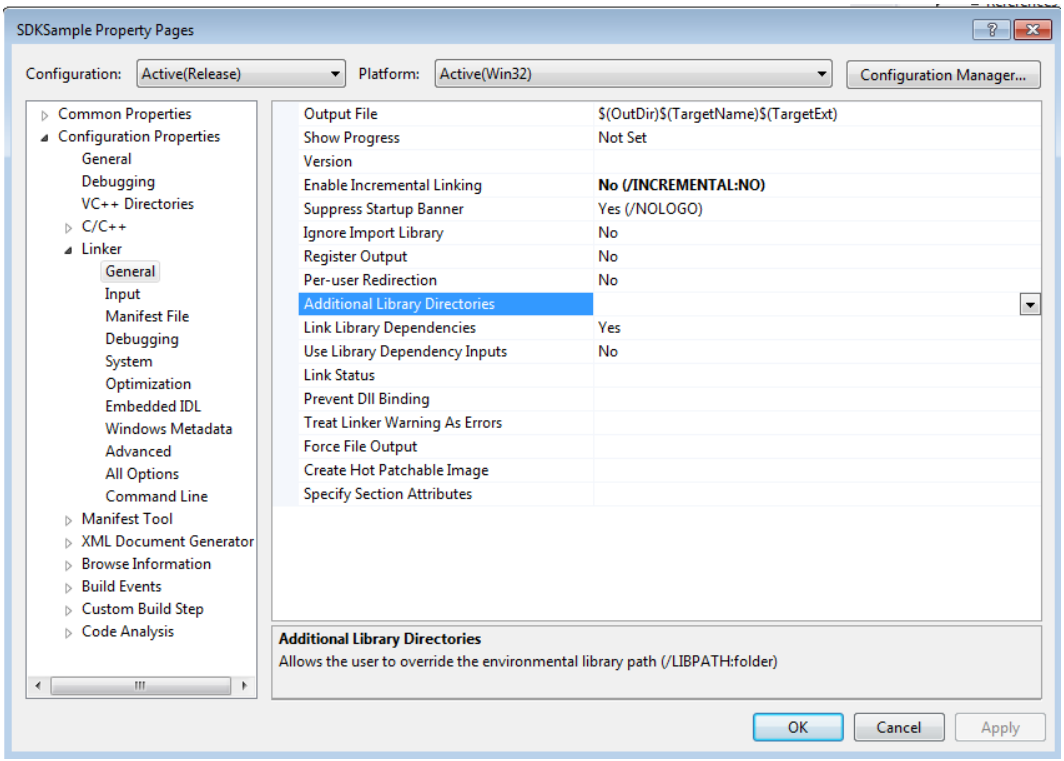
Returns:

0: Success; negative number: failed.

The following example is a reference for configurations:

1. Copy MDT_COMMAND_LIB_win32.dll to your program folder, and make sure your program folder path without any blank space string.

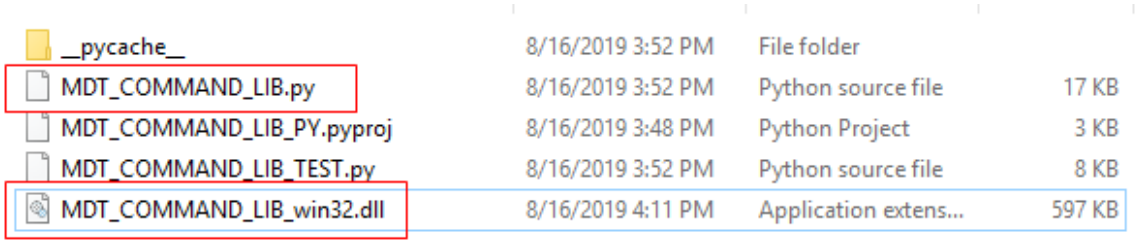
2. Set the Additional Library Directories to your folder which contains MDT_COMMAND_LIB_win32.dll, as seen below.



MDT69XB_Demo.cpp is example code files which you can also find in the sample directory. You can run it for testing.

Software Development (Python)

Python 3.6 or above is required. User can import MDT_COMMAND_LIB.py(can be found in Thorlabs_MDT69XB_PythonSDK)to your python project, that's the wrapper for MDT_COMMAND_LIB. Copy MDT_COMMAND_LIB_win32.dll(C/C++ development environment) to your program folder, and make sure the library file and MDT_COMMAND_LIB.py file are in the same folder. The " MDT_COMMAND_LIB_TEST.py" is the example code for how to use the python APIs.



User can also replace the reference win32 lib to x64 lib for 64-bit application.

MDT_COMMAND_LIB Namespace Reference

Functions

- def **mdtListDevices**
- def **mdtOpen**
- def **mdtIsOpen**
- def **mdtClose**
- def **mdtGetId**
- def **mdtGetLimtVoltage**
- def **mdtGetXAxisVoltage**
- def **mdtSetXAxisVoltage**
- def **mdtGetXAxisMinVoltage**
- def **mdtSetXAxisMinVoltage**
- def **mdtGetXAxisMaxVoltage**
- def **mdtSetXAxisMaxVoltage**
- def **mdtGetVoltageAdjustmentResolution**
- def **mdtSetVoltageAdjustmentResolution**
- def **mdtSetAllVoltage**
- def **mdtGetMasterScanEnable**
- def **mdtSetMasterScanEnable**
- def **mdtGetMasterScanVoltage**
- def **mdtSetMasterScanVoltage**
- def **mdtGetYAxisVoltage**
- def **mdtSetYAxisVoltage**
- def **mdtGetZAxisVoltage**
- def **mdtSetZAxisVoltage**
- def **mdtGetYAxisMinVoltage**
- def **mdtSetYAxisMinVoltage**
- def **mdtGetZAxisMinVoltage**
- def **mdtSetZAxisMinVoltage**
- def **mdtGetYAxisMaxVoltage**

- def **mdtSetYAxisMaxVoltage**
- def **mdtGetZAxisMaxVoltage**
- def **mdtSetZAxisMaxVoltage**
- def **mdtGetXYZAxisVoltage**
- def **mdtSetXYZAxisVoltage**

Function Documentation

def MDT_COMMAND_LIB.mdtListDevices ()

```
List all connected MDT devices
Returns:
   The mdt device list, each deice item is [serialNumber, mdtType]
```

def MDT_COMMAND_LIB.mdtOpen (*serialNo*, *nBaud*, *timeout*)

```
Open MDT device
Args:
    serialNo: serial number of MDT device
    nBaud: bit per second of port
    timeout: set timeout value in (s)
Returns:
    non-negative number: hdl number returned Successful; negative number: failed.
```

def MDT_COMMAND_LIB.mdtIsOpen (*serialNo*)

```
Check opened status of MDT device
Args:
    serialNo: serial number of MDT device
Returns:
    0: MDT device is not opened; 1: MDT device is opened.
```

def MDT_COMMAND_LIB.mdtClose (*hdl*)

```
Close opened MDT device
Args:
    hdl: the handle of opened MDT device
Returns:
    0: Success; negative number: failed.
```

def MDT_COMMAND_LIB.mdtGetId (*hdl*, *id*)

```
Get the product header and firmware version
Args:
    hdl: the handle of opened MDT device
    id: the output id string
Returns:
    0: Success; negative number: failed.
```

def MDT_COMMAND_LIB.mdtGetLimtVoltage (*hdl*, *voltage*)

```
Get output voltage limit setting.
Args:
    hdl: the handle of opened MDT device
    voltage: the output voltage
Returns:
    0: Success; negative number: failed.
```

def MDT_COMMAND_LIB.mdtGetXAxisVoltage (*hdl*, *voltage*)

```
Get the X axis output voltage.
Args:
    hdl: the handle of opened MDT device
    voltage: the output voltage
Returns:
    0: Success; negative number: failed.
```

def MDT_COMMAND_LIB.mdtSetXAxisVoltage (*hdl*, *voltage*)

```
Set the output voltage for the X axis.
Args:
    hdl: the handle of opened MDT device
    voltage: the input voltage range:(0 ~ limit voltage)
Returns:
    0: Success; negative number: failed.
```

def MDT_COMMAND_LIB.mdtGetXAxisMinVoltage (*hdl*, *voltage*)

```
Get the minimum output voltage limit for X axis.
Args:
    hdl: the handle of opened MDT device
    voltage: the output voltage
Returns:
    0: Success; negative number: failed.
```

def MDT_COMMAND_LIB.mdtSetXAxisMinVoltage (*hdl*, *voltage*)

```
Set the minimum output voltage limit for X axis.
Args:
    hdl: the handle of opened MDT device
    voltage: the input voltage range:(0 ~ limit voltage)
Returns:
    0: Success; negative number: failed.
```

def MDT_COMMAND_LIB.mdtGetXAxisMaxVoltage (*hdl*, *voltage*)

```
Get the maximum output voltage limit for X axis.
Args:
    hdl: the handle of opened MDT device
    voltage: the output voltage
Returns:
    0: Success; negative number: failed.
```

def MDT_COMMAND_LIB.mdtSetXAxisMaxVoltage (*hdl*, *voltage*)

```
Set the maximum output voltage limit for X axis.
Args:
    hdl: the handle of opened MDT device
    voltage: the input voltage range:(0 ~ limit voltage)
Returns:
    0: Success; negative number: failed.
```

def MDT_COMMAND_LIB.mdtGetVoltageAdjustmentResolution (*hdl*, *step*)

```
Get the current step resolution.
Args:
    hdl: the handle of opened MDT device
    step: current step solution
Returns:
    0: Success; negative number: failed.
```

def MDT_COMMAND_LIB.mdtSetVoltageAdjustmentResolution (*hdl*, *step*)

```
Set the step resolution when using up/down arrow keys.
Args:
    hdl: the handle of opened MDT device
    step: target step range:(1 ~ 1000)
Returns:
    0: Success; negative number: failed.
```

def MDT_COMMAND_LIB.mdtSetAllVoltage (*hdl*, *voltage*)

```
Set all outputs to desired voltage.
Args:
    hdl: the handle of opened MDT device
    voltage: the input voltage range:(0 ~ limit voltage)
Returns:
    0: Success; negative number: failed.
```

def MDT_COMMAND_LIB.mdtGetMasterScanEnable (*hdl*, *state*)

```
Get the state of the Master Scan enable.
Args:
    hdl: the handle of opened MDT device
    state: current master scan state.(1-enable,0-disable)
Returns:
    0: Success; negative number: failed.
```

def MDT_COMMAND_LIB.mdtSetMasterScanEnable (*hdl*, *state*)

```
Set Master Scan mode.
Args:
    hdl: the handle of opened MDT device
    state: current master scan state.(1-enable,0-disable)
Returns:
    0: Success; negative number: failed.
```

def MDT_COMMAND_LIB.mdtGetMasterScanVoltage (*hdl*, *voltage*)

```
Get the master scan voltage.
Args:
    hdl: the handle of opened MDT device
    voltage: the output voltage
Returns:
    0: Success; negative number: failed.
```

def MDT_COMMAND_LIB.mdtSetMasterScanVoltage (*hdl*, *voltage*)

```
Set a master scan voltage that adds to the x, y, and z axis voltages.
Args:
    hdl: the handle of opened MDT device
    voltage: the input voltage range:(0 ~ limit voltage)
Returns:
    0: Success; negative number: failed.
```

def MDT_COMMAND_LIB.mdtGetYAxisVoltage (*hdl*, *voltage*)

```
Get the Y axis output voltage.
Args:
    hdl: the handle of opened MDT device
    voltage: the output voltage
Returns:
    0: Success; negative number: failed.
```

def MDT_COMMAND_LIB.mdtSetYAxisVoltage (*hdl*, *voltage*)

```
Set the output voltage for the Y axis.
Args:
    hdl: the handle of opened MDT device
    voltage: the input voltage range:(0 ~ limit voltage)
Returns:
    0: Success; negative number: failed.
```

def MDT_COMMAND_LIB.mdtGetZAxisVoltage (*hdl*, *voltage*)

```
Get the Z axis output voltage.
Args:
    hdl: the handle of opened MDT device
    voltage: the output voltage
Returns:
    0: Success; negative number: failed.
```

def MDT_COMMAND_LIB.mdtSetZAxisVoltage (*hdl*, *voltage*)

```
Set the output voltage for the Z axis.
Args:
    hdl: the handle of opened MDT device
    voltage: the input voltage range:(0 ~ limit voltage)
```

```
Returns:
    0: Success; negative number: failed.
```

def MDT_COMMAND_LIB.mdtGetYAxisMinVoltage (*hdl*, *voltage*)

```
Get the minimum output voltage limit for Y axis.
Args:
    hdl: the handle of opened MDT device
    voltage: the output voltage
Returns:
    0: Success; negative number: failed.
```

def MDT_COMMAND_LIB.mdtSetYAxisMinVoltage (*hdl*, *voltage*)

```
Set the minimum output voltage limit for Y axis.
Args:
    hdl: the handle of opened MDT device
    voltage: the input voltage range:(0 ~ limit voltage)
Returns:
    0: Success; negative number: failed.
```

def MDT_COMMAND_LIB.mdtGetZAxisMinVoltage (*hdl*, *voltage*)

```
Get the minimum output voltage limit for Z axis.
Args:
    hdl: the handle of opened MDT device
    voltage: the output voltage
Returns:
    0: Success; negative number: failed.
```

def MDT_COMMAND_LIB.mdtSetZAxisMinVoltage (*hdl*, *voltage*)

```
Set the minimum output voltage limit for Z axis.
Args:
    hdl: the handle of opened MDT device
    voltage: the input voltage range:(0 ~ limit voltage)
Returns:
    0: Success; negative number: failed.
```

def MDT_COMMAND_LIB.mdtGetYAxisMaxVoltage (*hdl*, *voltage*)

```
Get the maximum output voltage limit for Y axis.
Args:
    hdl: the handle of opened MDT device
    voltage: the output voltage
Returns:
    0: Success; negative number: failed.
```

def MDT_COMMAND_LIB.mdtSetYAxisMaxVoltage (*hdl*, *voltage*)

```
Set the maximum output voltage limit for Y axis.
Args:
```

```
    hdl: the handle of opened MDT device
    voltage: the input voltage range:(0 ~ limit voltage)
Returns:
    0: Success; negative number: failed.
```

def MDT_COMMAND_LIB.mdtGetZAxisMaxVoltage (*hdl*, *voltage*)

```
Get the maximum output voltage limit for Z axis.
Args:
    hdl: the handle of opened MDT device
    voltage: the output voltage
Returns:
    0: Success; negative number: failed.
```

def MDT_COMMAND_LIB.mdtSetZAxisMaxVoltage (*hdl*, *voltage*)

```
Set the maximum output voltage limit for Z axis.
Args:
    hdl: the handle of opened MDT device
    voltage: the input voltage range:(0 ~ limit voltage)
Returns:
    0: Success; negative number: failed.
```

def MDT_COMMAND_LIB.mdtGetXYZAxisVoltage (*hdl*, *xyzVoltage*)

```
Get the x,y,z axis output voltages.
Args:
    hdl: the handle of opened MDT device
    xyzVoltage: the output x,y,z axis voltage
Returns:
    0: Success; negative number: failed.
```

def MDT_COMMAND_LIB.mdtSetXYZAxisVoltage (*hdl*, *xVoltage*, *yVoltage*, *zVoltage*)

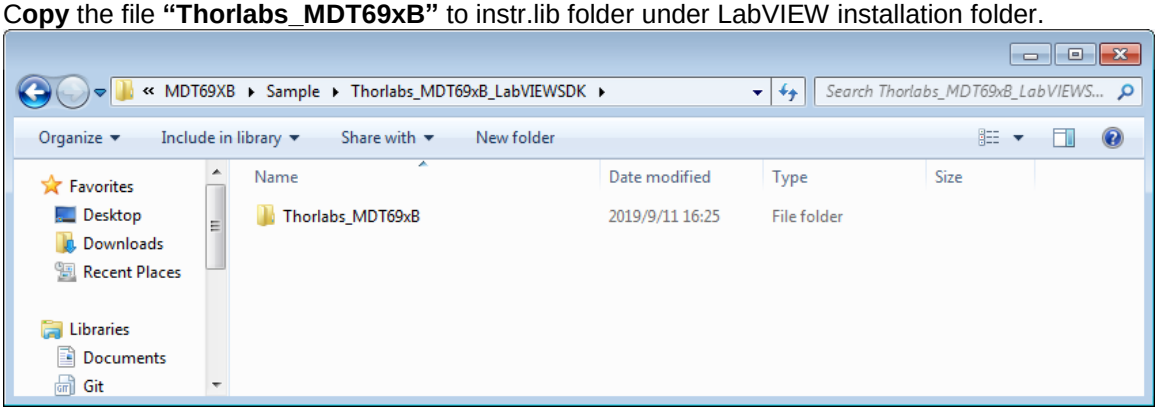
```
Set the x,y,z axis output voltages.
Args:
    hdl: the handle of opened MDT device
    xVoltage: the x axis input voltage range:(0 ~ limit voltage)
    yVoltage: the y axis input voltage range:(0 ~ limit voltage)
    zVoltage: the z axis input voltage range:(0 ~ limit voltage)
Returns:
    0: Success; negative number: failed.
```

Software Development (LabVIEW)

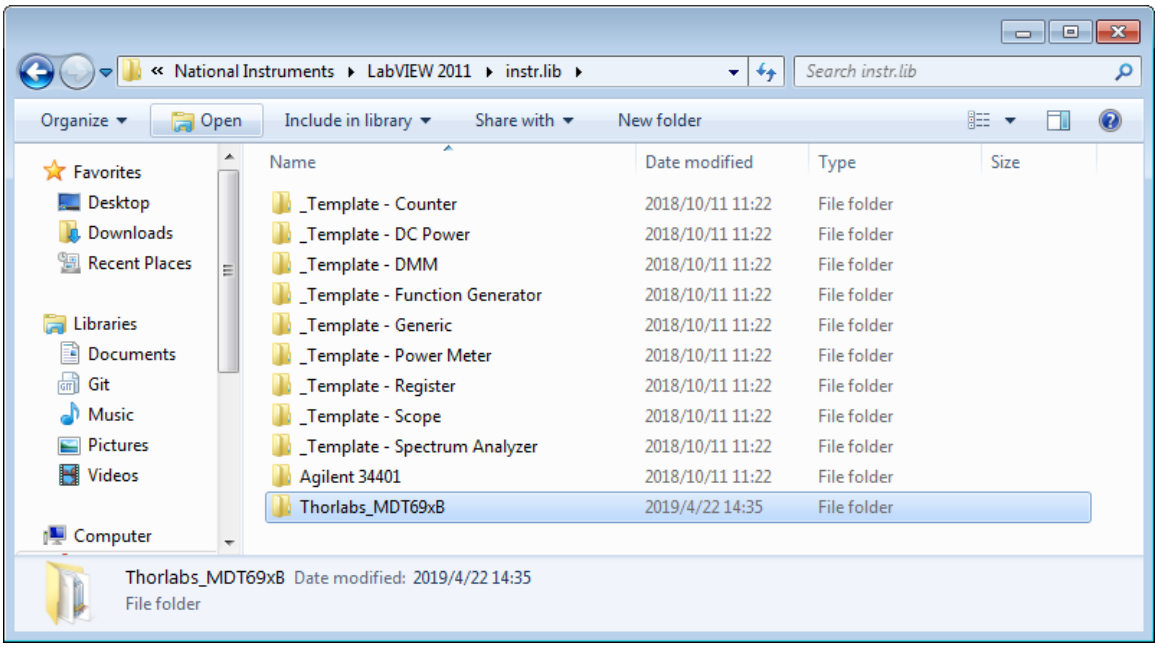
The user can start software development with LabVIEW 2011 or later versions based on LabVIEW instrument driver mechanism. The supported files are in *\LabVIEW SDK* under the **Sample** directory.

How to install

Copy the file **"Thorlabs_MDT69xB"** to instr.lib folder under LabVIEW installation folder.

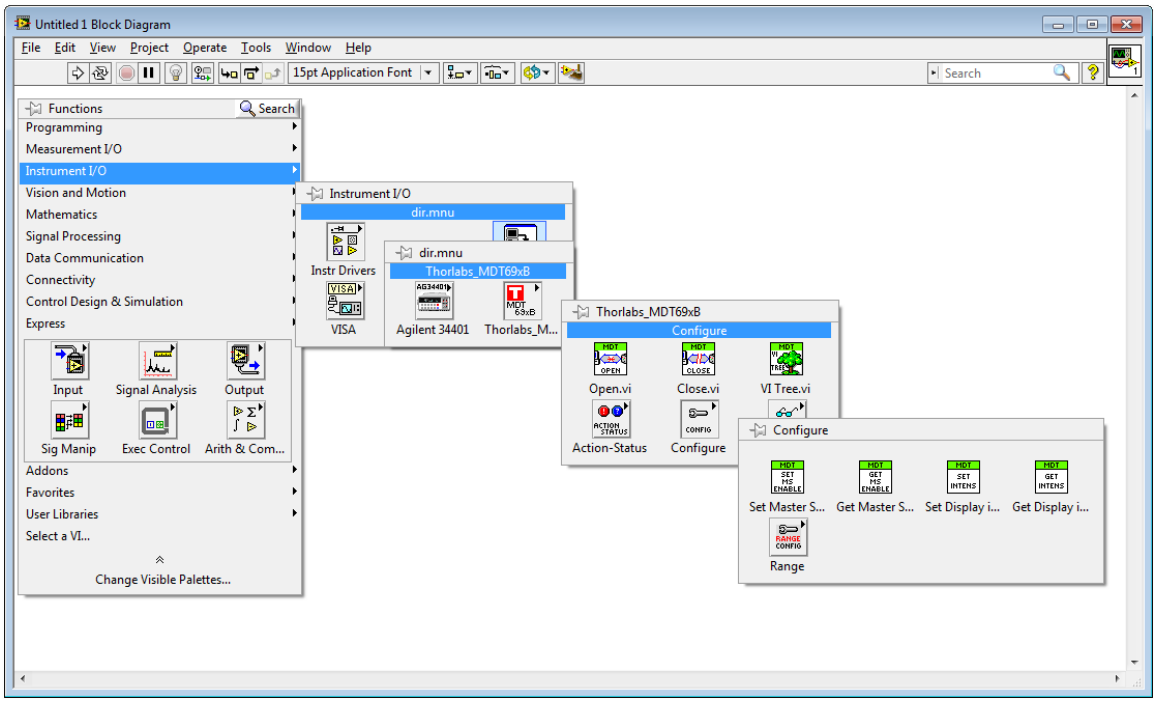


Destination folder: under %LabVIEW install path%\instr.lib
Typically, C:\Program Files (x86)\National Instruments\LabVIEW 2011\instr.lib
Note: LabVIEW 2011 or later LabVIEW versions are compatible.



How to find VI

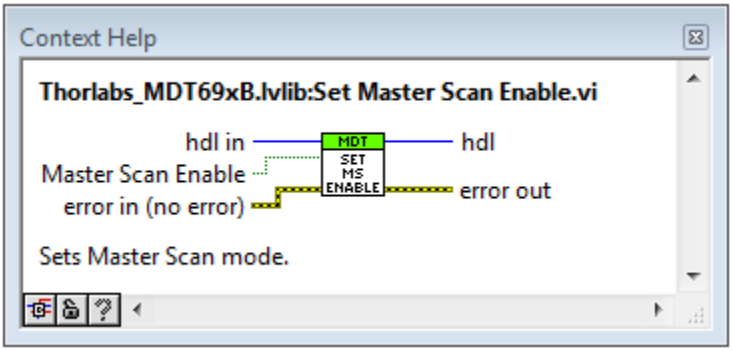
VI Could be found under: Functions\Instrument I/O\Instrument Drivers\



How to use

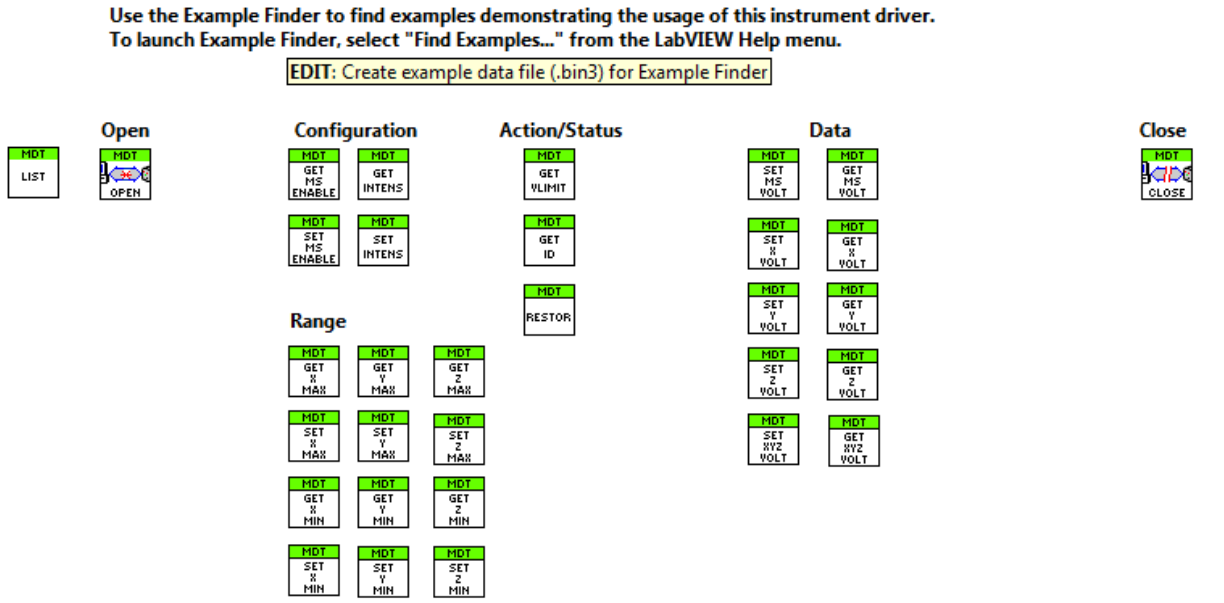
1. From VI

Note: Before you open the SDK LabVIEW project, make sure the device has been connected to the computer.



2. From VI tree

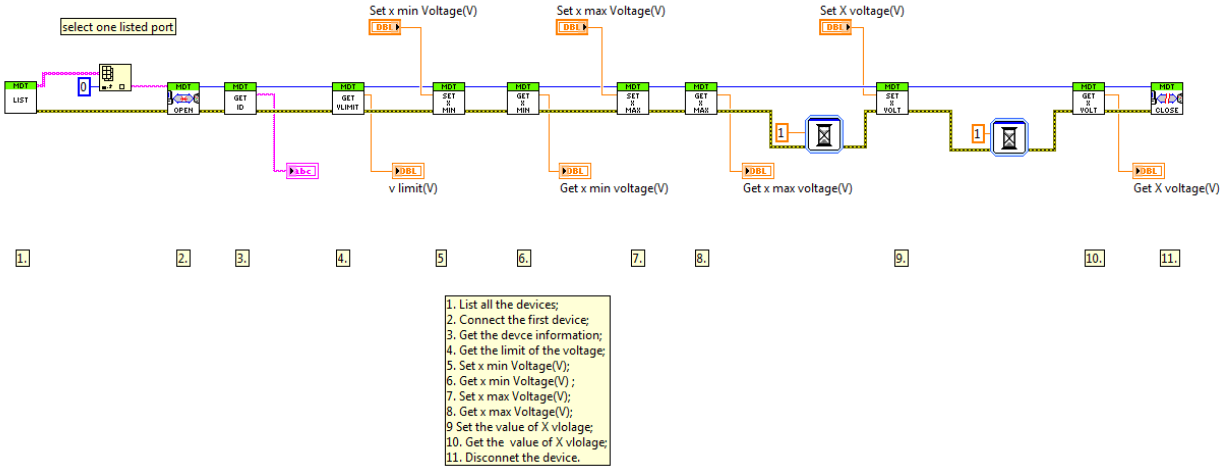
Some classic data flow in VI tree.



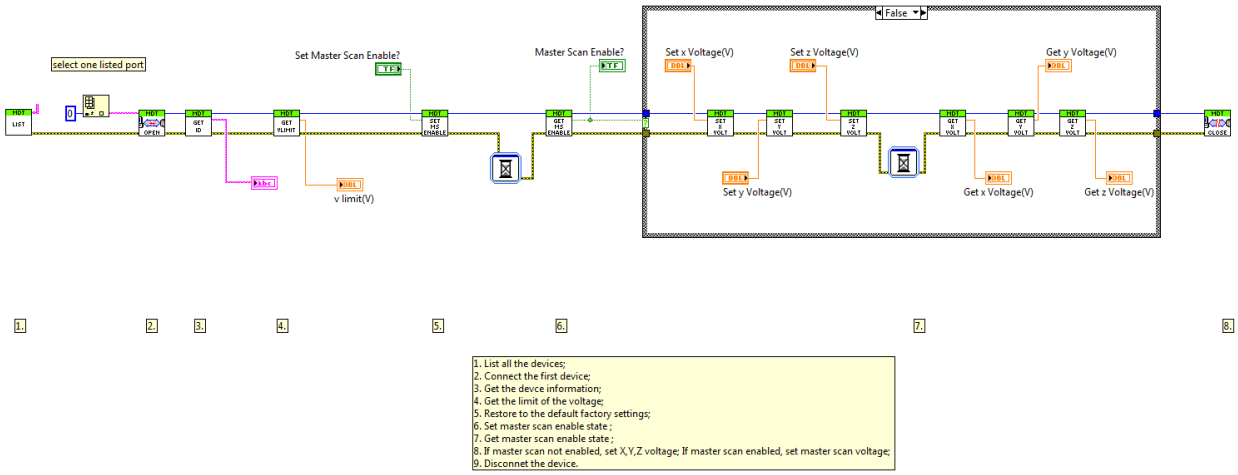
3. From example

An examples show the classic usage. Example path: instr.lib\Thorlabs_MDT69xB\Examples

(1) For single channel Open-Loop Piezo Controller



(2) For 3-channel Open-Loop Piezo Controller



Easy programming and detailed comment will help.