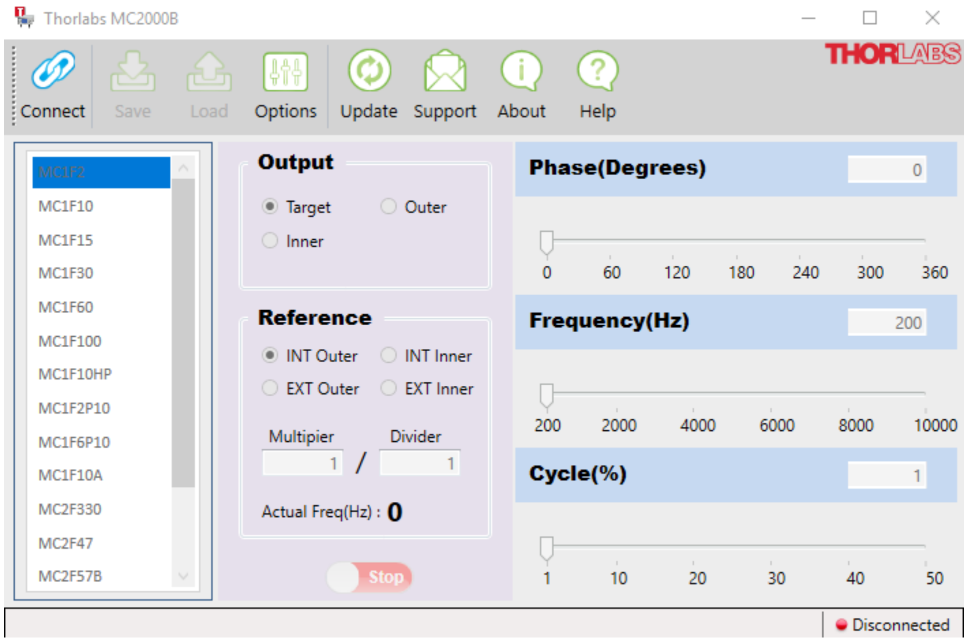
MC2000B®
Optical Chopper

SDK Manual



Output: (● Target) ○ Outer ○ Inner

Reference: (● INT Outer) ○ INT Inner ○ EXT Outer ○ EXT Inner

Multipier 1 / Divider 1

Actual Freq(Hz) : 0

Phase(Degrees) 0

Frequency(Hz) 200

Cycle(%) 1

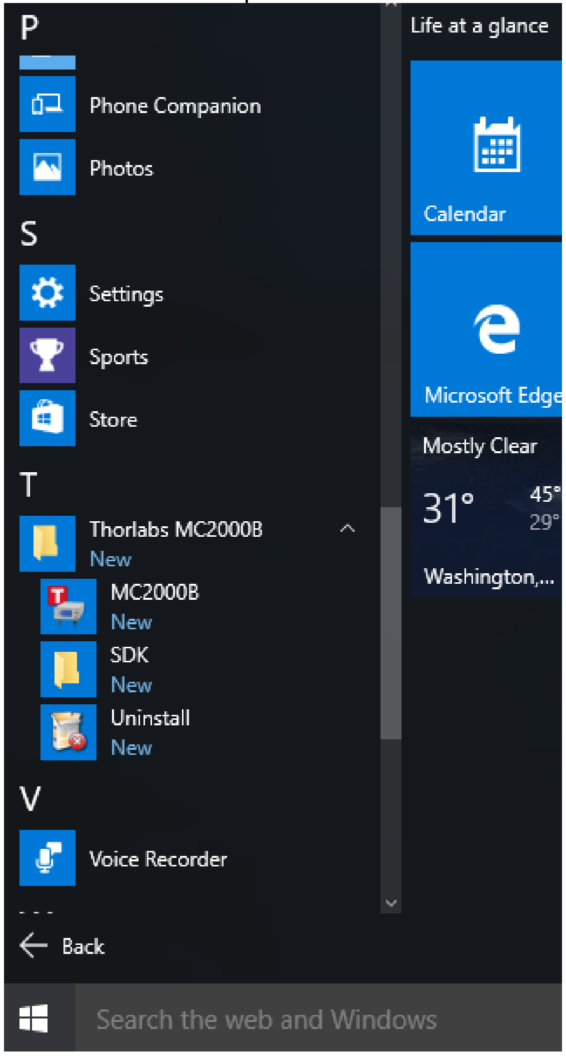
Disconnected

Table of Contents

Chapter 1 Introduction

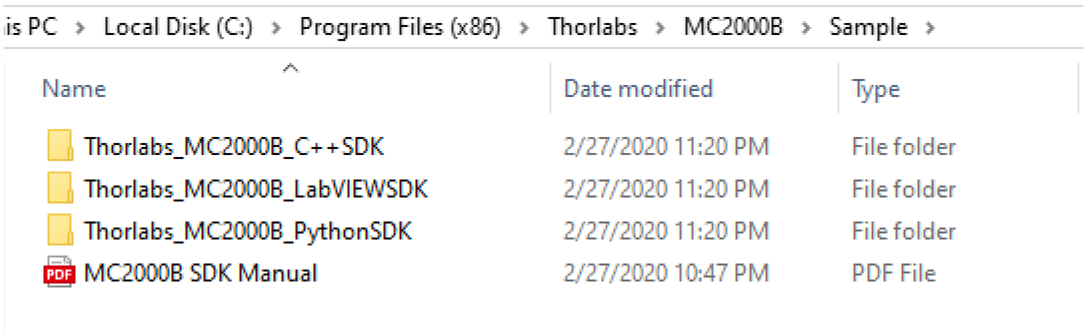
User can start software development in C/C++ develop environment, LabVIEW and Python.

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



or by clicking *Help* in software menu.

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



is PC › Local Disk (C:) › Program Files (x86) › Thorlabs › MC2000B › Sample ›

Name	Date modified	Type
Thorlabs_MC2000B_C++SDK	2/27/2020 11:20 PM	File folder
Thorlabs_MC2000B_LabVIEWSDK	2/27/2020 11:20 PM	File folder
Thorlabs_MC2000B_PythonSDK	2/27/2020 11:20 PM	File folder
MC2000B SDK Manual	2/27/2020 10:47 PM	PDF File

Chapter 2 C++ Software Development Kit

User can start software development with **MC2000CommandLibWin32.dll** or **MC2000CommandLibWin64.dll** in C/C++ development environment which can be found in **Thorlabs_MC2000B_C++SDK** under \Document directory. The corresponding header file is also in **Thorlabs_MC2000B_C++SDK** under \Document directory.

Copy **MC2000CommandLibWin32.dll** or **MC2000CommandLibWin64.dll** to your program folder, and make sure the library file and exe file are in the same folder.

Below is the description of the header file **MC2000CommandLib.h**

2.1. cmd_library.h File Reference

2.1.1. Functions

- **MC2000COMMANDLIB_API** int **List** (char *sn)

list all the possible port on this computer.

- **MC2000COMMANDLIB_API** int **Open** (char *serialNo, int nBaud, int timeout)

open port function.

- **MC2000COMMANDLIB_API** int **IsOpen** (char *serialNo)

check opened status of port

- **MC2000COMMANDLIB_API** int **Close** (int hdl)

close current opened port

- **MC2000COMMANDLIB_API** int **Read** (int hdl, char *b, int limit)
- **MC2000COMMANDLIB_API** int **Write** (int hdl, char *b, int size)
- **MC2000COMMANDLIB_API** int **GetId** (int hdl, char *id)

Get the model number and firmware version.

- **MC2000COMMANDLIB_API** int **Set** (int hdl, char *c, int var)
- **MC2000COMMANDLIB_API** int **Get** (int hdl, char *c, char *d)
- **MC2000COMMANDLIB_API** int **SetTimeout** (int hdl, int time)

set time out value for read or write process.

- **MC2000COMMANDLIB_API** int **Purge** (int hdl, int flag)

Purge the RX and TX buffer on port.

- **MC2000COMMANDLIB_API int SetFrequency (int hdl, int frequency)**

Set the desired internal reference frequency.

- **MC2000COMMANDLIB_API int GetFrequency (int hdl, int *frequency)**

Get the internal reference frequency.

- **MC2000COMMANDLIB_API int SetBladeType (int hdl, int type)**

Set the blade type.

- **MC2000COMMANDLIB_API** int **GetBladeType** (int hdl, int *type)

Get the blade type.

- **MC2000COMMANDLIB_API** int **SetHarmonicMultiplier** (int hdl, int nharmonic)

Set Harmonic Multiplier applied to external reference frequency.

- **MC2000COMMANDLIB_API** int **GetHarmonicMultiplier** (int hdl, int *nharmonic)

Get Harmonic Multiplier applied to external reference frequency.

- **MC2000COMMANDLIB_API** int **SetHarmonicDivider** (int hdl, int dharmonic)

Set the Harmonic Divider applied to external reference frequency.

- **MC2000COMMANDLIB_API** int **GetHarmonicDivider** (int hdl, int *dharmonic)

Get the Harmonic Divider applied to external reference frequency.

- **MC2000COMMANDLIB_API** int **SetPhase** (int hdl, int phase)

Set the Phase adjust.

- **MC2000COMMANDLIB_API** int **GetPhase** (int hdl, int *phase)

Get the Phase adjust.

- **MC2000COMMANDLIB_API** int **SetEnable** (int hdl, int enable)

Set enable or disable state of device.

- **MC2000COMMANDLIB_API** int **GetEnable** (int hdl, int *enable)

Get enable or disable state of device.

- **MC2000COMMANDLIB_API** int **SetReference** (int hdl, int ref)

Set the reference mode.

- **MC2000COMMANDLIB_API** int **GetReference** (int hdl, int *ref)

Get the reference mode.

- **MC2000COMMANDLIB_API** int **SetReferenceOutput** (int hdl, int output)

Set the output reference mode.

- **MC2000COMMANDLIB_API** int **GetReferenceOutput** (int hdl, int *output)

Get the output reference mode.

- **MC2000COMMANDLIB_API** int **SetDisplayIntensity** (int hdl, int intensity)

Set the display intensity (1-10).

- **MC2000COMMANDLIB_API** int **GetDisplayIntensity** (int hdl, int *intensity)

Get the display intensity (1-10).

- **MC2000COMMANDLIB_API** int **GetReferenceFrequency** (int hdl, int *frequence)

Get the current supplied external reference frequency.

- **MC2000COMMANDLIB_API** int **Restore** (int hdl)

Restore the factory default parameters.

- **MC2000COMMANDLIB_API** int **GetVerbose** (int hdl, int *verbose)

Get the verbose mode.

- **MC2000COMMANDLIB_API** int **SetVerbose** (int hdl, int verbose)

Set the verbose mode.

- **MC2000COMMANDLIB_API** int **SetLanguage** (int hdl, int lang)

Set display language in the device.

- **MC2000COMMANDLIB_API** int **GetLocked** (int hdl, int *lock)

Get the lock state.

- **MC2000COMMANDLIB_API** int **GetCycleAdjust** (int hdl, int *cycle)

Get the cycle adjustment.

- **MC2000COMMANDLIB_API** int **SetCycleAdjust** (int hdl, int cycle)

Set the cycle adjustment.

- **MC2000COMMANDLIB_API int GetReferenceOutputFrequency (int hdl, int * output)**

Gets actual frequency blade spinning.

2.1.2. Function Documentation

MC2000COMMANDLIB_API int Close (int *hdl*)

Close current opened port

Parameters:

hdl	Handle of port.

Returns:

0: success; negative number: failed.

MC2000COMMANDLIB_API int Get (int *hdl*, 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;
0xEA: CMD_NOT_DEFINED;
0xEB: time out;
0xED: invalid string buffer;

MC2000COMMANDLIB_API int GetBladeType (int *hdl*, int * *type*)

Get the blade type.

Parameters:

hdl	Handle of port.
type	Pointer to the blade type.

Returns:

Negative number: failed.
0: success; negative number: failed.
-1: invalid string buffer;
-2: time out;
-3: time out;
-4: CMD_NOT_DEFINED;

MC2000COMMANDLIB_API int GetCycleAdjust (int *hdl*, int * *cycle*)

Get the cycle adjustment.

Parameters:

hdl	Handle of port.
verbose	Pointer to the cycle adjustment

Returns:

Negative number: failed.
0: success; negative number: failed.
-1: invalid string buffer;
-2: time out;
-3: time out;
-4: CMD_NOT_DEFINED;

MC2000COMMANDLIB_API int GetDisplayIntensity (int *hdl*, int * *intensity*)

Get the display intensity (1-10).

Parameters:

hdl	Handle of port.
intensity	Pointer to display intensity (1-10).

Returns:

Negative number: failed.
0: success; negative number: failed.
-1: invalid string buffer;
-2: time out;
-3: time out;
-4: CMD_NOT_DEFINED;

MC2000COMMANDLIB_API int GetEnable (int *hdl*, int * *enable*)

Get enable or disable state of device.

Parameters:

hdl	Handle of port.
enable	Pointer to enable state 1: Enable 0: disable.

Returns:

Negative number: failed.
0: success; negative number: failed.
-1: invalid string buffer;
-2: time out;
-3: time out;
-4: CMD_NOT_DEFINED;

MC2000COMMANDLIB_API int GetFrequency (int *hdl*, int * *frequency*)

Get the internal reference frequency.

Parameters:

hdl	Handle of port.

Returns:

Negative number: failed.
0: success; negative number: failed.
-1: invalid string buffer;
-2: time out;
-3: time out;
-4: CMD_NOT_DEFINED;

MC2000COMMANDLIB_API int GetHarmonicDivider (int *hdl*, int * *dharmonic*)

Get the Harmonic Divider applied to external reference frequency.

Parameters:

hdl	Handle of port.
dharmonic	Pointer to the harmonic divider applied to external reference frequency.

Returns:

Negative number: failed.
0: success; negative number: failed.
-1: invalid string buffer;
-2: time out;
-3: time out;
-4: CMD_NOT_DEFINED;

MC2000COMMANDLIB_API int GetHarmonicMultiplier (int *hdl*, int * *nharmonic*)

Get Harmonic Multiplier applied to external reference frequency.

Parameters:

hdl	Handle of port.
nharmonic	Pointer to the harmonic multiplier.

Returns:

Negative number: failed.
0: success; negative number: failed.
-1: invalid string buffer;
-2: time out;
-3: time out;
-4: CMD_NOT_DEFINED;

MC2000COMMANDLIB_API int GetId (int *hdl*, char * *id*)

Get the model number and firmware version.

Parameters:

hdl	Handle of port.
id	Pointer to id string buffer.

Returns:

0: success; negative number: failed.
-1: invalid string buffer;
-2: time out;
-3: time out;
-4: CMD_NOT_DEFINED;

MC2000COMMANDLIB_API int GetLocked (int *hdl*, int * *lock*)

Get the lock state.

Parameters:

hdl	Handle of port.
verbose	pointer to the lock state

Returns:

Negative number: failed.
0: success; negative number: failed.
-1: invalid string buffer;
-2: time out;
-3: time out;
-4: CMD_NOT_DEFINED;

MC2000COMMANDLIB_API int GetPhase (int *hdl*, int * *phase*)

Get the Phase adjust.

Parameters:

hdl	Handle of port.
phase	oointer to the phase adjust.

Returns:

Negative number: failed.
0: success; negative number: failed.
-1: invalid string buffer;
-2: time out;
-3: time out;
-4: CMD_NOT_DEFINED;

MC2000COMMANDLIB_API int GetReference (int *hdl*, int * *ref*)

Get the reference mode.

Parameters:

hdl	Handle of port.
ref	Pointer to reference mode.

Returns:

Negative number: failed.
0: success; negative number: failed.
-1: invalid string buffer;
-2: time out;
-3: time out;
-4: CMD_NOT_DEFINED;

MC2000COMMANDLIB_API int GetReferenceFrequency (int *hdl*, int * *frequence*)

Get the current supplied external reference frequency.

Parameters:

hdl	Handle of port.
frequence	Pointer to the current supplied external reference frequency.

Returns:

Negative number: failed.
0: success; negative number: failed.
-1: invalid string buffer;
-2: time out;
-3: time out;
-4: CMD_NOT_DEFINED;

MC2000COMMANDLIB_API int GetReferenceOutputFrequency (int *hdl*, int * *freq*)

Get actual frequency blade spinning

Parameters:

hdl	Handle of port.
freq	Pointer to the actual frequency.

Returns:

Negative number: failed.
0: success; negative number: failed.
-1: invalid string buffer;
-2: time out;
-3: time out;
-4: CMD_NOT_DEFINED;

MC2000COMMANDLIB_API int GetReferenceOutput (int *hdl*, int * *output*)

Get the output reference mode.

Parameters:

hdl	Handle of port.
output	Pointer to the output mode.

Returns:

Negative number: failed.
0: success; negative number: failed.
-1: invalid string buffer;
-2: time out;
-3: time out;

-4: CMD_NOT_DEFINED;

MC2000COMMANDLIB_API int GetVerbose (int *hdl*, int * *verbose*)

Get the verbose mode.

Parameters:

hdl	Handle of port.
verbose	verbose mode

Returns:

Negative number: failed.
0: success; negative number: failed.
-1: invalid string buffer;
-2: time out;
-3: time out;
-4: CMD_NOT_DEFINED;

MC2000COMMANDLIB_API int IsOpen (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.

MC2000COMMANDLIB_API int List (char * *sn*)

List all the possible port on this computer.

Parameters:

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

MC2000COMMANDLIB_API int Open (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.

MC2000COMMANDLIB_API int Purge (int *hdl*, int *flag*)

Purge the RX and TX buffer on port.

Parameters:

hdl	Handle of port.
flag	

FT_PURGE_RX: 0x01

FT_PURGE_TX: 0x02

Returns:

0: success; negative number: failed.

MC2000COMMANDLIB_API int Read (int *hdl*, 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.

MC2000COMMANDLIB_API int Restore (int *hdl*)

Restore the factory default parameters.

Parameters:

hdl	Handle of port.

Returns:

Negative number: failed.
0: success; negative number: failed.
-1: invalid string buffer;
-2: time out;
-3: time out;
-4: CMD_NOT_DEFINED;

MC2000COMMANDLIB_API int Set (int *hdl*, char * *c*, int *var*)

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
var	length of input command string (<255)

Returns:

0: success;
0xEA: CMD_NOT_DEFINED;
0xEB: time out;
0xED: invalid string buffer;

MC2000COMMANDLIB_API int SetBladeType (int *hdl*, int *type*)

Set the blade type.

Parameters:

hdl	Handle of port.
type	Desired blade type.

Returns:

Negative number: failed.
0: success; negative number: failed.
-1: invalid string buffer;
-2: time out;

-3: time out;
-4: CMD_NOT_DEFINED;

MC2000COMMANDLIB_API int SetCycleAdjust (int *hdl*, int *cycle*)

Set the cycle adjustment.

Parameters:

hdl	Handle of port.
verbose	the desired cycle adjustment

Returns:

Negative number: failed.
0: success; negative number: failed.
-1: invalid string buffer;
-2: time out;
-3: time out;
-4: CMD_NOT_DEFINED;

MC2000COMMANDLIB_API int SetDisplayIntensity (int *hdl*, int *intensity*)

Set the display intensity (1-10).

Parameters:

hdl	Handle of port.
intensity	Display intensity (1-10).

Returns:

Negative number: failed.
0: success; negative number: failed.
-1: invalid string buffer;
-2: time out;
-3: time out;
-4: CMD_NOT_DEFINED;

MC2000COMMANDLIB_API int SetEnable (int *hdl*, int *enable*)

Set enable or disable state of device.

Parameters:

hdl	Handle of port.
enable	Enable state 1: enable, 0: disable.

Returns:

Negative number: failed.
0: success; negative number: failed.
-1: invalid string buffer;
-2: time out;
-3: time out;
-4: CMD_NOT_DEFINED;

MC2000COMMANDLIB_API int SetFrequency (int *hdl*, int *frequency*)

Set the desired internal reference frequency.

Parameters:

hdl	Handle of port.
id	Desired internal reference frequency.

Returns:

Negative number: failed.
0: success; negative number: failed.
-1: invalid string buffer;

-2: time out;
-3: time out;
-4: CMD_NOT_DEFINED;

MC2000COMMANDLIB_API int SetHarmonicDivider (int *hdl*, int *dharmonic*)

Set the Harmonic Divider applied to external reference frequency.

Parameters:

hdl	Handle of port.
dharmonic	Desired harmonic divider applied to external reference frequency.

Returns:

Negative number: failed.
0: success; negative number: failed.
-1: invalid string buffer;
-2: time out;
-3: time out;
-4: CMD_NOT_DEFINED;

MC2000COMMANDLIB_API int SetHarmonicMultiplier (int *hdl*, int *nharmonic*)

Set Harmonic Multiplier applied to external reference frequency.

Parameters:

hdl	Handle of port.
nharmonic	Desired harmonic multiplier.

Returns:

Negative number: failed.
0: success; negative number: failed.
-1: invalid string buffer;
-2: time out;
-3: time out;
-4: CMD_NOT_DEFINED;

MC2000COMMANDLIB_API int SetLanguage (int *hdl*, int *lang*)

Set display language in the device.

Parameters:

hdl	Handle of port.
lang	Language mode. 1: English, 0: Chinese

Returns:

Negative number: failed.
0: success; negative number: failed.
-1: invalid string buffer;
-2: time out;
-3: time out;
-4: CMD_NOT_DEFINED;

MC2000COMMANDLIB_API int SetPhase (int *hdl*, int *phase*)

Set the Phase adjust.

Parameters:

hdl	Handle of port.
phase	Desired phase adjust.

Returns:

Negative number: failed.
0: success; negative number: failed.

-1: invalid string buffer;
-2: time out;
-3: time out;
-4: CMD_NOT_DEFINED;

MC2000COMMANDLIB_API int SetReference (int *hdl*, int *ref*)

Set the reference mode.

Parameters:

hdl	Handle of port.
ref	Reference mode.

Returns:

Negative number: failed.
0: success; negative number: failed.
-1: invalid string buffer;
-2: time out;
-3: time out;
-4: CMD_NOT_DEFINED;

MC2000COMMANDLIB_API int SetReferenceOutput (int *hdl*, int *output*)

Set the output reference mode.

Parameters:

hdl	Handle of port.
output	Output mode.

Returns:

Negative number: failed.
0: success; negative number: failed.
-1: invalid string buffer;
-2: time out;
-3: time out;
-4: CMD_NOT_DEFINED;

MC2000COMMANDLIB_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

MC2000COMMANDLIB_API int SetVerbose (int *hdl*, int *verbose*)

Set the verbose mode.

Parameters:

hdl	Handle of port.
verbose	Verbose mode. When verbose mode is set to 1, status messages are output on the USB

Returns:

Negative number: failed.
0: success; negative number: failed.
-1: invalid string buffer;
-2: time out;
-3: time out;
-4: CMD_NOT_DEFINED;

MC2000COMMANDLIB_API int Write (int *hdl*, char * *b*, int *size*)

Write string to device through opened port.

Make sure the port was opened successful before call this function.

Parameters:

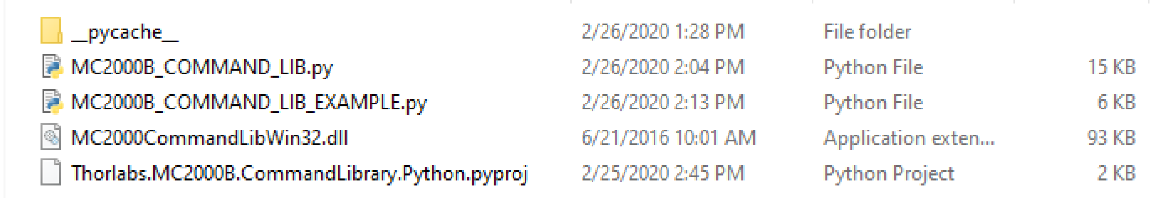
hdl	Handle of port.
b	input string
size	Size of string to be written.

Returns:

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

Chapter 3 Python Software Development Kit

Python 3.6 or above is required. User can import **MC2000B_COMMAND_LIB.py** to your python project, that's the wrapper for **MC2000B _COMMAND_LIB**(**MC2000BCommandLibraryWin32.dll** in C/C++ development environment). Copy **MC2000BCommandLibraryWin32.dll** to your program folder, and make sure the library file and **MC2000B_COMMAND_LIB.py** file are in the same folder. The " **MC2000B _COMMAND_LIB_EXAMPLE.py**" is example code for how to use the python APIs.



__pycache__	2/26/2020 1:28 PM	File folder	
MC2000B_COMMAND_LIB.py	2/26/2020 2:04 PM	Python File	15 KB
MC2000B_COMMAND_LIB_EXAMPLE.py	2/26/2020 2:13 PM	Python File	6 KB
MC2000CommandLibWin32.dll	6/21/2016 10:01 AM	Application exten...	93 KB
Thorlabs.MC2000B.CommandLibrary.Python.pyproj	2/25/2020 2:45 PM	Python Project	2 KB

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

3.1. LK220_COMMAND_LIB Namespace Reference

3.1.1. Functions

0)def **MC2000BListDevices**
1)def **MC2000BOpen**
2)def **MC2000BIsOpen**
3)def **MC2000BClose**
4)def **MC2000BSetFrequency**
5)def **MC2000BGetFrequency**
6)def **MC2000BSetBladeType**
7)def **MC2000BGetBladeType**
8)def **MC2000BSetHarmonicMultiplier**
9)def **MC2000BGetHarmonicMultiplier**
10)def **MC2000BSetHarmonicDivider**
11)def **MC2000BGetHarmonicDivider**
12)def **MC2000BSetPhase**
13)def **MC2000BGetPhase**
14)def **MC2000BSetEnable**
15)def **MC2000BGetEnable**
16)def **MC2000BSetReference**
17)def **MC2000BGetReference**
18)def **MC2000BSetReferenceOutput**
19)def **MC2000BGetReferenceOutput**
20)def **MC2000BSetDisplayIntensity**
21)def **MC2000BGetDisplayIntensity**
22)def **MC2000BGetReferenceFrequency**
23)def **MC2000BRestore**
24)def **MC2000BGetVerbose**
25)def **MC2000BSetVerbose**
26)def **MC2000BSetLanguage**
27)def **MC2000BGetLocked**
28)def **MC2000BGetCycleAdjust**
29)def **MC2000BSetCycleAdjust**
30)def **MC2000BGetReferenceOutFrequency**

3.1.2. Function Documentation

def MC2000B_COMMAND_LIB.MC2000BClose* (hdl*)

```
Close opened MC2000B device
Args:
    hdl: the handle of opened MC2000B device
Returns:
    0: Success; negative number: failed.
```

def MC2000B_COMMAND_LIB.MC2000BGetBladeType* (hdl, bladetype*)

```
Get the blade type
Args:
    hdl: the handle of opened MC2000B device
    bladetype: pointer to the blade type
Returns:
    0: Success; negative number: failed.
```

def MC2000B_COMMAND_LIB.MC2000BGetCycleAdjust* (hdl, cycle*)

```
Get the cycle adjustment.
Args:
    hdl: the handle of opened MC2000B device
    cycle: pointer to the cycle adjustment
Returns:
    0: Success; negative number: failed.
```

def MC2000B_COMMAND_LIB.MC2000BGetDisplayIntensity* (hdl, intensity*)

```
Get the display intensity (1-10).
Args:
    hdl: the handle of opened MC2000B device
    intensity: pointer to display intensity (1-10)
Returns:
    0: Success; negative number: failed.
```

def MC2000B_COMMAND_LIB.MC2000BGetEnable* (hdl, enable*)

```
Get enable or disable state of device
Args:
    hdl: the handle of opened MC2000B device
    enable: pointer to enable state 1:enable, 0:disable
Returns:
    0: Success; negative number: failed.
```

def MC2000B_COMMAND_LIB.MC2000BGetFrequency* (hdl, frequency*)

```
Get the internal reference frequency
Args:
    hdl: the handle of opened MC2000B device
    frequency:get internal reference frequency
```

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

def MC2000B_COMMAND_LIB.MC2000BGetHarmonicDivider (**hdl, dharmonic** *)*

```
Get the Harmonic Divider applied to external reference frequency
Args:
    hdl: the handle of opened MC2000B device
    dharmonic: pointer to the harmonic divider applied to external reference frequency
Returns:
    0: Success; negative number: failed.
```

def MC2000B_COMMAND_LIB.MC2000BGetHarmonicMultiplier (**hdl, nharmonic** *)*

```
Get Harmonic Multiplier applied to external reference frequency
Args:
    hdl: the handle of opened MC2000B device
    nharmonic: pointer to the harmonic multiplier
Returns:
    0: Success; negative number: failed.
```

def MC2000B_COMMAND_LIB.MC2000BGetLocked (**hdl, lock** *)*

```
Get the lock state.
Args:
    hdl: the handle of opened MC2000B device
    lock: pointer to the lock state
Returns:
    0: Success; negative number: failed.
```

def MC2000B_COMMAND_LIB.MC2000BGetPhase (**hdl, phase** *)*

```
Get the Phase adjust
Args:
    hdl: the handle of opened MC2000B device
    phase: pointer to the phase adjust
Returns:
    0: Success; negative number: failed.
```

def MC2000B_COMMAND_LIB.MC2000BGetReference (**hdl, ref** *)*

```
Get the reference mode
Args:
    hdl: the handle of opened MC2000B device
    ref: pointer to reference mode 1:external, 0:internal
                    reference mode high precision 0:InternalOuter 1:InternalInner 2:ExternalOuter
3:ExternalInner
Returns:
    0: Success; negative number: failed.
```

def MC2000B_COMMAND_LIB.MC2000BGetReferenceFrequency (**hdl, frequence** *)*

```
Get the current supplied external reference frequency.
Args:
    hdl: the handle of opened MC2000B device
    frequence: pointer to the current supplied external reference frequency.
Returns:
    0: Success; negative number: failed.
```

def MC2000B_COMMAND_LIB.MC2000BGetReferenceOutFrequency (hdl, freq*)*

```
Get actual frequency of blade spinning
Args:
    hdl: the handle of opened MC2000B device
    freq: pointer to the actual frequency
Returns:
    0: Success; negative number: failed.
```

def MC2000B_COMMAND_LIB.MC2000BGetReferenceOutput (hdl, output*)*

```
Get the output reference mode
Args:
    hdl: the handle of opened MC2000B device
    output: pointer to the output mode 1:actual, 0:target
Returns:
    0: Success; negative number: failed.
```

def MC2000B_COMMAND_LIB.MC2000BGetVerbose (hdl, verbose*)*

```
Get the verbose mode.
Args:
    hdl: the handle of opened MC2000B device
    verbose: verbose mode
Returns:
    0: Success; negative number: failed.
```

def MC2000B_COMMAND_LIB.MC2000BIsOpen (serialNo*)*

```
Check opened status of MC2000B device
Args:
    serialNo: serial number of MC2000B device
Returns:
    0: MC2000B device is not opened; 1: MC2000B device is opened.
```

def MC2000B_COMMAND_LIB.MC2000BListDevices ()

```
List all connected MC2000B devices
Returns:
   The MC2000B device list, each deice item is [serialNumber, MC2000BType]
```

def MC2000B_COMMAND_LIB.MC2000BOpen (serialNo, nBaud, timeout*)*

```
Open MC2000B device
```

```
Args:
    serialNo: serial number of MC2000B 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 MC2000B_COMMAND_LIB.MC2000BRestore (**hdl***)*

```
Restore the factory default parameters.
Args:
    hdl: the handle of opened MC2000B device
Returns:
    0: Success; negative number: failed.
```

def MC2000B_COMMAND_LIB.MC2000BSetBladeType (**hdl, bladetype***)*

```
Set the blade type
Args:
    hdl: the handle of opened MC2000B device
    bladetype: desired blade type
Returns:
    0: Success; negative number: failed.
```

def MC2000B_COMMAND_LIB.MC2000BSetCycleAdjust (**hdl, cycle***)*

```
Set the cycle adjustment.
Args:
    hdl: the handle of opened MC2000B device
    cycle: the desired cycle adjustment
Returns:
    0: Success; negative number: failed.
```

def MC2000B_COMMAND_LIB.MC2000BSetDisplayIntensity (**hdl, intensity***)*

```
Set the display intensity (1-10).
Args:
    hdl: the handle of opened MC2000B device
    intensity: display intensity (1-10)
Returns:
    0: Success; negative number: failed.
```

def MC2000B_COMMAND_LIB.MC2000BSetEnable (**hdl, enable***)*

```
Set enable or disable state of device
Args:
    hdl: the handle of opened MC2000B device
    enable: enable state 1:enable, 0:disable.
Returns:
    0: Success; negative number: failed.
```

def MC2000B_COMMAND_LIB.MC2000BSetFrequency (**hdl, frequency** *)*

```
Set the desired internal reference frequency.
Args:
    hdl: the handle of opened MC2000B device
    frequency:desired internal reference frequency
Returns:
    0: Success; negative number: failed.
```

def MC2000B_COMMAND_LIB.MC2000BSetHarmonicDivider (**hdl, dharmonic** *)*

```
Set the Harmonic Divider applied to external reference frequency
Args:
    hdl: the handle of opened MC2000B device
    dharmonic: desired harmonic divider applied to external reference frequency
Returns:
    0: Success; negative number: failed.
```

def MC2000B_COMMAND_LIB.MC2000BSetHarmonicMultiplier (**hdl, nharmonic** *)*

```
Set Harmonic Multiplier applied to external reference frequency
Args:
    hdl: the handle of opened MC2000B device
    nharmonic: desired harmonic multiplier
Returns:
    0: Success; negative number: failed.
```

def MC2000B_COMMAND_LIB.MC2000BSetLanguage (**hdl, lang** *)*

```
Set display language in the device.
Args:
    hdl: the handle of opened MC2000B device
    lang: language mode. 1: English, 0: Chinese
Returns:
    0: Success; negative number: failed.
```

def MC2000B_COMMAND_LIB.MC2000BSetPhase (**hdl, phase** *)*

```
Set the Phase adjust
Args:
    hdl: the handle of opened MC2000B device
    phase: desired phase adjust
Returns:
    0: Success; negative number: failed.
```

def MC2000B_COMMAND_LIB.MC2000BSetReference (**hdl, ref** *)*

```
Set the reference mode
Args:
    hdl: the handle of opened MC2000B device
    ref: reference mode 1:external, 0:internal
         reference high pre
Returns:
    0: Success; negative number: failed.
```

def MC2000B_COMMAND_LIB.MC2000BSetReferenceOutput **(hdl, output*)***

```
Set the output reference mode
Args:
    hdl: the handle of opened MC2000B device
    output: output mode 1:actual, 0:target
Returns:
    0: Success; negative number: failed.
```

def MC2000B_COMMAND_LIB.MC2000BSetVerbose **(hdl, verbose*)***

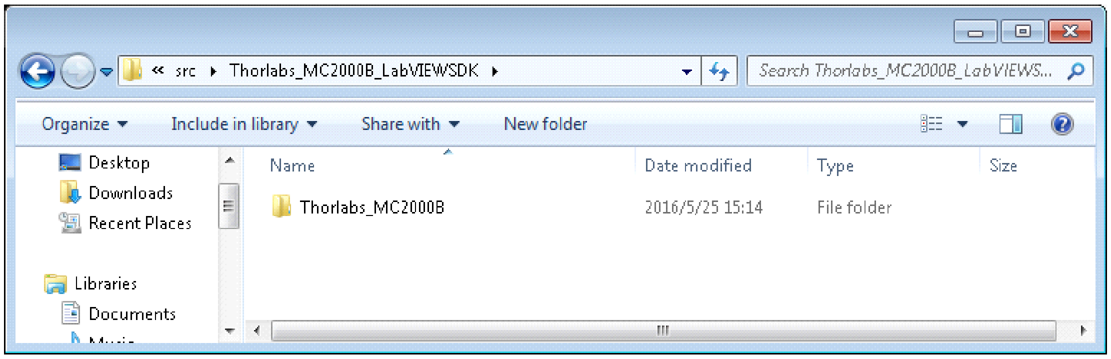
```
Set the verbose mode.
Args:
    hdl: the handle of opened MC2000B device
    verbose: verbose mode. When verbose mode is set to 1, status messages are output on the USB
Returns:
    0: Success; negative number: failed.
```

Chapter 4 LabVIEW Software Development Kit

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.

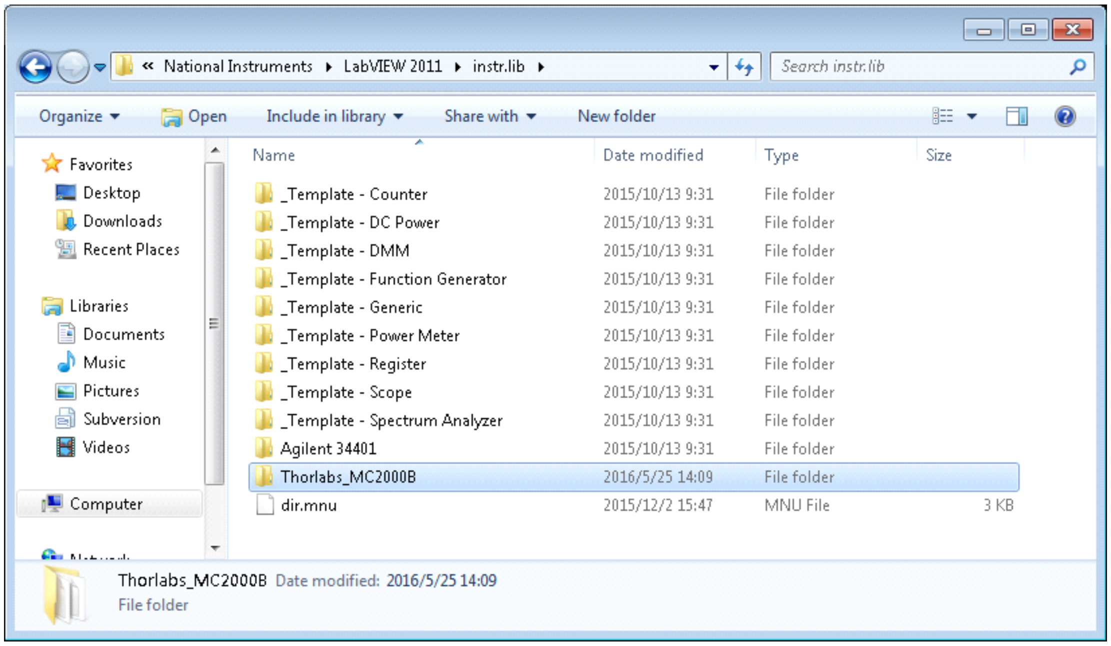
4.1. How to install

Unzip the zip file and **copy** to instr.lib folder under LabVIEW installation folder.



Name	Date modified	Type	Size
Thorlabs_MC2000B	2016/5/25 15:14	File 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.

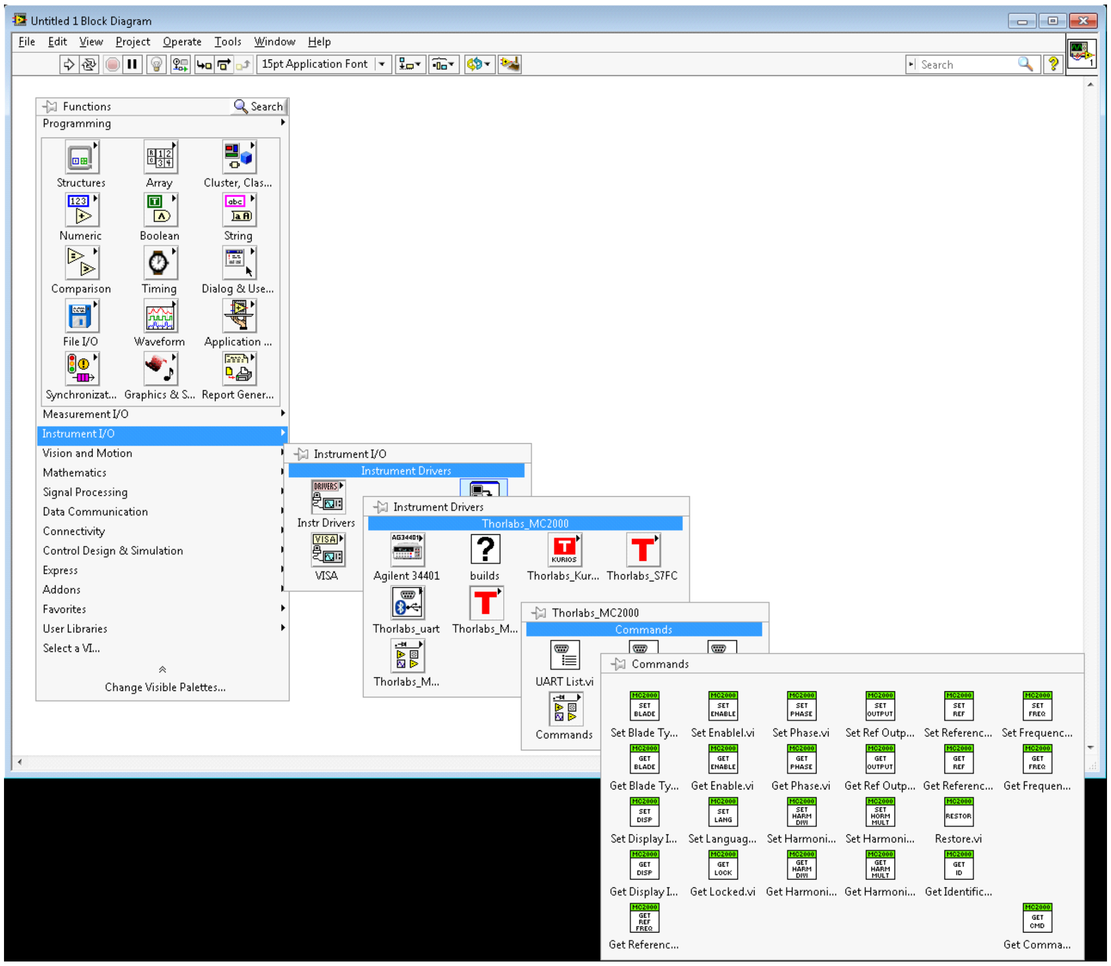


Name	Date modified	Type	Size
_Template - Counter	2015/10/13 9:31	File folder	
_Template - DC Power	2015/10/13 9:31	File folder	
_Template - DMM	2015/10/13 9:31	File folder	
_Template - Function Generator	2015/10/13 9:31	File folder	
_Template - Generic	2015/10/13 9:31	File folder	
_Template - Power Meter	2015/10/13 9:31	File folder	
_Template - Register	2015/10/13 9:31	File folder	
_Template - Scope	2015/10/13 9:31	File folder	
_Template - Spectrum Analyzer	2015/10/13 9:31	File folder	
Agilent 34401	2015/10/13 9:31	File folder	
Thorlabs_MC2000B	2016/5/25 14:09	File folder	
dir.mnu	2015/12/2 15:47	MNU File	3 KB

Thorlabs_MC2000B Date modified: 2016/5/25 14:09
File folder

4.2. How to find VI

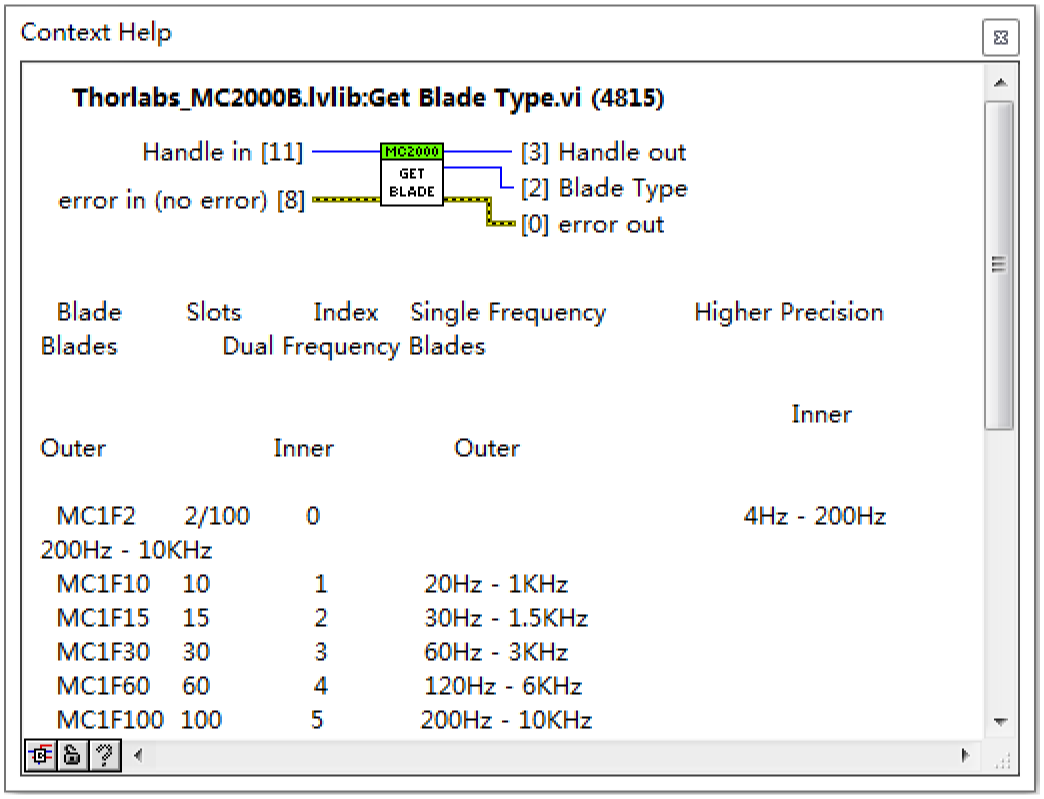
VI Could be found under: Functions\Instrument I/O\Instrument Drivers\



4.3. How to use

1. From VI

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



Thorlabs_MC2000B.lvlib:Get Blade Type.vi (4815)

Handle in [11] — [3] Handle out
[2] Blade Type
error in (no error) [8] — [0] error out

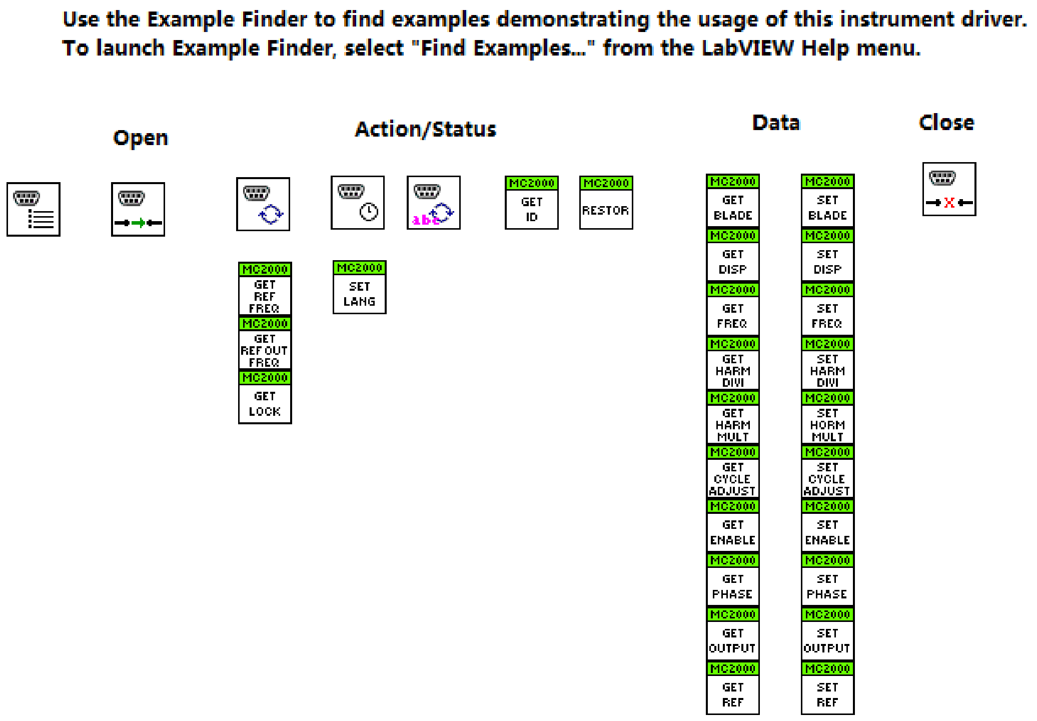
Blade Slots Index Single Frequency Higher Precision
Blades Dual Frequency Blades

Inner
Outer Inner Outer

MC1F2 2/100 0 4Hz - 200Hz
200Hz - 10KHz
MC1F10 10 1 20Hz - 1KHz
MC1F15 15 2 30Hz - 1.5KHz
MC1F30 30 3 60Hz - 3KHz
MC1F60 60 4 120Hz - 6KHz
MC1F100 100 5 200Hz - 10KHz

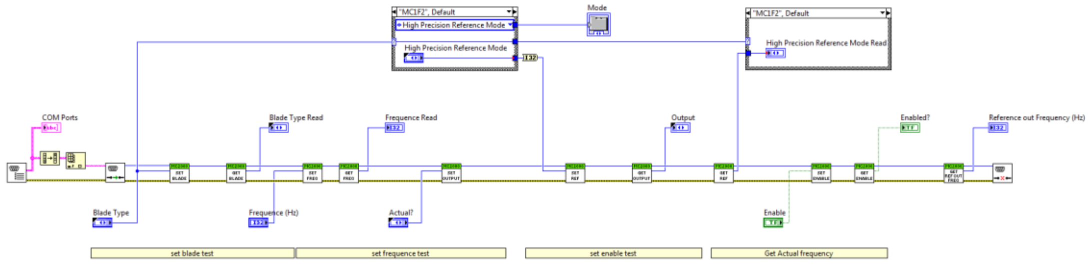
2. From VI tree

Some classic data flow in VI tree.



3. From example

An examples show the classic usage. Example path: instr.lib\Thorlabs_MC2000B\Examples



Easy programming and detailed comment will help.

THORLABS
www.thorlabs.com