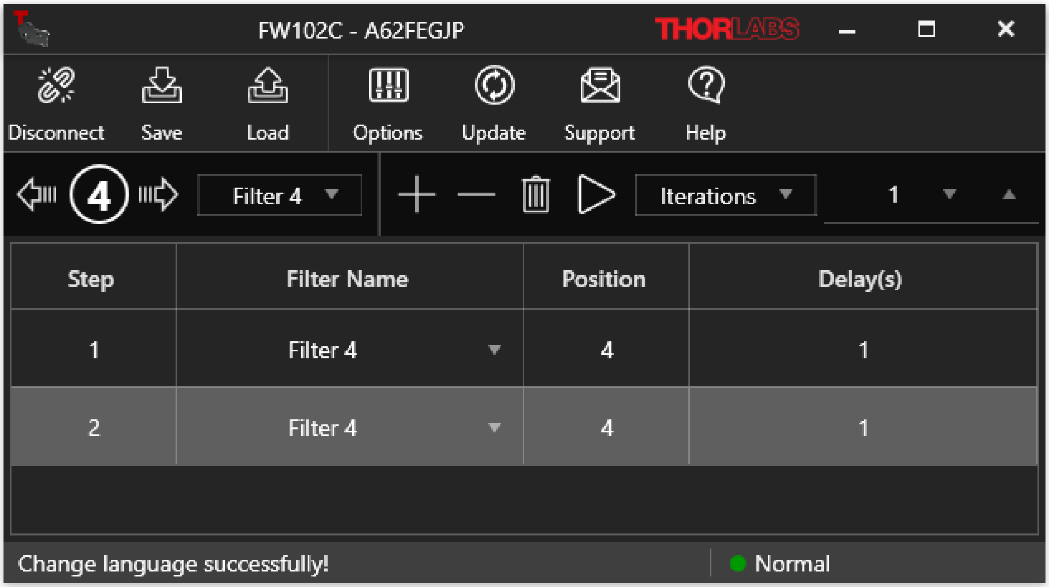
THORLABS

FWxC®

Motorized Filter Wheels

SDK Manual



Step	Filter Name	Position	Delay(s)
1	Filter 4	4	1
2	Filter 4	4	1

Change language successfully!

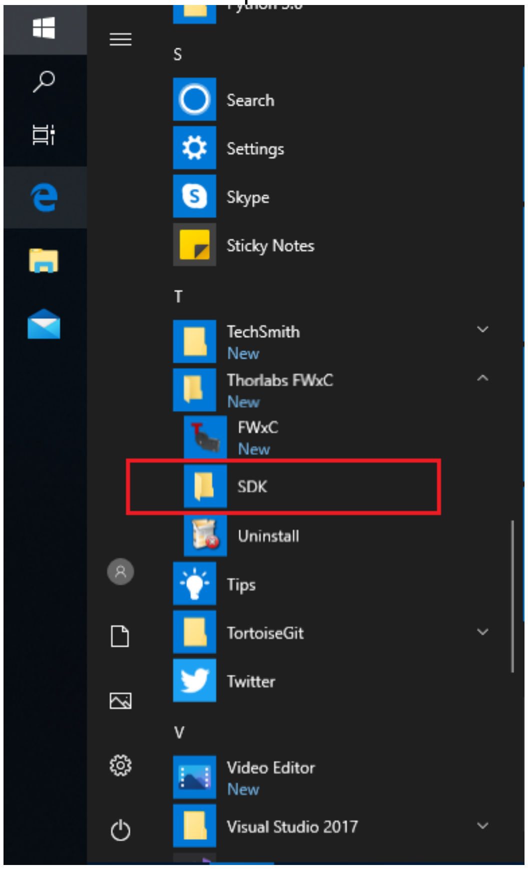
● Normal

Table of Contents

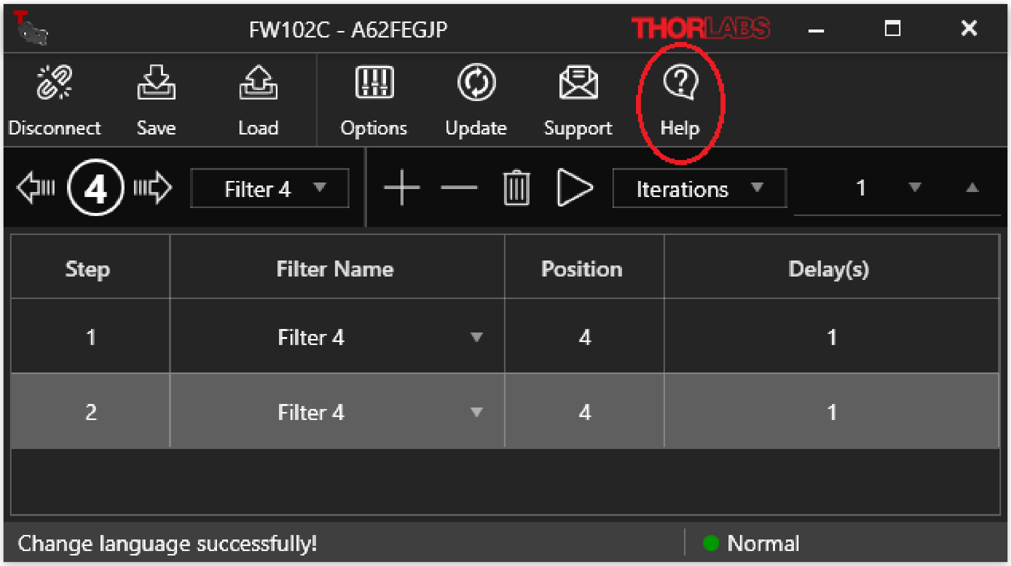
Chapter 1 Introduction

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.

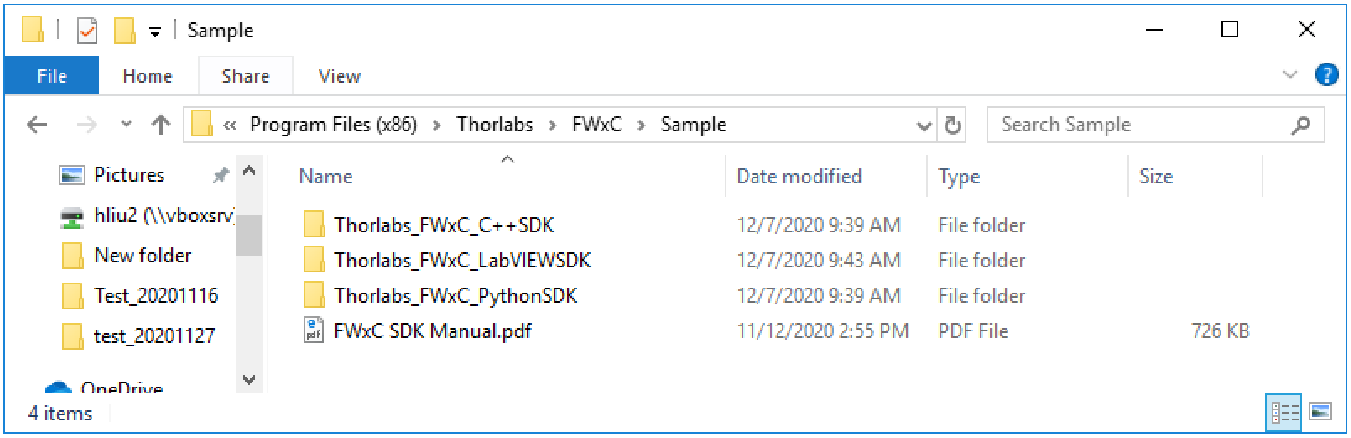


Step	Filter Name	Position	Delay(s)
1	Filter 4	4	1
2	Filter 4	4	1

Change language successfully!

● Normal

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



Name	Date modified	Type	Size
Thorlabs_FWxC_C++SDK	12/7/2020 9:39 AM	File folder	
Thorlabs_FWxC_LabVIEWSDK	12/7/2020 9:43 AM	File folder	
Thorlabs_FWxC_PythonSDK	12/7/2020 9:39 AM	File folder	
FWxC SDK Manual.pdf	11/12/2020 2:55 PM	PDF File	726 KB

Chapter 2 C++ Software Development Kit

User can start software development with **FilterWheel102_win32.dll** or **FilterWheel102_win64.dll** in C/C++ development environment which can be found in **Thorlabs_FWxC_C++SDK** under \Document directory. The corresponding header file is also in **Thorlabs_FWxC_C++SDK** under \Document directory.

Copy **FilterWheel102_win32.dll** or **FilterWheel102_win64.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 FWxCCommand.h

2.1. FWxCCommand.h File Reference

2.1.1. Functions

- **DllExport** int **List** (unsigned char *serialNo, unsigned int length)
  list all the possible port on this computer.
- **DllExport** int **Open** (char *serialNo, int nBaud, int timeout)
  open port function.
- **DllExport** int **IsOpen** (char *serialNo)
  check opened status of port
- **DllExport** int **Close** (int hdl)
  close current opend port
- **DllExport** int **Read** (int hdl, unsigned char *b, int limit)
- **DllExport** int **Write** (int hdl, char *b, int size)
- **DllExport** int **Set** (int hdl, char *c, int size)
- **DllExport** int **Get** (int hdl, unsigned char *c, unsigned char *d)
- **DllExport** int **Purge** (int hdl, int flag)
  Purge the RX and TX buffer on port.
- **DllExport** int **SetPosition** (int hdl, int pos)
- **DllExport** int **SetPositionCount** (int hdl, int count)
- **DllExport** int **SetSpeedMode** (int hdl, int mode)
- **DllExport** int **SetTriggerMode** (int hdl, int mode)
- **DllExport** int **SetSensorMode** (int hdl, int mode)
- **DllExport** int **Save** (int hdl)
- **DllExport** int **GetPosition** (int hdl, int *pos)
- **DllExport** int **GetPositionCount** (int hdl, int *poscount)
- **DllExport** int **GetSpeedMode** (int hdl, int *speed)
- </param > **DllExport** int **GetTriggerMode** (int hdl, int *triggermode)
- **DllExport** int **GetSensorMode** (int hdl, int *sensormode)
- **DllExport** int **GetTimeToCurrentPos** (int hdl, int *time)
- **DllExport** int **GetId** (int hdl, char *d)

2.1.2. Function Documentation

DllExport int Close (int hdl)

close current opend port

Parameters

hdl	handle of port.

Returns

0: success; negtive number : failed.

DllExport int Get (int hdl, unsigned char * c, unsigned 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;

DllExport int GetId (int hdl, char * d)

get the fiterwheel id
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.
d	output string (<255)

Returns

0: success;

0xEA: CMD_NOT_DEFINED;

0xEB: time out;

0xED: invalid string buffer;

DllExport int GetPosition (int hdl, int * pos)

get the fiterwheel current position
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.
pos	fiterwheel actual position

Returns

0: success;

0xEA: CMD_NOT_DEFINED;

0xEB: time out;
0xED: invalid string buffer;

DllExport int GetPositionCount (int **hdl***, int ** **poscount***)*

get the fiterwheel current position count
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.
poscount	fiterwheel actual position count

Returns

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

DllExport int GetSensorMode (int **hdl***, int ** **sensormode***)*

get the fiterwheel current sensor 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.
sensormode	fiterwheel actual sensor mode:0, Sensors turn off;1, Sensors remain active

Returns

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

DllExport int GetSpeedMode (int **hdl***, int ** **speed***)*

get the fiterwheel current speed 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.
speed	fiterwheel actual speed mode:0,slow speed:1,high speed

Returns

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

DllExport int GetTimeToCurrentPos (int **hdl***, int ** **time***)*

get the fiterwheel current sensor 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.
time	the time from last position to current position

Returns

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

</param> DllExport int GetTriggerMode (int **hdl***, int ** **triggermode***)*

get the fiterwheel current trigger 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.
triggermode	fiterwheel actual trigger mode:tr0, input mode;1, output mode

Returns

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

*DllExport 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.

*DllExport int List (unsigned char ** **serialNo***, unsigned int* **length***)*

list all the possible port on this computer.

Parameters

serialNo	port list returned string include serial number and device descriptor, separated by comma
length	max size of buf

Returns

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

*DllExport 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-negtive number: hdl number returned successfully; negtive number : failed.

DllExport 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.

DllExport 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.

DllExport int Save (int **hdl***)*

save all the settings as default on power up

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;

0xEA: CMD_NOT_DEFINED;

0xEB: time out;

0xED: invalid string buffer;

DllExport int Set (int **hdl***, 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;

DllExport int SetPosition (int **hdl***, int* **pos***)*

set fiterwheel's position

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.
pos	fiterwheel position

Returns

0: success;

0xEA: CMD_NOT_DEFINED;

0xEB: time out;

0xED: invalid string buffer;

DllExport int SetPositionCount (int **hdl***, int* **count***)*

set fiterwheel's position count

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.
count	fiterwheel PositionCount

Returns

0: success;

0xEA: CMD_NOT_DEFINED;

0xEB: time out;

0xED: invalid string buffer;

DllExport int SetSensorMode (int **hdl***, int* **mode***)*

set fiterwheel's sensor 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.
mode	fiterwheel sensor mode:sensors=0 Sensors turn off when wheel is idle to eliminate stray light;sensors=1 Sensors remain active

Returns

0: success;

0xEA: CMD_NOT_DEFINED;

0xEB: time out;

0xED: invalid string buffer;

DllExport int SetSpeedMode (int **hdl***, int* **mode***)*

set fiterwheel's speed 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.
mode	fiterwheel speed mode:speed=0 Sets the move profile to slow speed:speed=1 Sets the move profile to high speed

Returns

0: success;

0xEA: CMD_NOT_DEFINED;

0xEB: time out;

0xED: invalid string buffer;

DllExport int SetTriggerMode (int **hdl***, int* **mode***)*

set fiterwheel's trigger 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.
mode	fiterwheel trigger mode:trig=0 Sets the external trigger to the input mode, Respond to an active low pulse by advancing position by 1;trig=1 Sets the external trigger to the output mode, Generate an active high pulse when selected position arrived at

Returns

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

DllExport 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

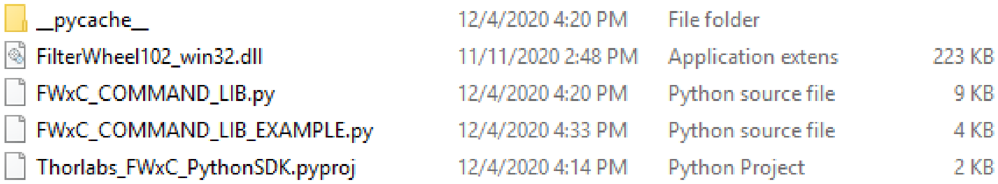
hdl	handle of port.
b	input string
size	size of string to be written.

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 **FWxC_COMMAND_LIB.py** to your python project, that's the wrapper for **FWxC_COMMAND_LIB**(**FilterWheel102_win32.dll** in C/C++ development environment). Copy **FilterWheel102_win32.dll** to your program folder, and make sure the library file and **FWxC_COMMAND_LIB.py** file are in the same folder. The " **FWxC_COMMAND_LIB_EXAMPLE.py**" is example code for how to use the python APIs.



__pycache__	12/4/2020 4:20 PM	File folder	
FilterWheel102_win32.dll	11/11/2020 2:48 PM	Application extens	223 KB
FWxC_COMMAND_LIB.py	12/4/2020 4:20 PM	Python source file	9 KB
FWxC_COMMAND_LIB_EXAMPLE.py	12/4/2020 4:33 PM	Python source file	4 KB
Thorlabs_FWxC_PythonSDK.pyproj	12/4/2020 4:14 PM	Python Project	2 KB

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

3.1. FWxC_COMMAND_LIB Namespace Reference

3.1.1. Functions

- def **FWxCListDevices** ()
- def **FWxCOpen** (serialNo, nBaud, timeout)
- def **FWxCIsOpen** (serialNo)
- def **FWxCClose** (hdl)
- def **FWxCSetPosition** (hdl, pos)
- def **FWxCSetPositionCount** (hdl, count)
- def **FWxCSetSpeedMode** (hdl, spmode)
- def **FWxCSetTriggerMode** (hdl, trimode)
- def **FWxCSetSensorMode** (hdl, senmode)
- def **FWxCSave** (hdl)
- def **FWxCGetId** (hdl, value)
- def **FWxCGetPosition** (hdl, pos)
- def **FWxCGetPositionCount** (hdl, poscount)
- def **FWxCGetSpeedMode** (hdl, spemode)
- def **FWxCGetTriggerMode** (hdl, triggermode)
- def **FWxCGetSensorMode** (hdl, sensormode)

3.1.2. Function Documentation

def FWxC_COMMAND_LIB.FWxCClose (* hdl*)

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

def FWxC_COMMAND_LIB.FWxCGetId (* hdl, value*)

```
get the FWxC id
Args:
    hdl: the handle of opened FWxC device
    value: the model number, hardware and firmware versions
Returns:
    0: Success; 0xEA: CMD_NOT_DEFINED; 0xEB: time out; 0xED: invalid string buffer.
```

def FWxC_COMMAND_LIB.FWxCGetPosition (hdl, pos)

```
get the fiterwheel current position
Args:
    hdl: the handle of opened FWxC device
    pos: fiterwheel actual position
Returns:
     0: Success; 0xEA: CMD_NOT_DEFINED; 0xEB: time out; 0xED: invalid string buffer.
```

def FWxC_COMMAND_LIB.FWxCGetPositionCount (hdl, poscount)

```
get the fiterwheel current position count
Args:
    hdl: the handle of opened FWxC device
    poscount: fiterwheel actual position count
Returns:
     0: Success; 0xEA: CMD_NOT_DEFINED; 0xEB: time out; 0xED: invalid string buffer.
```

def FWxC_COMMAND_LIB.FWxCGetSensorMode (hdl, sensormode)

```
get the fiterwheel current sensor mode
Args:
    hdl: the handle of opened FWxC device
    sensormode: fiterwheel actual sensor mode:0, Sensors turn off;1, Sensors remain active
Returns:
     0: Success; 0xEA: CMD_NOT_DEFINED; 0xEB: time out; 0xED: invalid string buffer.
```

def FWxC_COMMAND_LIB.FWxCGetSpeedMode (hdl, spemode)

```
get the fiterwheel current speed mode
Args:
    hdl: the handle of opened FWxC device
    spemode: 0,slow speed:1,high speed
Returns:
     0: Success; 0xEA: CMD_NOT_DEFINED; 0xEB: time out; 0xED: invalid string buffer.
```

def FWxC_COMMAND_LIB.FWxCGetTriggerMode (hdl, triggermode)

```
get the fiterwheel current position count
Args:
    hdl: the handle of opened FWxC device
    triggermode: fiterwheel actual trigger mode:0, input mode;1, output mode
Returns:
     0: Success; 0xEA: CMD_NOT_DEFINED; 0xEB: time out; 0xED: invalid string buffer.
```

def FWxC_COMMAND_LIB.FWxCIsOpen (serialNo)

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

def FWxC_COMMAND_LIB.FWxCListDevices ()

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

def FWxC_COMMAND_LIB.FWxCOpen (serialNo, nBaud, timeout*)*

```
Open FWxC device
Args:
    serialNo: serial number of FWxC 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 FWxC_COMMAND_LIB.FWxCSave (hdl*)*

```
save all the settings as default on power up
Args:
    hdl: the handle of opened FWxC device
Returns:
    0: Success; 0xEA: CMD_NOT_DEFINED; 0xEB: time out; 0xED: invalid string buffer.
```

def FWxC_COMMAND_LIB.FWxCSetPosition (hdl, pos*)*

```
set fiterwheel's position
Args:
    hdl: the handle of opened FWxC device
    pos: fiterwheel position
Returns:
    0: Success; 0xEA: CMD_NOT_DEFINED; 0xEB: time out; 0xED: invalid string buffer.
```

def FWxC_COMMAND_LIB.FWxCSetPositionCount (hdl, count*)*

```
set fiterwheel's position count
Args:
    hdl: the handle of opened FWxC device
    count: fiterwheel PositionCount
Returns:
   0: Success; 0xEA: CMD_NOT_DEFINED; 0xEB: time out; 0xED: invalid string buffer.
```

def FWxC_COMMAND_LIB.FWxCSetSensorMode (hdl, senmode*)*

```
set fiterwheel's sensor mode
Args:
    hdl: the handle of opened FWxC device
    senmode: fiterwheel sensor mode
             sensors=0 Sensors turn off when wheel is idle to eliminate stray light
             sensors=1 Sensors remain active
Returns:
   0: Success; 0xEA: CMD_NOT_DEFINED; 0xEB: time out; 0xED: invalid string buffer.
```

def FWxC_COMMAND_LIB.FWxCSetSpeedMode (hdl, spmode*)*

```
set fiterwheel's trigger mode
Args:
    hdl: the handle of opened FWxC device
    spmode: fiterwheel speed mode
            speed=0 Sets the move profile to slow speed
            speed=1 Sets the move profile to high speed
Returns:
   0: Success; 0xEA: CMD_NOT_DEFINED; 0xEB: time out; 0xED: invalid string buffer.
```

def FWxC_COMMAND_LIB.FWxCSetTriggerMode (hdl, trimode)

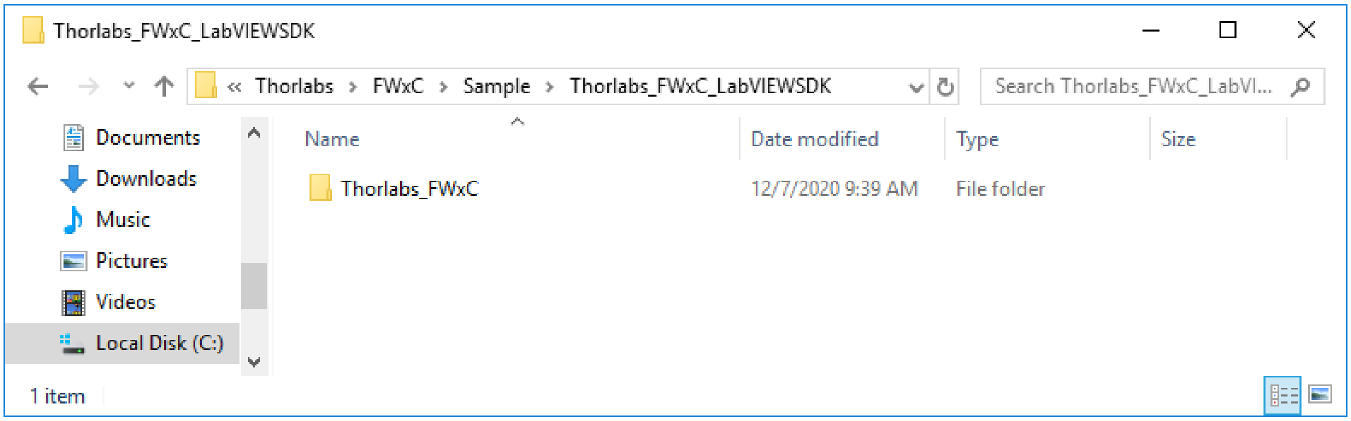
```
set fiterwheel's trigger mode
Args:
    hdl: the handle of opened FWxC device
    trimode: fiterwheel's trigger mode
             trig=0 Sets the external trigger to the input mode, Respond to an active low pulse by
advancing position by 1
             trig=1 Sets the external trigger to the output mode, Generate an active high pulse when
selected position arrived at
Returns:
   0: Success; 0xEA: CMD_NOT_DEFINED; 0xEB: time out; 0xED: invalid string buffer.
```

Chapter 4 LabVIEW Software Development Kit

The user can start software development with LabVIEW 2013 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.



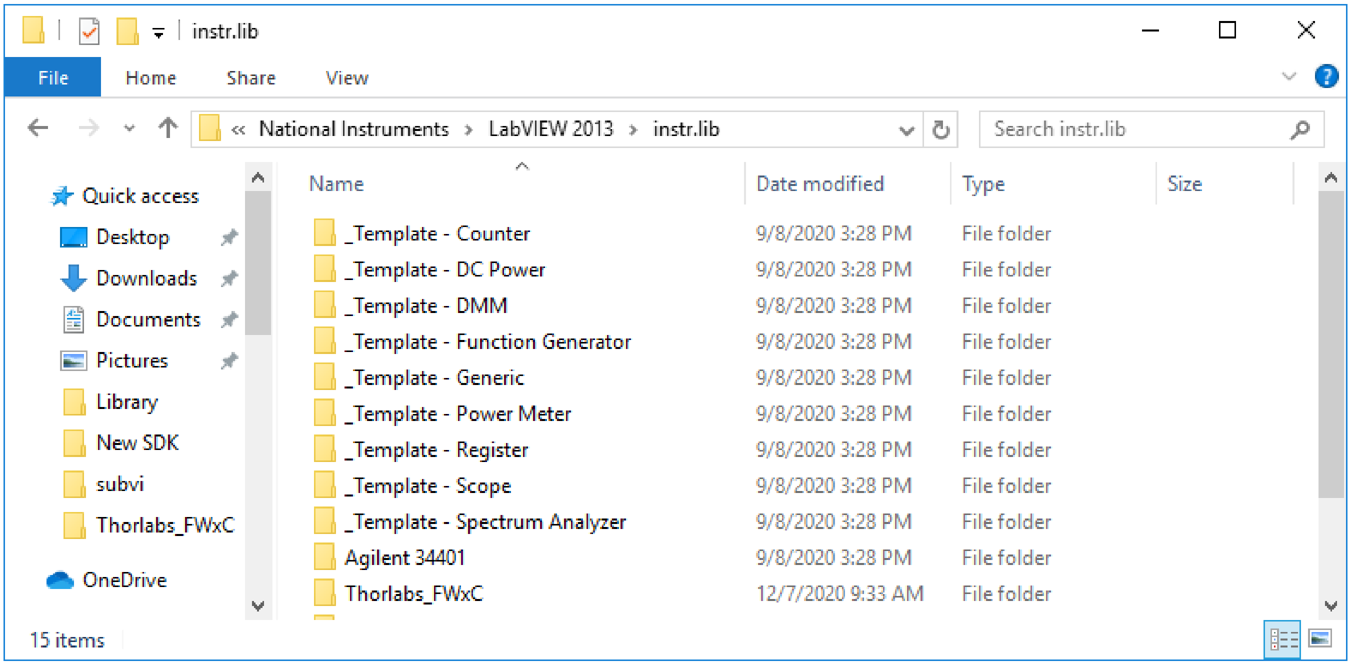
Thorlabs_FWxC_LabVIEWSDK

« Thorlabs › FWxC › Sample › Thorlabs_FWxC_LabVIEWSDK

Name	Date modified	Type	Size
Thorlabs_FWxC	12/7/2020 9:39 AM	File folder	

1 item

Destination folder: under %LabVIEW install path%\instr.lib
Typically, C:\Program Files (x86)\National Instruments\LabVIEW 2013\instr.lib
Note: LabVIEW 2013 or later LabVIEW versions are compatible.



instr.lib

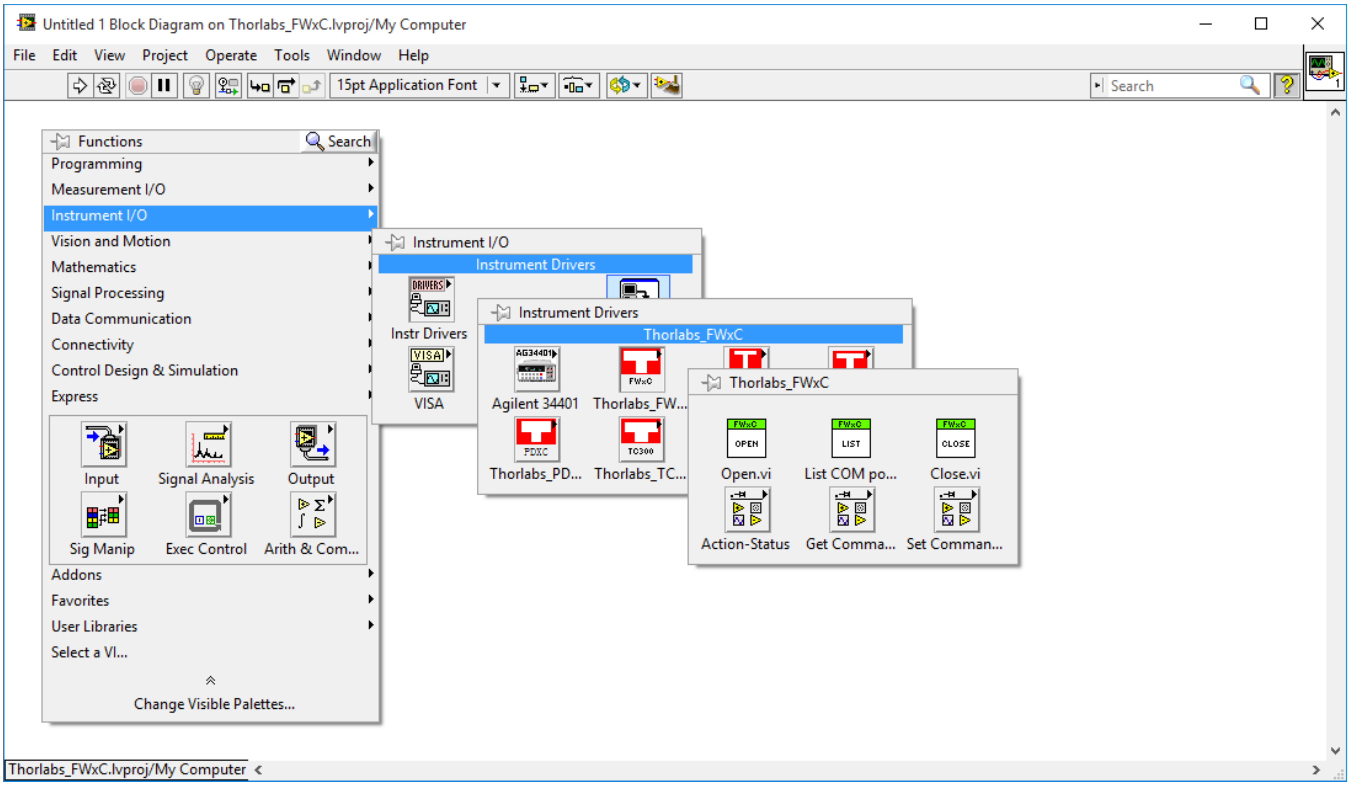
« National Instruments › LabVIEW 2013 › instr.lib

Name	Date modified	Type	Size
_Template - Counter	9/8/2020 3:28 PM	File folder	
_Template - DC Power	9/8/2020 3:28 PM	File folder	
_Template - DMM	9/8/2020 3:28 PM	File folder	
_Template - Function Generator	9/8/2020 3:28 PM	File folder	
_Template - Generic	9/8/2020 3:28 PM	File folder	
_Template - Power Meter	9/8/2020 3:28 PM	File folder	
_Template - Register	9/8/2020 3:28 PM	File folder	
_Template - Scope	9/8/2020 3:28 PM	File folder	
_Template - Spectrum Analyzer	9/8/2020 3:28 PM	File folder	
Agilent 34401	9/8/2020 3:28 PM	File folder	
Thorlabs_FWxC	12/7/2020 9:33 AM	File folder	

15 items

4.2. How to find VI

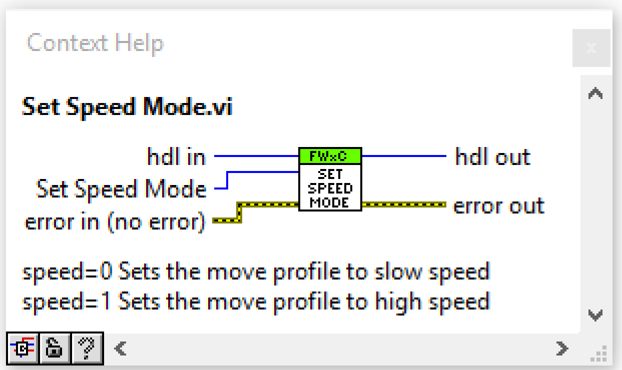
VI Could be found under: Functions\Instrument I/O\Instrument Drivers\



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.

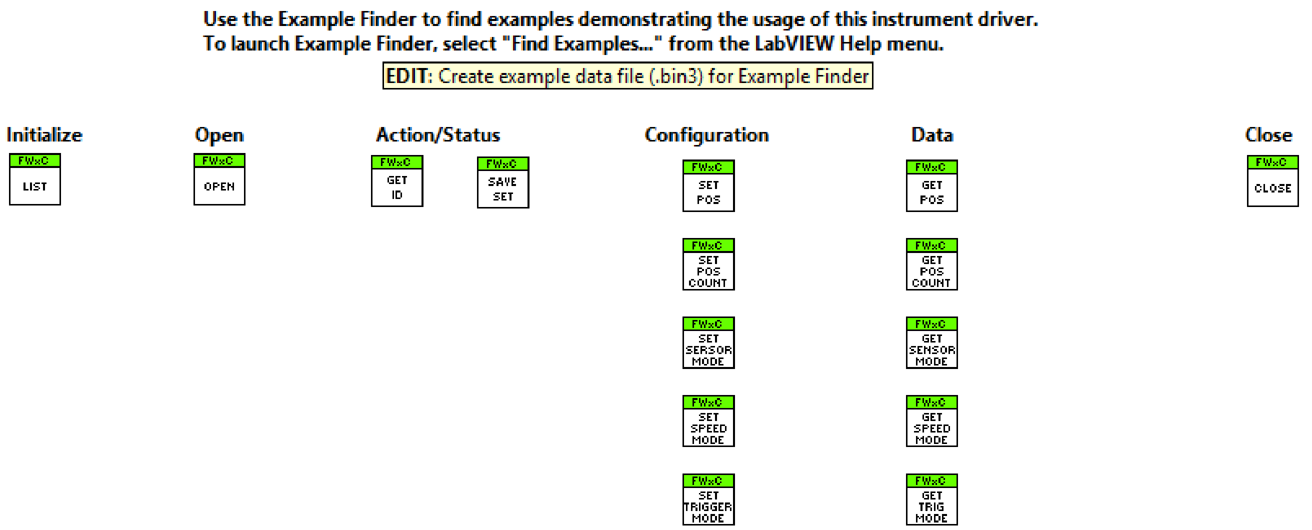


Set Speed Mode.vi

speed=0 Sets the move profile to slow speed
speed=1 Sets the move profile to high speed

2. From VI tree

Some classic data flow in VI tree.

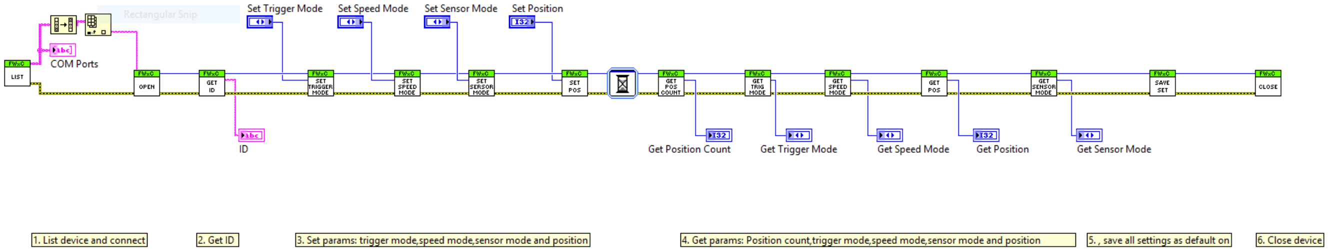


Use the Example Finder to find examples demonstrating the usage of this instrument driver.
To launch Example Finder, select "Find Examples..." from the LabVIEW Help menu.

EDIT: Create example data file (.bin3) for Example Finder

3. From example

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



1. List device and connect

2. Get ID

3. Set params: trigger mode,speed mode,sensor mode and position

4. Get params: Position count,trigger mode,speed mode,sensor mode and position

5. , save all settings as default on

6. Close device

Easy programming and detailed comment will help.

THORLABS
www.thorlabs.com