THORLABS

Application Note

PM100x Visual C++ Programming

This is a brief overview of how to get started making a custom program to communicate with a PM100x Compact Power and Energy Meter Console. While this application note is written for the PM100D Power and Energy Meter, the process is similar for our other Handheld (PM100A, PM160, and PM160T) Power Meters, as well as our Touchscreen (PM400 and PM200) and USB-Interface (PM100USB and PM16 series) Power Meters. The example program is for reference only and the user is encouraged to extend or modify the program to fit his or her specific needs.

Author: Sean Tanner
Date: 24-01-2017

Table of Contents

Author: Sean Tanner
Date: 24-01-2017

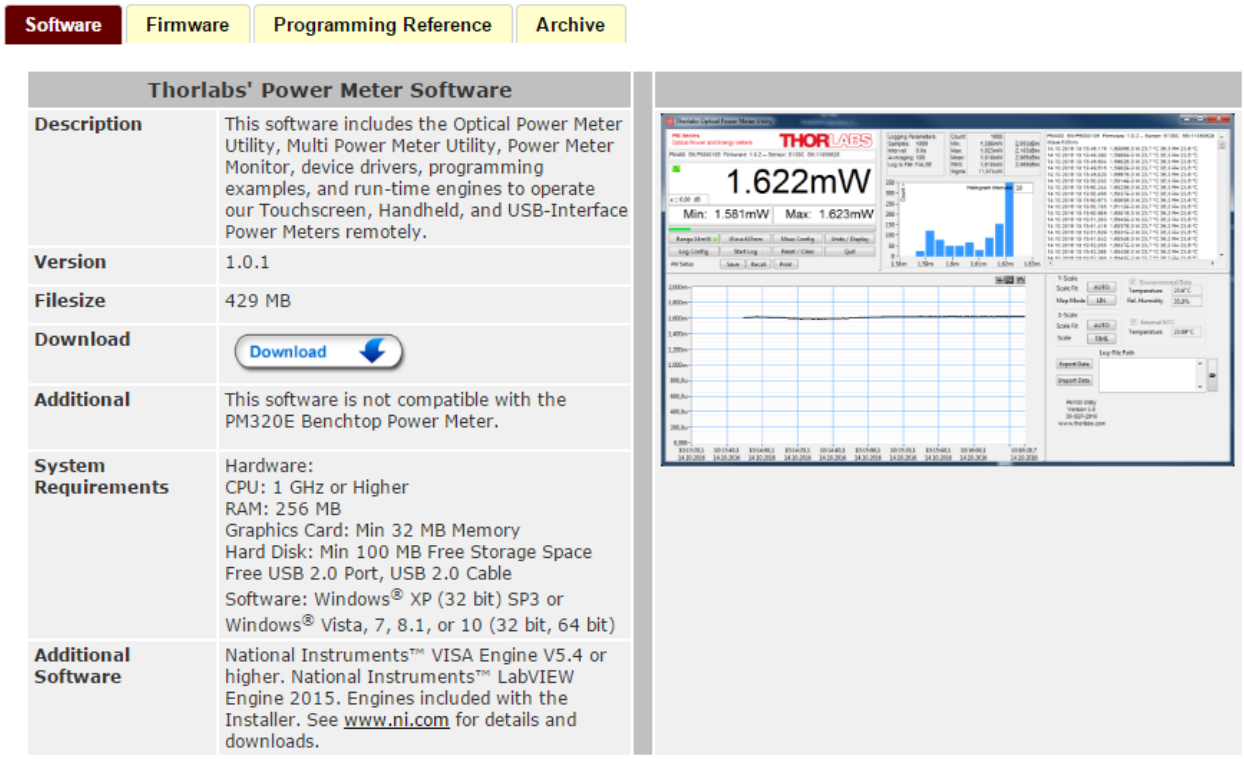
Part 1. Preface

This application note was written for the PM100D Digital Power Meter using the firmware and software versions detailed below. Functionality and procedures may vary when using other controllers or firmware/software versions.

- PM100D Firmware: Version 2.3.2
- Software for PM16, PM160, PM160T, PM100A, PM100D, PM100USB, and PM200 Series Optical Power and Energy Meters: Version 5.4
- Visual Studio: Version 11.0.61030.00 Update 4
- Microsoft .NET Framework: Version 4.5.50938

Part 2. Step by Step Instructions

1. Download and install the software for our Touchscreen (PM400 and PM200), Handheld (PM100D, PM100A, PM160, and PM160T), and USB-Interface (PM100USB and PM16 series) Power Meters located on the Software tab here: http://www.thorlabs.com/software_pages/ViewSoftwarePage.cfm?Code=PM100x

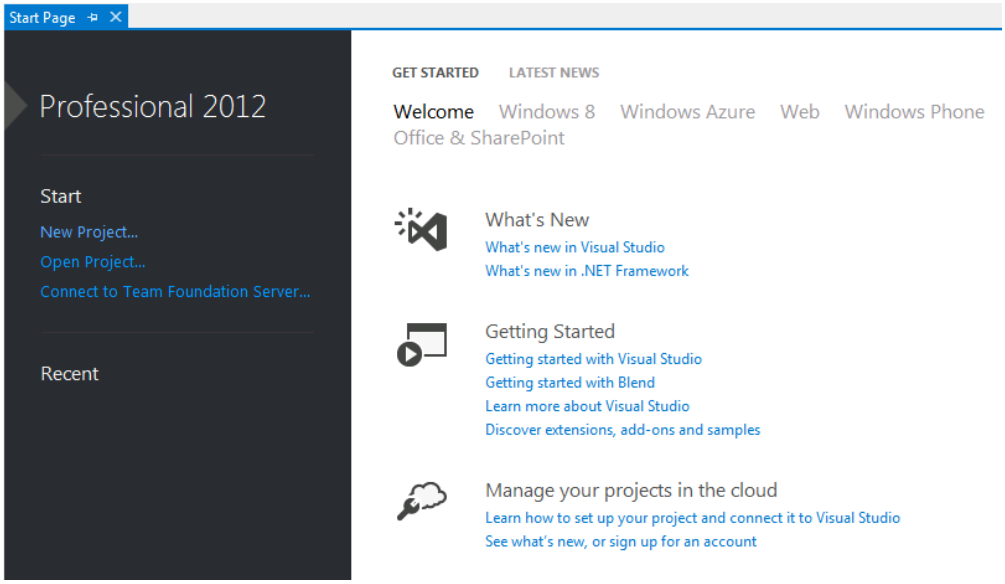


Software | Firmware | Programming Reference | Archive

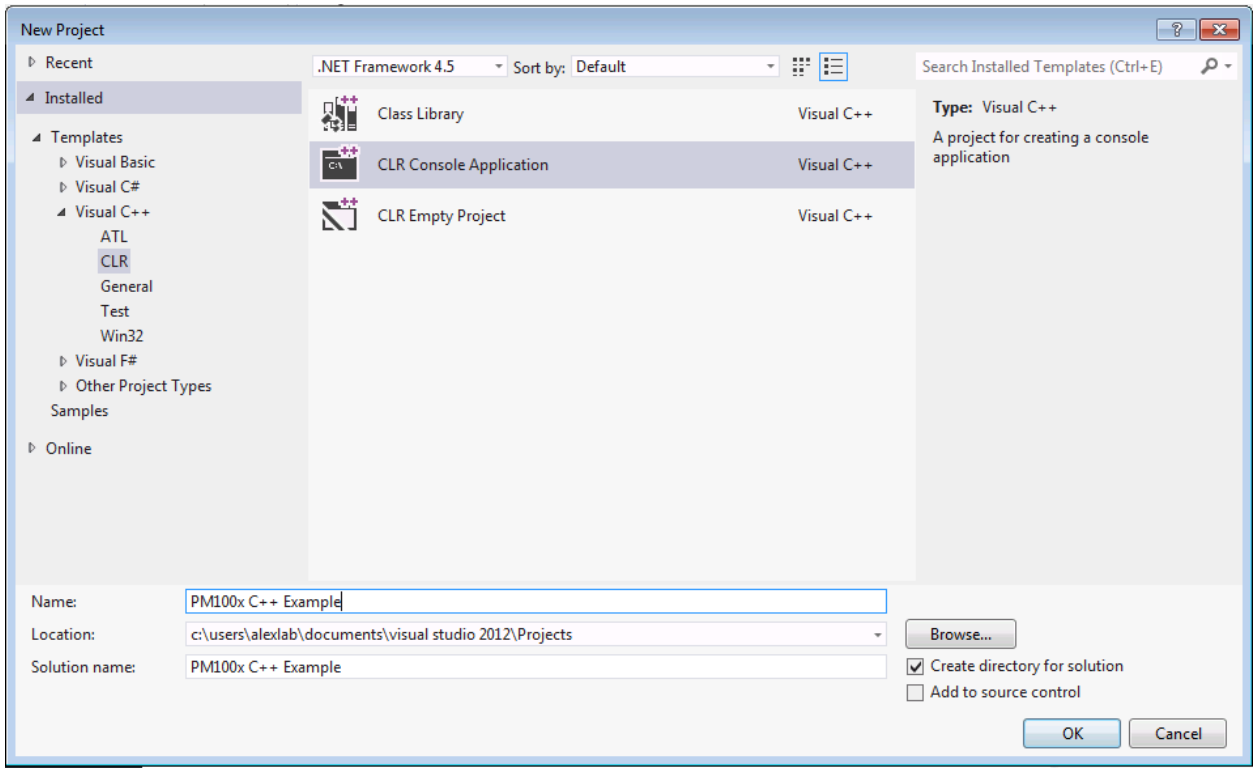
Thorlabs' Power Meter Software	
Description	This software includes the Optical Power Meter Utility, Multi Power Meter Utility, Power Meter Monitor, device drivers, programming examples, and run-time engines to operate our Touchscreen, Handheld, and USB-Interface Power Meters remotely.
Version	1.0.1
Filesize	429 MB
Download	Download
Additional	This software is not compatible with the PM320E Benchtop Power Meter.
System Requirements	Hardware: CPU: 1 GHz or Higher RAM: 256 MB Graphics Card: Min 32 MB Memory Hard Disk: Min 100 MB Free Storage Space Free USB 2.0 Port, USB 2.0 Cable Software: Windows® XP (32 bit) SP3 or Windows® Vista, 7, 8.1, or 10 (32 bit, 64 bit)
Additional Software	National Instruments™ VISA Engine V5.4 or higher. National Instruments™ LabVIEW Engine 2015. Engines included with the Installer. See www.ni.com for details and downloads.

2. Run the Thorlabs Optical Power Meter Utility to verify that your instrument is working with the computer correctly. Make note of your Device ID in the Connected Devices window (this will be used in step 12).

3. Open Microsoft Visual Studio and start a New Project.



4. Start a C++ CLR Console Application and give it an appropriate name. We used “PM100x C++ Example” for this application note.



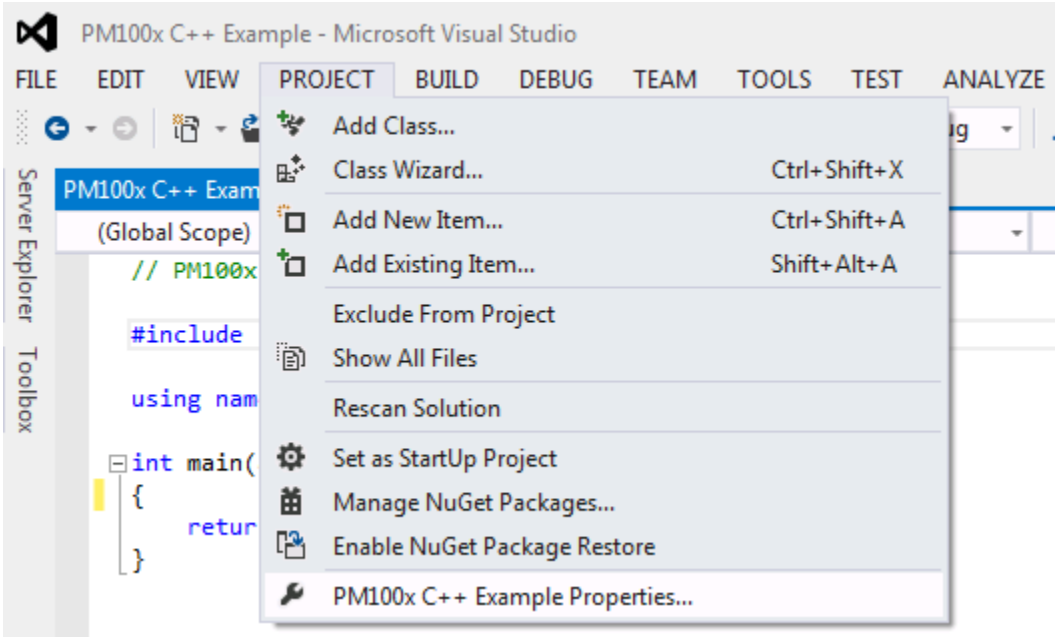
5. Your empty project will look similar to the empty project below.

```
// PM100x C++ Example.cpp : main project file.

#include "stdafx.h"

using namespace System;

int main(array<System::String ^> ^args)
{
    Console::WriteLine(L"Hello World");
    return 0;
}
```

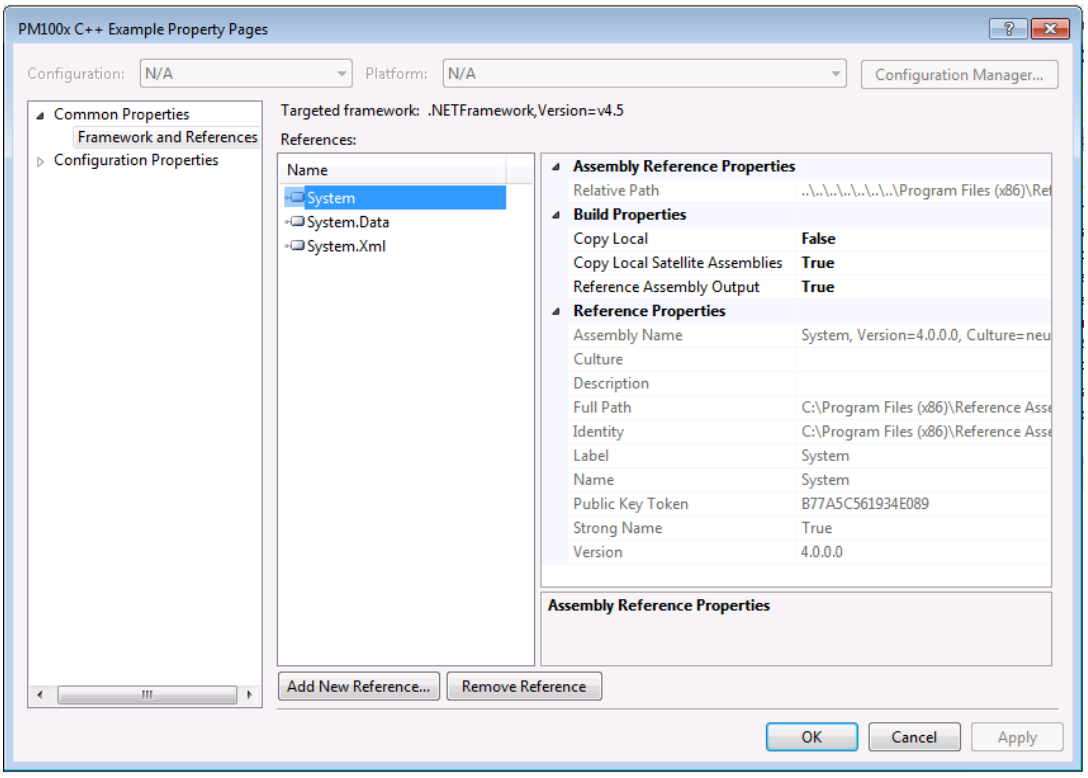
6. Remove the line that prints Hello World.

```
int main(array<System::String ^> ^args)
{
    return 0;
}
```

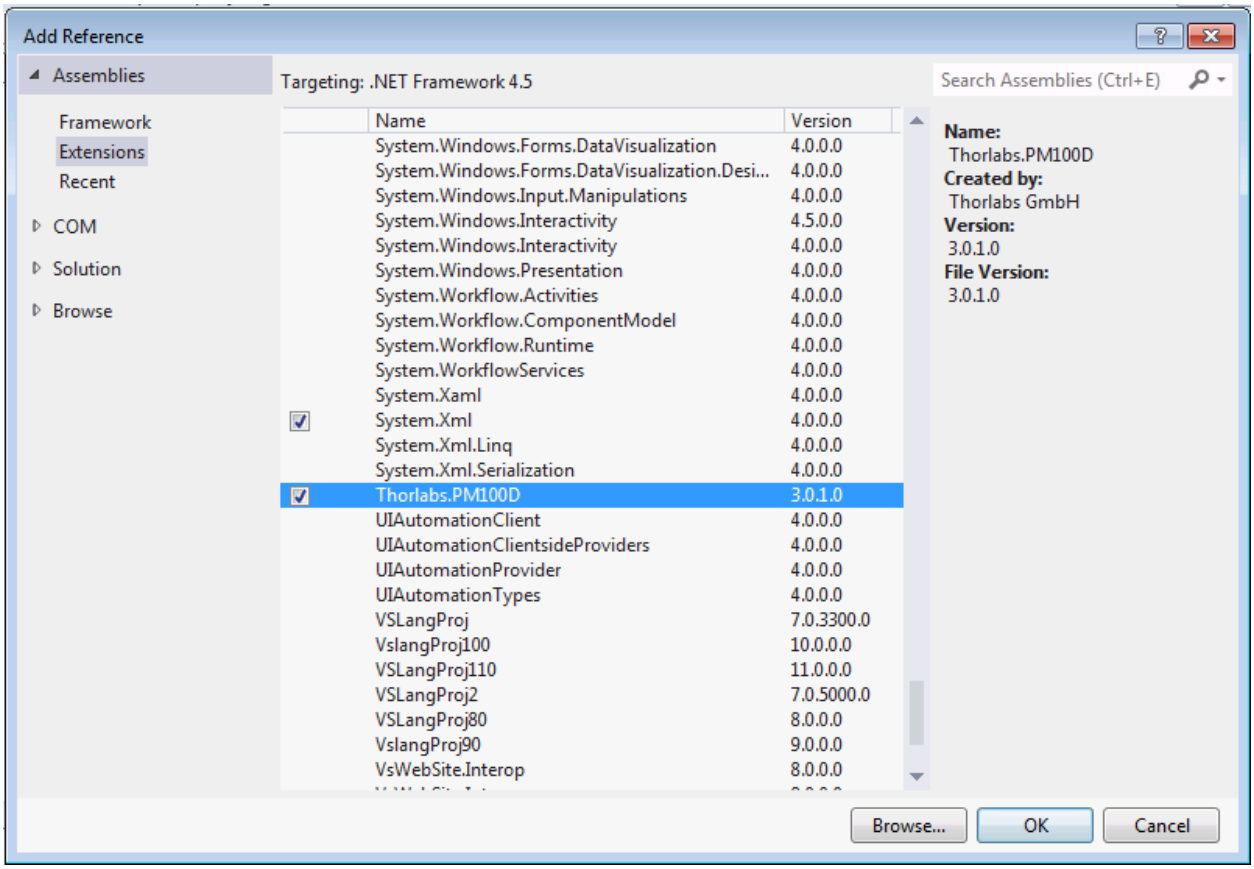
7. From the Project menu select your project name Properties, in this case PM100x C++ Example Properties.



8. Select Add New Reference.



9. Select Thorlabs.PM100D from the Extensions list and select OK (for both windows). This is used for the PM100A, PM100D, PM100USB, PM200, PM16, PM160, and PM160T.



10. Add a using statement for the Thorlabs.PM100D namespace.

```
using namespace System;
using namespace Thorlabs::PM100D;
```

11. Create a new PM100D object. The constructor parameters are a string that contains the device resource name and two Booleans which tell the device to do an ID query and to reset the device. The device resource name is listed by the Thorlabs Optical Power Meter Utility.

```
int main(array<System::String ^> ^args)
{
    //Create PM100D object.
    //The device resource name can be found by running the Thorlabs Optical Power Meter Utility.
    PM100D ^pm = gcnew PM100D("USB0::0x1313::0x8072::P2002734::INSTR",true,true);
```

12. Set the wavelength. The setWavelength method parameter is the wavelength in nm.

```
//set wavelength
pm->setWavelength(1064);
```

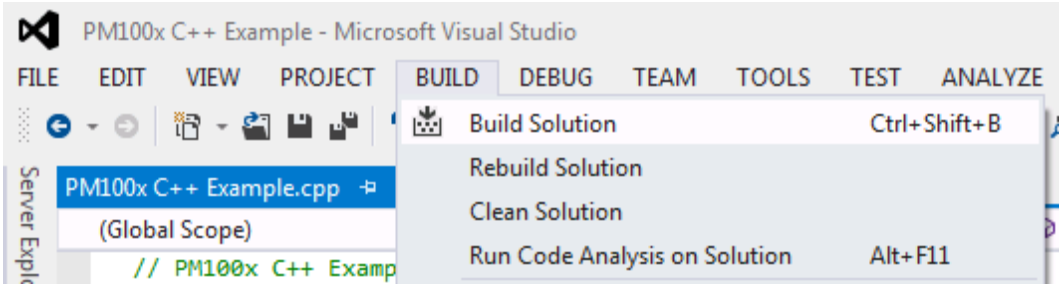
13. Get the power from the power meter and display it to the console. The measPower method parameter is a variable in which we will store the power measurement.

```
//measure and display power
double power;
pm->measPower(power);
Console::Write("The power is {0} W",power);
```

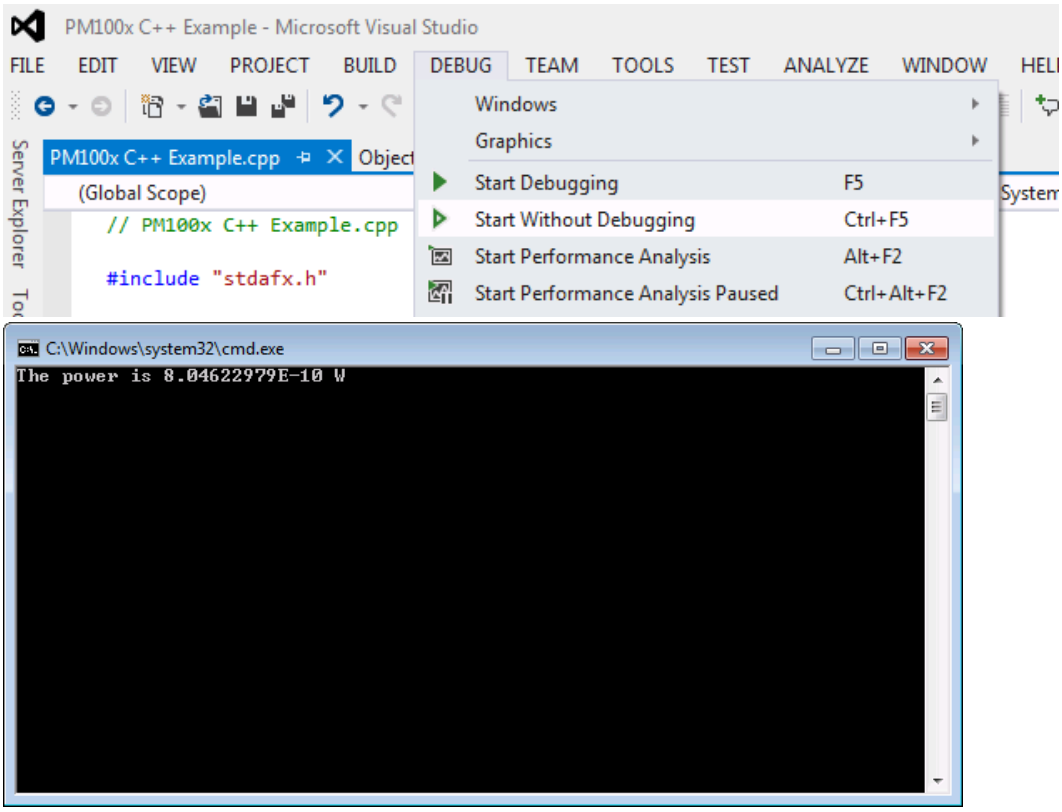
14. Wait for a button to be pressed so program does no exit immediately.

```
//Wait untill user presses a key to exit the program
Console::ReadKey();
```

15. Select Build Solution from the Build menu.



16. Select Start Without Debugging from the Debug menu to run the program.



Part 3. Methods Used

public PM100D(string Resource_Name, bool ID_Query, bool Reset_Device)

Summary:
This function initializes the instrument driver session and performs the following initialization actions: (1) Opens a session to the Default Resource Manager resource and a session to the specified device using the Resource Name specified. (2) Performs an identification query on the instrument. (3) Resets the instrument to a known state. (4) Sends initialization commands to the instrument. (5) Returns an instrument handle which is used to distinguish between different sessions of this instrument driver. Note: Each time this function is invoked a unique session is opened.

Parameters:
Resource_Name: This parameter specifies the device (resource) with which to establish a communication session.
ID_Query: This parameter specifies whether an identification query is performed during the initialization process. False - Skip query. True - Do query.
Reset_Device: This parameter specifies whether the instrument is reset during the initialization process. False - no reset. True - instrument is reset

Returns:
This function constructs a PM100D object which can be used to communicate with the device.

---

public int setWavelength(double Wavelength)

Summary:
This function sets the user's wavelength in nanometers [nm]. Remark: Wavelength set value is used for calculating power.

Parameters:
Wavelength: This parameter specifies the user's wavelength in nanometers [nm]. Remark: Wavelength set value is used for calculating power.

Returns:
The function returns an integer which indicates if an error occurred. A return of 0 indicates no error.

---

public int measPower(out double Power)

Summary:
This function is used to obtain power readings from the instrument.

Parameters:
Power: A variable to which the power measurement is written.

Returns:
The function returns an integer which indicates if an error occurred. A return of 0 indicates no error.

Part 4. Full Program

```
// PM100x C++ Example.cpp : main project file.

#include "stdafx.h"

using namespace System;
using namespace Thorlabs::PM100D;

int main(array<System::String ^> ^args)
{
       //Create PM100D object.
    //The device resource name can be found by running the Thorlabs Optical Power Meter
Utility.
       PM100D ^pm = gcnew PM100D("USB0::0x1313::0x8072::P2002734::INSTR",true,true);

       //set wavelength
    pm->setWavelength(1064);

    //measure and display power
    double power;
    pm->measPower(power);
    Console::Write("The power is {0} W",power);

    //Wait untill user presses a key to exit the program
    Console::ReadKey();

    return 0;
}
```

Part 5. Other Resources

The Object Browser in Visual Studio will list all of the classes and methods available in the Thorlabs.PM100D namespace. The Object Browser can be opened from the View menu in Visual Studio once a reference to Thorlabs.PM100D has been added (step 10).

Other programming examples can be found in the following directory once the drivers have been installed:
C:\Program Files (x86)\IVI Foundation\VISA\WinNT\PM100D\Samples\DotNet