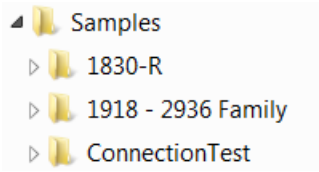
Power Meter Samples

March 22, 2012

Prepared by:
Newport Corporation
1791 Deere Avenue
Irvine, CA 92606

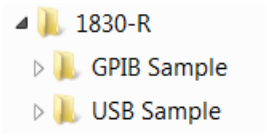
1 Introduction

The samples for the power meter are divided into the following folders: 1830-R, 1918 – 2936 Family, and ConnectionTest. The 1830-R folder includes samples that demonstrate how to interact with the 1830-R power meter. The 1918 – 2936 Family folder contains samples that show how to interact with the 1918 and 2936 family of power meters. The ConnectionTest folder has a sample utility (no source code) that tests the USB connection.



2 1830-R

This folder is divided into sub-folders where each folder that demonstrates how to interact with the 1830-R over a particular port, such as GPIB or USB.



2.1 GPIB Sample

This folder contains examples on how to communicate with the 1830-R power meter using the GPIB interface. The subfolders (LabVIEW 8.x, LabVIEW 2009, and LabVIEW 2010) each contain a different LabVIEW version of the same sample.

2.1.1 GPIBSample.lvproj

To view all of the VIs in the project, open the project file GPIBSample.lvproj.

2.1.1.1 GPIB_Query_With_SerialPoll.vi

This VI sends a command to the instrument, continuously polls the status register until a response is ready, and then reads the GPIB response data.

2.1.1.2 GPIB_Query_Without_SerialPoll.vi

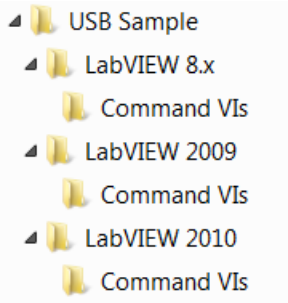
This VI sends a command to the instrument and then reads the GPIB response data. If the response data is not ready then a timeout error will occur.

2.1.1.3 GPIBCommunicationTest.vi

This VI allows the user to specify a command to be sent to the instrument over the GPIB, the number of times to send it, and the delay interval between each time the command is sent.

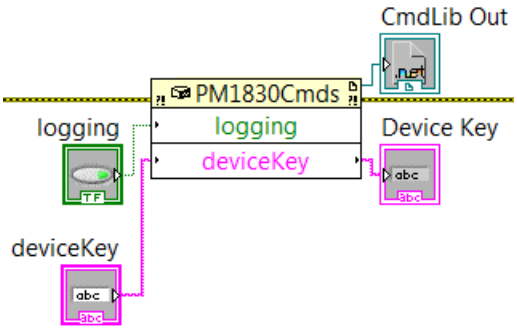
2.2 USB Sample

This folder contains examples on how to communicate with the 1830-R power meter using the USB port. The subfolders (LabVIEW 8.x, LabVIEW 2009, and LabVIEW 2010) each contain a different LabVIEW version of the same sample. The Command VIs folder contains VIs that wrap a single command that belongs to PowerMeterCommands.dll.

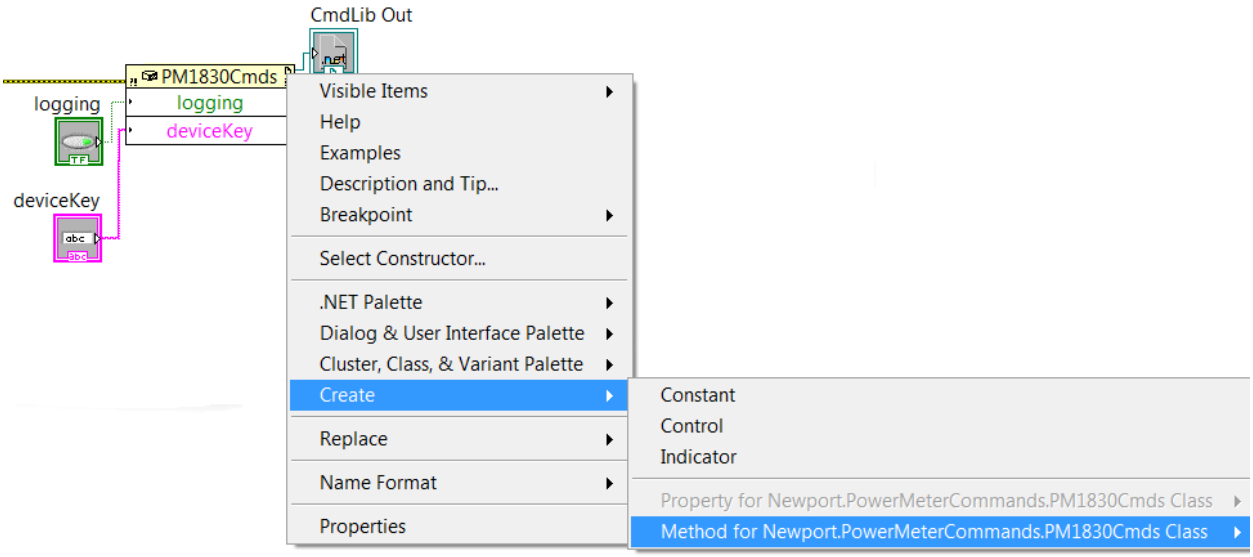


PowerMeterCommands.dll has a public method that can be called by LabVIEW, or any .NET development language, for most of the commands described in the user's manual. If a VI does not exist for a particular command then a method in PowerMeterCommands.dll may exist to perform this command. If neither exist, then the Read, Write, or Query methods in PowerMeterCommands.dll can be used to perform any of the commands described in the user's manual.

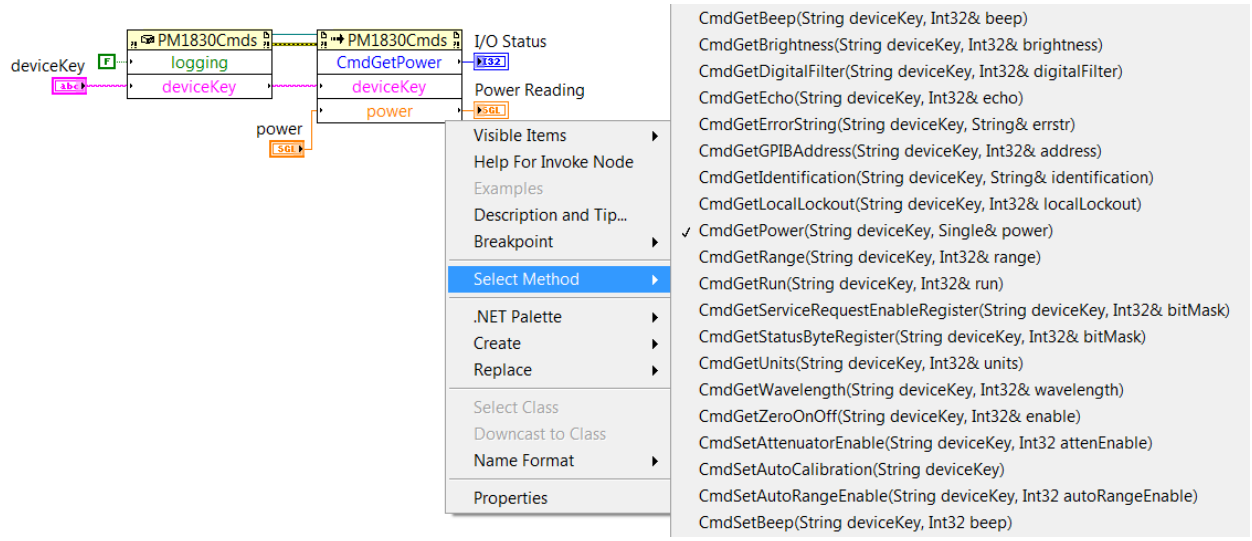
There are two basic steps necessary to send a command or query to the instrument. The first step is to initialize the USB interface. This can be done by calling InitCmdLib.vi (from the Command VIs folder) or by directly calling the LabVIEW constructor block that is called by InitCmdLib.vi. The PM1830Cmds constructor block initializes communication with an 1830-R power meter. This constructor also opens all instruments that are connected to the PC, automatically selects the first one found in the list of open devices, and returns the device key of the selected instrument. The device key is a unique identifier used by the USB interface to indicate which instrument is communicating with the PC. Logging can be turned on or off with the Boolean value passed into the constructor. Logging can be useful when trying to determine the cause of communication errors or what data is being transferred between the PC and the device.



The second step is to send a command or query to the instrument. This can be done by calling a VI from the Command VIs folder or by directly calling a method from PowerMeterCommands.dll. To call a method in PowerMeterCommands.dll, simply right-click the upper right hand corner of the constructor block (on the CmdLib Out wire) to display a context menu and then select “Create”, and then select “Method for Newport.PowerMeterCommands.PM1830Cmds Class”, and then select the name of the method to be called from the list in the context menu.



If a LabVIEW method block has been copied and pasted into a VI it can be modified by right-clicking the block, selecting “Select Method” from the context menu, and then selecting a method from the list in the context menu. If the data type changes on any of the inputs or outputs then this part will have to be rewired with the correct data type.



2.2.1 GetPower.lvproj

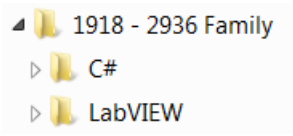
To view all of the VIs in the project, open the project file GetPower.lvproj.

2.2.1.1 SampleGetPower.vi

This VI demonstrates the two basic steps required to get a power reading from the power meter. First it initializes the USB interface, and then it calls a function in PowerMeterCommands.dll to get the power.

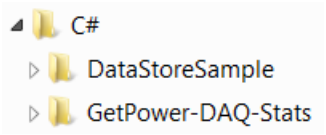
3 1918 - 2936 Family

This folder has samples for the 1918 and 2936 family of power meters.



3.1 C#

This folder has samples for the 1918 and 2936 family of power meters that have been developed in the C# programming language.



3.1.1 DataStoreSample

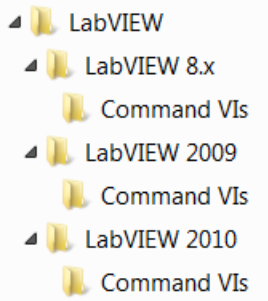
The DataStoreSample shows the low level details on how to program a small GUI application that allows the user to enter the sample size and perform data acquisition by clicking a button. The Response text box displays the number of samples collected by the power meter until it matches the sample size entered by the user. When the power meter has completed data acquisition, then the PC begins transferring the data from the power meter to a file on the PC and keeping track of the time for this operation.

3.1.2 GetPower-DAQ-Stats

This sample demonstrates how to use PowerMeterCommands.dll from a .NET programming language. It calls the PowerMeterCommands constructor to initialize the USB driver, and then calls methods to set the channel to A and perform data acquisition. After data acquisition is complete, it displays all the data collected. Then it displays the data acquisition statistics and the power reading (for up to two channels). This sample can be edited and debugged with Visual Studio Express 2008 which can be downloaded for free from Microsoft's web site.

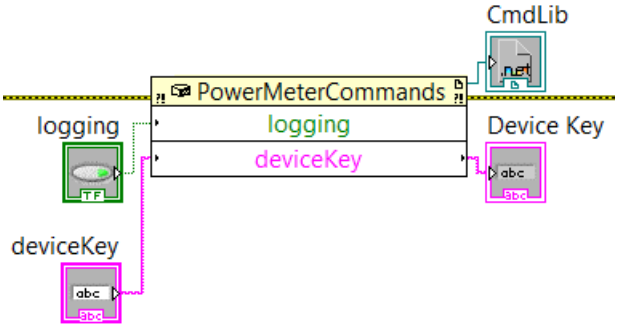
3.2 LabVIEW

This folder has samples for the 1918 and 2936 family of power meters that have been developed in the LabVIEW programming language. The subfolders (LabVIEW 8.x, LabVIEW 2009, and LabVIEW 2010) each contain a different LabVIEW version of the same sample. The Command VIs folder contains VIs that wrap a single command that belongs to PowerMeterCommands.dll.

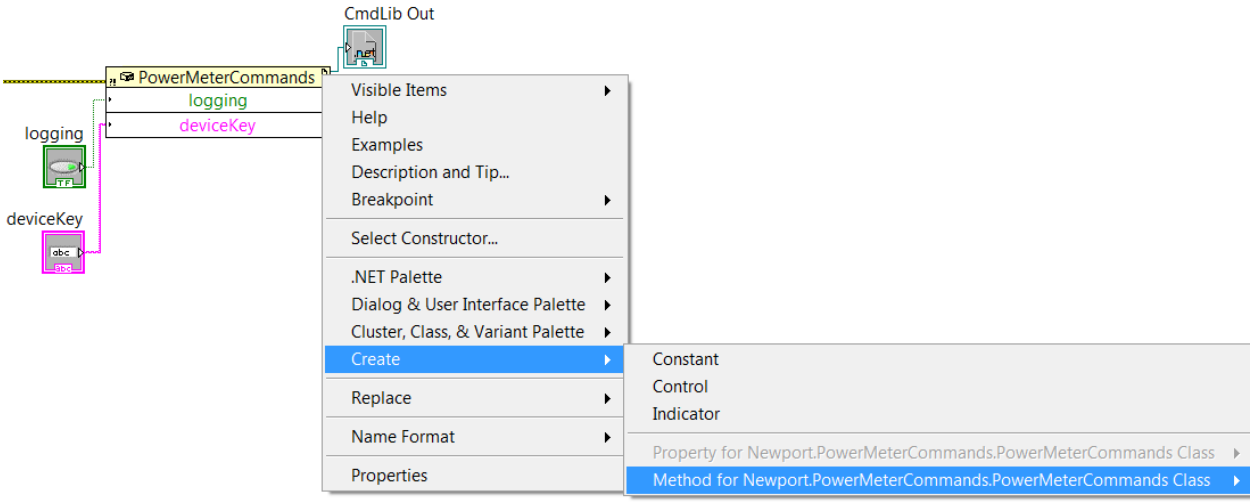


PowerMeterCommands.dll has a public method that can be called by LabVIEW, or any .NET development language, for almost all of the commands described in the user's manual. If a VI does not exist for a particular command then a method in PowerMeterCommands.dll may exist to perform this command. If neither exist, then the Read, Write, or Query methods in PowerMeterCommands.dll can be used to perform any of the commands described in the user's manual.

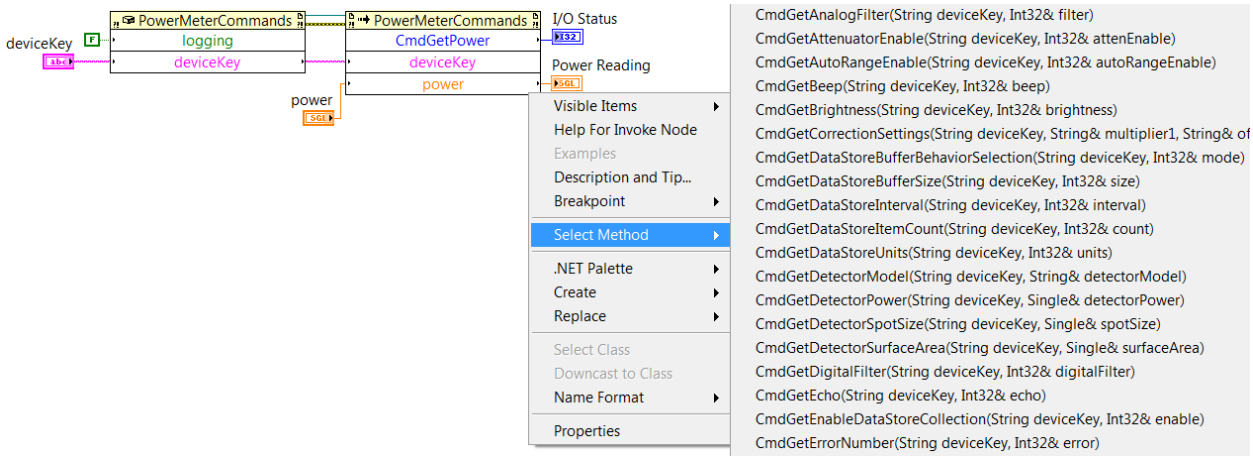
There are two basic steps necessary to send a command or query to the instrument. The first step is to initialize the USB interface. This can be done by calling InitCmdLib.vi (from the Command VIs folder) or by directly calling the LabVIEW constructor block that is called by InitCmdLib.vi. The PowerMeterCommands constructor block initializes communication with the power meter. This constructor also opens all instruments that are connected to the PC, automatically selects the first one found in the list of open devices, and returns the device key of the selected instrument. The device key is a unique identifier used by the USB interface to indicate which instrument is communicating with the PC. Logging can be turned on or off with the Boolean value passed into the constructor. Logging can be useful when trying to determine the cause of communication errors or what data is being transferred between the PC and the device.



The second step is to send a command or query to the instrument. This can be done by calling a VI from the Command VIs folder or by directly calling a method from PowerMeterCommands.dll. To call a method in PowerMeterCommands.dll, simply right-click the upper right hand corner of the constructor block (on the CmdLib Out wire) to display a context menu and then select “Create”, and then select “Method for Newport.PowerMeterCommands.PowerMeterCommands Class”, and then select the name of the method to be called from the list in the context menu.



If a LabVIEW method block has been copied and pasted into a VI it can be modified by right-clicking the block, selecting “Select Method” from the context menu, and then selecting a method from the list in the context menu. If the data type changes on any of the inputs or outputs then this part will have to be rewired with the correct data type.



3.2.1 Power-DAQ-Stats.lvproj

To view all of the VIs in the project, open the project file Power-DAQ-Stats.lvproj.

3.2.1.1 Sample Power-DAQ-Stats.vi

This VI demonstrates the two basic steps required to use PowerMeterCommands.dll. First it initializes the USB interface, and then it calls a method to perform a particular task, such as get the power. This sample initializes the USB interface, gets the power reading, sets the current channel to A, performs data acquisition, displays the collected data values, and then displays the statistics in two formats (the scaled decimal value with units, and the decimal value without units).

4 ConnectionTest

This folder has an application (ConnectionTest.exe) that tests the connection status of the instrument. It repeatedly attempts to connect until successful. If connected then it repeatedly sends the specified command to the instrument and displays any response. The command that is sent may be changed by entering a new command in the Command text box and clicking the Enter button. The current device may be changed by selecting a new instrument from the Device drop down control.