Interfacing PPMS, DynaCool, VersaLab, and MPMS 3 Systems with LabVIEW and Other .NET Languages

Some users of Quantum Design instruments design custom experiments using their own measurement electronics, especially when working on the general purpose platforms of the PPMS, DynaCool and VersaLab. This requires the user to have control of both their external device as well as the system's state (e.g., temperature and magnetic field). A very popular software package for general measurement automation used by our customers is LabVIEW from National Instruments. However, we at Quantum Design have designed our own software package called MultiVu to control the base system as well as measurement options.

This application note describes how we have made a bridge between these two software environments for the purpose of controlling our system from LabVIEW. Accompanying this note should be a ZIP archive entitled `QDInstrument_LabVIEW.zip` which contains all necessary programs described herein, assuming that the user has separately purchased a LabVIEW software installation. This document is intended for those with previous LabVIEW programming experience. For general LabVIEW questions and support, please consult National Instruments resources on the web.

Integrating LabVIEW and .NET Languages with MultiVu

The software package accompanying this application note provides a link between the LabVIEW and MultiVu programs. We have also posted software packages on our Pharos digital library that demonstrate this for C# and Visual Basic:

https://www.qdusa.com/pharos/view.php?fDocumentId=2704

as well as a socket server that allows remote control from non-Windows operating systems:

https://www.qdusa.com/pharos/view.php?fDocumentId=2703

The program responsible for this connection is called QDInstrument and must be installed on the computer running LabVIEW. In brief, the QDInstrument program is a translator with a .NET interface for communication with the LabVIEW VI and an OLE interface for communication with MultiVu. For more detailed information about this, please see Appendix B of this application note. There are a number of advantages to using this package:

1. It can be used across the Quantum Design systems of PPMS, DynaCool, Versalab and MPMS3. This is because the QDInstrument program communicates with the user-identified type of MultiVu and not with the system's hardware directly. The user is required simply to select from a list in a VI (OpenQDInstrument) which system type is in use.
2. It allows (requires) MultiVu to be running because it interacts at the software level with the OLE interface of MultiVu. Therefore it takes advantage of all the data redirection and safeguarding

built into MultiVu. Basically, MultiVu treats the LabVIEW requests the same way it treats direct user input from the keyboard at the MultiVu computer.

3. In addition to being able to run LabVIEW locally on the MultiVu computer, this software package can run in a "remote mode" where the LabVIEW and MultiVu computers communicate over the local area network (LAN). In this case, a simple server program `QDInstrument_server.exe` runs on the MultiVu computer and handles requests from LabVIEW via QDInstrument. The remote mode is appealing as it does not require additional LabVIEW installations and also keeps the MultiVu computer's resources dedicated to running the QD system.
4. It is compatible with all versions of LabVIEW from LabVIEW 8.2 up to the most recent (at this time that is LabVIEW 2014).

For more information about the software versions, compatibility, and bug fixes please see the `ReleaseNotes.txt` file included in the QDInstrument package accompanying this application note. Appendix A gives some helpful historical background and context regarding alternative methods for making custom experiments on Quantum Design instruments.

Operating in Remote Mode

As mentioned above, running the LabVIEW program on a separate computer from MultiVu has some advantages. The diagrams in Figure 1 and Figure 2 illustrate examples of a LabVIEW computer attached to an LCR meter via, for example, a GPIB or USB interface (yellow link). The QDInstrument VIs and `QDInstrument.dll` are installed on this computer and communicate over .NET (green link). The DLL is configured to address the `QDInstrument_server.exe` program running on the MultiVu computer. This is done via wired or wireless LAN (blue link). It is also possible to connect an Ethernet cable directly between the two computers in the absence of a LAN. The server program in turn communicates over OLE with the MultiVu application (red link). Lastly, MultiVu controls the instrument which in this case is either a PPMS (Figure 1) or a DynaCool (Figure 2).

Follow these steps to set up for remote mode:

1. Install Microsoft .NET 3.5 on the computer running MultiVu and the computer running LabVIEW computers. Microsoft .NET 3.5 can be downloaded at http://www.microsoft.com/en-us/download/details.aspx?id=21.
2. Delete any previous versions of the QDInstrument VIs, LLBs, and `QDInstrument.dll` on the LabVIEW computer.
3. Copy all files except for the `QDinstrument_server.exe` to a folder on the LabVIEW computer. Keep these files in one directory. Any directory is OK.
4. Copy `QDInstrument_Server.exe` to the MultiVu computer. We recommend creating a folder `C:\QDLabVIEW` but any directory is OK.
5. Determine the IP address of the MultiVu computer. One way to do this on Windows: Click Start→ Run... and type "cmd" then hit OK. At the command prompt, type "ipconfig" and locate the address (in format ###.###.###.###) next to the text "IP Address" or "IPv4 Address".

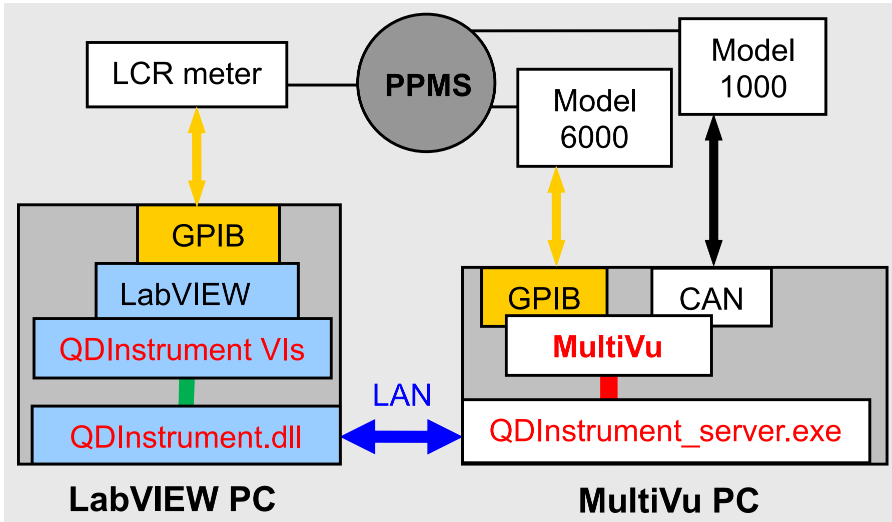

Figure 1: Example using LabVIEW in remote mode on a PPMS system

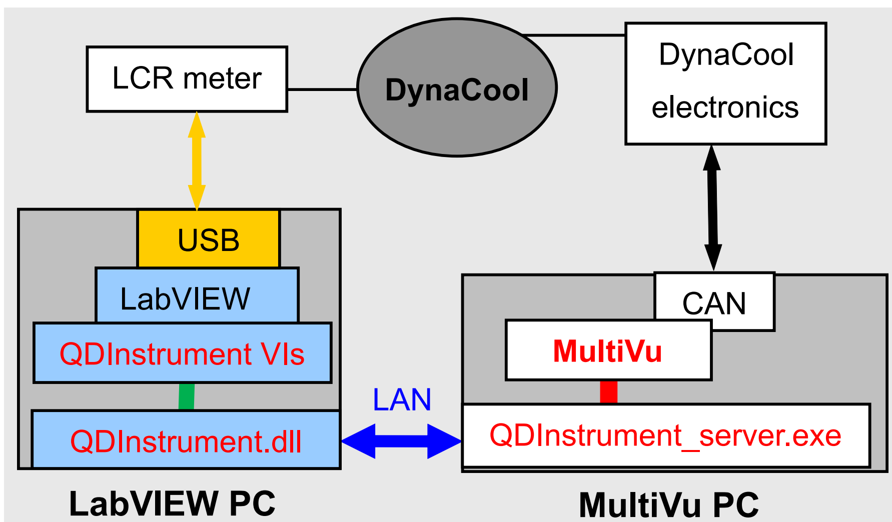

Figure 2: Example using LabVIEW in remote mode on a DynaCool system

Operating in Local Mode

In local mode, both LabVIEW as well as a communication bus to the user's external electronics (here, GPIB to an LCR meter) must be installed on the MultiVu PC. The .NET connection (green) between the VIs and the `QDInstrument.dll` is the same as above, but this time the DLL communicates directly with MultiVu over OLE (red). Note that an independent GPIB bus is recommended in this setup so that there are not traffic or settings conflicts in communicating with the LCR meter vs. the PPMS electronics.

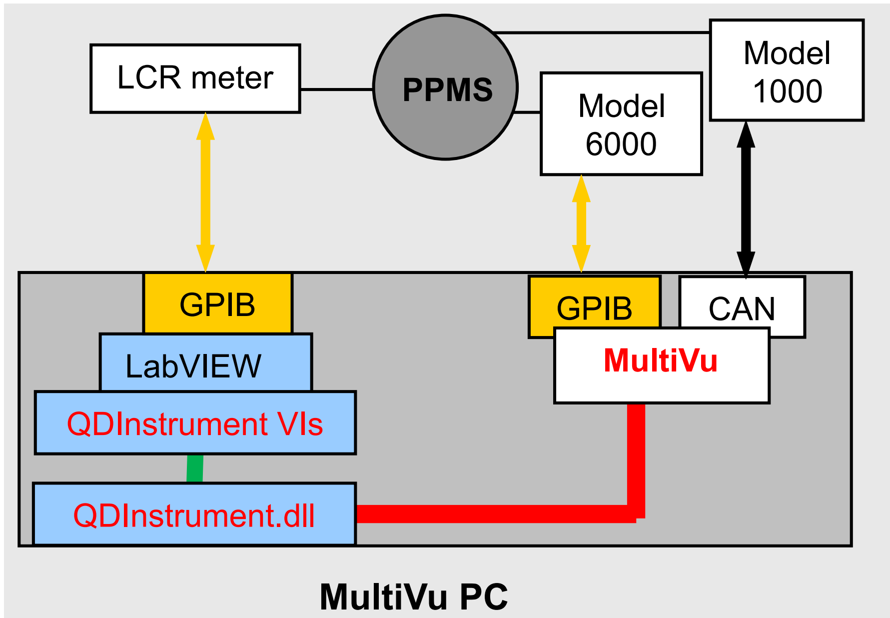

Figure 3: Example using LabVIEW in local mode on a PPMS system

Follow these steps to set up for local mode:

1. Install Microsoft .NET 3.5 on the computer running MultiVu and LabVIEW. Microsoft .NET 3.5 can be downloaded at: http://www.microsoft.com/en-us/download/details.aspx?id=21.
2. Delete any previous versions of these VIs, LLBs, and `QDInstrument.dll`.
3. Copy all files in the zip archive to a new folder on your PC which runs MultiVu and LabVIEW. We suggest creating a folder `C:\QDLabVIEW` but any directory is OK.

Testing the Installation by Running the Example VI

To make sure your connection is functioning properly and to see an example of how to use the QDInstrument VIs, load `QDInstrument_Example.vi` and run it. The example purges the sample chamber, increments temperature (295, 296, 297 K) and at each temperature steps the field from 0 to 500 Oe in 100 Oe steps. At the end it sets zero field (on PPMS the end mode for the magnet is Persistent) and a temperature of 300 K. Instructions for running this example are below:

1. After opening the VI, go to the front panel and select the appropriate instrument type: PPMS, VersaLab, DynaCool, or SVSM (MPMS3). Ignore warnings from LabVIEW about the version of the QDInstrument DLL if they come up.
2. If in remote mode, toggle the Remote button to True and enter the IP address of the MultiVu computer.
3. Launch MultiVu. If debugging the code, you may use Simulation Mode for MultiVu on a PC not connected to the QD instrument hardware.
4. If in remote mode, run `QDInstrument_Server.exe` on the MultiVu computer.
5. Run the VI and watch in MultiVu to see the chamber purge, then the temperature and field steps.

6. For troubleshooting remote connections, a log of the QDInstrument network connection is kept in the file `C:\QDLogs\QDInstrument\Event.log` on the LabVIEW computer.

Creating New LabVIEW Programs Using QDInstrument VIs

1. Start with an `OpenQDInstrument.vi`. Wire a Ring Control or Ring Constant to its "Instrument Type" input. Choose the appropriate instrument: PPMS, VersaLab, DynaCool, or SVSM (MPMS3). If you are using Local Mode, set the "Remote" input to False (the "IP Address" input is ignored in this case). If you are using Remote Mode, set the "Remote" input to True and set the "IP Address" input to the IP address of the MultiVu computer found in the setup section.
2. Wire the "Instrument Ref" output of `OpenQDInstrument.vi` to the other QDInstrument VIs, such as `SetTemperature.vi`. This LabVIEW RefNum is used by all QDInstrument VIs to refer to the instrument. You may use this RefNum for LabVIEW flow control.
3. You may wish to wire up the error clusters as well for additional flow control. This is especially useful for use with non-QD VIs because they do not use the QDInstrument reference for flow control.
4. Integer inputs and outputs of the VIs, such as approach modes and status codes, are enumerated. As a result, if you right-click on the connection for one of these values and select Create→ Control, Create→ Indicator, or Create→ Constant, LabVIEW will create enumerated rings to help you set and read these values.
5. At the end of your VI, connect the "Instrument Ref" RefNum to a .NET Close Reference VI in order to avoid a memory leak in LabVIEW. The example `QDInstrument_Example.vi` shows how to do this.

The library `QDInstrument.llb` is a collection of LabVIEW VIs which use QDInstrument and contains the following:

- `OpenQDInstrument.vi`: Gets a RefNum reference for communication to MultiVu for use by the rest of the QDInstrument VIs.
- `SetTemperature.vi`: Sets temperature, rate, and approach mode.
- `GetTemperature.vi`: Reads present temperature and temperature status.
- `SetField.vi`: Sets magnetic field, rate, approach mode, and end mode.
- `GetField.vi`: Reads present magnetic field and field status.
- `SetChamber.vi`: Sends sample chamber commands such as purge and seal.
- `GetChamber.vi`: Reads sample chamber status.
- `SetPosition.vi`: Sets rotator position.
- `GetPosition.vi`: Reads present rotator position.
- `WaitFor.vi`: Waits for stability of requested subsystems (Temperature, Field, Rotator Position, and Chamber) and then waits for a specified amount of time.

NOTE: the Position subsystem is compatible with DynaCool Release 1.0.4 and later. Other platforms do not support the position commands as of this revision (07/2020). Rotator position can still be controlled on PPMS using the SendPPMSCommand_Rotator VI (see the next section).

NOTE: when using the PPMS with temperature control at a user thermometer, a simple addition ($ALT_TEMP command) needs to be made to the user thermometer .CFG file before sending it to the Model 6000. Otherwise, `GetTemperature.vi` will read the block temperature instead of the probe temperature. Relevant cases include the rotator probe, multifunction probe (MFP), or any custom probe that uses a .cfg file with a USERTEMP command for the user thermometer. See the files in this folder on Pharos for details: https://www.qdusa.com/pharos/browse.php?fFolderId=417; this is only necessary on the PPMS system.

Getting PPMS Data Items and Low-Level PPMS Control in LabVIEW using QDInstrument VIs

For most applications, you can get the data you need from the PPMS and access normal PPMS controls with the basic QDInstrument VIs. However, you may need other data items (e.g. rotator position) or low level control (e.g. analog and digital outputs) from the PPMS Model 6000. The following examples show how to do this.

To get PPMS data items:

1. On the LabVIEW computer, you should have the PPMS folder which contains `GetPPMSItem.vi` and `GetPPMSItem_Example.vi`.
2. If using remote mode, start `QDInstrument_Server.exe` on the MultiVu computer.
3. Open `GetPPMSItem_Example.vi`. If using remote mode, set remote and set IP address to the address of the MultiVu computer.
4. Set the Channel to 3.
5. Run the VI. You should see the rotator position reported in "PPMS Data"
6. Use `GetPPMSItem_Example.vi` as a template for creating your own VIs. Consult Table A-1 of the PPMS GPIB Commands Manual available on Pharos at: https://www.qdusa.com/pharos/view.php?fDocumentId=328 for mapping the channel. Note in that table that the channel number is referred to as a bit.
7. You can use the same "Instrument Ref" for the `GetPPMSItem_Example.vi` as for the other QDInstrument VIs.

To access low-level Model 6000 controls by sending GPIB commands and receiving replies from the Model 6000, use the procedure described below. This functionality is a replacement for sending GPIB commands to the Model 6000 using other techniques, such as WinWrap scripts (using the SendPpmsCommand function) or custom software. On the LabVIEW computer, you should have the PPMS folder which contains `SendPPMSCommand.vi`, `SendPPMSCommand_Rotator.vi` and `SendPPMSCommand_Example.vi`. If using remote mode, start `QDInstrument_Server.exe` on the MultiVu computer. First we will cover the example of controlling the rotator on the PPMS:

1. Open `SendPPMSCommand_Rotator.vi`. If using remote mode, set remote and set IP address to the address of the MultiVu computer.
2. The dialog to set the rotator position allows you move to a position, go to the limit switch (called “go to index” in motion dialog of MultiVu) or redefine current position to the position value that is

entered. The slow down code is described in the PPMS GPIB Commands Manual mentioned above.

3. The MOVE command for setting rotator position, like temperature and field setting commands, does not contain a wait condition so we added a wait dialog in this VI that uses the GetPPMSItem command to query the PPMS status word.
4. The “Final Position” box demonstrates the use of the SendPPMSCommand to read the rotator position (instead of the usual GetPPMSItem VI).

For an example of controlling the digital output of the Model 6000, see this example:

1. Open `SendPPMSCommand_Example.vi`. If using remote mode, set remote and set IP address to the address of the MultiVu computer.
2. In MultiVu, select “Instrument->Digital Output…”. If “Aux Driver 1” reads “ON”, then uncheck “Aux Driver 1” and press “Set” below “Digital Outputs”.
3. Run `SendPPMSCommand_Example.vi`.
4. In MultiVu, you should see “Aux Driver 1:” reading “ON”. In LabVIEW,” PpmsReply” should be blank indicating a successful “DIGSET 1” GPIB command. “PpmsReply 2” should read “1”, the value returned from the GPIB query “DIGSET?”.
5. Use `SendPPMSCommand_Example.vi` as a template for creating your own VIs. Consult the PPMS GPIB Command Manual for details on GPIB commands you use with the Model 6000.
6. You can use the same "Instrument Ref" for the `QDInstrument_Example.vi` as for the other QDInstrument VIs.

Getting CAN Data Items in LabVIEW using QDInstrument VIs

For most applications, you can get the necessary data from the standard VIs provided. However, in case a CAN data item is needed, this example shows how it is done:

1. On the LabVIEW computer, you should have the CAN folder which contains `CAN_QDInstrument.llb` and `CAN_Float_Example.vi`.
2. If using remote mode, start `QDInstrument_Server.exe` on the MultiVu computer.
3. Open `CAN_Float_Example.vi`. Set the instrument type: PPMS, VersaLab, DynaCool or SVSM (MPMS3). If using remote mode, set remote and set IP address to the address of the MultiVu computer.
4. Set the CAN Module Node ID to an existing node on your system. For DynaCool and SVSM, you can use node 3 for the TCM. For VersaLab, use node 2 for the VersaLab controller.
5. Run the VI. The SDO preloaded in this example is that of the temperature readback of the temperature controller. You should read a nonzero value in the field "SDO Float Result".
6. Use `CAN_Float_Example.vi` as an example for creating your own VIs. You probably need to ask Quantum Design for information about which CAN SDOs to read (node, index and subindex).
7. You can use the same "Instrument Ref" for the CAN VIs as for the QDInstrument VIs.

Troubleshooting

Below are the most common solutions offered when customers report problems in communicating between computers or programs:

1. Make sure .NET 3.5 is installed on both computers. In our experience it is not always sufficient to have a newer .NET installation (e.g., 4.0).
2. Windows Firewall may be blocking the TCP call from the remote computer. Please see this screencast video on Pharos which shows how to open a TCP port for communication when Windows Firewall is activated: https://www.qdusa.com/pharos/view.php?fDocumentId=1394
3. (PPMS only) make sure you have the latest release of PPMS MultiVu available from www.qdusa.com
4. After installing MultiVu software, it needs to be listed as an OLE server in the Windows registry. This is done by right-clicking on the MultiVu icon and selecting “run as administrator”. After MultiVu launches, you can exit again. This only has to be done once.
5. On computers with UAC (user account controls) enabled, you must run MultiVu and `QDInstrument_server.exe` using the same administrator privilege level: either both as administrator or both as non-administrator. When in local mode, LabVIEW must have same privilege level as MultiVu.
6. (this case is probably quite rare) When in remote mode: in the event that the TCP port needs to be changed from its default value of 11000 due to port conflicts, it is accessible under Options in QDInstrument_server.exe and in the Block Diagram for `OpenQDInstrument.vi` in LabVIEW. The value obviously needs to match both these places and also be a port that is not in use on either computer.
7. Security settings on certain Windows-based PCs may flag the downloaded `QDInstrument.dll` as potentially unsafe and block execution. This will lead to VIs in the QDInstrument package failing to run properly, usually accompanied by a LabVIEW error stating ‘Error 1386 – The specified .NET class is not available in LabVIEW’.

   This issue can be resolved by navigating to the location where `QDInstrument.dll` is saved, right-clicking to access the properties menu, and clicking the ‘Unblock’ button at the bottom of the dialog:

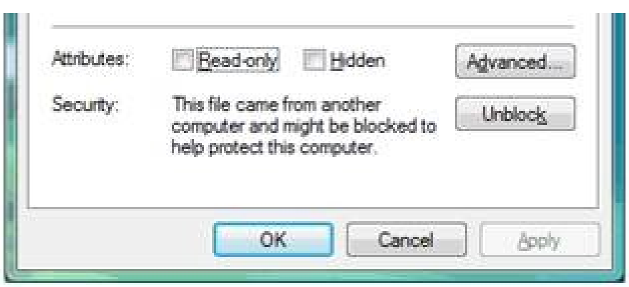

Figure 4: Windows dialog showing the 'Unblock' option.

Appendix A: Historical Overview of Custom Experiments on PPMS

If one wanted to perform any measurement outside the scope of MultiVu, there were historically 3 options:

1. Advisories (PPMS only): these are commands in MultiVu sequences which trigger an external program (C++, Delphi, Visual Basic) to perform a task. Shortcomings of this method are that it requires the user to write such a program and run it on the same MultiVu computer, it provides only primitive and one-way communication from MultiVu, and is only available on the PPMS. This method is described in PPMS Application Note 1070-202 at www.qdusa.com while 3rd party instrument sample programs are available at: http://www.qdusa.com/techsupport/softwareUpgrades.html.
2. Scripting within MultiVu using WinWrap Basic editor (available on all systems): think of scripts here as enhanced MultiVu sequences that have the full power of the Visual Basic programming language with its full command set. These are a very convenient way of enhancing an existing measurement sequence without going outside of MultiVu. A shortcoming for use on PPMS: there is currently (March 2013) only one GPIB bus that can be addressed within WinWrap so any external instrument must use another bus (USB, Ethernet, serial) or share the GPIB bus with the PPMS (not ideal due to possible GPIB settings conflicts and heavy traffic to Model 6000). This will be corrected in future versions of PPMS MultiVu. See Application Note 1070-209 at www.qdusa.com along with the attached example programs for more information on scripting.
3. LabVIEW (PPMS only): in the absence of a package provided by Quantum Design, users have written LabVIEW VIs (virtual instruments) which controlled the PPMS by issuing GPIB commands directly to the Model 6000. Drawbacks to this method are that:
    a. MultiVu is usually required to be closed due to conflicts between MultiVu and LabVIEW at the Model 6000. This presents a serious problem in cases where MultiVu is required to be running in order to handle data “redirection” such as temperature (e.g., with Helium-3 or Dilution Refrigerator options) or magnetic field (when using our new CAN-based magnet power supplies).
    b. The VIs will only work on the PPMS and not the CAN-based DynaCool, Versalab or MPMS3 systems.

It is clear that many customers prefer LabVIEW for running their custom experiments so a solution to LabVIEW-MultiVu integration was needed across all systems.

Appendix B: More detail about MultiVu, OLE and .NET

Modern Quantum Design instruments all use MultiVu software for control and monitoring of the instrument. Different instrument platforms (PPMS, DynaCool, VersaLab, and MPMS3) use different varieties of MultiVu, but much in MultiVu is the same on all systems, such as many user-interface elements. Most importantly for communicating with QD instruments from third party software, all four varieties of MultiVu make available the same interface for common operations including setting and reading temperature, magnetic field, and chamber gas state. The interface in MultiVu is Microsoft

Object Linking and Embedding (OLE, also known as ActiveX). MultiVu is an OLE server, so any program that can act as an OLE client can connect to the MultiVu OLE server to access temperature, field, and chamber gas functionality.

LabVIEW can act as an OLE client, so in principle it can connect directly to MultiVu. Due to variations in the way OLE is implemented, making a connection in this way between LabVIEW and MultiVu is not reliable. Another issue with this approach is that each variety of MultiVu looks like a different program to LabVIEW, so different LabVIEW VI files would be needed for each QD system. It is preferable to have one set of VI files that work with all QD systems.

LabVIEW works very well with a newer Microsoft technology, .NET. QDInstrument.dll provides a bridge between MultiVu (using OLE) and LabVIEW (using .NET). QDInstrument.dll communicates with MultiVu using OLE, and provides access to the temperature, field, and chamber gas functionality on its .NET interface for use by LabVIEW.

Additionally, `QDInstrument.dll` makes it possible to use the same LabVIEW VI files on all four modern QD platforms. It provides a generic type for all platforms called QDInstrument. When a VI creates an instance of this type in LabVIEW, the VI specifies what platform to connect to, and then `QDInstrument.dll` connects to the appropriate variety of MultiVu.

`QDInstrument.dll` provides another important function: the ability to make a remote connection, with LabVIEW and MultiVu on separate computers. An additional program, `QDInstrument_Server.exe`, provides remote access to the temperature, field, and chamber gas functionality of MultiVu. When instructed to use a remote connection, `QDInstrument.dll` communicates with `QDInstrument_Server.exe` (over the local area network) instead of MultiVu (using OLE). The LabVIEW VI files are the same for local and remote modes: the VI simply specifies remote mode (and the IP address of the MultiVu computer) when creating an instance of QDInstrument.