SHELDON INSTRUMENTS, INC.
QuVIEW For Use With LabVIEW
QuBASE For Use with Visual Basic

June 2011

Table of Contents

1.0 QuX Software Installation.

It is imperative that all QuX files be loaded into the correct location for proper operation.

1.1 QuX Files Organization.

All QuX discs contain at minimum these folders which are organized as follows:

Source Folder				Description
QuX->				QuBASE/QuVIEW files.
	QuBASE->			
		QuBase_net->		QuBASE files for Visual Basic.NET 2003.
	QuVIEW->			
		LabVIEW5->		
			si.lib->	All QuVIEW library files for LabVIEW 5.x
			menus\QuVIEW	QuVIEW 'root.mnu' menu file for LabVIEW 5.x.
		Sheldon_LV5->		All QuVIEW examples and demos for LabVIEW 5.x
		LabVIEW6->		
			si.lib->	All QuVIEW library files for LabVIEW 6.x
			menus\QuVIEW	QuVIEW 'root.mnu' menu file for LabVIEW 6.x.
		Sheldon_LV6->		All QuVIEW examples and demos for LabVIEW 6.x
		LabVIEW7->		
			si.lib->	All QuVIEW library files for LabVIEW 7.x
			menus\QuVIEW	QuVIEW 'root.mnu' menu file for LabVIEW 7.x.
		Sheldon_LV7->		All QuVIEW examples and demos for LabVIEW 7.x
		LabVIEW8->		
			si.lib->	All QuVIEW library files for LabVIEW 8.x
		Sheldon_LV8->		All QuVIEW examples and demos for LabVIEW 8.x
	SIC30DSP->			All low level QuX files, essential binaries for the host, and DSP COFF files. Used by both QuBASE as well as QuVIEW.

1.2 QuVIEW Files Installation.

***NOTE**:*
There are two steps that must be first performed in order for the SI hardware to operate correctly:
1) Drivers for all SI-DSP hardware. Please consult the '\DOCS\DRIVERS\README-DRIVERS.RTF' document for more details.
2) For certain SI hardware, the FPGA loader must be invoked in order to load the onboard logic. The FPGALoader must be run at least once before any DSP software can run; it is recommended that a batch file be executed just once every time the computer is booted. Please consult the '\DOCS\SIDSP\FPGALoad\FPGALOAD.RTF' document for more details.

Once the drivers and onboard logic have been successfully loaded, QuVIEW installation may begin. Please copy the directories from the CD onto your hard drive as follows:

Source Directory	Destination Directory on Hard Disk
\QuX\SIC30DSP	**C:\SIC30DSP** (this folder must reside on the C drive root)
\QuX\QuVIEW\SHELDON_LVx	**C:\..\SHELDON** (this folder can be placed anywhere)
\QuX\QuVIEW\LABVIEWx\SI.LIB	**C:\..\LABVIEWx\SI.LIB**
\QuX\QuVIEW\LABVIEWx\MENUS\QuVIEW	**C:\..\LABVIEWx\MENUS\QuVIEW** (LabVIEW 8.x has a different menu palette, so this step is ignored.)

***NOTE**:*
1) Files directly copied from CDs to hard drives retain their read-only attributes. This attribute needs to be removed once the files have been copied over to the hard drive. The quickest method is to highlight all files within a directory, right click with the mouse to invoke the 'Properties' menu item, and clear the adjacent check box. This must be performed for all subdirectories copied onto the hard drive.
2) Notice the QuVIEW path structure on the CD directly mirrors the path structure that must be followed with the installed files on your system. For example, the '..\QuVIEW\root.mnu' folder and file under the '\LabVIEWx\menus\' folder on the CD, must be placed under the same '\..\LabVIEW\menus\' folder on your system.
3) Once all of the QuVIEW libraries and demo files are installed, it is strongly advisable to add them to LabVIEW's search path defaults. Adding QuVIEW to the LabVIEW search path defaults enables their quick and automatic loading of applications as well as mass compiling, described in the next section.
4) Once QuVIEW files are added to LabVIEW's search paths, it is always advisable to perform a 'mass compile.' Mass compiling LabVIEW files ensures concurrency between the installed version of LabVIEW and the newly installed library files, described in the next section.
5) In order to view the QuVIEW icon palettes within the LabVIEW5-7 environment, select the 'QuVIEW' menu as 'default', described in the next section. This step is ignored with LabVIEW8.x.
6) Before running any of the QuVIEW demos and apps, be sure that the Windows drivers are installed and the subsequent the FPGALoader has successfully loaded the onboard logic.

1.2.1 Adding QuVIEW to LabVIEW Paths.

After QuVIEW files have been installed, it is strongly advised to include them into LabVIEW's search path defaults. This can be done in the following ways:

1) Using the LabVIEW's graphical editor.

LabVIEW 5.x.
Select 'Edit>>Preferences....>>Paths>>VI Search Path' which will invoke the 'Preferences' dialog box. Remove or unclick the 'Use Default' setting so that you are able to add both the '\..\LabVIEW\SI.LIB*' and 'C:\SIC30DSP*' paths to the overall list.

LabVIEW 6.x through LabVIEW 8.x.
Select 'Tools>>Options.....>>Paths>>VI Search Path' which will invoke the 'Preferences' dialog box. Remove or unclick the 'Use Default' setting so that you are able to add both the '\..\LabVIEW\SI.LIB*' and 'C:\SIC30DSP*' folder paths to the overall list.

NOTE:
1) The "/" characters must be added to the end of the selected paths in order to ensure that all items inside of this folder are accessed, and each folder's path must be separated by a semicolon.*

2) Manually editing the LabVIEW.INI file.
Look for the 'viSearchPath' line, and append the complete paths for the '\..\LabVIEW\SI.LIB*' and 'C:\SIC30DSP*' folders. An example of this line is as follows:

viSearchPath="<topvi>:*;<foundvi>:\;<vilib>:*;<userlib>:*;<instrlib>:*;C:\Program Files\LABVIEW\si.lib*;C:\SIC30dsp*"

Once the paths have been added, LabVIEW must be shut down and restarted in order for these changes to take effect. From now on, LabVIEW will automatically include these paths to load QuVIEW libraries for applications as well as mass compiling. A mass compile of all QuVIEW libraries and demos is also highly recommended in order to ensure concurrency between the installed version of LabVIEW and the newly installed QuVIEW files.

1.2.2 Viewing QuVIEW Menus Under LabVIEW.

1) LabVIEW 5.x.
The newly installed QuVIEW library menus can be seen as regular icons within the 'Controls' and 'Function' palettes in LabVIEW 5.x by manually selecting the “QuVIEW” palettes under the ‘Edit>>Select Palette Set’ option.

2) LabVIEW 6.x and LabVIEW 7.x.
The newly installed QuVIEW library menus can be seen as regular icons by manually selecting the “QuVIEW” palettes after pressing the ‘Options’ button. The ‘Options’ button appears as the right most or the last of three buttons on the Functions Palette, and can only be selected graphically.

3) LabVIEW 8.x.
The newly installed QuVIEW library function menus can be seen as a separate category with its own icons by following these steps:

a) Invoke the '>>Tools>>Advanced>>Edit Palette Set...' menu in order to access the “Edit Controls and Functions Palette Set” dialog box.
b) On the separate main 'Functions' menu box, right click on an empty area (preferably the last one typically labeled 'Select a VI...') in order to invoke the 'Insert Subpalette' dialog box.
c) Select the 'Link to an existing palette file (.mnu)' option, which in turn will direct you to search for a file with a standard dialog box. Search for and select the 'si.mnu' file located inside of the '..\LabVIEW 8.0\si.lib\simenu'. You should immediately see the 'QuVIEW' subpalette on the main functions palette.
d) Repeat this process for the main 'Controls' palette menu.
e) Lastly, select the “Save Changes” option on the 'Edit Controls and Functions Palette Set' dialog box.

Note that you will see a new subfolder created by LabVIEW 8, named 'C:\Documents and Settings\Administrator\My Documents\LabVIEW Data\8.0\Palettes\menus\default' inside of your Windows 'My Documents' folder. You will also notice changes to the 'LabVIEW.ini' file.

1.3 QuBASE Files Installation.

There is no specific folder to place the QuBASE files. However, despite the all of the solutions being delivered in working fashion, the library links pointing to the subfolders and files located inside of the QuBASE_lib folder, may be broken. These links must be fixed in order for the solutions to compile error free.

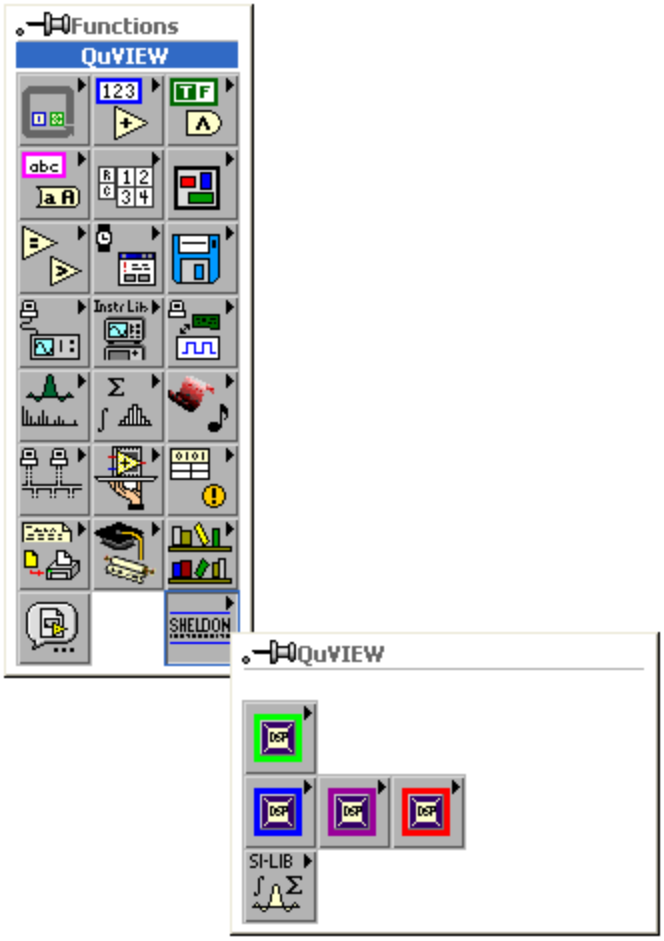
1.4 Supplemental Files Organization.

These supplemental files include the documentation as well as the drivers for your hardware.

Source Directory	Description
SI_Support->	
Docs->	All Sheldon Instruments documentation.
sidev->	Precompiled binaries (drivers), command line examples, and all source code for custom development.
binaries->	Precompiled drivers for all SI hardware, along with precompiled examples for custom development.
drivers->	All SI hardware drivers for Windows, used for QuX and custom development.
fpgaloader->	FPGA loader utility.
sihw_apps->	Command line utilities for board diagnosis used in custom development.
siddk->	Host side source code for custom development.
sidsp->	DSP side source code for custom development.

2.0 QuVIEW: DSP Real Time Acquisition, Signal Processing and Control.

The Sheldon Instruments' QuVIEW library VI menu is located at the bottom right corner of the Functions menu.



These VIs are used for implementing ultra high speed real time processing, with the added flexibility of using LabVIEW icons. A library of functions for real time acquisition, signal processing and control exists for algebra, calculus, digital filtering, Z transforms, frequency domain analysis, numerical analysis, waveform generation, and process control, which are constantly being expanded.

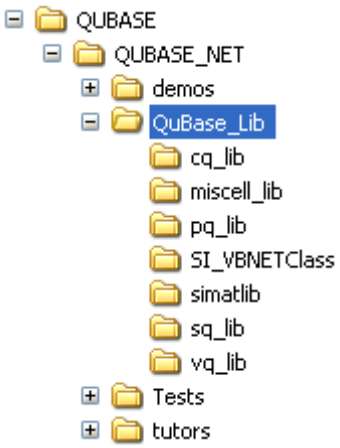
Two distinct methods of data processing within the DSP are implemented, and hence the libraries are divided in such a way so as to make this distinction more obvious. One method, categorized as "Point Queue" (VIs denoted with a "PQ_" prefix), is better suited for real time and continuous data processing in which data is processed on a point by point (sample by sample) basis and a result is returned before the next point is to be processed. These VIs are predominantly colored blue.

The other method, categorized as "Vector Queue" (VIs denoted with a "VQ_" prefix), is better suited for processing blocks of data points in which data is processed after the entire block has

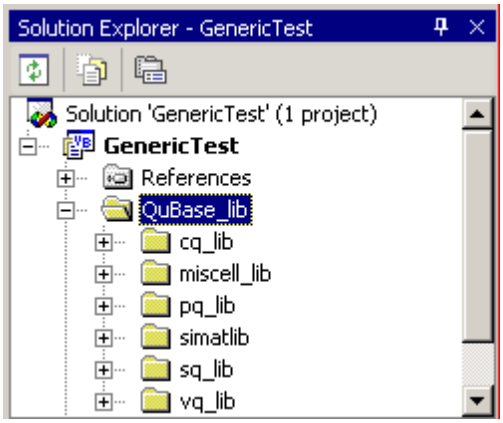
been received. These VIs are predominantly colored purple.

QuBASE Equivalent.

Path structure for QuBASE libraries.



QuBASE library as seen from Solution Explorer:



2.1 Building a QuVIEW Application.

The diagram for a QuVIEW application is very similar to that of a normal LabVIEW application. However, there are a few important issues which must be considered when constructing a VI that uses QuVIEW libraries.

Please refer to the file Tutorial.rtf in the same folder as this document. This document provides instructions for using the QuVIEW function libraries and building DSP programs.

2.2 Common Terminals to All QuVIEW VIs.

There are a number of terminals common to all QuVIEW VIs:

Board = Board Number of board installed. On all VIs, this is required terminal to wire.

Daughter Module = LabVIEW constant that defines the type of multifunction I/O module in place, it contains information about the daughter module converters' resolution and voltage scale.

Error In = Error input cluster connection that is daisy chained with the "Error Out" cluster terminals of previous VIs in a data flow chain. Besides being used for error detection, wiring these terminals ensures deterministic data flow operation.

Error Out = Error output cluster connection that is daisy chained with the "Error In" cluster terminals of subsequent VIs in a data flow chain. Besides being used for error detection, wiring these terminals ensures deterministic data flow operation.

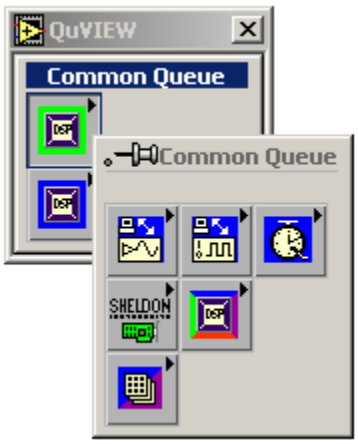
NOTE: *Input and Output connection terminal descriptions are color coded.* ***BLACK*** *is used to highlight input terminals, while* ***NAVY BLUE*** *highlights output terminals.*

2.3 Common Entry Equivalents for All QuBASE Applications.

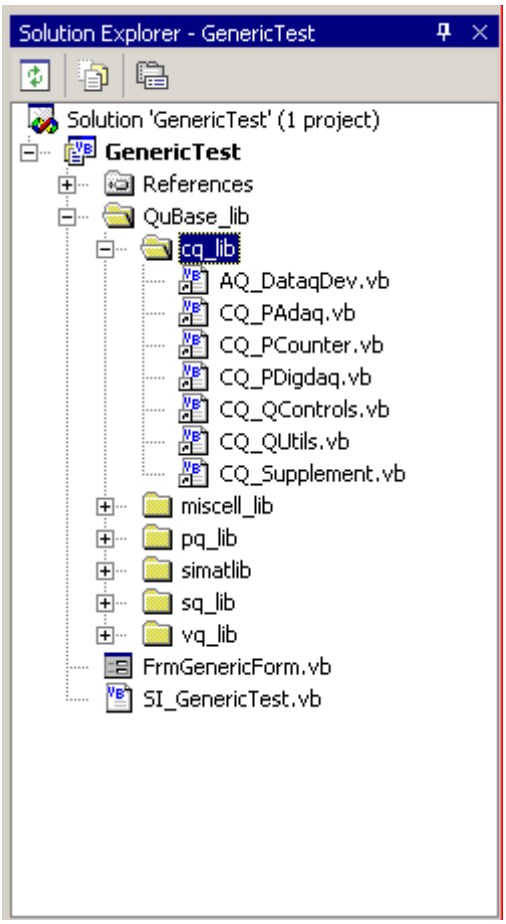
There are a number of equivalent entries common to all QuBASE applications:

3.0 Common Queue VIs

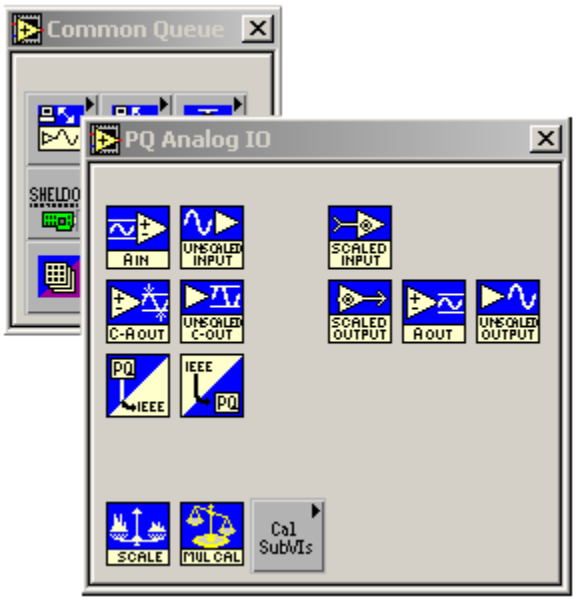
The Common Queue VIs are highlighted with a green border.



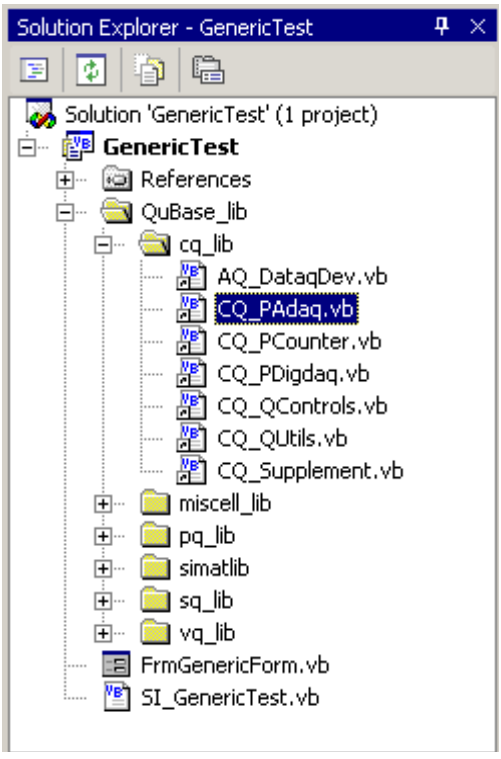
QuBASE Equivalent.



3.1 Common Queue: Analog I/O.

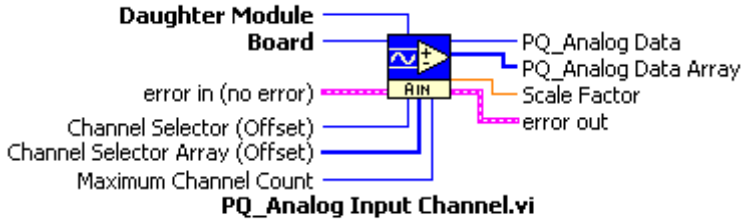


QuBASE Equivalent.



3.1.1 PQ_Analog Input Channel.

The PQ_Analog Input Channel VI integrates the complete functionality associated with reading one or more analog inputs. This functionality includes reading the raw, bipolar signed binary value, and then scaling and converting it to a floating point value that represent its voltage based on the ADC's resolution and signal range.



Daughter Module = LabVIEW constant that defines the type of multifunction I/O module in place, it contains information about the daughter module converters' resolution and voltage scale.

Channel Selector (Offset) = LabVIEW constant used to select a single analog input of interest. Ignored and not to be used if an array is wired to the 'Channel Selector Array (Offset)' terminal instead.

Channel Selector Array (Offset) = LabVIEW array of constants used to select an array of analog inputs. Convenient if many analog inputs are to be read simultaneously as it avoids the wiring of separate icons. If wired, it becomes the default channel selector and the single variable 'Channel Selector (Offset)' terminal is ignored.

Maximum Channel Count = LabVIEW constant used to limit the maximum number of elements in the 'Channel Selector Array' input. Allows the number of elements inside of the 'Channel Selector Array' to be altered in real time while a VI is running.

PQ_Analog Data = PQ output containing the actual scaled reading from a single selected analog input. If an array of channels is selected, only the first value of the array is returned.

PQ_Analog Data Array = PQ output array containing an array of scaled readings from the selected array of analog inputs. An empty array is returned if only the single input terminal is used.

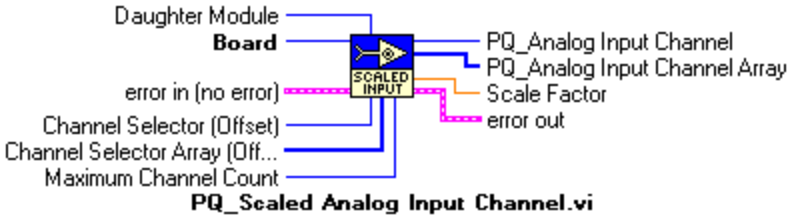
Scale Factor = LabVIEW output used to read the exact analog voltage scaling factor.

QuBASE Equivalent.

```
'**********************************************************************
'       Function PQ_AnalogInputChannel
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               daughterModule, Integer
'               channelSelectorArray, Integer
'               iMaxChannelCount, Integer
'
'       Output Parameters:
'               oPQAnalogDatalArray, Integer
'               oScaleFactor, Double
'
'**********************************************************************

  Function PQ_AnalogInputChannel( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef daughterModule As Integer, _
        ByRef channelSelectorArray() As Integer, _
        ByRef oPQAnalogDatalArray() As Integer, _
        Optional ByRef iMaxChannelCount As Integer = 8, _
        Optional ByRef oScaleFactor As Double = 0) _
        As T_Error
```

3.1.2 PQ_Scaled Analog Input Channel.

Same as PQ_Analog Input Channel.vi, except it uses several subVIs to perform same function, and therefore is not efficient. It is not recommended for use.

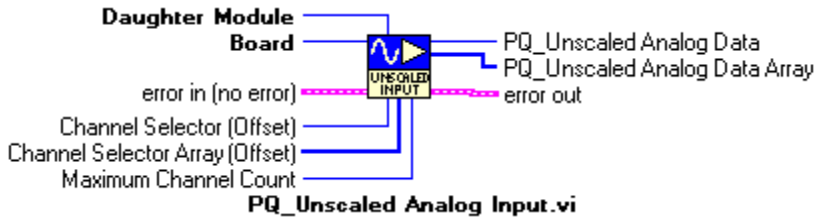


QuBASE Equivalent.

Unused, obsolete function.

3.1.3 PQ_Unscaled Analog Input.

The PQ_Unscaled Analog Input VI simply reads one or more analog inputs as raw, bipolar, signed binary values, no scaling or calibration takes place.



Daughter Module = LabVIEW constant that defines the type of multifunction I/O module in place, it contains information about the daughter module converters' resolution and voltage scale.

Channel Selector (Offset) = LabVIEW constant used to select a single analog input of interest. Ignored and not to be used if an array is wired to the 'Channel Selector Array (Offset)' terminal instead.

Channel Selector Array (Offset) = LabVIEW array of constants used to select an array of analog inputs. Convenient if many analog inputs are to be read simultaneously as it avoids the wiring of separate icons. If wired, it becomes the default channel selector and the single variable 'Channel Selector (Offset)' terminal is ignored.

Maximum Channel Count = LabVIEW constant used to limit the maximum number of elements in the 'Channel Selector Array' input. Allows the number of elements inside of the 'Channel Selector Array' to be altered in real time while a VI is running.

PQ_Unscaled Analog Data = PQ output containing the actual unscaled binary reading from a single selected analog input. If an array of channels is selected, only the first value of the array is returned.

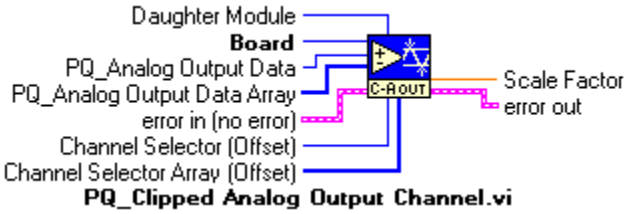
PQ_Unscaled Analog Data Array = PQ output array containing an array of unscaled binary readings from the selected array of analog inputs. An empty array is returned if only the single input terminal is used.

QuBASE Equivalent.

```
'**********************************************************************
'       Function PQ_UnscaledAnalogInput
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               iModule, Integer
'               channelSelectorArray, Integer
'               iMaxChannelCount, Integer
'
'       Output Parameters:
'               oPQUnscaledAnalogDataArray, Integer
'
'**********************************************************************

  Function PQ_UnscaledAnalogInput( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef iModule As Integer, _
        ByRef iChannelSelectorArray() As Integer, _
        ByRef oPQUnscaledAnalogDataArray() As Integer, _
        Optional ByRef iMaxChannelCount As Integer = 8) _
        As T_Error
```

3.1.4 PQ_Clipped Analog Output Channel.

The PQ_Clipped Analog Output Channel.vi integrates the complete functionality associated with writing one or more analog outputs. This functionality includes writing the voltage represented as a scaled floating point value, and then unscaling and converting it to the raw, bipolar signed binary value based on the DAC's resolution and signal range, as well as clipping the limits so as to avoid wrapping.



Daughter Module = LabVIEW constant that defines the type of multifunction I/O module in place, it contains information about the daughter module converters' resolution and voltage scale.

Channel Selector (Offset) = LabVIEW constant used to select a single analog output of interest. Ignored and not to be used if an array is wired to the 'Channel Selector Array (Offset)' terminal instead.

Channel Selector Array (Offset) = LabVIEW array of constants used to select an array of analog outputs. Convenient if many analog outputs are to be written simultaneously as it avoids the wiring of separate icons. If wired, it becomes the default channel selector and the single variable 'Channel Selector (Offset)' terminal is ignored.

PQ_Analog Output Data = PQ input containing the actual scaled value to be written to a single selected analog output. Ignored and not to be used if an array is wired to the 'PQ_Analog Output Data Array' terminal instead.

PQ_Analog Output Data Array = PQ input array containing an array of scaled values to be written to an array of analog outputs. Should be used if the Channel Selector Array terminal is used, since both array sizes must be the same and have a one to one correspondence.

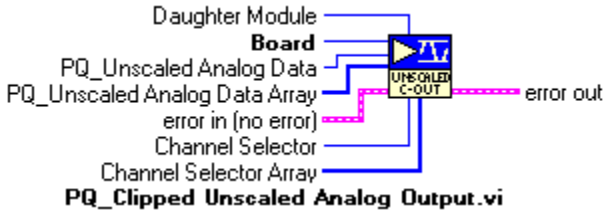
Scale Factor = LabVIEW output used to read the exact analog voltage scaling factor.

QuBASE Equivalent.

```
'**************************************************************************
'           Function PQ_ClippedAnalogOutputChannel
'
'           Description:
'
'           Input Parameters:
'                       board, Integer
'                       errIn, T_Error
'                       daughterModule, Integer
'                       channelSelectorArray, Integer
'                       pqOutputDataArray, Integer
'
'           Output Parameters:
'                       oScaleFactor, Double
'
'**************************************************************************

    Function PQ_ClippedAnalogOutputChannel( _
            ByRef board As Integer, _
            ByRef errIn As T_Error, _
            ByRef daughterModule As Integer, _
            ByRef pqOutputDataArray() As Integer, _
            ByRef channelSelectorArray() As Integer, _
            Optional ByRef oScaleFactor As Double = 0) _
            As T_Error
```

3.1.5 PQ_Clipped Unscaled Analog Output.

The PQ_Clipped Unscaled Analog Output VI simply writes one or more analog outputs as raw, bipolar, signed binary values, no scaling or calibration takes place except for the clipping of the limits so as to avoid wrapping.



Daughter Module = LabVIEW constant that defines the type of multifunction I/O module in place, it contains information about the daughter module converters' resolution and voltage scale.

Channel Selector (Offset) = LabVIEW constant used to select a single analog output of interest. Ignored and not to be used if an array is wired to the 'Channel Selector Array (Offset)' terminal instead.

Channel Selector Array (Offset) = LabVIEW array of constants used to select an array of analog outputs. Convenient if many analog outputs are to be written simultaneously as it avoids the wiring of separate icons. If wired, it becomes the default channel selector and the single variable 'Channel Selector (Offset)' terminal is ignored.

PQ_Unscaled Analog Data = PQ input containing the actual unscaled signed binary value to be written to a single selected analog output. Ignored and not to be used if an array is wired to the 'PQ_Unscaled Analog Data Array' terminal instead.

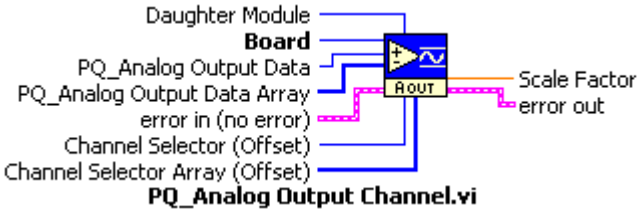
PQ_Unscaled Analog Data Array = PQ input array containing an array of unscaled signed binary values to be written to an array of analog outputs. Should be used if the Channel Selector Array terminal is used, since both array sizes must be the same and have a one to one correspondence.

QuBASE Equivalent.

```
'*********************************************************************
'       Function PQ_ClippedUnscaledAnalogOutput
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               iModule, Integer
'               channelSelectorArray, Integer
'               pqOutputDataArray, Integer
'
'       Output Parameters:
'
'*********************************************************************

   Function PQ_ClippedUnscaledAnalogOutput( _
          ByRef board As Integer, _
          ByRef errIn As T_Error, _
          ByRef iModule As Integer, _
          ByRef channelSelectorArray() As Integer, _
          ByRef pqOutputDataArray() As Integer) _
          As T_Error
```

3.1.6 PQ_Analog Output Channel.

The PQ_Analog Output Channel VI integrates the complete functionality associated with writing one or more analog outputs. This functionality includes writing the voltage represented as a scaled floating point value, and then unscaling and converting it to the raw, bipolar signed binary value based on the DAC's resolution and signal range.



Daughter Module = LabVIEW constant that defines the type of multifunction I/O module in place, it contains information about the daughter module converters' resolution and voltage scale.

Channel Selector (Offset) = LabVIEW constant used to select a single analog output of interest. Ignored and not to be used if an array is wired to the 'Channel Selector Array (Offset)' terminal instead.

Channel Selector Array (Offset) = LabVIEW array of constants used to select an array of analog outputs. Convenient if many analog outputs are to be written simultaneously as it avoids the wiring of separate icons. If wired, it becomes the default channel selector and the single variable 'Channel Selector (Offset)' terminal is ignored.

PQ_Analog Output Data = PQ input containing the actual scaled value to be written to a single selected analog output. If an array of channels is selected, this terminal is ignored.

PQ_Analog Output Data Array = PQ input array containing an array of scaled values to be written to an array of analog outputs. Should be used if the Channel Selector Array terminal is used, since both array sizes must be the same and have a one to one correspondence.

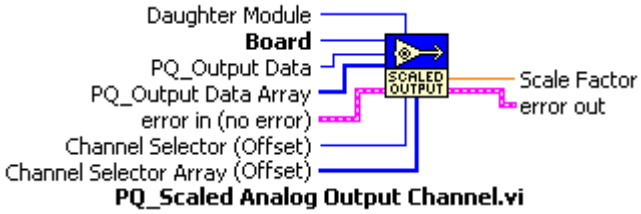
Scale Factor = LabVIEW output used to read the exact analog voltage scaling factor.

QuBASE Equivalent.

```
'**************************************************************************
'           Function PQ_AnalogOutputChannel
'
'           Description:
'
'           Input Parameters:
'                       board, Integer
'                       errIn, T_Error
'                       daughterModule, Integer
'                       channelSelectorArray, Integer
'                       pqOutputDataArray, Integer
'
'           Output Parameters:
'                       oScaleFactor, Double
'
'**************************************************************************

    Function PQ_AnalogOutputChannel( _
            ByRef board As Integer, _
            ByRef errIn As T_Error, _
            ByRef daughterModule As Integer, _
            ByRef pqOutputDataArray() As Integer, _
            ByRef channelSelectorArray() As Integer, _
            Optional ByRef oScaleFactor As Double = 0) _
            As T_Error
```

3.1.7 PQ_Scaled Analog Output Channel.

Same as PQ_Analog Output Channel.vi, except it uses several subVIs to perform the same function, and therefore is not efficient. It is not recommended for use.

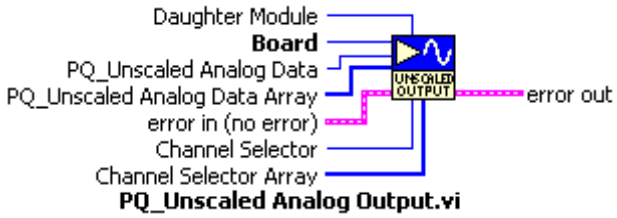


QuBASE Equivalent.

```
'*********************************************************************
'       Function PQ_ScaledAnalogOutput
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               daughterModule, Integer
'               channelSelectorArray, Integer
'               pqOutputDataArray, Integer
'
'       Output Parameters:
'               oScaleFactor, Double
'
'*********************************************************************

  Function PQ_ScaledAnalogOutput( _
        ByRef board As Integer, _
        ByRef daughterModule As Integer, _
        ByRef oScaleFactor As Double, _
        ByRef pqOutputData() As Integer, _
        ByRef channelSelectorArray() As Integer) _
        As T_Error
```

3.1.8 PQ_Unscaled Analog Output.

The PQ_Unscaled Analog Output VI simply writes one or more analog outputs as raw, bipolar, signed binary values, no scaling or calibration takes place.



Daughter Module = LabVIEW constant that defines the type of multifunction I/O module in place, it contains information about the daughter module converters' resolution and voltage scale.

Channel Selector (Offset) = LabVIEW constant used to select a single analog output of interest. Ignored and not to be used if an array is wired to the 'Channel Selector Array (Offset)' terminal instead.

Channel Selector Array (Offset) = LabVIEW array of constants used to select an array of analog outputs. Convenient if many analog outputs are to be written simultaneously as it avoids the wiring of separate icons. If wired, it becomes the default channel selector and the single variable 'Channel Selector (Offset)' terminal is ignored.

PQ_Unscaled Analog Data = PQ input containing the actual unscaled signed binary value to be written to a single selected analog output. If an array of channels is selected, this terminal is ignored.

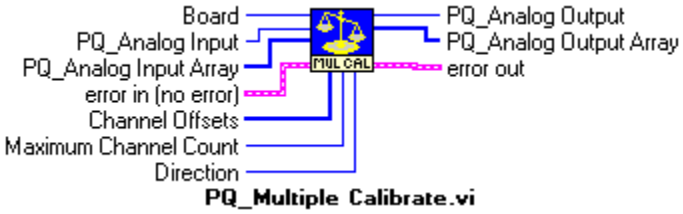
PQ_Unscaled Analog Data Array = PQ input array containing an array of unscaled signed binary values to be written to an array of analog outputs. Should be used if the Channel Selector Array terminal is used, since both array sizes must be the same and have a one to one correspondence.

QuBASE Equivalent.

```
'*************************************************************************
'       Function PQ_UnscaledAnalogOutput
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               iModule, Integer
'               channelSelectorArray, Integer
'               pqOutputDataArray, Integer
'
'       Output Parameters:
'
'*************************************************************************

   Function PQ_UnscaledAnalogOutput( _
          ByRef board As Integer, _
          ByRef errIn As T_Error, _
          ByRef iModule As Integer, _
          ByRef channelSelectorArray() As Integer, _
          ByRef pqOutputDataArray() As Integer) _
          As T_Error
```

3.1.9 PQ_Multiple Calibrate.

The PQ_Multiple Calibrate VI performs gain and offset calibration on an array of analog I/Os. The calibration error values are contained in a file with the *.TBL extension. For advanced users and is not recommended for use.

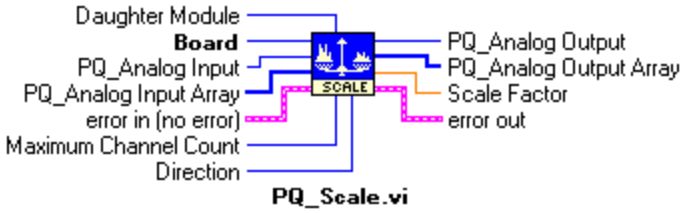


QuBASE Equivalent.

```
'*************************************************************************
'       Function PQ_MultipleCalibrate
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqAnalogInputArray, Integer
'               channelOffsetArray, Integer
'               iDirection, Integer
'               iMaxChannelsCount, Integer
'
'       Output Parameters:
'               oPQAnalogOutputArray, Integer
'
'*************************************************************************

  Function PQ_MultipleCalibrate( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef oPQAnalogOutputArray() As Integer, _
        ByRef pqAnalogInputArray() As Integer, _
        ByRef channelOffsetArray() As Integer, _
        Optional ByVal iDirection As Integer = 0, _
        Optional ByVal iMaxChannelsCount As Integer = 8) _
        As T_Error
```

3.1.10 PQ_Scale.

The PQ_Scale VI performs a conversion between binary and voltage values associated with ADCs and DACs. For advanced users and is not recommended for use.



QuBASE Equivalent.

```
'*************************************************************************
'       Function PQ_Scale
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               daughterModule, Integer
'               pqAnalogInputArray, Integer
'               direction, Integer
'               iMaxChannelsCount, Integer
'
'       Output Parameters:
'               oPQAnalogOutputArray, Double
'               oScaleFactor, Double
'
'*************************************************************************

  Function PQ_Scale( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef daughterModule As Integer, _
        ByRef oPQAnalogOutputArray() As Integer, _
        ByRef oScaleFactor As Double, _
        ByRef pqAnalogInputArray() As Integer, _
        Optional ByVal direction As Integer = 0, _
        Optional ByVal iMaxChannelsCount As Integer = 8) _
        As T_Error
```

3.1.11 PQ_PQ Source to IEEE.

The PQ_PQ Source to IEEE VI serves to transfer any PQ value or array of PQ values from the DSP domain to the LabVIEW/host domain. The floating point PQ values are read from the DSP and converted to the IEEE floating point format supported by LabVIEW. This function is mostly performed with minimal DSP intervention. Intended for advanced users.

PQ_Source = PQ input source to be transferred to the LabVIEW domain for immediate use. Ignored and not to be used if an array is wired to the 'PQ_Source Array' terminal instead.

PQ_Source Array = PQ input source array to be transferred to the LabVIEW domain for immediate use. Convenient if an array of variables is used as it avoids the wiring of separate icons. If wired, it becomes the default channel selector and the single variable 'PQ_Source Array' terminal is ignored.

IEEE Data Point = LabVIEW output variable represented as a single precision floating point number. Not to be used if an array is wired.

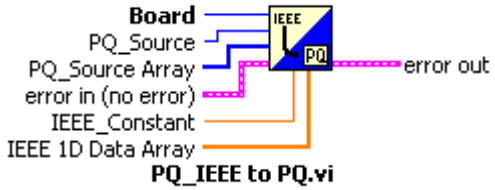
IEEE 1D Data Array = LabVIEW output array variable representing an array of single precision floating point numbers. If an input array was used, then the output array will be correspondingly used.

QuBASE Equivalent.

```
'*********************************************************************
'       Function PQ_PQToIEEE
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               ipqInputArray, Integer
'
'       Output Parameters:
'               oDataArray, Double
'
'*********************************************************************

  Function PQ_PQToIEEE( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef ipqInputArray() As Integer, _
        ByRef oDataArray() As Double) _
        As T_Error
```

3.1.12 PQ_IEEE to PQ Source.

The PQ_PQ Source to IEEE VI serves to transfer any PQ value or array of PQ values from the DSP domain to the LabVIEW/host domain. The floating point PQ values are read from the DSP and converted to the IEEE floating point format supported by LabVIEW. This function is mostly performed with minimal DSP intervention. Intended for advanced users.



PQ_Source = PQ source to receive the IEEE values for immediate use. Ignored and not to be used if an array is wired to the 'PQ_Source Array' terminal instead.

PQ_Source Array = PQ input source array to receive the IEEE array of values for immediate use. Convenient if an array of variables is used as it avoids the wiring of separate icons. If wired, it becomes the default channel selector and the single variable 'PQ_Source Array' terminal is ignored.

IEEE Constant = LabVIEW input variable represented as a single precision floating point number. Not to be used if an array is wired.

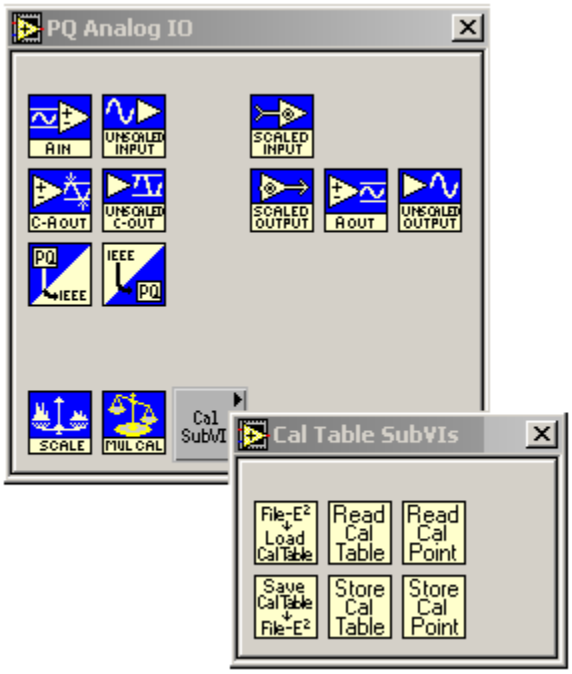
IEEE 1D Data Array = LabVIEW input array variable representing an array of single precision floating point numbers. Not used if the single 'IEEE Constant' input is the only terminal wired.

QuBASE Equivalent.

```
'*************************************************************************
'       Function PQ_IEEEToPQ
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               ipqInputArray, Integer
'
'       Output Parameters:
'               oDataArray, Double
'
'*************************************************************************

  Function PQ_IEEEToPQ( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef ipqInputArray() As Integer, _
        ByRef oDataArray() As Double) _
        As T_Error
```

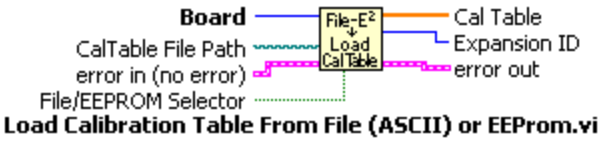
3.1.13 PQ Analog I/O: Calibration SubVIs.

A set of subVIs used for calibrating gain and offset errors present on analog I/O channels. The gain and offset errors are contained inside a file with a *.TBL extension which resides inside of the \SIC30DSP directory. These calibration errors may also be optionally burned into an EEPROM from this file. For advanced users and not intended for general use.



3.1.13.1 Load Calibration Table From File (ASCII) or EEProm.

The Load Calibration Table From File (ASCII) or EEProm VI Loads a Calibration Table from a file or optionally from an EEPROM. For advanced users and not intended for general use.



CalTable Path = Path to Calibration Table file.

File/EEPROM Selector = LabVIEW boolean input to define the source of the calibration table to be used by the DSP when calibrating analog IO in real time. The calibration table is first created and stored into a file typically named with a 'TBL' extension. Subsequently, the contents of the calibration file may also be optionally burned into an EEPROM on the DSP carrier card using this file as a template. The values of the calibration table reflect gain and offset errors of the SI-MOD's analog IO circuitry. When analog IO data are accessed from an SI-MOD card by the DSP, the calibration error adjustments are performed in real time.

False: File = Load Calibration Table from file.

True: EEPROM = Load Calibration Table from EEPROM.

NOTE:

1) This boolean is not available as an externally accessible wire terminal from the 'Initialize Board' VI, and therefore its desired position must be set as a default and then saved.

2) Please refer to the VI named 'Initialize Board' for more details on its use.

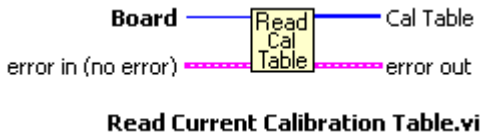
Cal Table = An array of values read from the Calibration Table file or EEPROM.

QuBASE Equivalent.

```
'*********************************************************************
'       Function CQ_LoadCalTablefromFileorEEPROM
'
'       Description:
'               loads entire cal table from cal file to host memory
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               calTable, Double
'               expansionID, Integer
'               fileEEPROM, Integer
'               calPath, String
'
'       Output Parameters:
'
'*********************************************************************

Function CQ_LoadCalTablefromFileorEEPROM( _
        ByVal Board As Long, _
        ByVal errIn As T_Error, _
        ByRef calTable() As Double, _
        Optional ByRef expansionID As Integer = 0, _
        Optional ByVal fileEEPROM As Integer = 0, _
        Optional ByVal calPath As String = kCalFileDefault) As T_Error
```

3.1.13.2 Read Current Calibration Table.

The Read Current Calibration Table VI reads from a Calibration Table already residing in LabVIEW/host memory. For advanced users and not intended for general use.



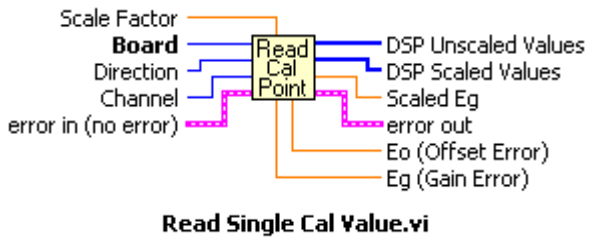
Cal Table = An array of calibration values.

QuBASE Equivalent.

```
'**************************************************************************
'       Function CQ_ReadCurrentCalTable
'
'       Description:
'               reads the entire cal table from host memory
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'
'       Output Parameters:
'               oCalTable, Double
'
'**************************************************************************

  Function CQ_ReadCurrentCalTable( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef oCalTable() As Double) _
        As T_Error
```

3.1.13.3 Read Single Cal Value.

The Read Single Cal Value VI reads a single calibration point for a given analog channel. For advanced users and not intended for general use.



Read Single Cal Value.vi

Direction = Denotes if the calibration is to apply to an analog input or an analog output.
Channel = Channel number of the analog I/O channel of interest.
Scale Factor = LabVIEW input that weights or scales the number of bits of the analog IO circuitry versus the voltage they represent.

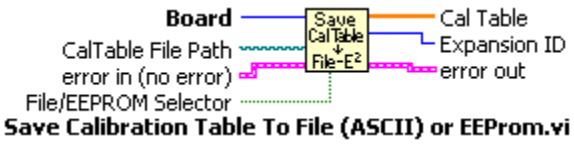
DSP Unscaled Values = A U32 array containing the Offset and Magnitude error values of a particular analog I/O channel of interest, converted to DSP format without any scaling.
DSP Scaled Values = A U32 array containing the Offset and Magnitude error values of a particular analog I/O channel of interest, converted to DSP format with scaling.
Scaled Eg = LabVIEW value representing the scaled gain or magnitude error.
Eo (Offset Error) = Offset error value to be applied to a particular analog channel of interest.
Eg (Gain Error) = Magnitude error value to be applied to a particular analog channel of interest.

QuBASE Equivalent.

```
'*********************************************************************
'       Function CQ_ReadSingleCalValue
'
'       Description:
'               reads a cal table value from host memory
'
'       Input Parameters:
'               iDirection, Integer
'               iChannel, Integer
'               iChannel, Double
'
'       Output Parameters:
'               oDSPUnscaledValues, Integer
'               oDSPScaledValues, Integer
'               oGain, Double
'               oScaledGain, Double
'               oOffset, Double
'
'*********************************************************************

   Function CQ_ReadSingleCalValue( _
            ByVal Board As Integer, _
            ByVal errIn As T_Error, _
            ByRef oDSPUnscaledValues() As Integer, _
            ByRef oDSPScaledValues() As Integer, _
            Optional ByVal iDirection As Integer = 1, _
            Optional ByVal iChannel As Integer = 0, _
            Optional ByVal iScaleFactor As Double = 1, _
            Optional ByRef oGain As Double = 1, _
            Optional ByRef oScaledGain As Double = 1, _
            Optional ByRef oOffset As Double = 0) As T_Error
```

3.1.13.4 Save Calibration Table to File (ASCII) or EEProm.

The Save Calibration Table to File (ASCII) or EEProm VI saves a Calibration Table into a file or into an EEPROM that resides on the DSP carrier card. Mainly used by the 'Save Calibration Table From File to the 66B EEProm' VI inside of the CALIBRATION.LLB library. For advanced users and not intended for general use.



CalTable File Path = Path to Calibration Table file.

File/EEPROM Selector = LabVIEW boolean input to define the source of the calibration table to be used by the DSP when calibrating analog IO in real time. The calibration table is first created and stored into a file typically named with a 'TBL' extension. Subsequently, the contents of the calibration file may also be optionally burned into an EEPROM on the DSP carrier card using this file as a template. The values of the calibration table reflect gain and offset errors of the SI-MOD's analog IO circutry. When analog IO data are accessed from an SI-MOD card by the DSP, the calibration error adjustments are performed in real time.

False: File = Save Calibration Table to file.
True: EEPROM = Save Calibration Table to EEPROM.

Cal Table = An array of values to be saved into the Calibration Table.

QuBASE Equivalent.

```
'*********************************************************************
'       Function CQ_SaveCalTabletoFileorEEPROM
'
'       Description:
'               saves/updates values from host memory to the cal file.
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               expansionID, Integer
'               iFileEEPROM, Integer
'               calPath, String
'
'       Output Parameters:
'               oCalTable, Double
'
'*********************************************************************

Function CQ_SaveCalTabletoFileorEEPROM( _
        ByVal Board As Long, _
        ByVal errIn As T_Error, _
        ByRef oCalTable() As Double, _
        Optional ByRef expansionID As Integer = 0, _
        Optional ByVal iFileEEPROM As Integer = 1, _
        Optional ByVal calPath As String = kCalFileDefault) As T_Error
```

3.1.13.5 Store Current Calibration Table.

The Store Current Calibration Table VI stores a calibration table into LabVIEW/host memory for current use. For advanced users and not intended for general use.

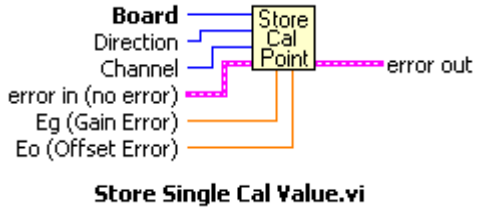
Cal Table = Array of Cal Table values.

QuBASE Equivalent.

```
'*************************************************************************
'       Function CQ_StoreCurrentCalTable
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               iCalTable, Double
'
'       Output Parameters:
'
'*************************************************************************

  Function CQ_StoreCurrentCalTable( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef iCalTable() As Double) _
        As T_Error
```

3.1.13.6 Store Single Cal Value.

Stores a calibration value of a single analog channel. For advanced users and not intended for general use.



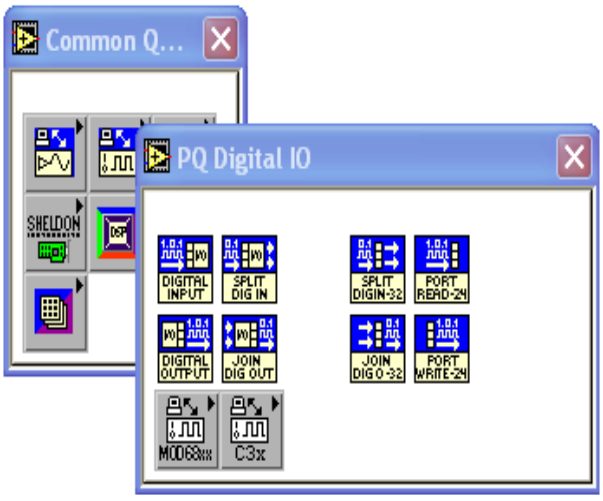
Direction = Denotes if the calibration is to apply to an analog input or an analog output.
Channel = Channel number of the analog I/O channel of interest.

Eo (Offset Error) = Offset error value to be applied to a particular analog channel of interest.
Eg (Magnitude Error) = Magnitude error value to be applied to a particular analog channel of interest.

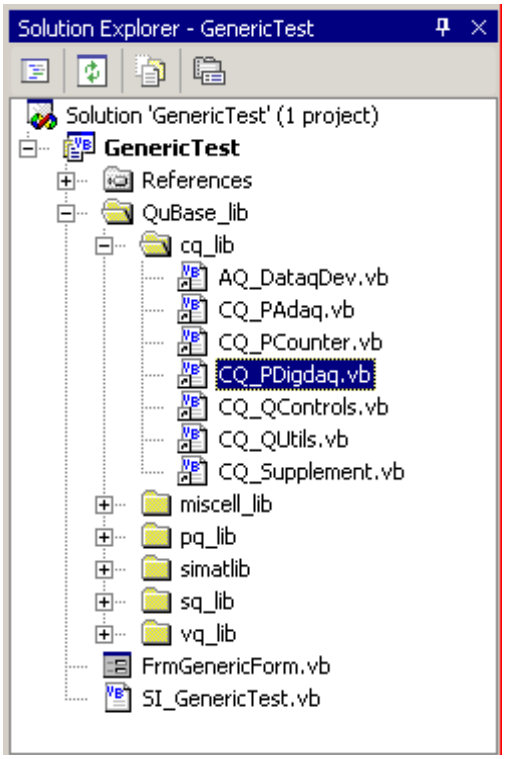
QuBASE Equivalent.

```
'**********************************************************************
'       Function CQ_StoreSingleCalValue
'
'       Description:
'               stores/updates a Double cal value to host memory
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               direction, Integer
'               channel, Integer
'               magnitude, Double
'               offset, Double
'
'       Output Parameters:
'
'**********************************************************************

    Function CQ_StoreSingleCalValue( _
            ByRef board As Integer, _
            ByRef errIn As T_Error, _
            Optional ByVal direction As Integer = 1, _
            Optional ByVal channel As Integer = 0, _
            Optional ByVal magnitude As Double = 0, _
            Optional ByVal offset As Double = 0) _
            As T_Error
```

3.2 Common Queue: Digital I/O.

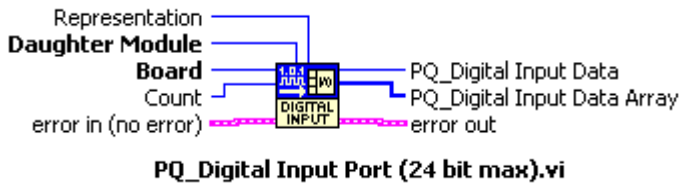


QuBASE Equivalent.



3.2.1 PQ_Digital Input Port (24 bit max).

The PQ_Digital Input Port (24 bit max) VI reads up to 24 bits of raw binary data present on the digital I/O port on the SI-MOD68xx module when it is configured as an input. The reads may be specified to occur sequentially on the same port within a single 'PQ' sample period.



Count = LabVIEW input, determines the size of the output array. A count of '1' returns a single element, while a count greater than '1' returns an array of data formed from sequentially reading from the same port within a single 'PQ' sample period. Defaulted to a value of '1'.

Representation = LabVIEW input, selects to read either signed or unsigned binary data from the digital I/O port; defaulted to read unsigned integer values.

PQ_Digital Input Data = PQ output, up to 24 bits of binary data read from the digital I/O port, represented as a floating point value. If an output array is selected, only the first value of the output array is returned.

PQ_Digital Input Data Array = PQ output array, up to 24 bits of a binary data read from an array of digital port, represented as floating point values. If only a single value is selected, the output array only returns a single element.

NOTE: *Although the SI-MOD68xx's digital IO port is 32 bits wide, only the lower 24 bits are accessible because of the DSP's floating point format using a 24 bit mantissa. For operations requiring the complete 32 bit widths, data must be split into smaller portions with each portion not to exceed the 24 bit limit.*

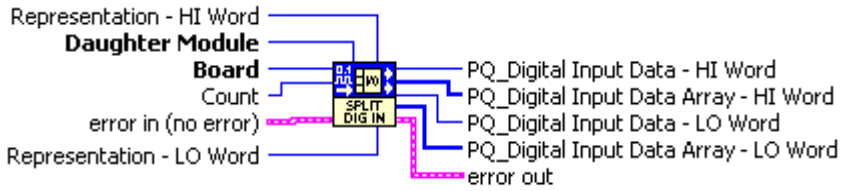
QuBASE Equivalent.

```
'*************************************************************************
'       Function PQ_DigitalInputPort
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqTrigger, Integer
'               pqCounterReset, Integer
'               pqCountDirection, Integer
'               pqWrapMode, Integer
'               pqCounterThreshold, Integer
'
'       Output Parameters:
'               oPQYn, Integer
'               oPQFinalCountSync, Integer
'
'*************************************************************************

   Function PQ_DigitalInputPort( _
          ByRef board As Integer, _
          ByRef errIn As T_Error, _
          ByRef intModule As Integer, _
          ByRef oPQDigitalInput() As Integer, _
          Optional ByRef representation As Integer = 0) _
          As T_Error
```

3.2.2 PQ_Digital Input Port (Split to 16 bit Pair).

The 'PQ_Digital Input Port (Split to 16 bit Pair)' VI reads 32 bits of raw binary data present on the digital I/O port on the SI-MOD68xx module when it is configured as an input, and then separates the reading into a pair of 16 bit values. The reads may be specified to occur sequentially on the same port within a single 'PQ' sample period.

NOTE: *This split is necessary since the DSP is only able to represent a 24 bit integer when that value is encoded as a PQ compatible floating point value.*



PQ_Digital Input Port (Split to 16 bit Pair).vi

Count = LabVIEW input, determines the size of the output array. A count of '1' returns a single element, while a count greater than '1' returns an array of data formed from sequentially reading from the same port within a single 'PQ' sample period. Defaulted to a value of '1'.

Representation - HI Word = LabVIEW input, selects to read either signed or unsigned binary data from the upper half 16 bit word of a 32 bit digital I/O port; defaulted to read unsigned integer values.

Representation - LO Word = LabVIEW input, selects to read either signed or unsigned binary data from the lower half 16 bit word of a 32 bit digital I/O port; defaulted to read unsigned integer values.

PQ_Digital Input Data - HI Word = PQ output, returns the upper 16 bits from a single 32 bit digital I/O port access, represented as a floating point 'PQ' value. If an array is selected, only the first value of the array is returned.

PQ_Digital Input Data Array - HI Word = PQ output array, returns the array of upper 16 bits from sequentially accessing a single 32 bit digital I/O port within a single 'PQ' sample period. If only a single value is selected with a count of '1', the array returns a single element.

PQ_Digital Input Data - LO Word = PQ output, returns the lower 16 bits from a single 32 bit digital I/O port access, represented as a floating point 'PQ' value. If an array is selected, only the first value of the array is returned.

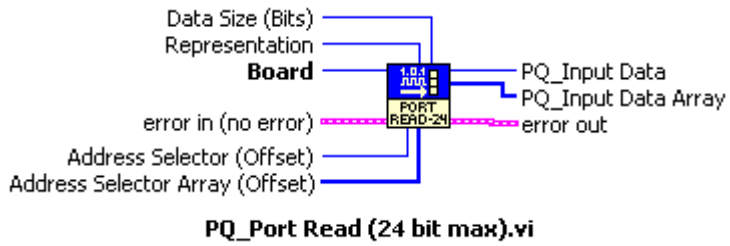
PQ_Digital Input Data Array - LO Word = PQ output array, returns the array of lower 16 bits from sequentially accessing a single 32 bit digital I/O port within a single 'PQ' sample period. If only a single value is selected with a count of '1', the array returns a single element.

QuBASE Equivalent.

```
'*********************************************************************
'       Function PQ_DigitalInputPortSplitTo16BitPair
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               daughter, Integer
'               hiRepresentation, Integer
'               loRepresentation, Integer
'               pqInputDataHi, Integer
'               pqInputDataLo, Integer
'
'       Output Parameters:
'
'*********************************************************************

  Function PQ_DigitalInputPortSplitTo16BitPair( _
    ByRef board As Integer, _
    ByRef errIn As T_Error, _
    ByVal daughter As Integer, _
    ByVal hiRepresentation As Integer, _
    ByVal loRepresentation As Integer, _
    ByRef pqInputDataHi As Integer, _
    ByRef pqInputDataLo As Integer) _
    As T_Error
```

3.2.3 PQ_Port Read (24 bit max).

The PQ_Port Read VI reads up to 24 bits of raw binary data from any location within the DSP memory space. Primarily used as a building block for other functions.



Data Size (Bits) = LabVIEW input, specifies the width of the port in bits ranging from 1 to 24 bits, and masks away the unwanted upper bits.

Representation = LabVIEW input, selects to read either signed or unsigned binary data from the digital I/O port; defaulted to read unsigned integer values.

Address Selector (Offset) = LabVIEW input, specifies the actual location or address of the port to be read in the DSP's memory space. If not wired, then the 'Address Selector Array (Offset)' array must be wired instead.

Address Selector Array (Offset) = LabVIEW input array, which specifies an array of actual locations or addresses of ports to be read in the DSP's memory space. If wired, it becomes the default terminal used, and the single variable 'Address Selector (Offset)' terminal is ignored..

PQ_Input Data = PQ output, up to 24 bits of binary data read from the port, represented as a floating point value. If an array is selected, only the first value of the array is returned.

PQ_Input Data Array = PQ output array, up to 24 bits of a binary data to be read from one or more ports, represented as floating point values. If only a single value is selected, the array only returns a single element.

NOTE: *Although all DSP as well as the SI-MOD68xx's digital IO ports are 32 bits wide, only 24 bits can be accessed because of the DSP's floating point format using a 24 bit mantissa. For operations requiring the complete 32 bit widths, data must be separated into smaller 16 bit portions thereby complying with the 24 bit limit.*

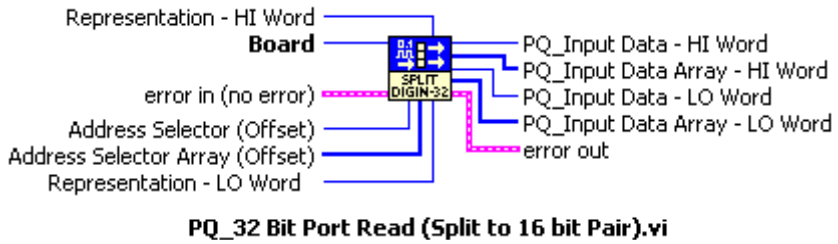
QuBASE Equivalent.

```
'*********************************************************************
'       Function PQ_PortRead
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               channelSelectorArray, Integer
'               representation, Integer
'               dataSize, Integer
'
'       Output Parameters:
'               oPQDigitalInput, Integer
'
'*********************************************************************

   Function PQ_PortRead( _
          ByRef board As Integer, _
          ByRef errIn As T_Error, _
          ByRef oPQDigitalInput() As Integer, _
          ByRef channelSelectorArray() As Integer, _
          Optional ByRef representation As Integer = 0, _
          Optional ByRef dataSize As Integer = 0) _
          As T_Error
```

3.2.4 PQ_32 Bit Port Read (Split to 16 bit Pair).

The 'PQ_32 Bit Port Read (Split to 16 bit Pair)' VI reads one or more 32 bit ports as raw binary data, and separates the reading into a pair of 16 bit values.

NOTE: *This split is necessary since the DSP is only able to represent a 24 bit integers when that value is encoded as a PQ compatible floating point value.*



PQ_32 Bit Port Read (Split to 16 bit Pair).vi

Representation - HI Word = LabVIEW input, selects to read either signed or unsigned binary data from the upper half 16 bit word of a 32 bit digital I/O port; defaulted to read unsigned integer values.

Representation - LO Word = LabVIEW input, selects to read either signed or unsigned binary data from the lower half 16 bit word of a 32 bit digital I/O port; defaulted to read unsigned integer values.

Address Selector (Offset) = LabVIEW input, specifies the actual location or address of the port to be read in the DSP's memory space. If not wired, then the 'Address Selector Array (Offset)' array must be wired instead.

Address Selector Array (Offset) = LabVIEW input array, which specifies an array of actual locations or addresses of ports to be read in the DSP's memory space. If wired, it becomes the default terminal used, and the single variable 'Address Selector (Offset)' terminal is ignored.

PQ_Input Data - HI Word = PQ output, up to 16 bits of binary data read from the upper half 16 bit word of a 32 bit digital I/O port, represented as a floating point 'PQ' value. If an array is selected, only the first value of the array is returned.

PQ_Input Data Array - HI Word = PQ output array, up to 16 bits of binary data read from the upper half 16 bit word of a 32 bit digital I/O port, represented as a floating point 'PQ' value. If only a single value is selected, the array only returns a single element.

PQ_Input Data - LO Word = PQ output, up to 16 bits of binary data read from the lower half 16 bit word of a 32 bit digital I/O port, represented as a floating point 'PQ' value. If an array is selected, only the first value of the array is returned.

PQ_Input Data Array - LO Word = PQ output array, up to 16 bits of binary data read from the lower half 16 bit word of a 32 bit digital I/O port, represented as a floating point 'PQ' value. If only a single value is selected, the array only returns a single element.

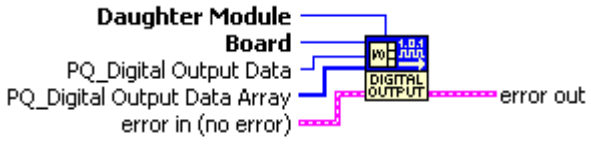
QuBASE Equivalent.

```
'*************************************************************************
'       Function PQ_32BitPortReadSplitTo16BitPair
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               channelSelectorArray, Integer
'               representation, Integer
'               dataSize, Integer
'
'       Output Parameters:
'               oPQDigitalInput, Integer
'
'*************************************************************************

   Function PQ_32BitPortReadSplitTo16BitPair( _
          ByRef board As Integer, _
          ByRef errIn As T_Error, _
          ByRef oPQDigitalInput() As Integer, _
          ByRef channelSelectorArray() As Integer, _
          Optional ByRef representation As Integer = 0, _
          Optional ByRef dataSize As Integer = 0) _
          As T_Error
```

3.2.5 PQ_Digital Output Port (24 bit max).

The PQ_Digital Output Port (24 bit max) VI writes up to 24 bits of raw binary data to the digital I/O port when it is configured as an output.

NOTE: *Although the SI-MOD68xx's digital IO port is 32 bits wide, only 24 bits can be accessed because of the DSP's floating point format using a 24 bit mantissa. For operations requiring the complete 32 bit widths, data must be split into smaller portions with each portion not exceeding the 24 bit limit.*



PQ_Digital Output Port (24 bit max).vi

PQ_Digital Output Data = PQ input, up to 24 bits of binary data can be written to the digital I/O port, represented as a floating point value. Ignored and not to be used if an array is wired to the 'PQ_Output Data Array' terminal instead.

PQ_Digital Output Data Array = PQ input array, with each element consisting of up to 24 bits of binary data that can be written to the digital I/O port, represented as floating point values. Should be used if an array of data is to be written to the digital I/O port within a single sample period.

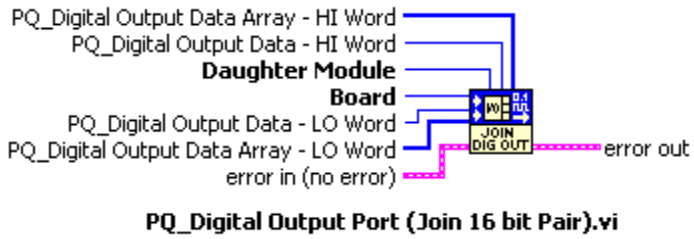
QuBASE Equivalent.

```
'*********************************************************************
'       Function PQ_DigitalOutputPort
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               intModule, Integer
'               pqOutputData, Integer
'
'       Output Parameters:
'
'*********************************************************************

  Function PQ_DigitalOutputPort( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef intModule As Integer, _
        ByRef pqOutputData() As Integer) _
        As T_Error
```

3.2.6 PQ_Digital Output Port (Join 16 bit Pair).

The 'PQ_Digital Output Port (Join 16 bit Pair)' VI joins a pair of 16 bit values to write a single 32 bit value to one or more ports.

NOTE: *This concatenation is necessary since the DSP is only able to represent a 24 bit integer when that value is encoded as a PQ compatible floating point value.*



PQ_Digital Output Port (Join 16 bit Pair).vi

PQ_Digital Output Data - HI Word = PQ input, the upper 16 bits of binary data to be written to a single 32 bit digital I/O port, represented as a floating point 'PQ' value. Ignored if the 'PQ_Digital Output Data Array - HI Word' array terminal is wired.

PQ_Digital Output Data Array - HI Word = PQ input array, the upper 16 bits of a binary data array to be sequentially written to a single 32 bit digital I/O port, within a single 'PQ' sample period. If wired, it becomes the default terminal used, and the single variable 'PQ_Digital Output Data - HI Word' terminal is ignored.

PQ_Digital Output Data - LO Word = PQ input, the lower 16 bits of binary data to be written to a single 32 bit digital I/O port, represented as a floating point 'PQ' value. Ignored if the 'PQ_Digital Output Data Array - HI Word' array terminal is wired.

PQ_Digital Output Data Array - LO Word = PQ input array, the lower 16 bits of a binary data array to be sequentially written to a single 32 bit digital I/O port, within a single 'PQ' sample period. If wired, it becomes the default terminal used, and the single variable 'PQ_Digital Output Data - LO Word' terminal is ignored.

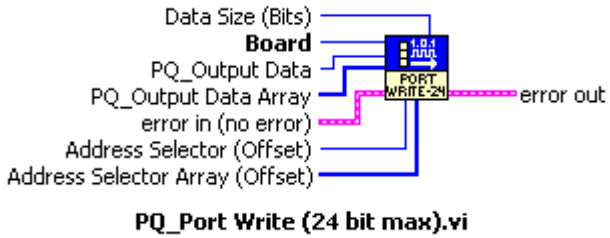
QuBASE Equivalent.

```
'*********************************************************************
'       Function PQ_DigitalOutputPortJoin6BitPair
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               daughter, Integer
'               hiRepresentation, Integer
'               loRepresentation, Integer
'               pqInputDataHi, Integer
'               pqInputDataLo, Integer
'
'       Output Parameters:
'
'*********************************************************************

  Function PQ_DigitalOutputPortJoin16BitPair( _
    ByRef board As Integer, _
    ByRef errIn As T_Error, _
    ByVal daughter As Integer, _
    ByVal hiRepresentation As Integer, _
    ByVal loRepresentation As Integer, _
    ByRef pqInputDataHi As Integer, _
    ByRef pqInputDataLo As Integer) _
    As T_Error
```

3.2.7 PQ_Port Write (24 bit max).

The 'PQ_Port Write (24 bit max)' VI writes up to 24 bits of raw binary data to any location within the DSP memory space. Primarily used as a building block for other functions.

NOTE: *Although all DSP as well as the SI-MOD68xx's digital IO ports are 32 bits wide, only 24 bits can be accessed because of the DSP's floating point format using a 24 bit mantissa. For operations requiring the complete 32 bit widths, data must be separated into smaller 16 bit portions thereby complying with the 24 bit limit.*



Data Size (Bits) = LabVIEW input, specifies the width of the port ranging from 1 to 24 bits, and masks away the unwanted upper bits.

Address Selector (Offset) = LabVIEW input, specifies the actual address or location of the port to write in the DSP's memory space. Ignored if the 'Address Selector Array (Offset)' array terminal is wired.

Address Selector Array (Offset) = LabVIEW input array, specifies an array of actual port addresses or locations to write in the DSP's memory space. If the array terminal for sourcing the data was used, then the Channel Selector Array should be correspondingly used and the single variable 'Address Selector (Offset)' terminal is ignored.

PQ_Output Data = PQ input, up to 24 bits of binary data can be written to the port, represented as a floating point value. Ignored if the 'PQ_Output Data Array' array terminal is wired.

PQ_Output Data Array = PQ input array, up to 24 bits of a binary data can be written to one or more ports, represented as floating point values. Convenient if an array of variables is used as it avoids the wiring of separate icons. If wired, it becomes the default terminal used, and the single variable 'PQ_Output Data' terminal is ignored.

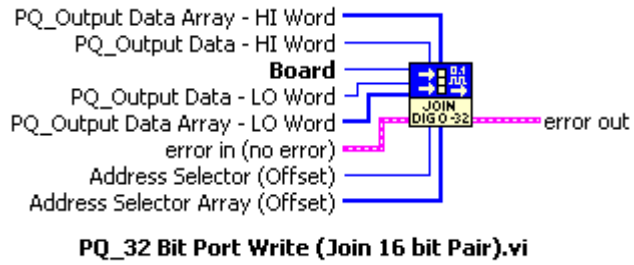
QuBASE Equivalent.

```
'*********************************************************************
'       Function PQ_PortWrite
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqDigitalOutput, Integer
'               channelSelectorArray, Integer
'               dataSize, Integer
'
'       Output Parameters:
'
'*********************************************************************
   Function PQ_PortWrite( _
            ByRef board As Integer, _
            ByRef errIn As T_Error, _
            ByRef pqDigitalOutput() As Integer, _
            ByRef channelSelectorArray() As Integer, _
            Optional ByRef dataSize As Integer = 16) _
            As T_Error
```

3.2.8 PQ_32 Bit Port Write (Join 16 bit Pair).

The 'PQ_32 Bit Port Write (Join 16 bit Pair)' VI joins a pair of 16 bit values to write a single 32 bit value to one or more ports.

NOTE: *This concatenation is necessary since the DSP is only able to represent a 24 bit integer when that value is encoded as a PQ compatible floating point value.*



Address Selector (Offset) = LabVIEW input, specifies the actual address or location of the port to write in the DSP's memory space. Ignored if the 'Address Selector Array (Offset)' array terminal is wired.

Address Selector Array (Offset) = LabVIEW input array, specifies an array of actual port addresses or locations to write in the DSP's memory space. If the array terminal for sourcing the data was used, then the Channel Selector Array should be correspondingly used and the single variable 'Address Selector (Offset)' terminal is ignored.

PQ_Output Data - HI Word = PQ input, up to 16 bits of binary data can be written to the upper half 16 bit word of a 32 bit port, represented as a floating point 'PQ' value. Ignored if the 'PQ_Output Data Array - HI Word' array terminal is wired.

PQ_Output Data Array - HI Word = PQ input array, up to 16 bits of a binary data array can be written to the upper half of an array of 32 bit ports, represented as floating point values. Convenient if an array of variables is used as it avoids the wiring of separate icons. If wired, it becomes the default terminal used, and the single variable 'PQ_Output Data - HI Word' terminal is ignored.

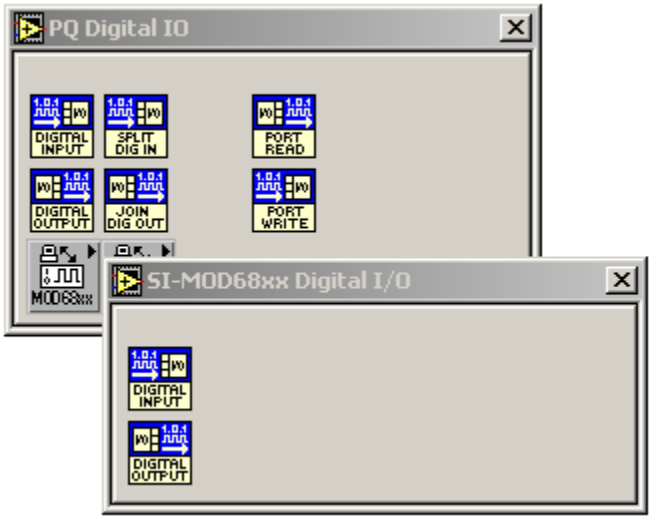
PQ_Output Data - LO Word = PQ input, up to 16 bits of binary data can be written to the lower half 16 bit word of a 32 bit port, represented as a floating point 'PQ' value. Ignored if the 'PQ_Output Data Array - LO Word' array terminal is wired.

PQ_Output Data Array - LO Word = PQ input array, up to 16 bits of a binary data array can be written to the lower half of an array of 32 bit ports, represented as floating point values. Convenient if an array of variables is used as it avoids the wiring of separate icons. If wired, it becomes the default terminal used, and the single variable 'PQ_Output Data - LO Word' terminal is ignored.

QuBASE Equivalent.

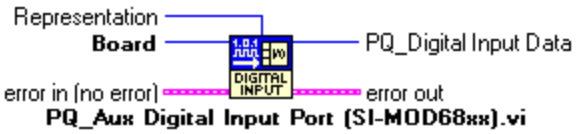
```
'*************************************************************************
'       Function PQ_32BitPortWriteJoin16BitPair
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqDigitalOutput, Integer
'               channelSelectorArray, Integer
'               dataSize, Integer
'
'       Output Parameters:
'
'*************************************************************************
   Function PQ_32BitPortWriteJoin16BitPair( _
            ByRef board As Integer, _
            ByRef errIn As T_Error, _
            ByRef pqDigitalOutput() As Integer, _
            ByRef channelSelectorArray() As Integer, _
            Optional ByRef dataSize As Integer = 16) _
            As T_Error
```

3.2.9 SI-MOD68xx Digital I/O.



3.2.9.1 PQ_Aux Digital Input Port (SI-MOD68xx).

The PQ_Aux Digital Input Port (SI-MOD68xx).vi reads up to 4 bits of raw binary data present on the auxiliary digital I/O port of the SI-MOD68xx module when it is configured as an input.



Representation = LabVIEW input, selects to read either signed or unsigned binary data from the digital I/O port; defaulted to read unsigned integer values.

PQ_Digital Input Data = PQ output, up to 4 bits of binary data read from the auxiliary digital I/O port, represented as a floating point value.

QuBASE Equivalent.

```
'**************************************************************************
'       Function PQ_AuxDigitalInputPort_SI_MOD68xx
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqDigitalInputData, Integer
'               representation, Integer
'
'       Output Parameters:
'
'**************************************************************************

  Function PQ_AuxDigitalInputPort_SI_MOD68xx( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef pqDigitalInputData As Integer, _
        Optional ByVal representation As Integer = 0) _
        As T_Error
```

3.2.9.2 PQ_Aux Digital Output Port (SI-MOD68xx).

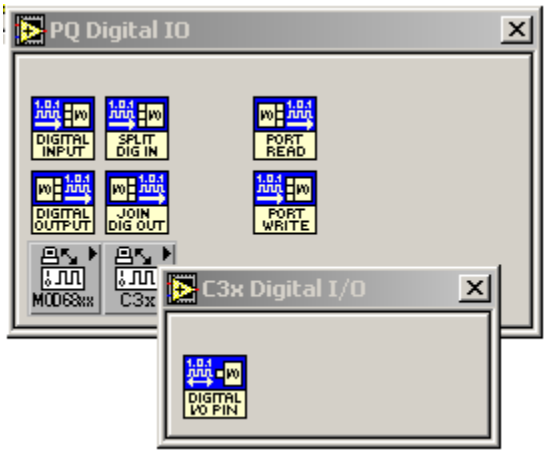
The PQ_Aux Digital Output Port (SI-MOD68xx).vi writes up to 4 bits of raw binary data to the auxiliary digital I/O port when it is configured as an output.

PQ_Output Data = PQ input, up to 4 bits of binary data can be written to the auxiliary digital I/O port, represented as a floating point value.

QuBASE Equivalent.

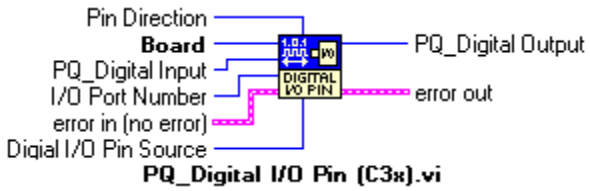
```
'*********************************************************************
'       Function PQ_AuxDigitalOutputPort_SI_MOD68xx
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqDigitalOutputData, Integer
'
'       Output Parameters:
'
'*********************************************************************

  Function PQ_AuxDigitalOutputPort_SI_MOD68xx( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByVal pqDigitalOutputData As Integer) _
        As T_Error
```

3.2.10 C3x Digital I/O.



3.2.10.1 PQ_Digital I/O Pin (C3x).

The PQ_Digital I/O Pin (C3x) VI makes use of the C3x DSP's peripheral port programmed as a general purpose digital I/O port, where each pin's direction may be individually controlled.



Pin Direction = LabVIEW input, specifies if a particular line will operate as an input or an output.

PQ_Digital Input = PQ Input, unscaled value to be written if the pin of interest is configured as an output.

I/O Port Number = LabVIEW input, specifies one of two ports where the digital I/O pin is located. Only valid for those DSP's that have more than a single peripheral port; otherwise defaulted to the only port available.

Digital I/O Pin Source = LabVIEW input, selects a specific pin of the DSP's port for use as a digital I/O.

PQ_Digital Output = PQ output representing the value present on a requested digital I/O pin.

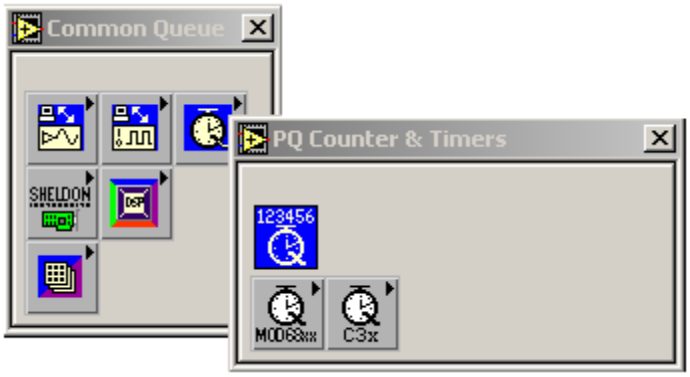
NOTE: Only valid with C3x DSPs.

QuBASE Equivalent.

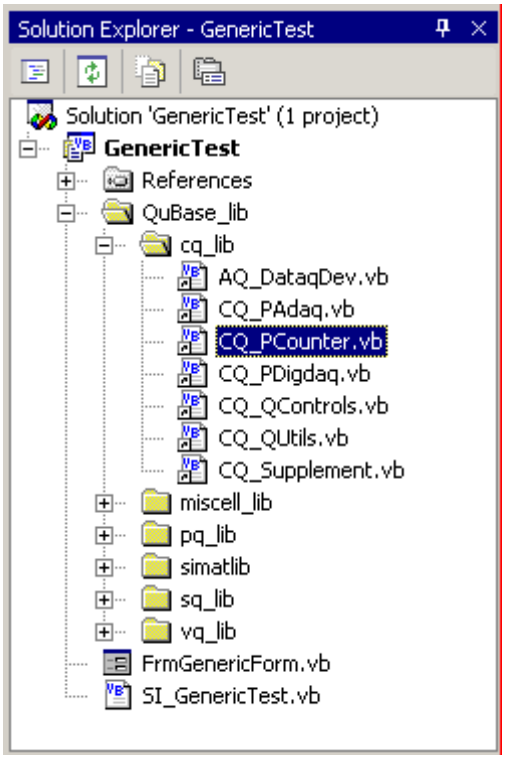
```
'**********************************************************************
'       Function PQ_DigitalIOPin_C3x
'
'       Description:
'               This function works exclusively for C3x DSPs only
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqDigitalInput, Integer
'               pinDirection, Integer
'               ioPortNumber, Integer
'               ioPinSource, Integer
'
'       Output Parameters:
'               oPQDigitalOutput, Integer
'
'**********************************************************************

   Function PQ_DigitalIOPin_C3x( _
          ByRef board As Integer, _
          ByRef errIn As T_Error, _
          Optional ByRef oPQDigitalOutput As Integer = 0, _
          Optional ByRef pqDigitalInput As Integer = 0, _
          Optional ByRef pinDirection As Integer = 0, _
          Optional ByRef ioPortNumber As Integer = 0, _
          Optional ByRef ioPinSource As Integer = 0) _
          As T_Error
```

3.3 Common Queue: PQ Counters & Timers.

These functions are internal to the DSP, and make use of its peripheral port for unique digital sampling functions.

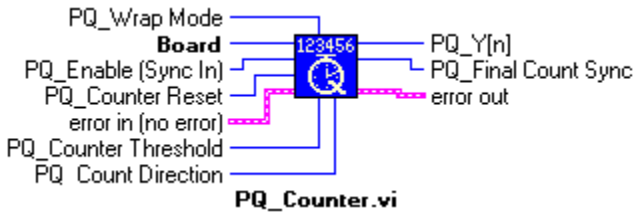


QuBASE Equivalent.



3.3.1 PQ_Counter.

The PQ_Counter VI invokes the software implementation of an event counter that resides inside the DSP binary executable or COFF file, where the clock source is the PQ interrupt rate used to trigger ADC samples. It operates as a traditional event counter, where the events can be defined by boolean pulses derived from any one of the comparison VIs.



PQ_Wrap Mode = PQ input, used to control if the counter must autowrap to the reset state when the count reaches the threshold value:

=0 = When set to a 'PQ Zero' value, the counter is reset or wrapped when the threshold is reached as long as there is a tick source (default).

nonzero = Otherwise, when set to a 'PQ Nonzero' value the counter output PQ_Y[n] is latched at the threshold value, until the 'PQ_Counter Reset' terminal is cleared with a 'PQ Zero' value fed into it.

PQ_Enable (Sync In) = PQ input that determines of the counts are enabled or not:

=0 = when set to a 'PQ Zero' value, the counter is enabled to increment or decrement (default).

nonzero = when set to a 'PQ Nonzero' value, the counter is detained with the current value and NOT cleared.

PQ_Counter Reset = PQ input, used to clear the counter

=0 = when set to a 'PQ Zero' value, the counter increments or decrements normally (default).

nonzero = when set to a 'PQ Nonzero' value, the counter is cleared irrespective of the state of the 'PQ_Enable' line.

PQ_Counter Threshold = PQ input constant that determines the last count value before wrapping or latching:

=0 = the counter increments or decrements freely until 2^32 approaches, after which it wraps to a 'PQ Zero' value or latches, as set by the 'PQ_Wrap Mode' input terminal, (default).

nonzero = the counter increments or decrements to this PQ value before wrapping or latching must be always positive.

PQ Count Direction = PQ input that determines the direction of the count.

=0 = When set to a 'PQ Zero' value the counter increments to +threshold, (default).

nonzero = When set to a 'PQ Nonzero' value, the counter decrements to -threshold.

PQ_Y[n] = PQ output that returns actual count value.

PQ_Final Count Sync = PQ output used to indicate the state of the counter output, useful for synchronizing events.

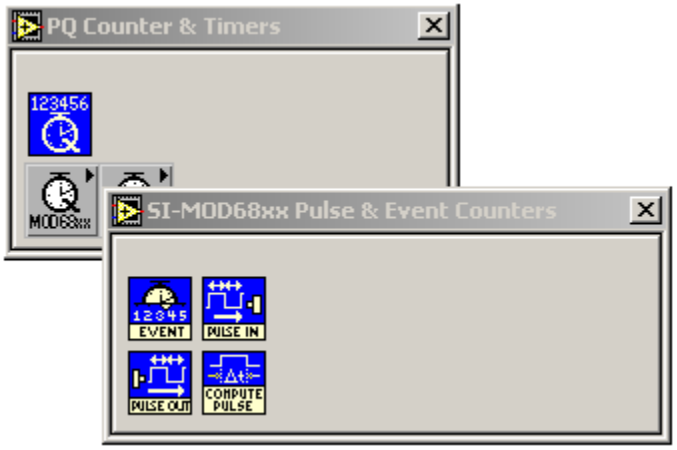
0 = The counter output has reached its final value, and returns a 'PQ zero' value

1 = The counter is busy and returns a 'PQ One' while counting or reset.

QuBASE Equivalent.

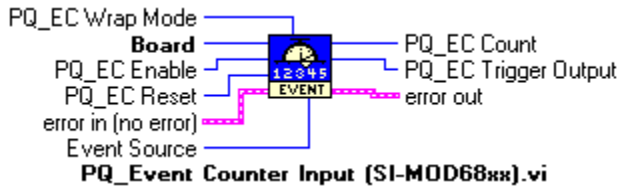
```
'**********************************************************************
'       Function PQ_Counter
'
'       Description:
'               Counts on PQ interrupts
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqTrigger, Integer
'               pqCounterReset, Integer
'               pqCountDirection, Integer
'               pqWrapMode, Integer
'               pqCounterThreshold, Integer
'
'       Output Parameters:
'               oPQYn, Integer
'               oPQFinalCountSync, Integer
'
'**********************************************************************

  Function PQ_Counter( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef oPQYn As Integer = 0, _
        Optional ByRef oPQFinalCountSync As Integer = 0, _
        Optional ByRef pqTrigger As Integer = 0, _
        Optional ByRef pqCounterReset As Integer = 0, _
        Optional ByRef pqCountDirection As Integer = 0, _
        Optional ByRef pqWrapMode As Integer = 0, _
        Optional ByRef pqCounterThreshold As Integer = 0) _
        As T_Error
```

3.3.2 SI-MOD68xx Pulse & Event Counters.



3.3.2.1 PQ_Event Counter Input (SI-MOD68xx).

The PQ_Event Counter Input (SI-MOD68xx) VI invokes the hardware implementation of an event counter that resides inside the SI-MOD68xx hardware. It operates as a traditional event counter, and is very flexible and especially useful for managing a variety of clocking and event schemes to trigger a variety of operations.



PQ_EC Wrap Mode = PQ input, used to control if the EC counter must autowrap to the reset state when the count reaches the threshold value.

=0 = When set to a 'PQ Zero' value, the counter is reset or wrapped when the threshold is reached as long as there is a tick source (default).

nonzero = Otherwise, when set to a 'PQ Nonzero' value the counter output PQ_Y[n] is latched at the threshold value, until the 'PQ_Counter Reset' terminal is cleared with a 'PQ Zero' value fed into it.

PQ_EC Enable = PQ input that determines of the counts are enabled or not.

=0 = when set to a 'PQ Zero' value, the counter is enabled to increment or decrement (default).

nonzero = when set to a 'PQ Nonzero' value, the counter is detained with the current value and NOT cleared.

PQ_EC Reset = PQ input, used to clear the EC counter.

=0 = when set to a 'PQ Zero' value, the counter increments or decrements normally (default).

nonzero = when set to a 'PQ Nonzero' value, the counter is cleared irrespective of the state of the 'PQ_Enable' line.

Event Source = LabVIEW input, selects one of four Event Counter circuits that reside on the SI-MOD68xx card.

PQ_EC Count = PQ output that returns actual count value.

PQ_EC Trigger Output = PQ output used to indicate the state of the counter output, useful for synchronizing events. This signal may also be used as a software source to other devices (reverse from bit reading).

0 = The counter output has reached its final value, and returns a 'PQ zero' value.

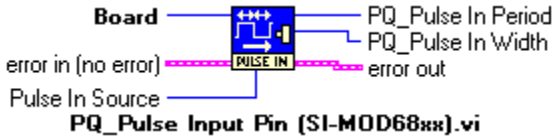
1 = The counter is busy and returns a 'PQ One' while counting or reset.

QuBASE Equivalent.

```
'*********************************************************************
'       Function PQ_EventCounterInput_SI_MOD68xx
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               eventSource, Integer
'               pqECReset, Integer
'               pqECEnable, Integer
'               pqECWrapMode, Integer
'               pqECCount, Integer
'               pqECTriggerOutput, Integer
'
'       Output Parameters:
'
'*********************************************************************

   Function PQ_EventCounterInput_SI_MOD68xx( _
          ByRef board As Integer, _
          ByRef errIn As T_Error, _
          Optional ByVal eventSource As Integer = 0, _
          Optional ByVal pqECReset As Integer = 0, _
          Optional ByVal pqECEnable As Integer = 0, _
          Optional ByVal pqECWrapMode As Integer = 0, _
          Optional ByRef pqECCount As Integer = 0, _
          Optional ByRef pqECTriggerOutput As Integer = 0) _
          As T_Error
```

3.3.2.2 PQ_Pulse Input (SI-MOD68xx).

The PQ_Pulse Input (SI-MOD68xx) VI invokes the hardware implementation of a frequency counter that resides on a pair of lines available on the auxiliary digital I/O ports on the SI-MOD68xx hardware.



Pulse In Source = LabVIEW input, selects one of the available pulse input pins present on the auxiliary digital I/O port.

PQ_Pulse In Period = PQ output that returns the last overall period that occured between consecutive positive edges derived from the external TTL level pulse train input. The period is relative to the number of H clock periods sourced from the DSP carrier card.

PQ_Pulse In Width = PQ output that returns the number of H clock periods that occured between every rising and falling edge of an external TTL level pulse train. The period is relative to the number of H clock periods sourced from the DSP carrier card.

NOTE: *The H Clock rate varies depending upon the particular DSP installed on the carrier card. Currently, all Sheldon Instruments hardware supports the following H clock rates:*
1) SI-C31DSP-PCI cards. The H clock rate for the C31 DSP ranges from 20Mhz to 30Mhz.
2) SI-C33DSP-PCI cards. The H clock rate for the C33 DSP is fixed at 37.5Mhz.
3) SI-C67xDSP-PCI cards. The H clock rate for the C67x DSP is fixed at 37.5Mhz.

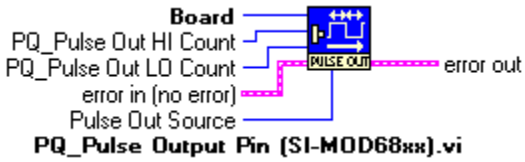
QuBASE Equivalent.

```
'*********************************************************************
'       Function PQ_PulseInputPin_SI_MOD68xx
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               iPulseInSource, Integer
'
'       Output Parameters:
'               opqPulseInPeriod, Integer
'               opqPulseInWidth, Integer
'
'*********************************************************************

  Function PQ_PulseInputPin_SI_MOD68xx( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef iPulseInSource As Integer = 0, _
        Optional ByRef opqPulseInPeriod As Integer = 0, _
        Optional ByRef opqPulseInWidth As Integer = 0) _
        As T_Error
```

3.3.2.3 PQ_Pulse Output (SI-MOD68xx).

The PQ_Pulse Output (SI-MOD68xx) invokes the hardware implementation of a pulse generator available on a pair of lines on the SI-MOD68xx's auxiliary digital I/O ports. It serves to generate TTL level output pulse trains, used in PWM applications.

NOTE: *The pulse output generators are setup as two separate VIs:*
1) DSP based PQ_Pulse Output (SI-MOD68xx), implemented in hardware inside the SI-MOD68xx, except that initialization values are updated by the DSP.
2) Conventional host based computation application, such as the LabVIEW VI. Especially useful for computing non time critical parameters that can be passed from the host to the DSP.



PQ_Pulse Out HI Count = PQ input, Width/Duration of the output pulse asserted HI (the number of periods between the rising and falling edges), relative to the number of DSP H clock periods. The period is relative to the number of H clock periods sourced from the DSP carrier card.

PQ_Pulse Out LO Count = PQ input, Width/Duration of the output pulse asserted LO (the number of periods between the falling and rising falling edges), relative to the number of DSP H clock periods. The period is relative to the number of H clock periods sourced from the DSP carrier card.

NOTE: *The sum of these two values is equivalent to the overall pulse period.*

Pulse Out Source = LabVIEW input, selects one of the available pulse output pins present on the auxiliary digital I/O port.

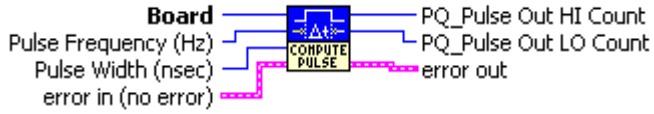
NOTE: *The H Clock rate varies depending upon the particular DSP installed on the carrier card. Currently, all Sheldon Instruments hardware supports the following H clock rates:*
1) SI-C31DSP-PCI cards. The H clock rate for the C31 DSP ranges from 20Mhz to 30Mhz.
2) SI-C33DSP-PCI cards. The H clock rate for the C33 DSP is fixed at 37.5Mhz.
3) SI-C67xDSP-PCI cards. The H clock rate for the C67x DSP is fixed at 37.5Mhz.

QuBASE Equivalent.

```
'*************************************************************************
'       Function PQ_PulseOutputPin_SI_MOD68xx
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               ipqPulseOutHighCounts, Integer
'               ipqPulseOutLowCounts, Integer
'               ipulseOutSource, Integer
'
'       Output Parameters:
'
'*************************************************************************

  Function PQ_PulseOutputPin_SI_MOD68xx( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByVal ipqPulseOutHighCounts As Integer, _
        ByVal ipqPulseOutLowCounts As Integer, _
        ByVal ipulseOutSource As Integer) _
        As T_Error
```

3.3.2.4 PQ_Pulse Width Computation (SI-MOD68xx).

The PQ_Pulse Width Computation VI computes the conversion values to be downloaded to 'PQ_Pulse Output (SI-MOD68xx)' VI.



PQ_Pulse Width Computation (SI-MOD68xx).vi

Pulse Width (nsec) = LabVIEW input to define output pulse HI asserted level duration in nanoseconds.

Pulse Frequency (Hz) = LabVIEW input to define the overall pulse frequency in hertz.

PQ_Pulse Out HI Count = PQ output, Width/Duration of the output pulse asserted HI (the number of periods between the rising and falling edges), relative to the number of DSP H clock periods. The period is relative to the number of H clock periods sourced from the DSP carrier card.

PQ_Pulse Out LO Count = PQ output, Width/Duration of the output pulse asserted LO (the number of periods between the falling and rising falling edges), relative to the number of DSP H clock periods. The period is relative to the number of H clock periods sourced from the DSP carrier card.

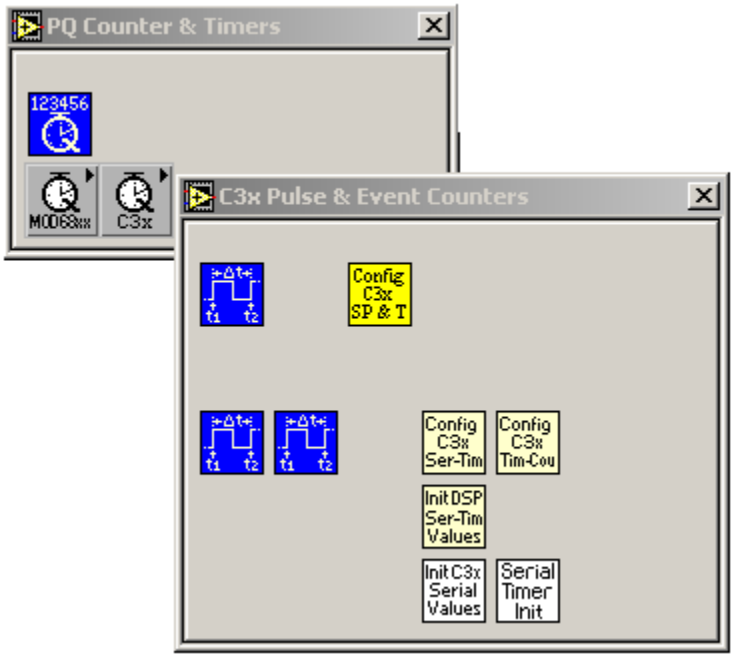
NOTE: *The sum of these two values is equivalent to the overall pulse period.*

NOTE: *The H Clock rate varies depending upon the particular DSP installed on the carrier card. Currently, all Sheldon Instruments hardware supports the following H clock rates:*
1) SI-C31DSP-PCI cards. The H clock rate for the C31 DSP ranges from 20Mhz to 30Mhz.
2) SI-C33DSP-PCI cards. The H clock rate for the C33 DSP is fixed at 37.5Mhz.
3) SI-C67xDSP-PCI cards. The H clock rate for the C67x DSP is fixed at 37.5Mhz.

QuBASE Equivalent.

```
'*************************************************************************
'       Function PQ_PulseWidthComputation
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               iFrequency, Integer
'               iPulseWidthnsec, Integer
'
'       Output Parameters:
'               oPQPulseUpCounts, Integer
'               oPQPulseDownCounts, Integer
'
'*************************************************************************

   Function PQ_PulseWidthComputation( _
          ByRef board As Integer, _
          ByRef errIn As T_Error, _
          Optional ByRef oPQPulseUpCounts As Integer = 0, _
          Optional ByRef oPQPulseDownCounts As Integer = 0, _
          Optional ByRef iFrequency As Integer = 0, _
          Optional ByRef iPulseWidthnsec As Integer = 0) _
          As T_Error
```

3.3.4 C3x Pulse & Event Counters.

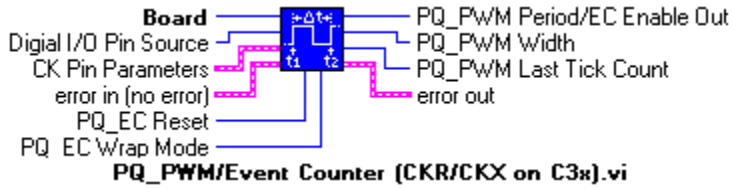


3.3.4.1 PQ_PWM/Event Counter (CLKx on C3x).

The PQ_PWM/Event Counter (CLKx on C3x) is an extremely powerful function that operates as a frequency counter on one of the four serial port clock pins of the C3x DSP.

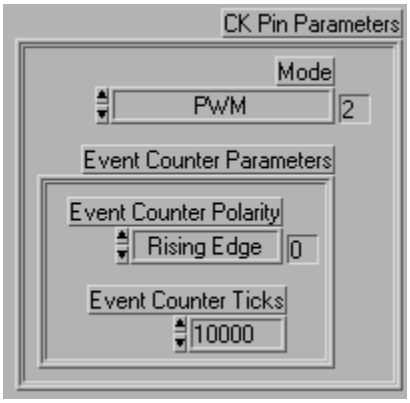
It returns a PQ_Last Pulse Width value and a PQ_Last Edge Time value based on the C3x's internal free running Timer 0 clock. In essence, the PQ_Last Pulse Width is the number of Timer 0 counts that occur between every rising or falling edge of an external TTL level pulse train. In order to compute the actual frequency of the external pulse train, divide the PQ_Last Pulse Width value into one fourth the C3x's clock speed. For example, if the C3x's clock speed is rated at 60Mhz, divide 15Mhz by the PQ_Last Pulse Width value.

The PQ_Last Edge Time simply returns the last value of the C3x's internal Timer 0 value, as opposed to a net difference provided by the PQ_Last Pulse Width output. Please consult the C3x User's Guide for further details about the C3x's internal architecture.



Digital I/O Pin Source (0-3) = LabVIEW input, ranging from 0 to 1 for a C31 and C32 or 0 to 3 for a C30, that determines which of the four serial clock port pins of the C3x is to be the PWM input.

CK Pin Paramerters = LabVIEW cluster input, defines the operation of the C3x's clock pins.



Mode = LabVIEW input, selects the operation mode of the C3x's clock pin I/Os.

0 = No operation.
1 = Digital I/O.
2 = PWM or Pulse input.
3 = Event Counter.

Event Counter Parameters = LabVIEW subcluster input, used to define the operation of the Event Counter.

Event Counter Polarity = LabVIEW input, selects the polarity of the Event Counter tick source.

0 = selects the tick source to increment the Event Counter on the positive/rising edges.

1 = selects the tick source to increment the Event Counter on the negative/falling edges.

Event Counter Ticks = LabVIEW input, defines the threshold or count limit for the Event counter.

PQ_EC Reset = PQ input, used to clear the PWM/Event Counter.

<=0 = when set to a 'PQ Zero' value or less, the counter increments or decrements normally (default).

nonzero = when set to a 'PQ Nonzero' value, the counter is cleared irrespective of the state of the 'PQ_Enable' line.

PQ_EC Wrap Mode = PQ input, used to control if the EC counter must autowrap to the reset state when the count reaches the threshold value.

<=0 = When set to a 'PQ Zero' value or less, the counter is reset or wrapped when the threshold is reached as long as there is a tick source (default).

nonzero = Otherwise, when set to a 'PQ Nonzero' value the counter output PQ_Y[n] is latched at the threshold value, until the 'PQ_Counter Reset' terminal is cleared with a 'PQ Zero' value fed into it.

PQ_PWM Period/EC Enable Out = PQ output whose value depends on the operation mode selected for this C3x digital I/O pin.

PQ_PWM Period = If selected to operate as a pulse input device, the PQ ouputs returns the last overall period relative to the number of Timer 0 counts that occured between the edges of an external TTL level pulse train

Event Counter Enable Out = If selected to operate as an Event Counter, the PQ output used to indicate the state of the counter output, useful for synchronizing events. This signal may also be used as a software source to other devices (reverse from bit reading).

0 = The counter output has reached its final value, and returns a 'PQ zero' value.

nonzero = The counter is busy and returns a 'PQ One' while counting or reset.

PQ_PWM Width = If the C3x digital I/O pin is selected to operate as a pulse input device, this PQ output returns the number of Timer 0 counts that occur between every rising or falling edge of an external TTL level pulse train. Not used if the pin is selected to operate as digital I/O or an Event Counter.

PQ_PWM Last Tick Count = If the C3x digital I/O pin is selected to operate as a pulse input device, PQ output that returns the last value of the C3x's internal Timer 0 value (Hx clock/2 value). Not used if the pin is selected to operate as digital I/O or an Event Counter.

NOTE: *The H Clock rate varies depending upon the particular DSP installed on the carrier card. Currently, all Sheldon Instruments hardware supports the following H clock rates:*
1) SI-C31DSP-PCI cards. The H clock rate for the C31 DSP ranges from 20Mhz to 30Mhz.

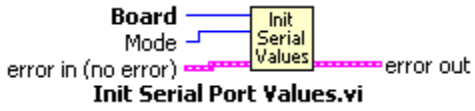
2) SI-C33DSP-PCI cards. The H clock rate for the C33 DSP is fixed at 37.5Mhz.
3) SI-C67xDSP-PCI cards. The H clock rate for the C67x DSP is fixed at 37.5Mhz.

QuBASE Equivalent.

```
'*********************************************************************
'       Function PQ_PWMEventCounterCKRCKXonC3x
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               ckPinParameters, CKPinParams
'               digitalIOPinSource, Integer
'               pqReset, Integer
'               pqWrapMode, Integer
'
'       Output Parameters:
'               oPQLastTickCount, Integer
'               oPQWidth, Integer
'               oPQPeriod, Integer
'
'*********************************************************************

  Function PQ_PWMEventCounterCKRCKXonC3x( _
        ByVal board As Integer, _
        ByRef errIn As T_Error, _
        ByRef ckPinParameters As CKPinParams, _
        Optional ByRef oPQLastTickCount As Integer = 0, _
        Optional ByRef oPQWidth As Integer = 0, _
        Optional ByRef oPQPeriod As Integer = 0, _
        Optional ByRef digitalIOPinSource As Integer = 0, _
        Optional ByRef pqReset As Integer = 0, _
        Optional ByRef pqWrapMode As Integer = 0) _
        As T_Error
```

3.3.4.2 Init Serial Port Values.

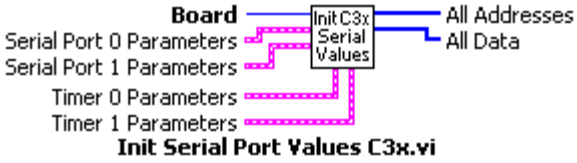
Initializes the C3x's serial port, whose pins are defaulted to be used as general purpose digital IO port or it can be reconfigured to be used to implement digital counters.



Mode = LabVIEW input, determines of the serial port is to be used for general purpose digital IO or if it is to be used as a frequency counter. To be used in conjunction with the PQ_Digital IO VIs or the PQ_Counter VIs.

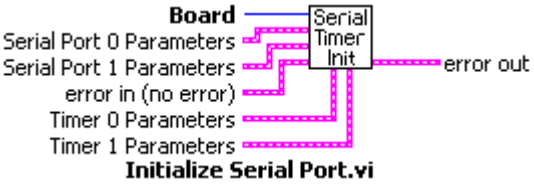
3.3.4.3 Init Serial Port Values C3x.

Initialized the C3x's serial port, whose pins are defaulted to be used as general purpose digital IO port or it can be reconfigured to be used to implement digital counters.



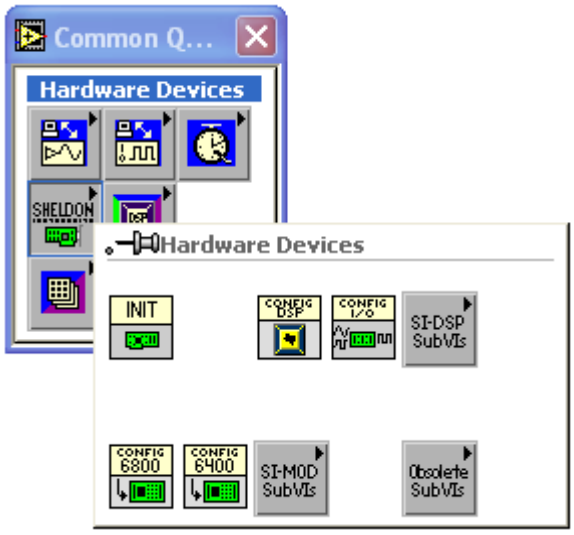
3.3.4.4 Initialize Serial Port.

Initialized the C3x's serial port, whose pins are defaulted to be used as general purpose digital IO port or it can be reconfigured to be used to implement digital counters.

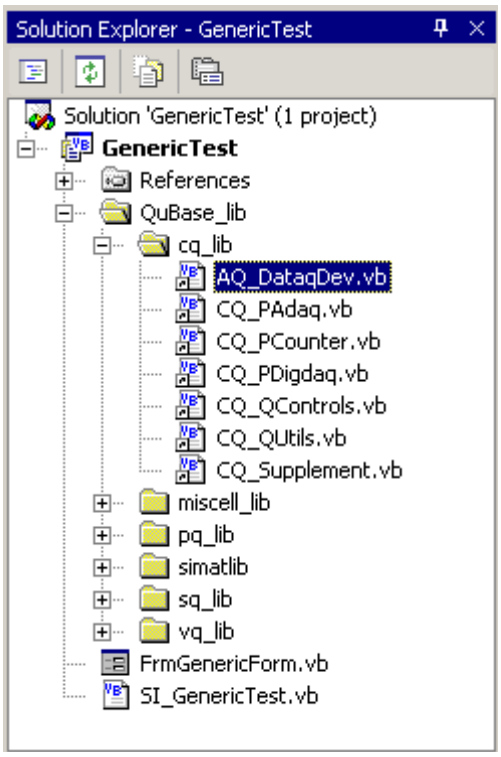


3.4 Common Queue: Hardware Devices.

These VIs control all SI-DSP and SI-MOD hardware.



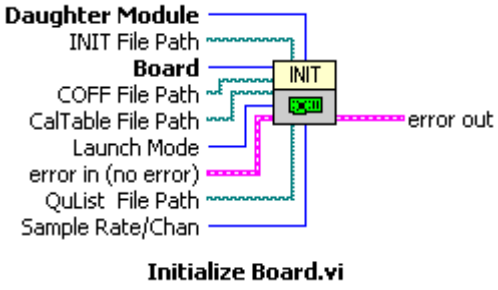
QuBASE Equivalent.



3.4.1 Initialize Board.

The Initialize Board VI performs several tasks in the LabVIEW/host and DSP domains to prepare for a QuVIEW application.

***NOTE**: The Initialize Board VI must only be used once in the beginning of a QuVIEW application.*



INIT File Path = LabVIEW filepath input used to indicate the location of the initialization COFF file for the DSP. For all DSPs, COFF files have a *.OUT extension and reside inside of the \SIC30DSP directory.

COFF File Path = LabVIEW filepath input used to indicate the location of the QuVIEW library COFF file for the DSP. For all DSPs, COFF files have a *.OUT extension and reside inside of the \SIC30DSP directory.

CalTable File Path = LabVIEW filepath input used to indicate the location of the Calibration Table for the given daughter module. Please notice this table is unique to each board, and cannot be used among different boards. The values in this table reflect each analog I/O channel's gain and offset errors. The calibration table is first placed into a file, and then may be optionally loaded into an EEPROM on the DSP carrier card if a file is not later desired. The calibration file has a *.TBL extension, and along with optional EEPROM loading, are implemented using the CALIBRATION.LLB library.

Launch Mode = LabVIEW input used to define the method by which to download files and launch DSP code execution:

0 = Mode 0: Normal COFF Load & Launch. The base COFF file is downloaded from the host to the DSP's boot memory, then the DSP is taken out of its idle or 'reset' state in order to launch DSP code execution. Default run mode, irrespective of the type of boot memory present on the DSP card (SRAM or Flash).

1 = Mode 1: Create QuList File, Load & Launch. A separate 'Netlist' file is created, which essentially contains the base COFF file along with the appended QuX 'netlist' generated from the application being run. Once the QuList file is created, the operation continues as in 'Mode 0' where file contents are downloaded to the DSP's boot memory, then the DSP is taken out of its idle or 'reset' state in order to launch DSP code execution.

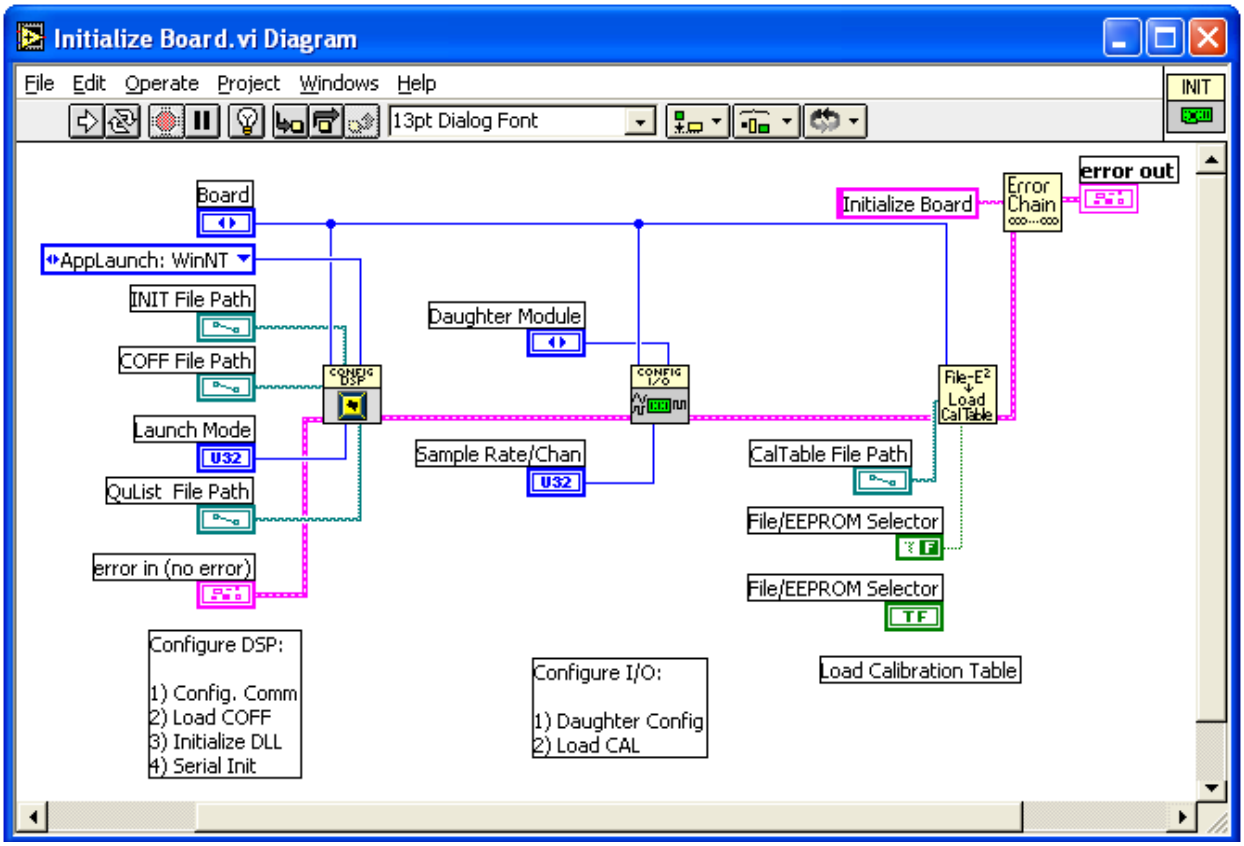
2 = Mode 2: QuList Load & Launch. Similar to 'Mode 0', except that instead of the base COFF file, the host downloads an existing 'QuList' file previously created in 'Mode 1' to the DSP's boot memory, then the DSP is taken out of its idle or 'reset' state in order to launch DSP code execution.

NOTE:

1) The QuList file is completely self contained since the entire target application resides inside of it. Particularly useful in situations where there is little or no DSP-host interaction such as the case with standalone SI hardware or if LabVIEW is absent.
2) One must take caution with the naming of this QuList file. It is recommended that the '_QuList' designation be appended to the end of the filename. Once created, the QuList File may be invoked as often as required.

QuList File Path = LabVIEW filepath input used to indicate the location of the QuList file. The QuList file essentially contains the QuX netlist appended to the base DSP COFF file.

Sample Rate/Chan = LabVIEW input used to define the sample rate per channel of a multifunction analog I/O daughter module.



File/EEPROM Selector = LabVIEW boolean input to define the source of the calibration table to be used by the DSP when calibrating analog IO in real time. The calibration table is first created and stored into a file typically named with a 'TBL' extension. Subsequently, the contents of the calibration file may also be optionally burned into an EEPROM on the DSP carrier card using this file as a template. The values of the calibration table reflect gain and offset errors of the SI-MOD's analog IO circuitry. When analog IO data are accessed from an SI-MOD card by the DSP, the calibration error adjustments are performed in real time.

False: File = Load Calibration Table from file.

True: EEPROM = Load Calibration Table from EEPROM.

NOTE:

1) This boolean is not available as an externally accessible wire terminal from the 'Initialize Board' VI, and therefore its desired position must be set as a default and then saved.

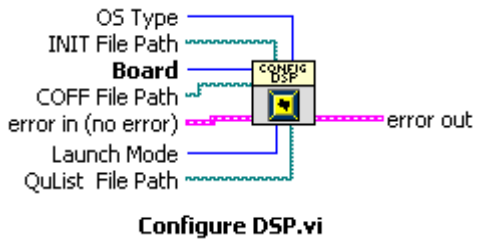
2) Please refer to the VI named 'Load Calibration Table from File (ASCII) or EEProm' for more details.

QuBASE Equivalent.

```
'*********************************************************************
'       Function AQ_InitializeBoard
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               initPath, String
'               coffPath, String
'               calPath, String
'               intModule, Integer
'               coffType, Integer
'               samplerateperchannel, Integer:
'               defaulted to 5khz
'               flashFilename, String
'
'       Output Parameters:
'
'*********************************************************************

  Function AQ_InitializeBoard( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef initPath As String, _
        ByRef coffPath As String, _
        ByRef calPath As String, _
        Optional ByRef intModule As Integer = 2, _
        Optional ByRef samplerateperchannel As Integer = 5000, _
        Optional ByVal coffType As Integer = 0, _
        Optional ByVal flashFilename As String = "", _
        Optional ByVal fileEEPROM As Integer = 0) _
        As T_Error
```

3.4.2 Configure DSP.

Configures the DSP carrier card by loading its corresponding COFF file. The COFF file contains all communications between the host PC and the DSP, along with all of QuVIEW's DSP resident functions. Intended for advanced users.



OS Type = LabVIEW input, Operating System Type. It is important to indicate what type of operating system is running, since different operating system uses different schemes for memory allocations and function calls.

INIT File Path = LabVIEW filepath input used to indicate the location of the initialization COFF file for the DSP. For C3x, it is typically C3xINI32.OUT, for the C6x it is C6711PLXINI.OUT.

COFF File Path = LabVIEW filepath input used to indicate the location of the VQLx.OUT COFF file for the DSP.

Launch Mode = LabVIEW input used to define the method by which to download files and launch DSP code execution. Please refer to the 'Initialize Board' VI for more details.

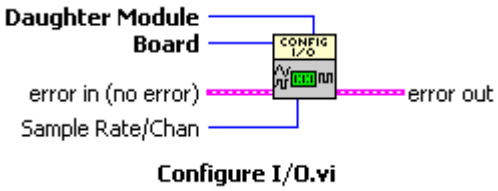
QuList File Path = LabVIEW filepath input used to indicate the location of the QuList file. The QuList file essentially contains the QuX netlist appended to the base DSP COFF file.

QuBASE Equivalent.

```
'*********************************************************************
'       Function AQ_ConfigureDSP
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               initPath, String
'               coffPath, String
'               coffType, Integer
'               flashFilename, String
'
'       Output Parameters:
'
'*********************************************************************

   Function AQ_ConfigureDSP( _
           ByRef board As Integer, _
           ByRef errIn As T_Error, _
           ByRef initPath As String, _
           ByRef coffPath As String, _
           Optional ByVal coffType As Integer = 0, _
           Optional ByVal flashFilename As String = "") _
           As T_Error
```

3.4.3 Configure I/O.

Configures the multifunction I/O modules that reside as daughter cards to the DSP carrier cards. The multifunction I/O cards contain ADCs, DACs, as well as digital I/O circuitry. Intended for advanced users.



Sample Rate/Chan = LabVIEW input used to define the sample rate per channel of a multifunction analog I/O carried by the DSP board.

Daughter Module = LabVIEW constant that defines the type of multifunction I/O module in place, it contains information about the daughter module converters' resolution and voltage scale.

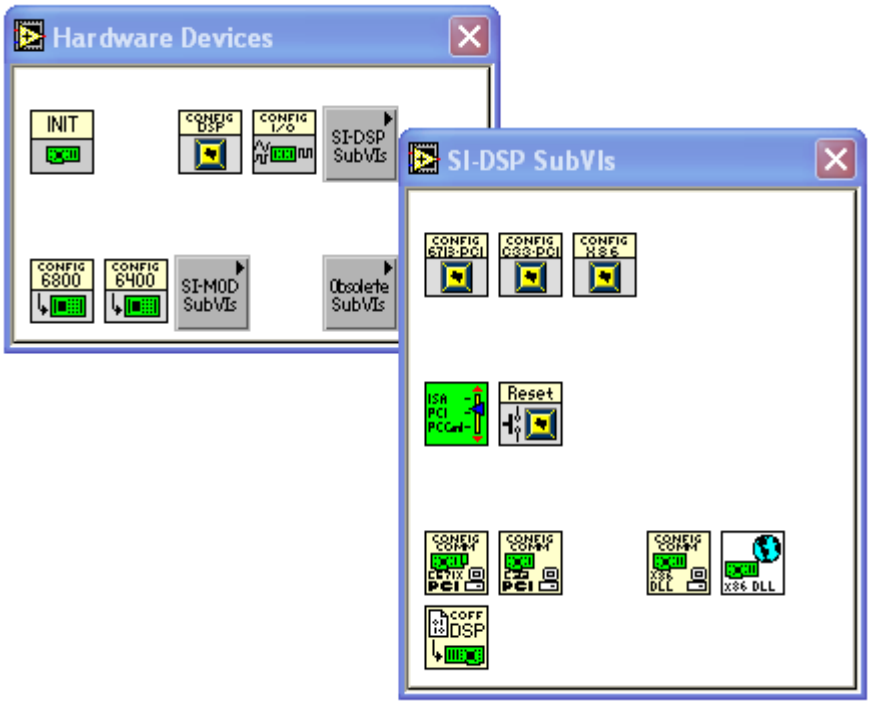
NOTE: *The H Clock rate varies depending upon the particular DSP installed on the carrier card. Currently, all Sheldon Instruments hardware supports the following H clock rates:*
1) SI-C31DSP-PCI cards. The H clock rate for the C31 DSP ranges from 20Mhz to 30Mhz.
2) SI-C33DSP-PCI cards. The H clock rate for the C33 DSP is fixed at 75Mhz.
3) SI-C67xDSP-PCI cards. The H clock rate for the C67x DSP is fixed at 75Mhz.

QuBASE Equivalent.

```
'************************************************************************
'       Function AQ_ConfigureIO
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               intModule, Integer
'               samplerateperchannel, Integer:
'               defaulted to 5k
'
'       Output Parameters:
'
'************************************************************************

  Function AQ_ConfigureIO( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef intModule As Integer, _
        Optional ByRef samplerateperchannel As Integer = 5000) _
        As T_Error
```

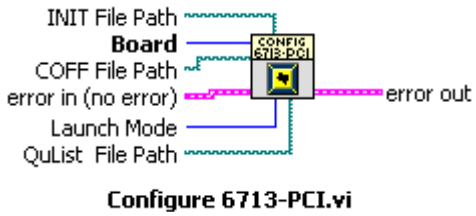
3.4.4 SI-DSP SubVIs.

These VIs are for configuring SI hardware. They are unique to specific hardware installed.



3.4.4.1 Configure C6713-PCI.

Configures the SI-C6713DSP-PCI carrier card by loading its corresponding COFF file. The COFF file contains all communications between the host PC and the DSP, along with all of QuVIEW's DSP resident functions. Intended for advanced users.



INIT File Path = LabVIEW filepath input used to indicate the location of the initialization COFF file for the DSP. For the C6713, it is C6711PLXINI.OUT.

COFF File Path = LabVIEW filepath input used to indicate the location of the VQLx.OUT COFF file for the DSP.

Launch Mode = LabVIEW input used to define the method by which to download files and launch DSP code execution. Please refer to the 'Initialize Board' VI for more details.

QuList File Path = LabVIEW filepath input used to indicate the location of the QuList file. The QuList file essentially contains the QuX netlist appended to the base DSP COFF file.

QuBASE Equivalent.

```
'*********************************************************************
'       Function AQ_ConfigureC6713PCI
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               initPath, String
'               coffPath, String
'               coffType, Integer
'               flashFilename, String
'
'       Output Parameters:
'
'*********************************************************************

  Function AQ_ConfigureC6713PCI( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef initPath As String = kInitFileC6713, _
        Optional ByRef coffPath As String = kCoffFileC6713, _
        Optional ByVal coffType As Integer = 0, _
        Optional ByVal flashFilename As String = "") _
        As T_Error
```

3.4.4.2 Configure C33-PCI.

Configures the SI-C33DSP-PCI carrier card by loading its corresponding COFF file. The COFF file contains all communications between the host PC and the DSP, along with all of QuVIEW's DSP resident functions. Intended for advanced users.

INIT File Path = LabVIEW filepath input used to indicate the location of the initialization COFF file for the DSP. For the C33, it is C33INI.OUT.

COFF File Path = LabVIEW filepath input used to indicate the location of the VQLx.OUT COFF file for the DSP.

Launch Mode = LabVIEW input used to define the method by which to download files and launch DSP code execution. Please refer to the 'Initialize Board' VI for more details.

QuList File Path = LabVIEW filepath input used to indicate the location of the QuList file. The QuList file essentially contains the QuX netlist appended to the base DSP COFF file.

QuBASE Equivalent.

```
'************************************************************************
'       Function AQ_ConfigureC33PCI
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               initPath, String
'               coffPath, String
'
'       Output Parameters:
'
'************************************************************************

  Function AQ_ConfigureC33PCI( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef initPath As String = kInitFileC33, _
        Optional ByRef coffPath As String = kCoffFileC33) _
        As T_Error
```

3.4.4.3 Configure x86DLL.

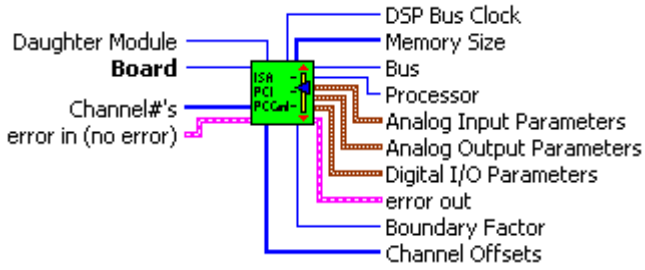
The Configure X86DLL VI invokes the 'vqlx86.dll' which acts as a DSP simulator to run QuVIEW VIs when a DSP board is absent, very convenient for debugging purposes.

QuBASE Equivalent.

```
'*************************************************************************
'       Function AQ_ConfigureX86DLL
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'
'       Output Parameters:
'
'*************************************************************************

   Function AQ_ConfigureX86DLL( _
         ByRef board As Integer, _
         ByRef errIn As T_Error, _
         As T_Error
```

3.4.4.4 Get Board Info.

The Get Board Info VI returns hardware identification as well as various useful parameters for proper hardware functionality. Intended for advanced users only.



Daughter Module = LabVIEW input that defines the SI-MOD expansion IO module present on an SI-DSP carrier; used to calculate various IO parameters such as converter resolution and scaling.

Channel#'s = LabVIEW input array, passed on directly to Channel Offsets output array.

DSP Bus Clock = LabVIEW output, defines the DSP's external bus clock rate mostly used for calculating time bases.

Memory Size = LabVIEW output array, describes the amount of memory present on each of the DSP's memory banks.

Bus = LabVIEW output that describes the bus type of the DSP card in place:

0 = ISA
1 = PCI (AMCC)
2 = PCMCIA
3 = PCI (PLX)
4 = Parallel (C6711)
5 = DLL

Processor = LabVIEW output that describes the DSP processor on the carrier card:

0 = C30
1 = C31
2 = C32
3 = C33
4 = C6711_75
5 = C671x_75
6 = C671x_66
7 = X86 (Host)

Analog Input Parameters = LabVIEW output cluster that defines the base address of the analog input hardware as seen by the DSP.

Analog Output Parameters = LabVIEW output cluster that defines the base address of the analog output hardware as seen by the DSP.

Digital I/O Parameters = LabVIEW output cluster that defines the base address of the digital I/O hardware as seen by the DSP.

Boundary Factor = LabVIEW output to describe the native address boundary, in terms

of bytes, used by the DSP.

Channel Offsets = LabVIEW output array that directly reflects the array present on the Channels#'s input terminal.

QuBASE Equivalent.

```
'*************************************************************************
'       Function AQ_GetBoardInfo
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               iChannelArray, Integer
'               intModule, Integer
'
'       Output Parameters:
'               oBus, Integer
'               oProcessor, Integer
'               oDSPClockSpeed, Integer
'               oBoundaryFactor, Integer
'               oMemoryBank0Size, Integer
'               oMemoryBank1Size, Integer
'               oAnalogInputParams, AnalogIOParams
'               oAnalogOutputParams, AnalogIOParams
'               oDigitalIOParams, Integer
'               oChannelOffsets, Integer
'
'*************************************************************************

  Function AQ_GetBoardInfo( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef iChannelArray() As Integer, _
        ByRef oAnalogInputParams As AnalogIOParams, _
        ByRef oAnalogOutputParams As AnalogIOParams, _
        ByRef oDigitalIOParams() As Integer, _
        ByRef oChannelOffsets() As Integer, _
        Optional ByRef intModule As Integer = 3, _
        Optional ByRef oBus As Integer = 1, _
        Optional ByRef oProcessor As Integer = 1, _
        Optional ByRef oDSPClockSpeed As Integer = 5, _
        Optional ByRef oBoundaryFactor As Integer = 1, _
        Optional ByVal oMemoryBank0Size As Integer = 0, _
        Optional ByVal oMemoryBank1Size As Integer = 0) _
        As T_Error
```

3.4.4.5 Reset DSP.

The Reset DSP VI enables or disables the DSP, and only used once during card initialization. Intended for advanced users only.

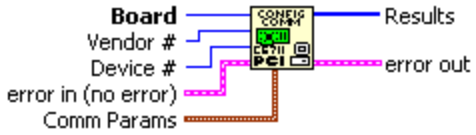
Reset? = LabVIEW input which selects to Reset the DSP or not:
Reset?: 0 = Assert, disables the DSP.
Reset?: 1 = Deassert, activates the DSP.
Reset?: 2 = Toggle, default setting that toggles the DSP's Reset line in order to restart the COFF file loading process.

QuBASE Equivalent.

```
'*********************************************************************
'       Function AQ_ResetDSP
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               intReset, T_Error
'
'       Output Parameters:
'
'*********************************************************************

  Function AQ_ResetDSP( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef intReset As Integer) _
        As T_Error
```

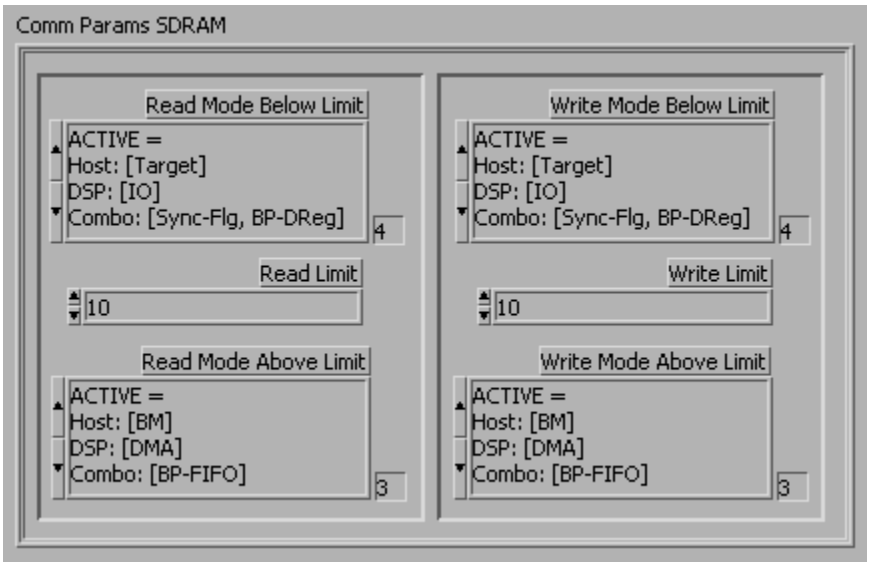
3.4.4.6 Configure PCI Communications (C671x PLX).

Configures the communication mode to be used between the host and the DSP for the SI-C6xDSP-PCI based boards, and only used once during card initialization.



Configure PCI Communications (C6711 PLX).vi

Vendor # = Ignored.
Device # = Ignored.
Comm Params = A LabVIEW input cluster of various parameters to specify what communication mode to be used. The user can define the type of communication mode that best suits a particular application between a specific boundary. The parameters to set are as follows:



Read Mode Below Limit = Communication mode for reading below boundary limit.

Read Mode Above Limit = Communication mode for reading below boundary limit.

Read Limit = Boundary value for host reads.

Write Mode Below Limit = Communication mode for writing below boundary limit.

Write Mode above Limit = Communication mode for writing above boundary limit.

Write limit = Boundary value for host writes.

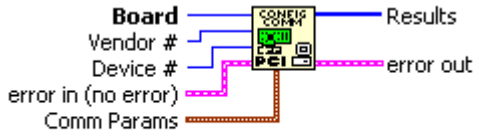
Results = LabVIEW output array returning the communication parameters selected.

QuBASE Equivalent.

```
'**********************************************************************
'       Function AQ_ConfigurePCICommunicationsPLXC671x
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vendorID, Integer
'               deviceID, Integer
'               readModeBelowLimit, Integer
'               readModeAboveLimit,, Integer
'               readDMALimit, Integer
'               writeModeBelowLimit, Integer
'               writeModeAboveLimit, Integer
'               writeDMALimit, Integer
'
'       Output Parameters:
'               oResults, Integer
'
'**********************************************************************

  Function AQ_ConfigurePCICommunicationsPLXC671x( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef oResults() As Integer, _
        Optional ByRef vendorID As Integer = VendorID, _
        Optional ByRef deviceID As Integer = 0, _
        Optional ByRef readModeBelowLimit As Integer = 4, _
        Optional ByRef readModeAboveLimit As Integer = 3, _
        Optional ByRef readDAMLimit As Integer = 10, _
        Optional ByRef writeModeBelowLimit As Integer = 4, _
        Optional ByRef writeModeAboveLimit As Integer = 3, _
        Optional ByRef writeDAMLimit As Integer = 10) As T_Error
```

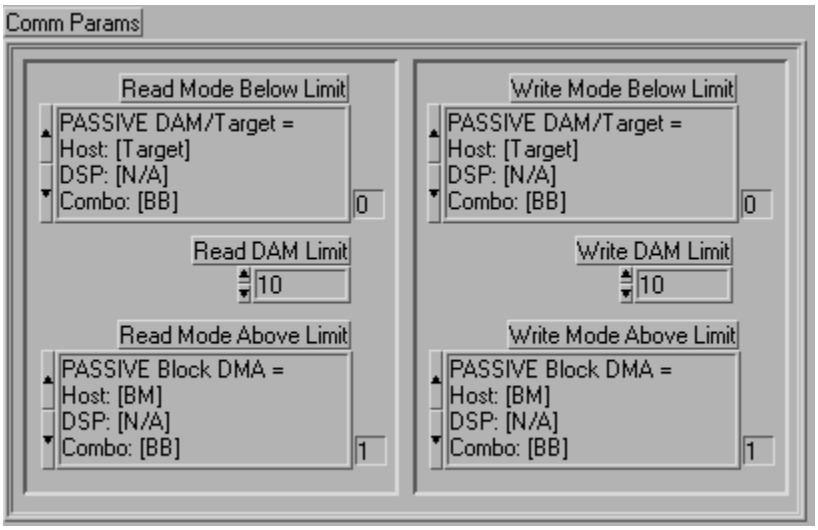
3.4.4.7 Configure PCI Communications (C33 PLX).

The Configure PCI Communications (C33 PLX) VI configures the communication mode to be used between the host and the DSP for the SI-C33DSP-PCI based boards, and only used once during card initialization.



Configure PCI Communications (C33 PLX).vi

Vendor # = Ignored.
Device # = Ignored.
Comm Params = A LabVIEW input cluster of various parameters to specify what communication mode to be used. The user can define the type of communication mode that best suits a particular application between a specific boundary. The parameters to set are as follows:



Read Mode Below Limit = Communication mode for reading below boundary limit.

Read Mode Above Limit = Communication mode for reading below boundary limit.

Read Limit = Boundary value for host reads.

Write Mode Below Limit = Communication mode for writing below boundary limit.

Write Mode above Limit = Communication mode for writing above boundary limit.

Write limit = Boundary value for host writes.

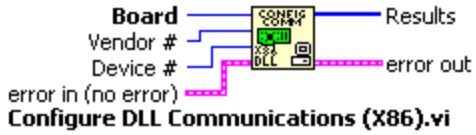
Results = LabVIEW output array returning the communication parameters selected.

QuBASE Equivalent.

```
'**********************************************************************
'       Function AQ_ConfigurePCICommunicationsPLX
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vendorID, Integer
'               deviceID, Integer
'               readModeBelowLimit, Integer
'               readModeAboveLimit,, Integer
'               readDMALimit, Integer
'               writeModeBelowLimit, Integer
'               writeModeAboveLimit, Integer
'               writeDMALimit, Integer
'
'       Output Parameters:
'               oResults, Integer
'
'**********************************************************************

  Function AQ_ConfigurePCICommunicationsPLX( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef oResults() As Integer, _
        Optional ByRef vendorID As Integer = VendorID, _
        Optional ByRef deviceID As Integer = 0, _
        Optional ByRef readModeBelowLimit As Integer = 0, _
        Optional ByRef readModeAboveLimit As Integer = 1, _
        Optional ByRef readDAMLimit As Integer = 10, _
        Optional ByRef writeModeBelowLimit As Integer = 0, _
        Optional ByRef writeModeAboveLimit As Integer = 1, _
        Optional ByRef writeDAMLimit As Integer = 10) As T_Error
```

3.4.4.8 Configure DLL Communications (x86).

The Configure DLL Communications (x86) VI configures communications for the host x86, used mainly for simulation purposes when an SI-DSP card is absent.



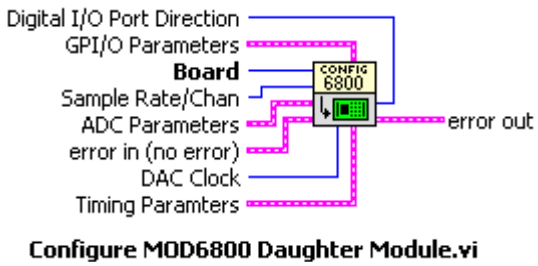
Vendor # = Ignored.
Device # = Ignored.
Results = Ignored.

QuBASE Equivalent.

```
'*********************************************************************
'       Function AQ_DLLInitialize
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'
'       Output Parameters:
'
'*********************************************************************

  Function AQ_DLLInitialize( _
        ByRef board As Integer, _
        ByRef errIn As T_Error) _
        As T_Error
```

3.4.5 Configure MOD6800 Daughter Module.

Configures the SI-MOD6800 daughter module, please consult the SI-MOD6800 documentation for more details.

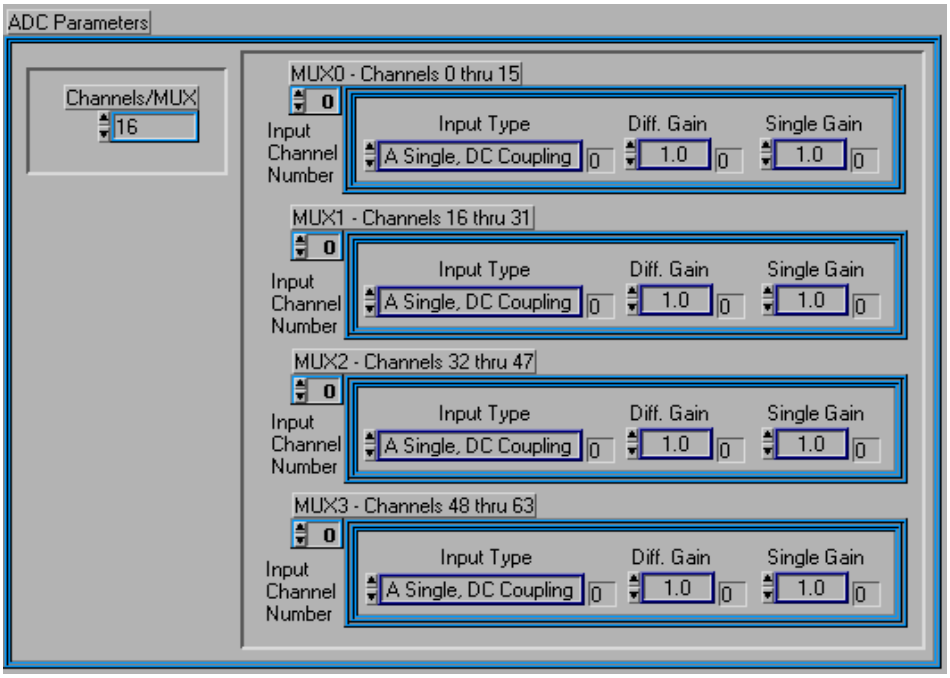


Digital I/O Port Direction = LabVIEW input, defines the direction of each of the 16 bit digital I/O pairs, which forms a complete 32 bit digital IO port.

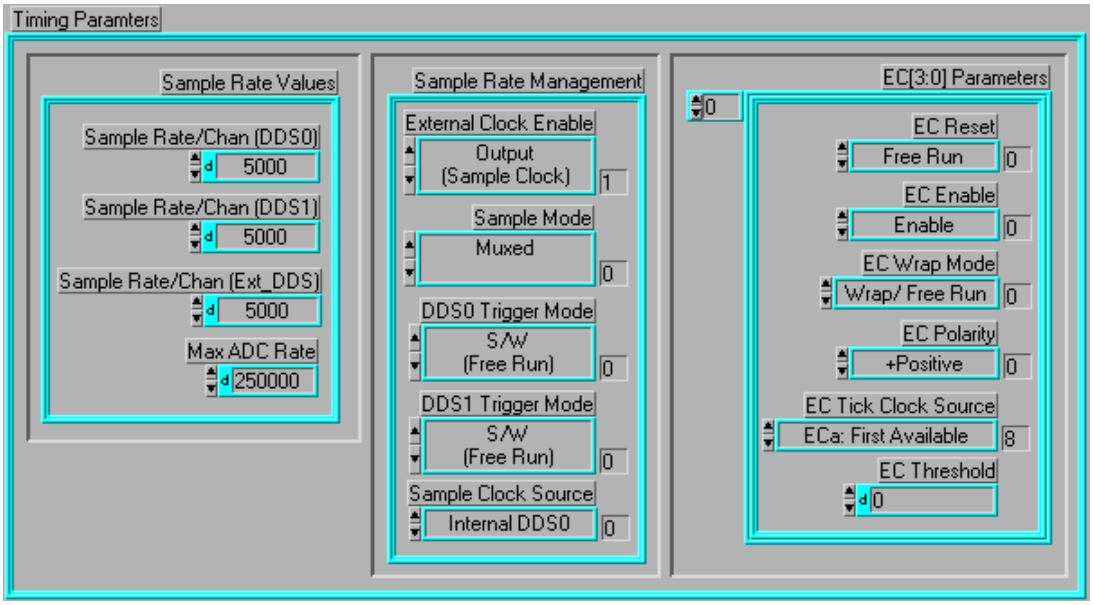
Sample Rate/Chan = LabVIEW input used to define the sample rate per channel.

DAC Clock = LabVIEW input, not implemented.

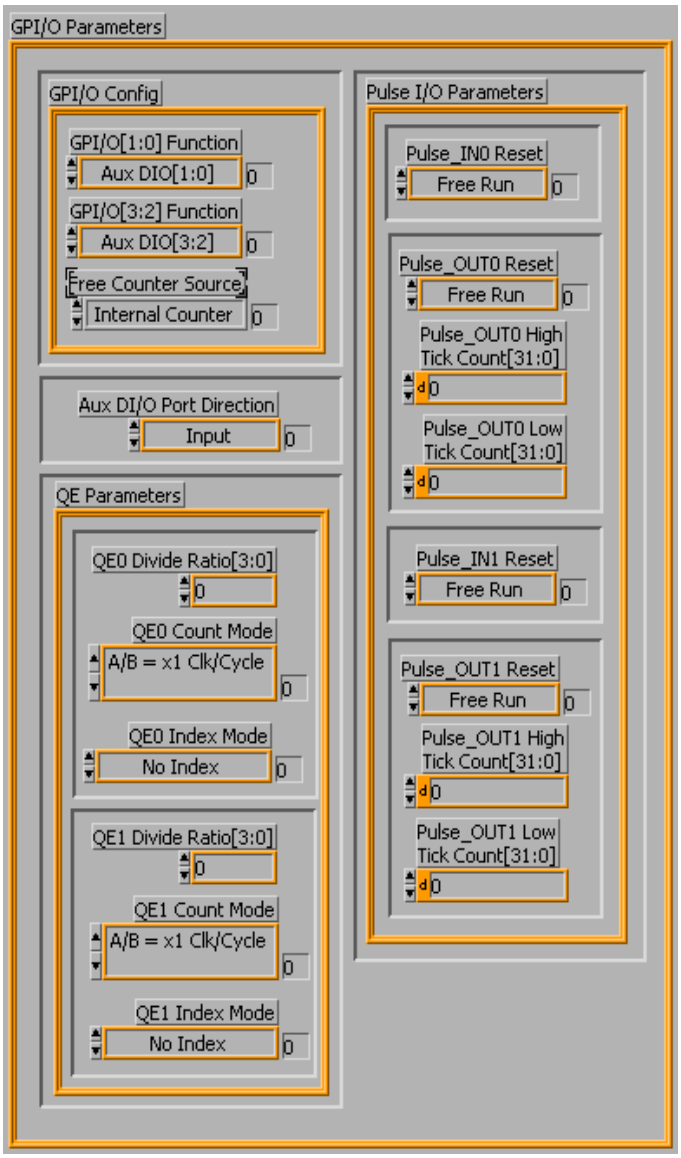
ADC Parameters = LabVIEW input cluster, defines the channel parameters as well as the number of channels to multiplex per ADC group.



Timing Parameters = LabVIEW input cluster, defines the overall timing parameters.



GP I/O Parameters = LabVIEW input cluster, defines the operation of the on board digital peripherals.

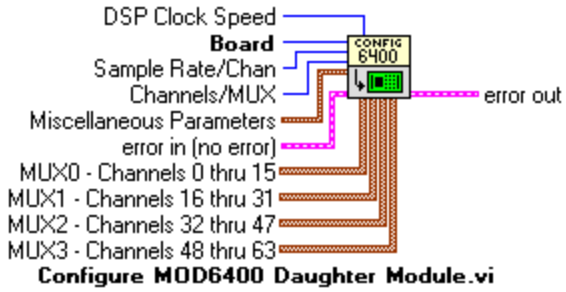


QuBASE Equivalent.

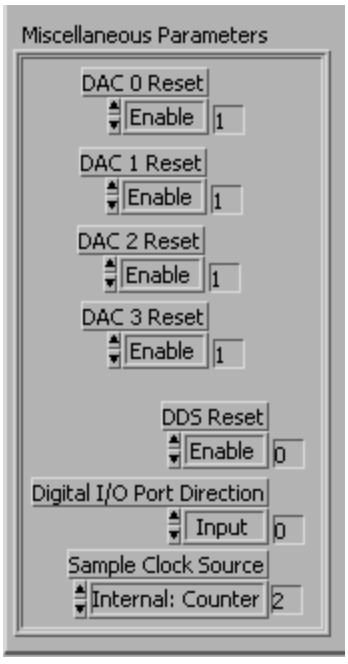
```
'*********************************************************************
'       Function AQ_ConfigureMOD6800
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               iGPIOParams, T_6800GPIOParameters
'               iTimingParams, T_6800TimingParameters
'               iADCParams, T_6800ADCParameters
'               iDigitalIOPortDirection, Integer
'               iDACClock, Integer
'               iSamplerateperchannel, Integer
'
'       Output Parameters:
'
'*********************************************************************

   Function AQ_ConfigureMOD6800( _
            ByRef board As Integer, _
            ByRef errIn As T_Error, _
            ByRef iGPIOParams As T_6800GPIOParameters, _
            ByRef iTimingParams As T_6800TimingParameters, _
            ByRef iADCParams As T_6800ADCParameters, _
            Optional ByVal iDigitalIOPortDirection As Integer = 0, _
            Optional ByVal iDACClock As Integer = 0, _
            Optional ByRef iSamplerateperchannel As Integer = 5000) _
            As T_Error
```

3.4.6 Configure MOD6400 Daughter Module.

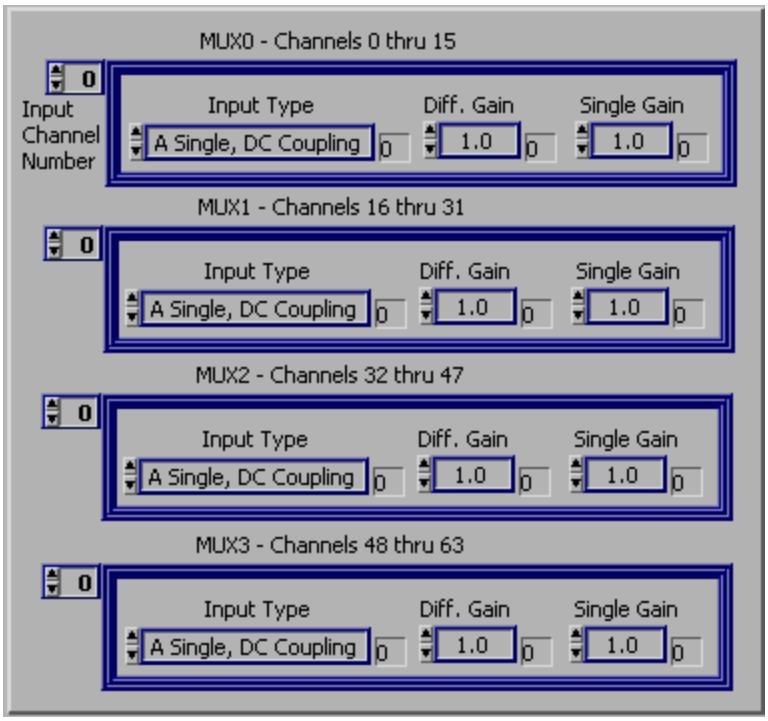
Configures the SI-MOD6400 daughter module, please consult the SI-MOD6400 documentation for more details.



DSP Clock Speed = LabVIEW input to define the DSP clock speed.
Sample Rate/Chan = LabVIEW input used to define the sample rate per channel.
Channels/MUX = LabVIEW input, defines the number of channels to multiplex per ADC group.
Miscellaneous Parameters = LabVIEW input cluster, configures miscellaneous hardware peripherals such as optional DDS and digital I/O port direction, as well as DAC enabling.



MUXn - Channels XX through YY = LabVIEW input cluster, configures the analog input hardware operation.


MUX0 - Channels 0 thru 15
0
Input Channel Number
Input Type
A Single, DC Coupling
0
Diff. Gain
1.0
0
Single Gain
1.0
0
MUX1 - Channels 16 thru 31
0
Input Type
A Single, DC Coupling
0
Diff. Gain
1.0
0
Single Gain
1.0
0
MUX2 - Channels 32 thru 47
0
Input Type
A Single, DC Coupling
0
Diff. Gain
1.0
0
Single Gain
1.0
0
MUX3 - Channels 48 thru 63
0
Input Type
A Single, DC Coupling
0
Diff. Gain
1.0
0
Single Gain
1.0
0

QuBASE Equivalent.

```
'*************************************************************************
'       Function AQ_ConfigureMOD6400
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               mux0, T_MuxParams
'               mux1, T_MuxParams
'               mux2, T_MuxParams
'               mux3, T_MuxParams
'               miscparams, T_MuxParams
'               samplerateperchannel, Integer
'               channelspermux, Integer
'
'       Output Parameters:
'
'*************************************************************************

   Function AQ_ConfigureMOD6400( _
          ByRef board As Integer, _
          ByRef errIn As T_Error, _
          ByRef mux0() As T_MuxParams, _
          ByRef mux1() As T_MuxParams, _
          ByRef mux2() As T_MuxParams, _
          ByRef mux3() As T_MuxParams, _
          ByRef miscparams As T_MiscParams, _
          Optional ByRef samplerateperchannel As Integer = 5000, _
          Optional ByRef channelspermux As Integer = 16) _
          As T_Error
```

3.4.7 SI-MOD SubVIs

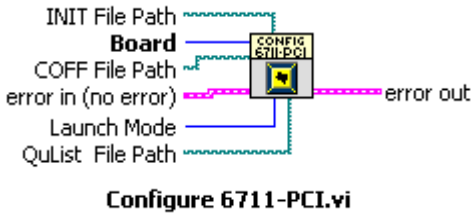
These are subVIs used by the SI-MOD hardware and generally not intended to be modified.

3.4.8 Obsolete SubVIS

These SubVIs are no longer used, but remain for legacy users.

3.4.8.1 Configure C6711-PCI.

Configures the SI-C6711DSP-PCI carrier card by loading its corresponding COFF file. The COFF file contains all communications between the host PC and the DSP, along with all of QuVIEW's DSP resident functions. Intended for advanced users.



INIT File Path = LabVIEW filepath input used to indicate the location of the initialization COFF file for the DSP. For the C6711, it is C6711PLXINI.OUT.

COFF File Path = LabVIEW filepath input used to indicate the location of the VQLx.OUT COFF file for the DSP.

Launch Mode = LabVIEW input used to define the method by which to download files and launch DSP code execution. Please refer to the 'Initialize Board' VI for more details.

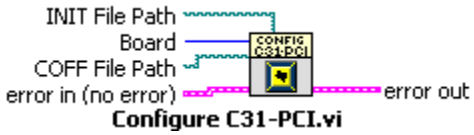
QuList File Path = LabVIEW filepath input used to indicate the location of the QuList file. The QuList file essentially contains the QuX netlist appended to the base DSP COFF file.

QuBASE Equivalent.

```
'*********************************************************************
'       Function AQ_ConfigureC6711PCI
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               initPath, String
'               coffPath, String
'               coffType, Integer
'               flashFilename, String
'
'       Output Parameters:
'
'*********************************************************************

  Function AQ_ConfigureC6711PCI( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef initPath As String = kInitFileC6711, _
        Optional ByRef coffPath As String = kCoffFileC6711, _
        Optional ByVal coffType As Integer = 0, _
        Optional ByVal flashFilename As String = "") _
        As T_Error
```

3.4.8.2 Configure C31-PCI.

Configures the SI-C31DSP-PCI carrier card by loading its corresponding COFF file. The COFF file contains all communications between the host PC and the DSP, along with all of QuVIEW's DSP resident functions. Intended for advanced users.

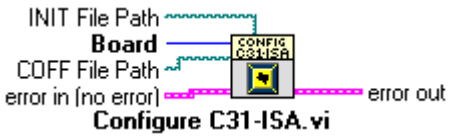


INIT File Path = LabVIEW filepath input used to indicate the location of the initialization COFF file for the DSP. For the C31, it is PCIINI32.OUT.

COFF File Path = LabVIEW filepath input used to indicate the location of the VQLx.OUT COFF file for the DSP.

3.4.8.3 Configure C31-ISA.

Configures the SI-C31DSP-ISA carrier card by loading its corresponding COFF file. The COFF file contains all communications between the host PC and the DSP, along with all of QuVIEW's DSP resident functions. Intended for advanced users.

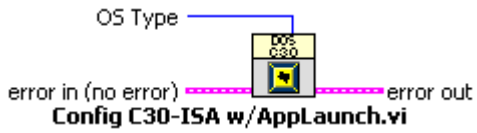


INIT File Path = LabVIEW filepath input used to indicate the location of the initialization COFF file for the DSP. For the C31, it is C31INI32.OUT.

COFF File Path = LabVIEW filepath input used to indicate the location of the VQLx.OUT COFF file for the DSP.

3.4.8.4 Configure C30-ISA w/ AppLaunch.

Alternate method to configure the SI-C30DSP-ISA carrier card by loading its corresponding COFF file using a command line executable. The COFF file contains all communications between the host PC and the DSP, along with all of QuVIEW's DSP resident functions. Intended for advanced users.



OS Type = LabVIEW input, Operating System Type. It is important to indicate what type of operating system is running, since different operating system uses different schemes for memory allocations and function calls.

3.4.8.5 Configure C30-ISA.

Configures the SI-C30DSP-ISA carrier card by loading its corresponding COFF file. The COFF file contains all communications between the host PC and the DSP, along with all of QuVIEW's DSP resident functions. Intended for advanced users.

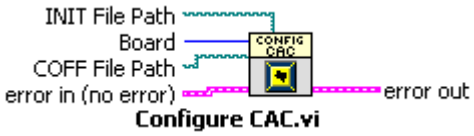
OS Type = LabVIEW input, Operating System Type. It is important to indicate what type of operating system is running, since different operating system uses different schemes for memory allocations and function calls.

INIT File Path = LabVIEW filepath input used to indicate the location of the initialization COFF file for the DSP. the the C30, it is C30INI32.OUT.

COFF File Path = LabVIEW filepath input used to indicate the location of the VQLx.OUT COFF file for the DSP.

3.4.8.6 Configure CAC.

Configures the SI-BULLET PCMCIA card by loading its corresponding COFF file. The COFF file contains all communications between the host PC and the DSP, along with all of QuVIEW's DSP resident functions. Intended for advanced users.

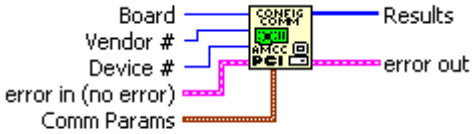


INIT File Path = Unused LabVIEW filepath input, placed for compatibility with other similar VIs.

COFF File Path = LabVIEW filepath input used to indicate the location of the VQLx.OUT COFF file for the DSP.

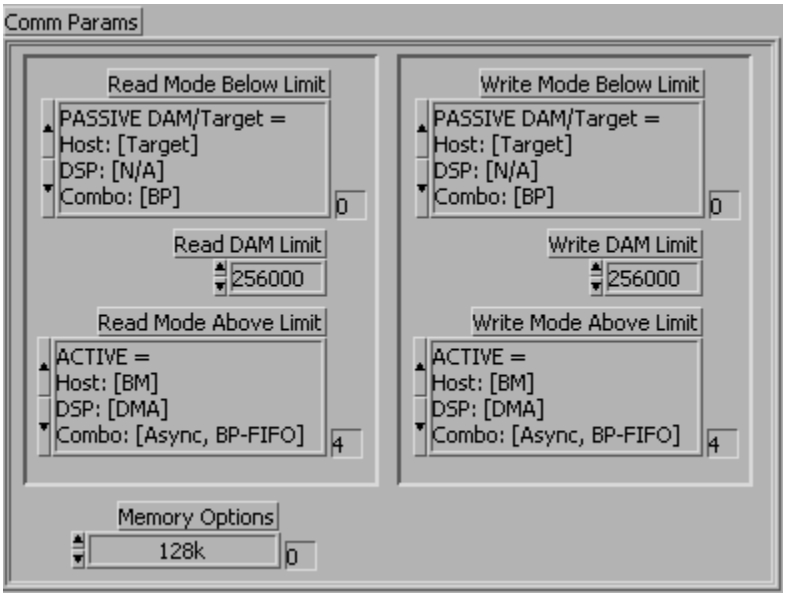
3.4.8.7 Configure PCI Communications (AMCC).

Configures the communication mode to be used between the host and the DSP for the SI-C31DSP-PCI based boards. Only executed once during the COFF file loading to the DSP.



Configure PCI Communications (AMCC).vi

Vendor # = Ignored.
Device # = Ignored.
Comm Params = A LabVIEW input cluster of various parameters to specify what communication mode to be used. The user can define the type of communication mode that best suits a particular application between a specific boundary. The parameters to set are as follows:



Read Mode Below Limit = Communication mode for reading below boundary limit.

Read Mode Above Limit = Communication mode for reading below boundary limit.

Read Limit = Boundary value for host reads.
Write Mode Below Limit = Communication mode for writing below boundary limit.

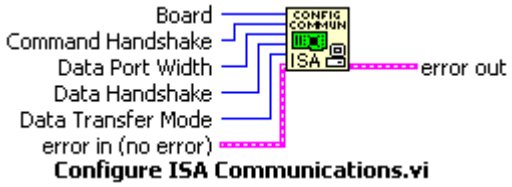
Write Mode above Limit = Communication mode for writing above boundary limit.

Write limit = Boundary value for host writes.
Memory options = Memory size installed on DSP carrier card.

Results = LabVIEW output array returning the communication parameters selected.

3.4.8.8 Configure ISA Communications.

Configures the communication mode to be used between the host and the DSP for the SI-C3xDSP-ISA based boards, and only used once during card initialization.



This is a low level subVI that is not typically altered by the user. However, when the IO and memory mapping is different from the software default values, these software values can be changed so as to coincide with the base addresses in hardware.

Command Handshake = LabVIEW input, handshaking protocol used only during the command phase of communications.

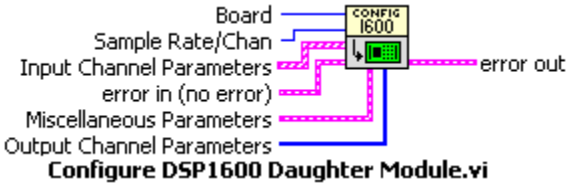
Data Port Width = LabVIEW input, specifies the data/communication port width present in hardware, either 16 or 32 bits wide.

Data Handshake = LabVIEW input, handshaking protocol used only during the data phase of communications.

Data Transfer Mode = LabVIEW input, specifies whether the DSP performs transfers to the host using programmed IO or it DMA engine.

3.4.8.9 Configure MOD1600 Daughter Module.

Configures the SI-MOD1600 daughter module, please consult the SI-MOD1600 documentation for more details.



Sample Rate/Chan = LabVIEW input that specifies the sample rate per channel.

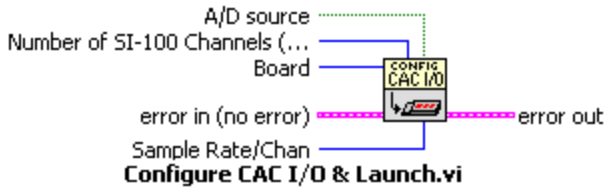
Input Channel Parameters = LabVIEW input cluster, configures the analog input hardware operation.

Output Channel Parameters = LabVIEW input array, that specifies the maximum output voltage swing to be set at either ±3Vpp or ±5Vpp.

Miscellaneous Parameters = LabVIEW input cluster, configures miscellaneous hardware peripherals such as optional DDS and DAC enabling.

3.4.8.10 Configure CAC I/O & Launch.

Configures the SI-Bullet I/O module, please consult the SI-Bullet documentation for more details.

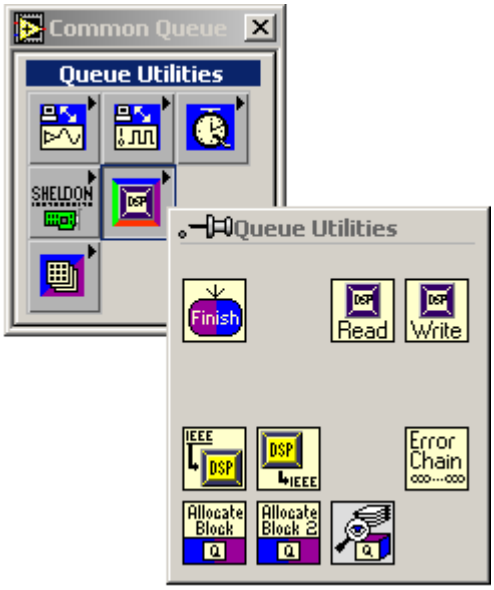


AD source = LabVIEW input, determines of the analog I/O source is the SI-100 or the onboard analog I/O stereo Codecs.

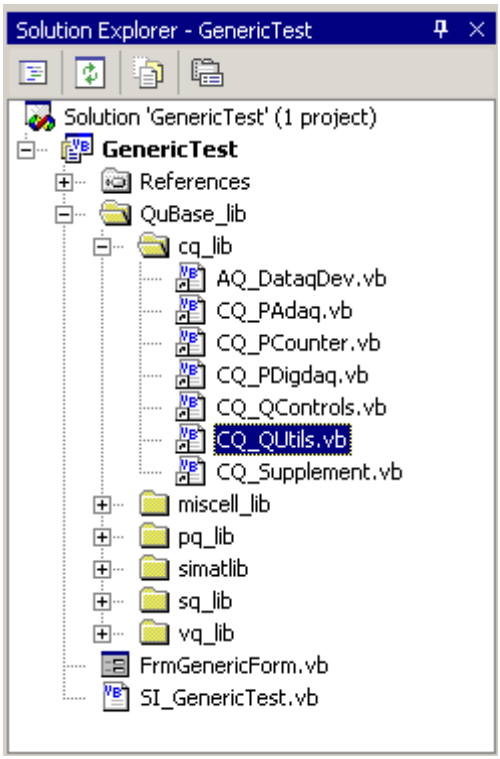
Number of SI-100 Channels = LabVIEW input, if the SI-100 is selected, specifies the number of channels transferred from the SI-100.

Sample Rate/Chan = LabVIEW input, if the onboard analog I/O stereo Codecs are selected, the sample rate per channel is specified.

3.5 Common Queue: Queue Utilities.



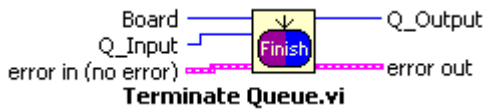
QuBASE Equivalent.



3.5.1 Terminate Queue.

The Terminate Queue VI function is used as an indicator to the DSP that the function list is complete. After receiving this indicator, the DSP commences code execution.

***NOTE**:*
1) Only a single 'Terminate Queue' VI is required per QuVIEW application.
2) It may be necessary to maintain precedence within LabVIEW by wiring in the 'Error In' terminal. In this way, the 'Terminate Queue' VI will be guaranteed to execute last.



Q_Input = Queuing input, may be either VQ or PQ variables, and optionally used to force precedence. The use of the 'error in' and 'error out' cluster terminals is recommended for imposing deterministic precedence.

Q_Output = Queuing output, may be either VQ or PQ variables, and optionally used to force precedence.

QuX Image Size (DWords) = LabVIEW output, the number of actual DWords (four bytes per DWord) contained in the runtime image that is downloaded from the host to the DSP's boot memory. The host creates a runtime image that is derived from either the base COFF or QuList files, which reflects the actual machine code loaded by the DSP at runtime. Particularly useful for verifying that the DSP carrier is configured with sufficient boot memory.

QuBASE Equivalent.

```
'************************************************************************
'       Function CQ_TerminateQueue
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'
'       Output Parameters:
'
'************************************************************************
  Function CQ_TerminateQueue( _
        ByRef board As Integer, _
        ByRef errIn As T_Error) _
        As T_Error
```

3.5.2 Read from DSP board.

The Read from DSP board VI allows the host to read one or more 32 bit words from any location within the DSP's memory space. Intended for advanced users.

Count = LabVIEW input that sets the number of sequential DWord locations that are read from the DSP.

Start Address = LabVIEW input used to indicate an address location within the DSP's address space from where the reads are to begin.

RW Mode = LabVIEW input that determines the communications mode. The following options apply:

RW Mode: 0 = Default. Uses the mode specified in the 'Configure Communications' subVI.

RW Mode: 1 = Hostpoll. Forces the "hostpoll" mode of communications between the host and DSP; please consult the SICommModes documentation for more details.

RW Mode: 2 = Debug. Uses the actual address as seen from the host side. The Address field no longer refers to the address as seen from the DSP. Useful for debugging.

Data Read = LabVIEW array containing the requested data from the DSP.

QuBASE Equivalent.

```
'*************************************************************************
'       Function CQ_ReadFromDSPBoard
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               rwMode, Integer
'               startAddress, Integer
'               datacount, Integer
'
'       Output Parameters:
'               dataRead, Integer
'
'*************************************************************************

   Function CQ_ReadFromDSPBoard( _
          ByRef board As Integer, _
          ByRef errIn As T_Error, _
          ByRef dataRead() As Integer, _
          Optional ByRef rwMode As Integer = 0, _
          Optional ByRef startAddress As Integer = 0, _
          Optional ByRef datacount As Integer = 0) _
          As T_Error
```

3.5.3 Write to DSP board.

The Write to DSP board VI allows the host to write one or more 32 bit words to any location within the DSP's memory space. Intended for advanced users.

Data to Write = LabVIEW input array containing the hex data to be written to the DSP's address space.

Start Address = LabVIEW input used to indicate an address location within the DSP's address space from where the writes are to begin.

RW Mode = LabVIEW input that determines the communications mode. The following options apply:

RW Mode: 0 = Default. Uses the mode specified in the 'Configure Communications' subVI.

RW Mode: 1 = Hostpoll. Forces the "hostpoll" mode of communications between the host and DSP; please consult the SICommModes documentation for more details.

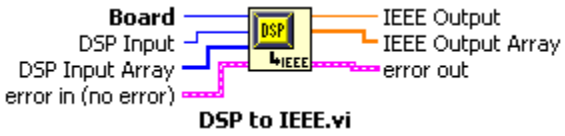
RW Mode: 2 = Debug. Uses the actual address as seen from the host side. The Address field no longer refers to the address as seen from the DSP. Useful for debugging.

QuBASE Equivalent.

```
'*************************************************************************
'       Function CQ_WriteToDSPBoard
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               dataWrite, Integer
'               rwMode, Integer
'               startAddress, Integer
'
'       Output Parameters:
'
'*************************************************************************

  Function CQ_WriteToDSPBoard( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef dataWrite() As Integer, _
        Optional ByRef rwMode As Integer = 0, _
        Optional ByRef startAddress As Integer = &HD) _
        As T_Error
```

3.5.4 DSP to IEEE.

This VI is strictly performed on the host and serves to translate the floating point format from the DSP domain (if necessary) to the native IEEE floating point format used on the host.



DSP Input = LabVIEW input formatted with the DSP floating point scheme. Ignored if the 'DSP Input Array' array terminal is wired instead.

DSP Input Array = LabVIEW input array formatted with the DSP floating point scheme. Convenient if an array of variables is used as it avoids the wiring of separate icons. If wired, it becomes the default terminal used, and the single variable 'DSP Input' terminal is ignored.

IEEE Output = LabVIEW output formatted with the IEEE floating point scheme for use by LabVIEW VIs. Not used if an array terminal was wired.

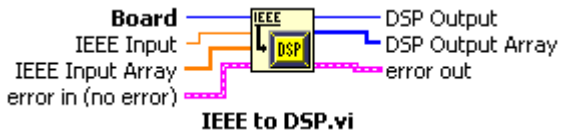
IEEE Output Array = LabVIEW output array formatted with the IEEE floating point scheme for use by LabVIEW VIs. If an input array terminal was used, then the output array should be correspondingly used.

QuBASE Equivalent.

```
'*********************************************************************
'       Function CQ_DSPToIEEE
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               intDSPInputArray, Integer
'
'       Output Parameters:
'               dblIEEEOutputArray, Double
'
'*********************************************************************

  Function CQ_DSPToIEEE( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef intDSPInputArray() As Integer, _
        ByRef dblIEEEOutputArray() As Double) _
        As T_Error
```

3.5.5 IEEE to DSP.

This VI is strictly performed on the host and serves to translate the floating point format used on the host to the floating point format used on the DSP (if necessary).



IEEE Input = LabVIEW input formatted with the IEEE floating point scheme. Ignored if the 'IEEE Input Array' array terminal is wired instead.

IEEE Input Array = LabVIEW input array formatted with the IEEE floating point scheme. Convenient if an array of variables is used as it avoids the wiring of separate icons. If wired, it becomes the default terminal used, and the single variable 'IEEE Input' terminal is ignored.

DSP Output = LabVIEW output formatted with the DSP floating point scheme for use by the DSP. Not to be used if an array terminal is wired.

DSP Output Array = LabVIEW output array formatted with the DSP floating point scheme for use by the DSP. If an input array terminal was used, then the output array should be correspondingly used.

QuBASE Equivalent.

```
'*************************************************************************
'       Function CQ_IEEEToDSP
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               dblIEEEInputArray, Double
'
'       Output Parameters:
'               intDSPOutputArray, Integer
'
'*************************************************************************

  Function CQ_IEEEToDSP( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef dblIEEEInputArray() As Double, _
        ByRef intDSPOutputArray() As Integer) _
        As T_Error
```

3.5.6 Error Chainer.

Daisy chains error codes and messages from previous VIs to the following set of VIs. If no errors ocurr, outputs simply mirror inputs.

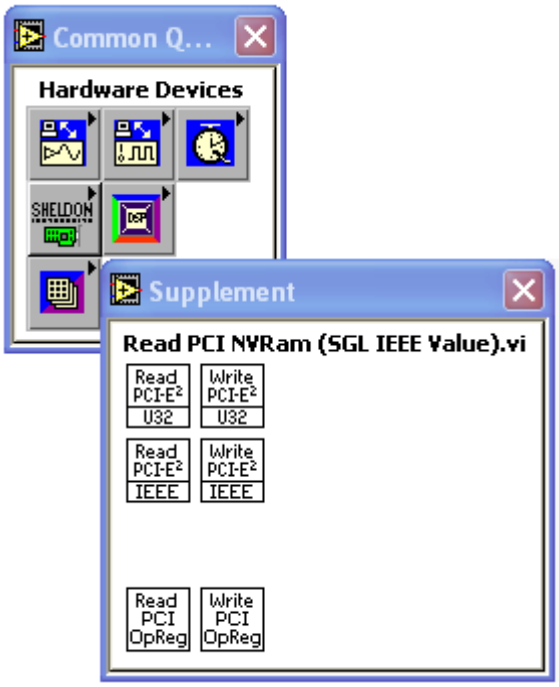
Error = LabVIEW input, error code reference number.
Error Source = A LabVIEW string or message describing the current error source.

Error Out = A LabVIEW output formatted as a cluster.

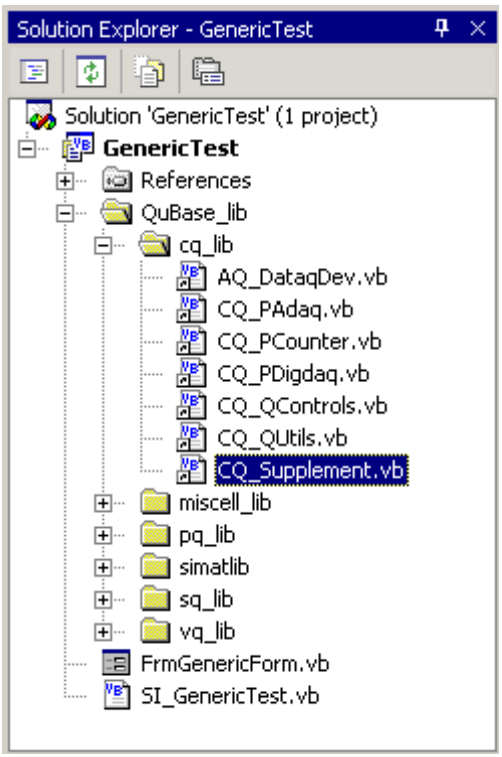
QuBASE Equivalent.

```
'*************************************************************************
'       Function CQ_ErrorChainer
'
'       Description:
'               Used for passing error messages between functions.
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'
'       Output Parameters:
'
'*************************************************************************

   Function CQ_ErrorChainer( _
          ByRef errorCode As Integer, _
          ByRef errorSource As String, _
          ByRef errorIn As T_Error) _
          As T_Error
```

3.6 Supplemental VIs.

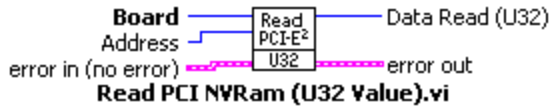


QuBASE Equivalent.



3.6.1 Read PCI NVRAM (U32 Value).

The Read PCI NVRAM (U32 Value) VI is used to read four bytes in U32 format that reside inside of the PCI bridge device's configuration NVRAM. Intended for advanced users only.



Address = LabVIEW input, byte boundary address of NVRAM.

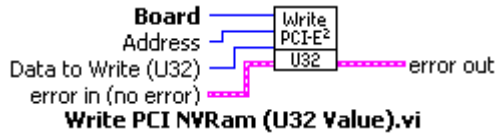
Data Read (U32) = LabVIEW output, depending upon the PCI bridge device, either byte or DWord values are read from NVRAM.

QuBASE Equivalent.

```
'*********************************************************************
'       Function LIB_ReadPCINvram
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               address, Integer
'
'       Output Parameters:
'               oDataRead, Integer
'
'*********************************************************************

  Function LIB_ReadPCINvram( _
        ByVal board As Integer, _
        ByVal errIn As T_Error, _
        ByRef oDataRead As Integer, _
        ByVal address As Integer) _
        As T_Error
```

3.6.2 Write PCI NVRAM (U32 Value).

The Write PCI NVRAM (U32 Value) VI is used to write four bytes in U32 format to the PCI bridge device's configuration NVRAM. Intended for advanced users only.



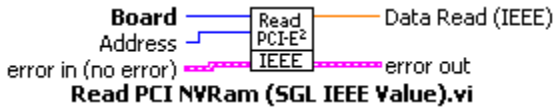
Address = LabVIEW input, byte boundary address of NVRAM.
Data to Write (U32) = LabVIEW input, depending upon the PCI bridge device, either byte or DWord value is written to the NVRAM.

QuBASE Equivalent.

```
'*************************************************************************
'       Function LIB_WritePCINvram
'
'       Description:
'
'       Input Parameters:
'               address, Integer
'               dataToWrite, Integer
'
'       Output Parameters:
'
'*************************************************************************

  Function LIB_WritePCINvram( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef address As Integer, _
        ByRef dataToWrite As Integer) _
        As T_Error
```

3.6.3 Read PCI NVRAM (SGL IEEE Value).

The Read PCI NVRAM (SGL IEEE Value) VI is used to read four bytes in IEEE floating point format that reside inside of the PCI bridge device's configuration NVRAM. Intended for advanced users only.



Address = LabVIEW input, byte boundary address of NVRAM.

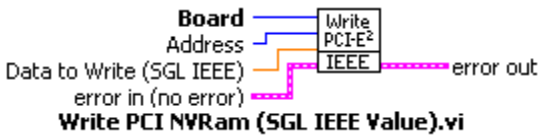
Data Read (IEEE) = LabVIEW output, depending upon the PCI bridge device, either byte or DWord values are read from NVRAM.

QuBASE Equivalent.

```
'*********************************************************************
'       Function LIB_ReadPCINvramTypecastIEEE
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               address, Integer
'
'       Output Parameters:
'               oDataRead, Double
'
'*********************************************************************

  Function LIB_ReadPCINvramTypecastIEEE( _
        ByVal Board As Long, _
        ByVal errIn As T_Error, _
        ByRef oDataRead As Double, _
        ByVal address As Long) As T_Error
```

3.6.4 Write PCI NVRAM (SGL IEEE Value).

The Write PCI NVRAM (SGL IEEE Value) VI is used to write four bytes in IEEE floating point format to the PCI bridge device's configuration NVRAM. Intended for advanced users only.



Address = LabVIEW input, byte boundary address of NVRAM.

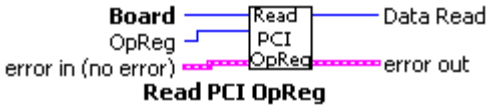
Data to Write (SGL IEEE) = LabVIEW input, depending upon the PCI bridge device, either byte or DWord value is written to the NVRAM.

QuBASE Equivalent.

```
'*********************************************************************
'       Function LIB_WritePCINvramTypecastIEEE
'
'       Description:
'
'       Input Parameters:
'               address, Integer
'               dataToWrite, Double
'
'       Output Parameters:
'
'*********************************************************************

  Function LIB_WritePCINvramTypecastIEEE( _
        ByVal Board As Long, _
        ByVal errIn As T_Error, _
        ByVal address As Long, _
        ByVal dataToWrite As Double) As T_Error
```

3.6.5 Read PCI Opreg.

The Read PCI Opreg VI is used to read data contained in the PCI bridge device's Operation Registers. Intended for advanced users only.



OpReg = LabVIEW input, OpReg address in bytes.

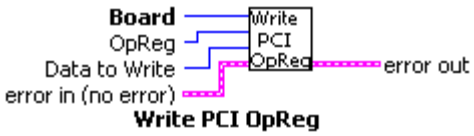
Data Read = LabVIEW output, DWord data read from OpReg.

QuBASE Equivalent.

```
'*************************************************************************
'       Function LIB_ReadPCIOpreg
'
'       Description:
'
'       Input Parameters:
'               address, Integer
'
'       Output Parameters:
'               oDataRead, Integer
'
'*************************************************************************

  Function LIB_ReadPCIOpreg( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef oDataRead As Integer, _
        ByRef address As Integer) _
        As T_Error
```

3.6.6 Write PCI Opreg.

The Write PCI Opreg VI is used to write data to the PCI bridge device's Operation Registers. Intended for advanced users only.



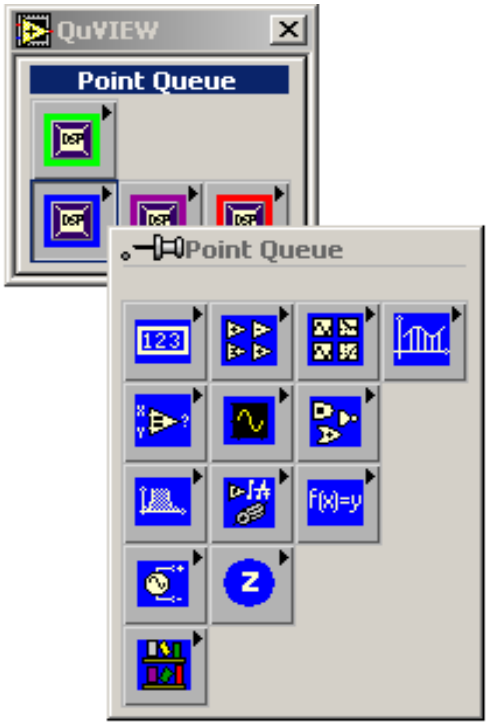
OpReg = LabVIEW input, OpReg address in bytes.

Data to Write = LabVIEW input, DWord data to write to OpReg.

QuBASE Equivalent.

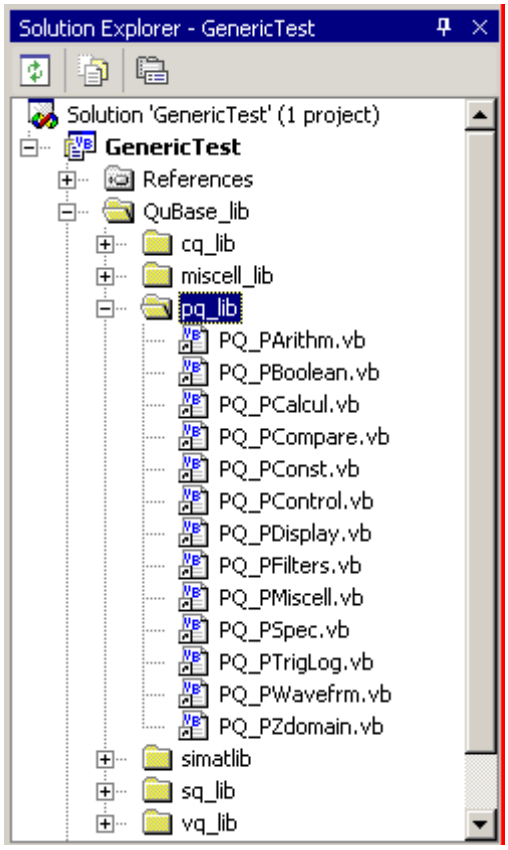
```
'*********************************************************************
'       Function LIB_WritePCIOpreg
'
'       Description:
'
'       Input Parameters:
'               address, Integer
'               dataToWrite, Integer
'
'       Output Parameters:
'
'*********************************************************************

  Function LIB_WritePCIOpreg( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef address As Integer, _
        ByRef dataToWrite As Integer) _
        As T_Error
```

4.0 Point Queuing VIs.



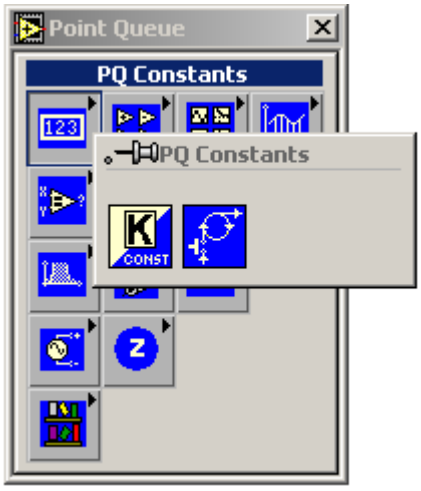
These VIs incorporate DSP functionality when a single data conversion/update cycle occurs for one or more channels of inputs and/or outputs. They are to be used when DSP computation is required on a point by point basis.

QuBASE Equivalent.

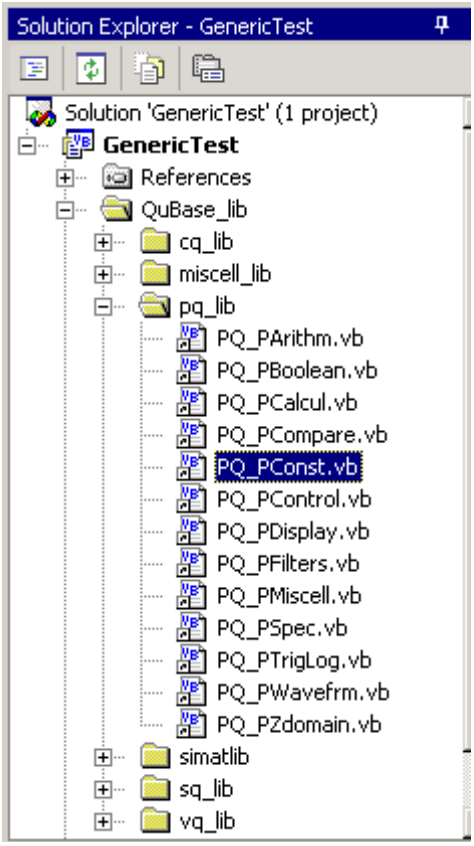


4.1 PQ Constants.

These VIs are necessary because the interconnecting wires within the QuVIEW/DSP domain represent “queueing variables" instead of direct LabVIEW numbers. Therefore, the PQ_Constant is convenient for transferring LabVIEW domain numeric constants and translating them to queueing constants used within the DSP domain.

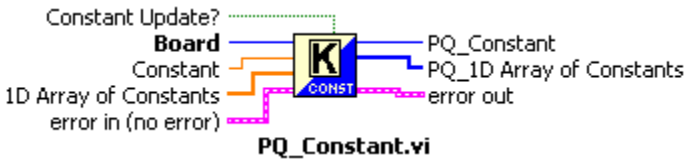


QuBASE Equivalent.



4.1.1 PQ_Constant.

This VI serves to pass single precision floating point constants from the LabVIEW domain to the DSP domain. The values in the LabVIEW domain are passed onto the DSP every time it is executed by LabVIEW, thereby rendering the ability to update constants in real time to the DSP.



Constant = LabVIEW constant input. Not to be used if an array terminal is wired.

1D Array of Constants = LabVIEW input array, useful in condensing wires. Convenient if an array of variables is used as it avoids the wiring of separate icons. If wired, it becomes the default terminal used, and the single variable terminal is ignored.

Constant Update? = LabVIEW boolean input that determines if the LabVIEW variables are to be constantly updated or not. When left in the 'false' position, the LabVIEW constants are only updated once during the queueing phase thereby avoiding DSP-Host communications overhead. When left in the default 'true' position, LabVIEW constants are continuously updated thereby implementing real time variable updates for maximum flexibility.

PQ_Constant = PQ constant output to be used by other "PQ" inputs within the DSP domain. Not to be used if an array terminal is wired.

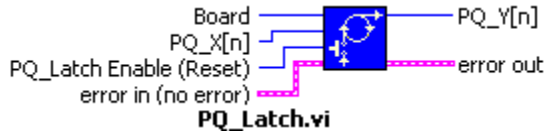
PQ_1D Array of Constants = PQ output array of constants, useful in condensing wires. If an input array terminal was used, then the output array should be correspondingly used.

QuBASE Equivalent.

```
'*********************************************************************
'       Function PQ_Constant
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               constantArrayIn, Double
'
'       Output Parameters:
'               oPQConstant, Integer
'
'*********************************************************************

  Function PQ_Constant( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef constantArrayIn() As Double, _
        ByRef oPQConstant() As Integer) _
        As T_Error
```

4.1.2 PQ_Latch.

The PQ_Latch function allows for a PQ variable to be passed on to its output terminal as long as it is enabled to do so. If disabled, the PQ output variable is latched and remains the same, ignoring the PQ input variable.



PQ_X[n] = PQ input variable to be passed on to the output terminal.

PQ_Latch Enable (Reset) = PQ input that determines if the value at the PQ_Y[n] output is a latched value or a real time value from PQ_X[n] input.

0 = Passive Output: The PQ_Y[n] output will be a direct reflection and follow the input.

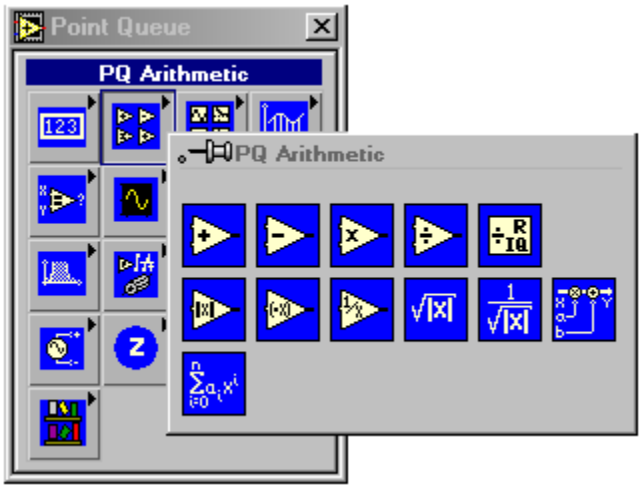
1 = Latch Input: latches the PQ_Y[n] output and ignores the input.

PQ_Y[n] = PQ output.

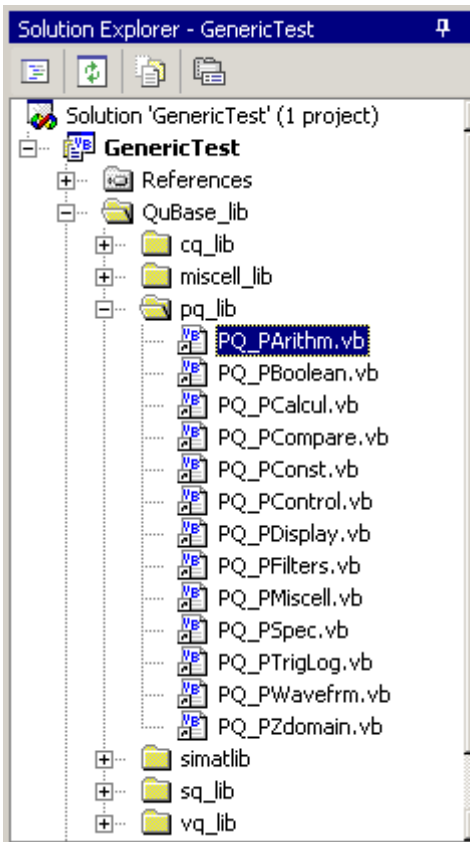
QuBASE Equivalent.

```
'**************************************************************************
'       Function PQ_Latch
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqXn, Integer
'               pqLatchEnable, Integer:
'               0 = Enable, non0 = Disable
'
'       Output Parameters:
'               oPQYn, Integer
'
'**************************************************************************

  Function PQ_Latch( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef oPQYn As Integer = 0, _
        Optional ByRef pqXn As Integer = 0, _
        Optional ByRef pqLatchEnable As Integer = 1) _
        As T_Error
```

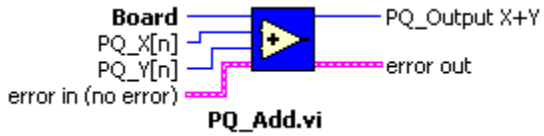
4.2 PQ Arithmetic.



QuBASE Equivalent.



4.2.1 PQ_Add.



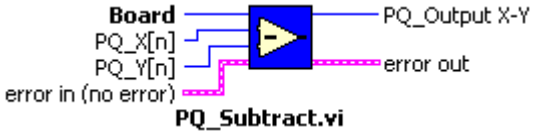
PQ_X[n] = PQ input.
PQ_Y[n] = PQ input.

PQ_Output X+Y = PQ output, sum of X[n] and Y[n].

QuBASE Equivalent.

```
'************************************************************************
'       Function PQ_Add
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqX, Integer
'               pqY, Integer
'
'       Output Parameters:
'               oPQAdd, Integer
'
'************************************************************************

   Function PQ_Add( _
          ByRef board As Integer, _
          ByRef errIn As T_Error, _
          Optional ByRef oPQAdd As Integer = 0, _
          Optional ByRef pqX As Integer = 0, _
          Optional ByRef pqY As Integer = 0) _
          As T_Error
```

4.2.2 PQ_Subtract.



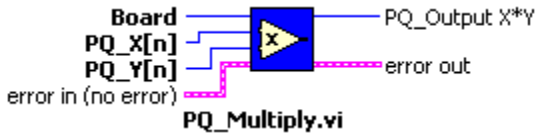
PQ_X[n] = PQ input.
PQ_Y[n] = PQ input.

PQ_Output X-Y = PQ output, difference of X[n] and Y[n].

QuBASE Equivalent.

```
'*********************************************************************
'       Function PQ_Subtract
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqX, Integer
'               pqY, Integer
'
'       Output Parameters:
'               oPQSub, Integer
'
'*********************************************************************

  Function PQ_Subtract( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef oPQSub As Integer = 0, _
        Optional ByRef pqX As Integer = 0, _
        Optional ByRef pqY As Integer = 0) _
        As T_Error
```

4.2.3 PQ_Multiply.



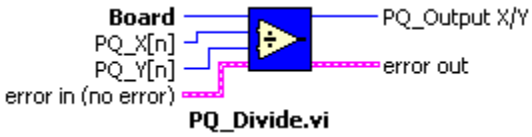
PQ_X[n] = PQ input.
PQ_Y[n] = PQ input.

PQ_Output X*Y = PQ output, product of X[n] and Y[n].

QuBASE Equivalent.

```
'*********************************************************************
'       Function PQ_Multiply
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqX, Integer
'               pqY, Integer
'
'       Output Parameters:
'               oPQProduct, Integer
'
'*********************************************************************

  Function PQ_Multiply( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef oPQProduct As Integer = 0, _
        Optional ByRef pqX As Integer = 0, _
        Optional ByRef pqY As Integer = 0) _
        As T_Error
```

4.2.4 PQ_Divide.



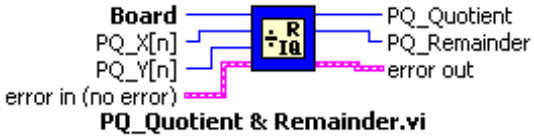
PQ_X[n] = PQ input.
PQ_Y[n] = PQ input.

PQ_Output X/Y = PQ output, ratio of X[n] over Y[n].

QuBASE Equivalent.

```
'*************************************************************************
'       Function PQ_Divide
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqX, Integer
'               pqY, Integer
'
'       Output Parameters:
'               oPQDivide, Integer
'
'*************************************************************************

   Function PQ_Divide( _
          ByRef board As Integer, _
          ByRef errIn As T_Error, _
          Optional ByRef oPQDivide As Integer = 0, _
          Optional ByRef pqX As Integer = 0, _
          Optional ByRef pqY As Integer = 0) _
          As T_Error
```

4.2.5 PQ_Quotient & Remainder.



PQ_X[n] = PQ input.
PQ_Y[n] = PQ input.

PQ_Quotient = PQ output, quotient of X[n] divided by Y[n].
PQ_Remainder = PQ output, remainder of X[n] divided by Y[n].

QuBASE Equivalent.

```
'**********************************************************************
'       Function PQ_QuotientRemainder
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqX, Integer
'               pqY, Integer
'
'       Output Parameters:
'               oPQQuotient, Integer
'               oPQRemainder, Integer
'
'**********************************************************************

  Function PQ_QuotientRemainder( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef pqX As Integer = 0, _
        Optional ByRef pqY As Integer = 0) _
        Optional ByRef oPQQuotient As Integer = 0, _
        Optional ByRef oPQRemainder As Integer = 0, _
        As T_Error
```

4.2.6 PQ_Absolute Value.

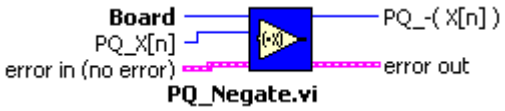
PQ_X[n] = PQ input.

PQ_|X[n]| = PQ output, absolute value of X[n].

QuBASE Equivalent.

```
'*************************************************************************
'       Function PQ_AbsoluteValue
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
                errIn, T_Error
                pqX, Integer
'
'       Output Parameters:
                oPQAbs, Integer
'
'*************************************************************************

   Function PQ_AbsoluteValue( _
          ByRef board As Integer, _
          ByRef errIn As T_Error, _
          Optional ByRef oPQAbs As Integer = 0, _
          Optional ByRef pqX As Integer = 0) _
          As T_Error
```

4.2.7 PQ_Negate.



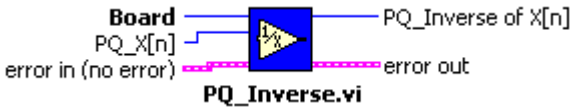
PQ_X[n] = PQ input.

PQ_-(X[n]) = PQ output, negative of X[n].

QuBASE Equivalent.

```
'*********************************************************************
'       Function PQ_Negate
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqX, Integer
'
'       Output Parameters:
'               oPQNegate, Integer
'
'*********************************************************************

   Function PQ_Negate( _
          ByRef board As Integer, _
          ByRef errIn As T_Error, _
          Optional ByRef oPQNegate As Integer = 0, _
          Optional ByRef pqX As Integer = 0) _
          As T_Error
```

4.2.8 PQ_Inverse.



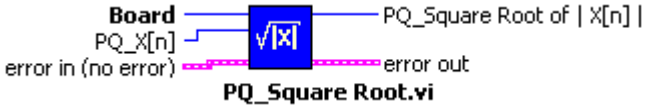
PQ_X[n] = PQ input.

PQ_Inverse of X[n] = PQ output, inverse of X[n].

QuBASE Equivalent.

```
'*************************************************************************
'       Function PQ_Inverse
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqX, Integer
'
'       Output Parameters:
'               oPQInverse, Integer
'
'*************************************************************************

   Function PQ_Inverse( _
          ByRef board As Integer, _
          ByRef errIn As T_Error, _
          Optional ByRef oPQInverse As Integer = 0, _
          Optional ByRef pqX As Integer = 0) _
          As T_Error
```

4.2.9 PQ_Square Root.



PQ_X[n] = PQ input.

PQ_Square Root of |X[n]| = PQ output, square root of absolute value of X[n].

QuBASE Equivalent.

```
'*************************************************************************
'       Function PQ_SquareRoot
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqX, Integer
'
'       Output Parameters:
'               oPQSquareRoot, Integer
'
'*************************************************************************

   Function PQ_SquareRoot( _
            ByRef board As Integer, _
            ByRef errIn As T_Error, _
            Optional ByRef oPQSquareRoot As Integer = 0, _
            Optional ByRef pqX As Integer = 0) _
            As T_Error
```

4.2.10 PQ_Inverse Square Root.

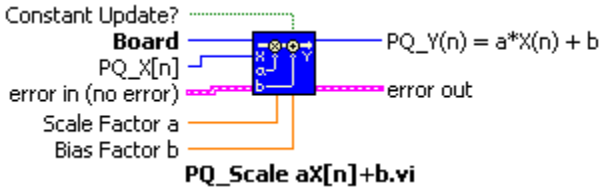
PQ_X[n] = PQ input.

PQ_Inverse Square Root of |X[n]| = PQ output, inverse square root of absolute value of X[n].

QuBASE Equivalent.

```
'**************************************************************************
'       Function PQ_InverseSquareRoot
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqX, Integer
'
'       Output Parameters:
'               oPQInvSqRoot, Integer
'
'**************************************************************************

   Function PQ_InverseSquareRoot( _
            ByRef board As Integer, _
            ByRef errIn As T_Error, _
            Optional ByRef oPQInvSqRoot As Integer = 0, _
            Optional ByRef pqX As Integer = 0) _
            As T_Error
```

4.2.11 PQ_Scale aX+b.



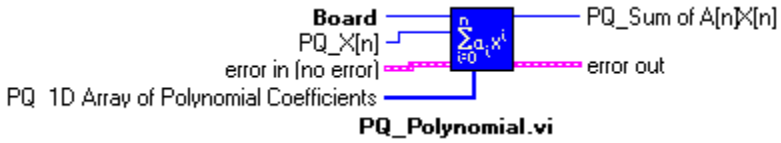
PQ_X[n] = PQ input.
Scale Factor a = LabVIEW constant that multiplies X[n].
Bias Factor b = LabVIEW constant that is added to the product of a*X[n].
Constant Update? = LabVIEW boolean input that determines if the LabVIEW variables are to be constantly updated or not. When left in the 'false' position, the LabVIEW constants are only updated once during the queueing phase thereby avoiding DSP-Host communications overhead. When left in the default 'true' position, LabVIEW constants are continuously updated thereby implementing real time variable updates for maximum flexibility.

PQ_Scale a*X[n]+b = PQ output, single step procedure for scaling and biasing X[n].

QuBASE Equivalent.

```
'**************************************************************************
'       Function PQ_ScaleaXPlusB
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqX, Integer
'               factorA, Double
'               factorB, Double
'
'       Output Parameters:
'               oPQScale, Integer
'
'**************************************************************************

  Function PQ_ScaleaXPlusB( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef oPQScale As Integer = 0, _
        Optional ByRef pqX As Integer = 0, _
        Optional ByRef factorA As Double = 1, _
        Optional ByRef factorB As Double = 0) _
        As T_Error
```

4.2.12 PQ_Polynomial.



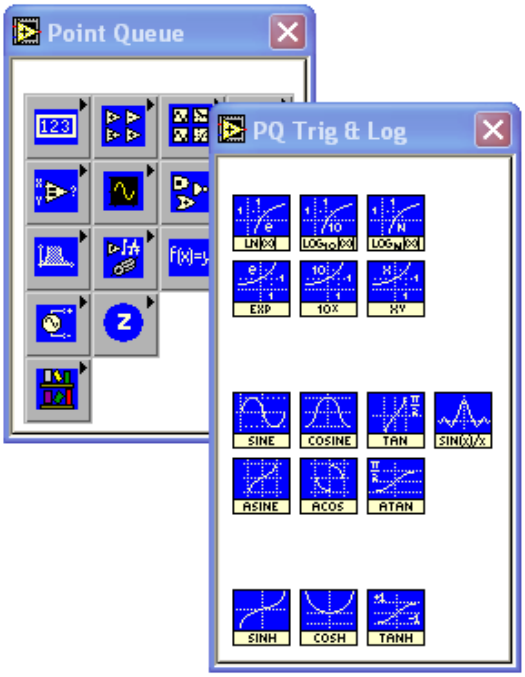
PQ_X[n] = PQ input.
PQ_1D Array of Polynomial Coefficients = PQ 1D Array input, where the nth position of the coefficient in the array corresponds to the nth power term.

PQ_Sum of A[n]X[n] = PQ output, the result of the summation of all the polynomial terms.

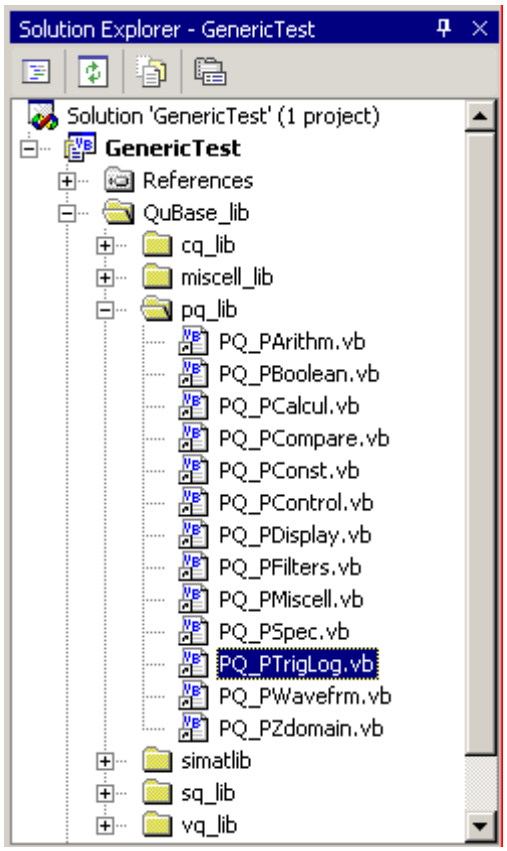
QuBASE Equivalent.

```
'*************************************************************************
'       Function PQ_Polynomial
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqX, Integer
'               ArrayPolynCoeff, Integer
'
'       Output Parameters:
'               oPQSumAnXn, Integer
'
'*************************************************************************
```

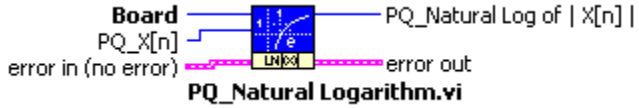
4.3 PQ Trig and Log.



QuBASE Equivalent.



4.3.1 PQ_Natural Logarithm.



PQ_X[n] = PQ input.

PQ_Natural Log of |X[n]| = PQ output, computes natural logarithm base "e" with argument as the absolute value of X[n].

QuBASE Equivalent.

```
'**************************************************************************
'       Function PQ_NaturalLog
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqXn, Integer
'
'       Output Parameters:
'               oPQLn, Integer
'
'**************************************************************************

  Function PQ_NaturalLog( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef oPQLn As Integer = 0, _
        Optional ByRef pqX As Integer = 0) _
        As T_Error
```

4.3.2 PQ_Logarithm Base 10.

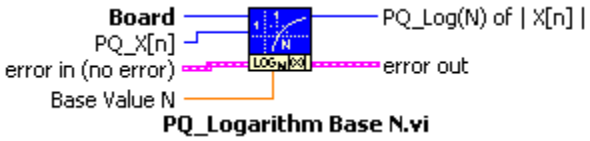
PQ_X[n] = PQ input.

PQ_Log(10) of |X[n]| = PQ output, computes the logarithm base "10" with argument as the absolute value of X[n].

QuBASE Equivalent.

```
'**********************************************************************
'       Function PQ_LogarithmBase10
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqXn, Integer
'
'       Output Parameters:
'               oPQLogBase10, Integer
'
'**********************************************************************

Function PQ_LogarithmBase10( _
        ByVal board As Integer, _
        ByVal errIn As T_Error, _
        Optional ByRef oPQLogBase10 As Integer = 0, _
        Optional ByVal pqX As Integer = 0) As T_Error
```

4.3.3 PQ_Logarithm Base N.



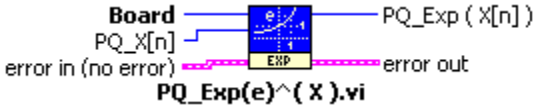
PQ_X[n] = PQ input.
Base Value N = LabVIEW input, the base value for which to compute the Logarithm.

PQ_Log(N) of |X[n]| = PQ output, computes the logarithm base "N" with argument as the absolute value of X[n].

QuBASE Equivalent.

```
'*************************************************************************
'       Function PQ_LogarithmBaseN
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqXn, Integer
'               baseN, Double
'
'       Output Parameters:
'               oPQLogBaseN, Integer
'
'*************************************************************************

Function PQ_LogarithmBaseN( _
        ByVal board As Integer, _
        ByVal errIn As T_Error, _
        Optional ByRef oPQLogBaseN As Integer = 0, _
        Optional ByVal pqX As Integer = 0, _
        Optional ByVal baseN As Double = 10) As T_Error
```

4.3.4 PQ_Exp(e)^(X).



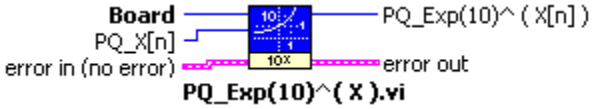
PQ_X[n] = PQ input.

PQ_Exp(X[n]) = PQ output, computes exponential base "e" with power of X[n].

QuBASE Equivalent.

```
'*************************************************************************
'       Function PQ_ExponentofX
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqXn, Integer
'
'       Output Parameters:
'               oPQExp, Integer
'
'*************************************************************************

  Function PQ_ExponentofX( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef oPQExp As Integer = 0, _
        Optional ByRef pqX As Integer = 0) _
        As T_Error
```

4.3.5 PQ_Exp(10)^(X).



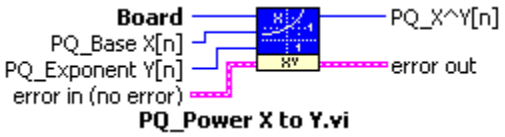
PQ_X[n] = PQ input.

PQ_Exp(10)(X[n]) = PQ output, computes exponential base "10" with power of X[n].

QuBASE Equivalent.

```
'************************************************************************
'       Function PQ_LogarithmBase10
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqXn, Integer
'
'       Output Parameters:
'               oPQLogBase10, Integer
'
'************************************************************************

Function PQ_LogarithmBase10( _
        ByVal board As Integer, _
        ByVal errIn As T_Error, _
        Optional ByRef oPQLogBase10 As Integer = 0, _
        Optional ByVal pqX As Integer = 0) As T_Error
```

4.3.6 PQ_Power X to Y.



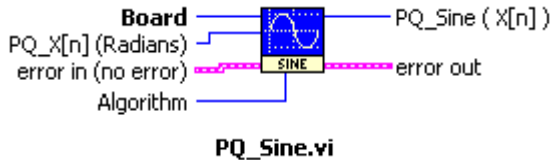
PQ_Base X[n] = PQ input, base.
PQ_Exponent Y[n] = PQ input, exponent.

PQ_X^Y[n] = PQ output, computes the exponent of base "X[n]" to the power of Y[n].

QuBASE Equivalent.

```
'************************************************************************
'       Function PQ_PowerXtoY
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqBaseX, Integer
'               exponentN, Integer
'
'       Output Parameters:
'               oPQ_PowerXtoY, Integer
'
'************************************************************************

Function PQ_PowerXtoY( _
        ByVal board As Integer, _
        ByVal errIn As T_Error, _
        Optional ByRef oPQ_PowerXtoY As Integer = 0, _
        Optional ByVal pqBaseX As Integer = 0, _
        Optional ByVal exponentN As Integer = 10) As T_Error
```

4.3.7 PQ_Sine.



PQ_X[n] (Radians) = PQ input, represented as radians.
Algorithm = LabVIEW input, selects the method for results as follows:

1) *Generate* = values are computed by the DSP using an accurate but relatively time consuming iterative process.

2) *Table* = values are sourced from a look up table for higher speed and interpolated for acceptable accuracy.

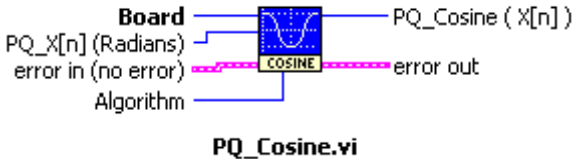
Table Resolution = LabVIEW input that determines the size of the look up table; defaulted to 1024.

PQ_Sine(X[n]) = PQ output, computes sine of radians of X[n].

QuBASE Equivalent.

```
'**************************************************************************
'       Function PQ_Sine
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqXn, Integer
'
'       Output Parameters:
'               oPQSine, Integer
'
'**************************************************************************

   Function PQ_Sine( _
          ByRef board As Integer, _
          ByRef errIn As T_Error, _
          Optional ByRef oPQSine As Integer = 0, _
          Optional ByRef pqX As Integer = 0) _
          As T_Error
```

4.3.8 PQ_Cosine.



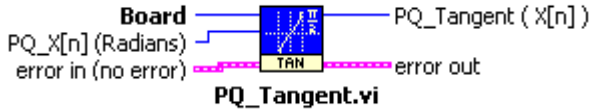
PQ_X[n] (Radians) = PQ input, represented as radians.
Algorithm = LabVIEW input, selects the method for results as follows:
1) *Generate* = values are computed by the DSP using an accurate but relatively time consuming iterative process.
2) *Table* = values are sourced from a look up table for higher speed and interpolated for acceptable accuracy.
Table Resolution = LabVIEW input that determines the size of the look up table; defaulted to 1024.

PQ_Cosine(X[n]) = PQ output, computes cosine of radians of X[n].

QuBASE Equivalent.

```
'**********************************************************************
'       Function PQ_Cosine
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqXn, Integer
'
'       Output Parameters:
'               oPQCosine, Integer
'
'**********************************************************************

  Function PQ_Cosine( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef oPQCosine As Integer = 0, _
        Optional ByRef pqX As Integer = 0) _
        As T_Error
```

4.3.9 PQ_Tangent.



PQ_X[n] (Radians) = PQ input, represented as radians.

PQ_Tangent(X[n]) = PQ output, computes tangent of radians of X[n].

QuBASE Equivalent.

```
'************************************************************************
'       Function PQ_Tangent
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqXn, Integer
'
'       Output Parameters:
'               oPQTangent, Integer
'
'************************************************************************

   Function PQ_Tangent( _
          ByRef board As Integer, _
          ByRef errIn As T_Error, _
          Optional ByRef oPQTangent As Integer = 0, _
          Optional ByRef pqX As Integer = 0) _
          As T_Error
```

4.3.10 PQ_Sinc.

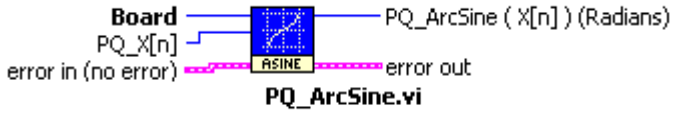
PQ_X[n] (Radians) = PQ input, represented as radians.

PQ_Sine(X[n])/X[n] = PQ output, computes the Sinc of radians of X[n].

QuBASE Equivalent.

```
'**************************************************************************
'       Function PQ_Sinc
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqXn, Integer
'
'       Output Parameters:
'               oPQSinc, Integer
'
'**************************************************************************

   Function PQ_Sinc( _
          ByRef board As Integer, _
          ByRef errIn As T_Error, _
          Optional ByRef oPQSinc As Integer = 0, _
          Optional ByRef pqX As Integer = 0) _
          As T_Error
```

4.3.11 PQ_ArcSine.



PQ_X[n] = PQ input, represented as any floating point value where -1 =< X[n] =<1. Values that fall out of this range are replaced with a '-1' for negative values beyond '-1', and similarly replaced with a '+1' for positive values beyond '+1'.

PQ_ArcSine(X[n]) = PQ output in radians, computes the Inverse Sine of X[n].

QuBASE Equivalent.

```
'**********************************************************************
'       Function PQ_ArcSine
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqXn, Integer
'
'       Output Parameters:
'               oPQArcSine, Integer
'
'**********************************************************************

  Function PQ_ArcSine( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef oPQArcSine As Integer = 0, _
        Optional ByRef pqX As Integer = 0) _
        As T_Error
```

4.3.12 PQ_ArcCosine.

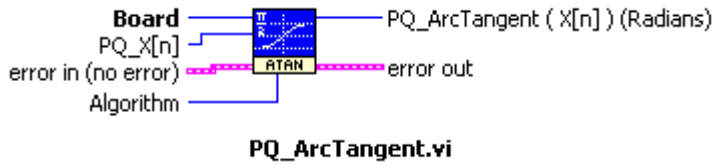
PQ_X[n] = PQ input, represented as any floating point value where -1 =< X[n] =<1. Values that fall out of this range are replaced with a '-1' for negative values beyond '-1', and similarly replaced with a '+1' for positive values beyond '+1'.

PQ_ArcSine(X[n]) = PQ output in radians, computes the Inverse Cosine of X[n].

QuBASE Equivalent.

```
'**************************************************************************
'       Function PQ_ArcCosine
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqXn, Integer
'
'       Output Parameters:
'               oPQArcCosine, Integer
'
'**************************************************************************

  Function PQ_ArcCosine( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef oPQArcCosine As Integer = 0, _
        Optional ByRef pqX As Integer = 0) _
        As T_Error
```

4.3.13 PQ_ArcTangent.



PQ_X[n] = PQ input, represented as any floating point value.
Algorithm = LabVIEW input, selects the method for results as follows:
1) *Generate* = values are computed by the DSP using an accurate but relatively time consuming iterative process.
2) *Table* = values are sourced from a look up table for higher speed and interpolated for acceptable accuracy.
Table Resolution = LabVIEW input that determines the size of the look up table; defaulted to 1024.

PQ_ArcTangent(X[n]) = PQ output in radians, computes the Inverse Tangent of X[n].

QuBASE Equivalent.

```
'**********************************************************************
'       Function PQ_ArcTangent
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqXn, Integer
'
'       Output Parameters:
'               oPQArcTangent, Integer
'
'**********************************************************************

  Function PQ_ArcTangent( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef oPQArcTangent As Integer = 0, _
        Optional ByRef pqX As Integer = 0) _
        As T_Error
```

4.3.14 PQ_Hyperbolic Sine.

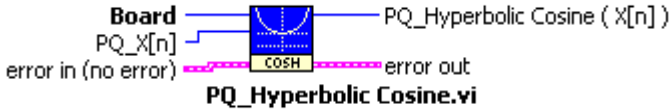
PQ_X[n] = PQ input, represented as any floating point value.

PQ_Hyperbolic Sine(X[n]) = PQ output, computes the Hyperbolic Sine of X[n].

QuBASE Equivalent.

```
'**************************************************************************
'       Function PQ_HyperbolicSine
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqXn, Integer
'
'       Output Parameters:
'               oPQHyperbolicSine, Integer
'
'**************************************************************************

  Function PQ_HyperbolicSine( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef oPQHyperbolicSine As Integer = 0, _
        Optional ByRef pqX As Integer = 0) _
        As T_Error
```

4.3.15 PQ_Hyperbolic Cosine.



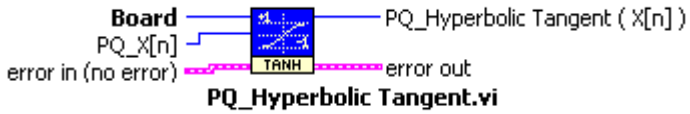
PQ_X[n] = PQ input, represented as any floating point value.

PQ_Hyperbolic Cosine(X[n]) = PQ output, computes the Hyperbolic Cosine of X[n].

QuBASE Equivalent.

```
'**************************************************************************
'       Function PQ_HyperbolicCosine
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqXn, Integer
'
'       Output Parameters:
'               oPQHyperbolicCosine, Integer
'
'**************************************************************************

   Function PQ_HyperbolicCosine( _
          ByRef board As Integer, _
          ByRef errIn As T_Error, _
          Optional ByRef oPQHyperbolicCosine As Integer = 0, _
          Optional ByRef pqX As Integer = 0) _
          As T_Error
```

4.3.16 PQ_Hyperbolic Tangent.



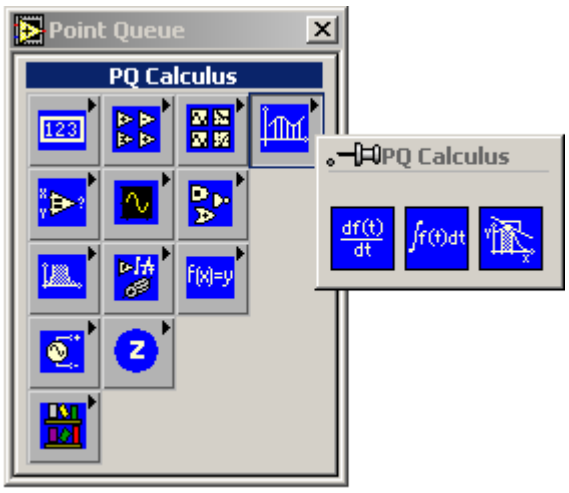
PQ_X[n] = PQ input, represented as any floating point value.

PQ_Hyperbolic Tangent(X[n]) = PQ output, computes the Hyperbolic Tangent of X[n].

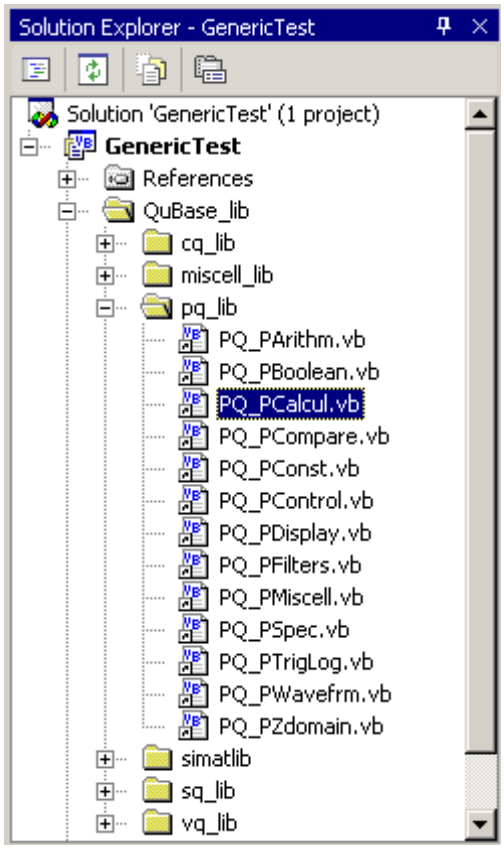
QuBASE Equivalent.

```
'**************************************************************************
'       Function PQ_HyperbolicTangent
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqXn, Integer
'
'       Output Parameters:
'               oPQHyperbolicTangent, Integer
'
'**************************************************************************

  Function PQ_HyperbolicTangent( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef oPQHyperbolicTangent As Integer = 0, _
        Optional ByRef pqX As Integer = 0) _
        As T_Error
```

4.4 PQ Calculus.



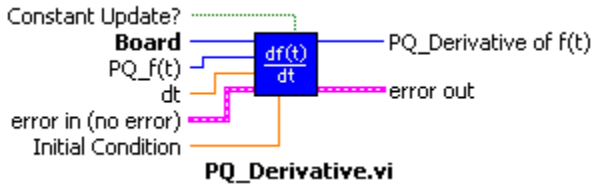
QuBASE Equivalent.



4.4.1 PQ_Derivative.

The PQ_Derivative VI performs a discrete differentiation of the sampled signal represented by a PQ Input. The differentiation of PQ_X[n], d/dt X[n], is computed using the following equation:

d/dtX[n][= (1/dt) * (X[n] – X[n-1]).



PQ_f(t) = PQ input.

dt = LabVIEW input that determines width at which to divide the current and previous values of f(t).

Initial Condition = LabVIEW floating point constant, used to fill in an absent point when computing the first slope value.

Constant Update? = LabVIEW boolean input that determines if the LabVIEW variables are to be constantly updated or not. When left in the 'false' position, the LabVIEW constants are only updated once during the queueing phase thereby avoiding DSP-Host communications overhead. When left in the default 'true' position, LabVIEW constants are continuously updated thereby implementing real time variable updates for maximum flexibility.

PQ_Derivative of f(t) = PQ output that returns a continuous differentiating output.

QuBASE Equivalent.

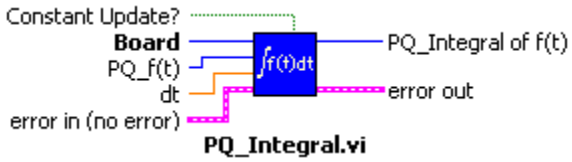
```
'**********************************************************************
'       Function PQ_Derivative
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqFt, Integer
'               dt, Double
'               initialCond, Double
'
'       Output Parameters:
'               oPQDerivative, Integer
'
'**********************************************************************

   Function PQ_Derivative( _
           ByRef board As Integer, _
           ByRef errIn As T_Error, _
           Optional ByRef oPQDerivative As Integer = 0, _
           Optional ByRef pqFt As Integer = 0, _
           Optional ByRef dt As Double = 1, _
           Optional ByRef initialCond As Double = 0) _
           As T_Error
```

4.4.2 PQ_Integral.

The PQ_Integral VI performs the discrete integration of the sampled signal represented by a PQ Input. The integration is obtained using the following equation:

Y[n] = Y[n-1] + (X[n] * dt).

For 0=<n<=k-1.



PQ_f(t) = PQ input.

dt = LabVIEW input, determines the width or discrete time base between adjacent points used to multiply the current and previous values of PQ_f(t).

Constant Update? = LabVIEW boolean input that determines if the LabVIEW variables are to be constantly updated or not. When left in the 'false' position, the LabVIEW constants are only updated once during the queueing phase thereby avoiding DSP-Host communications overhead. When left in the default 'true' position, LabVIEW constants are continuously updated thereby implementing real time variable updates for maximum flexibility.

PQ_Integral of f(t) = PQ output that returns a continuous additive output.

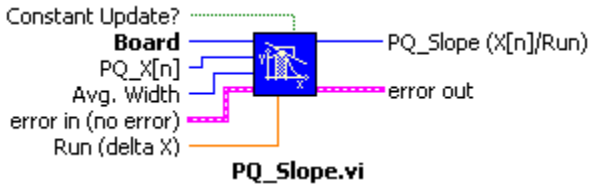
NOTE: It is possible to observe a wrap around effect (also known as wind up) when using the Integration function. This is not an error in the DSP code, but rather the compounded effect of the input signal's offset added back onto itself. The Calibrate function within the miscellaneous operations subset may be used to digitally correct this initial input signal offset.

QuBASE Equivalent.

```
'*************************************************************************
'       Function PQ_Integral
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqFt, Integer
'               dt, Double
'
'       Output Parameters:
'               oPQIntegral, Integer
'
'*************************************************************************

   Function PQ_Integral( _
           ByRef board As Integer, _
           ByRef errIn As T_Error, _
           Optional ByRef oPQIntegral As Integer = 0, _
           Optional ByRef pqFt As Integer = 0, _
           Optional ByRef dt As Double = 0) _
           As T_Error
```

4.4.3 PQ_Slope.

The PQ_Slope VI computes the slope of a sequence of incoming PQ inputs.



PQ_X[n] = PQ input.

Avg. Width = LabVIEW input that defines the number of points to be used to compute the slope.

Run (delta X) = LabVIEW input that defines each abscissa increment or the difference in the X axis between subsequent points.

Constant Update? = LabVIEW boolean input that determines if the LabVIEW variables are to be constantly updated or not. When left in the 'false' position, the LabVIEW constants are only updated once during the queueing phase thereby avoiding DSP-Host communications overhead. When left in the default 'true' position, LabVIEW constants are continuously updated thereby implementing real time variable updates for maximum flexibility.

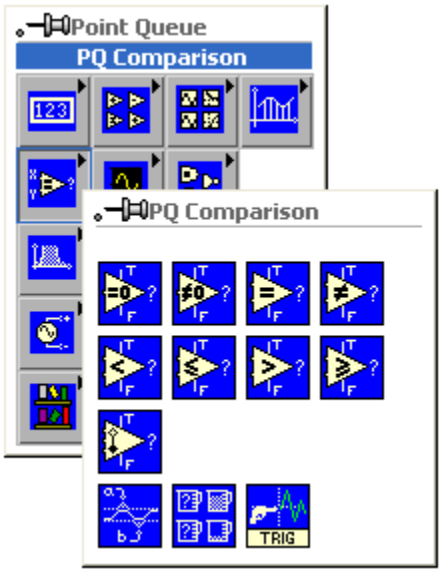
PQ_Slope (X[n]/Run) = PQ Output, slope obtained by dividing X[n]/Run.

QuBASE Equivalent.

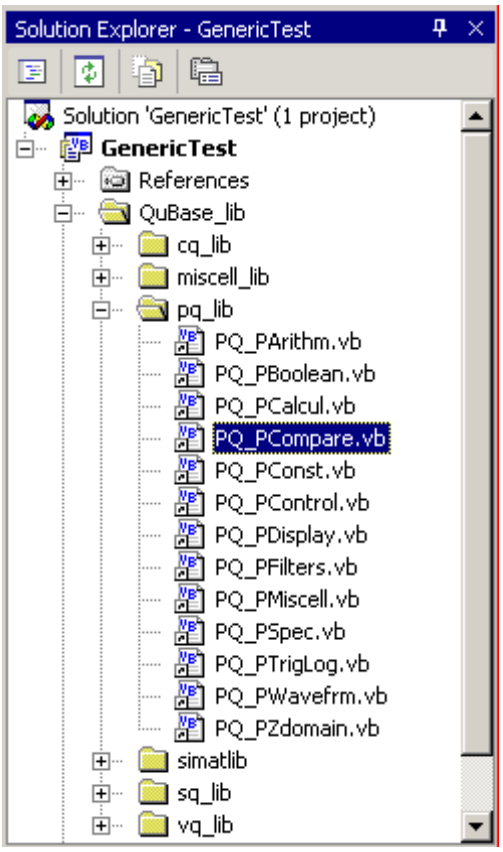
```
'*************************************************************************
'       Function PQ_Slope
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqX, Integer
'               run, Double
'               avgWidth, Double
'
'       Output Parameters:
'               oPQSlope, Integer
'
'*************************************************************************

  Function PQ_Slope( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef oPQSlope As Integer = 0, _
        Optional ByRef pqX As Integer = 0, _
        Optional ByRef run As Double = 0, _
        Optional ByRef avgWidth As Double = 0) _
        As T_Error
```

4.5 PQ Comparison.

The 'Q_Equal?', 'Q_Not equal?', 'Q_Less Than', 'Q_Less Than or Equal?', 'Q_Greater Than?', 'Q_Greater Than or Equal?' functions take in a pair of queueing inputs that will be subjected to the comparison, along with a pair of queueing inputs that define the boolean TRUE/FALSE values to be returned as the result. In other words, the output returns the value wired to the "TRUE" input if the case is a true, or the output returns the value wired to the "FALSE" input if the case is false. Each TRUE or FALSE state is defined as a pair of queuing inputs; i.e. a TRUE state can be defined as one value and a FALSE state as another value. Either state definition can be derived from any queuing function.

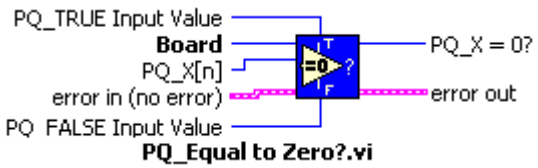


QuBASE Equivalent.



4.5.1 PQ_Equal to Zero?

Compares the PQ input X[n] to PQ "zero" value and returns a "PQ True" if equal, and a "PQ False" if not.



PQ_X[n] = PQ Input value to be compared to a PQ zero.

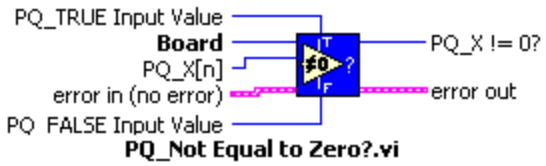
PQ_X = 0? = PQ output, PQ TRUE/FALSE result of the comparison.

QuBASE Equivalent.

```
'*********************************************************************
'       Function PQ_EqualToZero
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqX, Integer
'               pqY, Integer
'               pqTrueInputValue, Integer
'               pqFalseInputValue, Integer
'
'       Ouput Parameters:
'               oPQEqualtoZero, Integer
'
'*********************************************************************

  Function PQ_EqualtoZero( _
      ByRef board As Integer, _
      ByRef errIn As T_Error, _
      Optional ByRef oPQEqualtoZero As Integer = 0, _
      Optional ByRef pqX As Integer = 0, _
      Optional ByRef pqTrueInputValue As Integer = 0, _
      Optional ByRef pqFalseInputValue As Integer = 0) _
      As T_Error
```

4.5.2 PQ_Not Equal to Zero?

Compares the PQ input X[n] to a PQ "zero" and returns a "PQ False" if equal, and a "PQ True" if NOT equal.



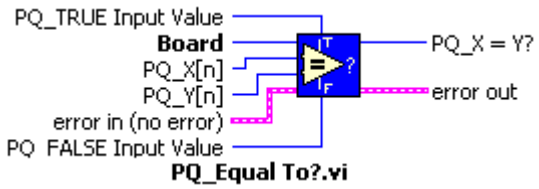
PQ_X[n] = PQ Input value to be compared to a PQ non-zero value.

PQ_X != 0? = PQ output, PQ TRUE/FALSE result of the comparison.

QuBASE Equivalent.

```
'*************************************************************************
'       Function PQ_NotEqualTo
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqX, Integer
'               pqY, Integer
'               pqTrueInputValue, Integer
'               pqFalseInputValue, Integer
'
'       Ouput Parameters:
'               oPQNotEqualTo, Integer
'
'*************************************************************************

  Function PQ_NotEqualTo( _
      ByRef board As Integer, _
      ByRef errIn As T_Error, _
      Optional ByRef oPQNotEqualTo As Integer = 0, _
      Optional ByRef pqX As Integer = 0, _
      Optional ByRef pqY As Integer = 0, _
      Optional ByRef pqTrueInputValue As Integer = 0, _
      Optional ByRef pqFalseInputValue As Integer = 0) _
      As T_Error
```

4.5.3 PQ_Equal?



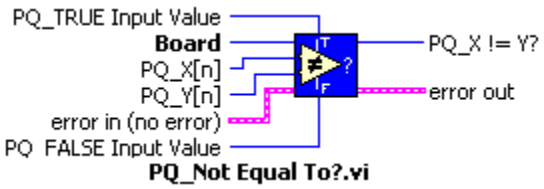
PQ_X[n] = First PQ input to compare.
PQ_Y[n] = Second PQ input to compare.

PQ_X=Y? = PQ output that returns the value wired to the PQ_TRUE if X=Y, or the value wired to the PQ_FALSE if X!=Y.

QuBASE Equivalent.

```
'*************************************************************************
'       Function PQ_EqualTo
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqX, Integer
'               pqY, Integer
'               pqTrueInputValue, Integer
'               pqFalseInputValue, Integer
'
'       Ouput Parameters:
'               oPQEqualTo, Integer
'
'*************************************************************************

  Function PQ_EqualTo( _
      ByRef board As Integer, _
      ByRef errIn As T_Error, _
      Optional ByRef oPQEqualTo As Integer = 0, _
      Optional ByRef pqX As Integer = 0, _
      Optional ByRef pqY As Integer = 0, _
      Optional ByRef pqTrueInputValue As Integer = 0, _
      Optional ByRef pqFalseInputValue As Integer = 0) _
      As T_Error
```

4.5.4 PQ_Not Equal?



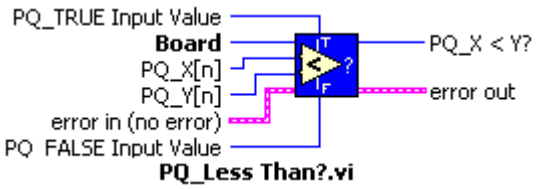
PQ_X[n] = First PQ input to compare.
PQ_Y[n] = Second PQ input to compare.

PQ_X != Y? = PQ output that returns the value wired to the PQ_TRUE if X!=Y, or the value wired to the PQ_FALSE if X=Y.

QuBASE Equivalent.

```
'*********************************************************************
'       Function PQ_NotEqualTo
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqX, Integer
'               pqY, Integer
'               pqTrueInputValue, Integer
'               pqFalseInputValue, Integer
'
'       Ouput Parameters:
'               oPQNotEqualTo, Integer
'
'*********************************************************************

  Function PQ_NotEqualTo( _
      ByRef board As Integer, _
      ByRef errIn As T_Error, _
      Optional ByRef oPQNotEqualTo As Integer = 0, _
      Optional ByRef pqX As Integer = 0, _
      Optional ByRef pqY As Integer = 0, _
      Optional ByRef pqTrueInputValue As Integer = 0, _
      Optional ByRef pqFalseInputValue As Integer = 0) _
      As T_Error
```

4.5.5 PQ_Less Than?



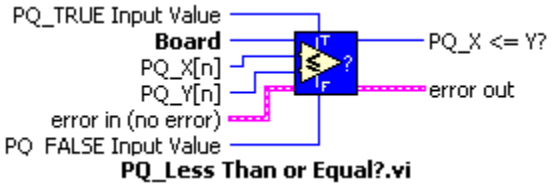
PQ_X[n] = First PQ input to compare.
PQ_Y[n] = Second PQ input to compare.

PQ_X<Y? = PQ output that returns the value wired to the PQ_TRUE if X<Y, or the value wired to the PQ_FALSE if X >= Y.

QuBASE Equivalent.

```
'*************************************************************************
'       Function PQ_LessThan
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqX, Integer
'               pqY, Integer
'               pqTrueInputValue, Integer
'               pqFalseInputValue, Integer
'
'       Ouput Parameters:
'               oPQLessThan, Integer
'
'*************************************************************************

  Function PQ_LessThan( _
      ByRef board As Integer, _
      ByRef errIn As T_Error, _
      Optional ByRef oPQLessThan As Integer = 0, _
      Optional ByRef pqX As Integer = 0, _
      Optional ByRef pqY As Integer = 0, _
      Optional ByRef pqTrueInputValue As Integer = 0, _
      Optional ByRef pqFalseInputValue As Integer = 0) _
      As T_Error
```

4.5.6 PQ_Less Than or Equal?



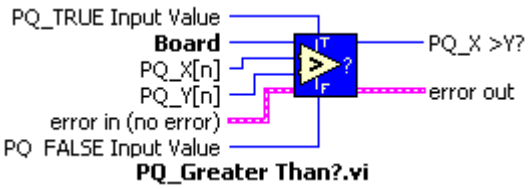
PQ_X[n] = First PQ input to compare.
PQ_Y[n] = Second PQ input to compare.

PQ_X<=Y? = PQ output that returns the value wired to the PQ_TRUE if X<=Y, or the value wired to the PQ_FALSE if X>Y.

QuBASE Equivalent.

```
'**************************************************************************
'       Function PQ_LessThanOrEqual
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqX, Integer
'               pqY, Integer
'               pqLo, Integer
'               pqTrueInputValue, Integer
'               pqFalseInputValue, Integer
'
'       Ouput Parameters:
'               oPQLessThanOrEqual, Integer
'
'**************************************************************************

  Function PQ_LessThanOrEqual( _
      ByRef board As Integer, _
      ByRef errIn As T_Error, _
      Optional ByRef oPQLessThanOrEqual As Integer = 0, _
      Optional ByRef pqX As Integer = 0, _
      Optional ByRef pqY As Integer = 0, _
      Optional ByRef pqTrueInputValue As Integer = 0, _
      Optional ByRef pqFalseInputValue As Integer = 0) _
      As T_Error
```

4.5.7 PQ_Greater Than?



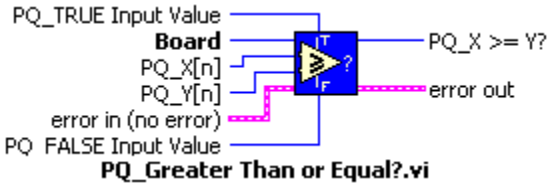
PQ_X[n] = First PQ input to compare.
PQ_Y[n] = Second PQ input to compare.

PQ_X>Y? = PQ output that returns the value wired to the PQ_TRUE if X>Y, or the value wired to the PQ_FALSE if X<=Y.

QuBASE Equivalent.

```
'*************************************************************************
'       Function PQ_GreaterThan
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqX, Integer
'               pqY, Integer
'               pqTrueInputValue, Integer
'               pqFalseInputValue, Integer
'
'       Ouput Parameters:
'               oPQGreaterThan, Integer
'
'*************************************************************************

  Function PQ_GreaterThan( _
      ByRef board As Integer, _
      ByRef errIn As T_Error, _
      Optional ByRef oPQGreaterThan As Integer = 0, _
      Optional ByRef pqX As Integer = 0, _
      Optional ByRef pqY As Integer = 0, _
      Optional ByRef pqTrueInputValue As Integer = 0, _
      Optional ByRef pqFalseInputValue As Integer = 0) _
      As T_Error
```

4.5.8 PQ_Greater Than or Equal?



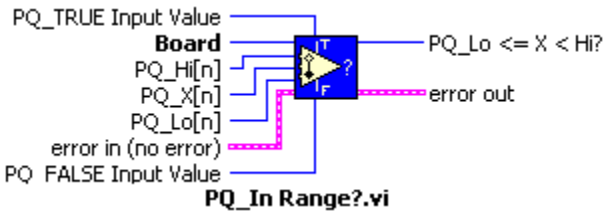
PQ_X[n] = First PQ input to compare.
PQ_Y[n] = Second PQ input to compare.

PQ_X>=Y? = PQ output that returns the value wired to the PQ_TRUE if X>=Y, or the value wired to the PQ_FALSE if X<Y.

QuBASE Equivalent.

```
'*********************************************************************
'       Function PQ_GreaterThanOrEqual
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqX, Integer
'               pqY, Integer
'               pqTrueInputValue, Integer
'               pqFalseInputValue, Integer
'
'       Ouput Parameters:
'               oPQGreaterThanOrEqual, Integer
'
'*********************************************************************

  Function PQ_GreaterThanOrEqual( _
      ByRef board As Integer, _
      ByRef errIn As T_Error, _
      Optional ByRef oPQGreaterThanOrEqual As Integer = 0, _
      Optional ByRef pqX As Integer = 0, _
      Optional ByRef pqY As Integer = 0, _
      Optional ByRef pqTrueInputValue As Integer = 0, _
      Optional ByRef pqFalseInputValue As Integer = 0) _
      As T_Error
```

4.5.9 PQ_In Range?



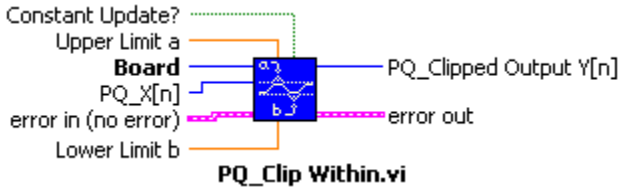
PQ_Hi[n] = PQ input, upper limit value.
PQ_X[n] = PQ input to compare.
PQ_Lo[n] = PQ input, lower limit value.

PQ_Lo <= X < Hi? = PQ output that returns the value wired to the PQ_TRUE if X falls within the two limits, or the value wired to the PQ_FALSE if X does not.

QuBASE Equivalent.

```
'**************************************************************************
'       Function PQ_InRange
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqX, Integer
'               pqHi, Integer
'               pqLo, Integer
'               pqTrueInputValue, Integer
'               pqFalseInputValue, Integer
'
'       Ouput Parameters:
'               oPQInRange, Integer
'
'**************************************************************************

  Function PQ_InRange( _
      ByRef board As Integer, _
      ByRef errIn As T_Error, _
      Optional ByRef oPQInRange As Integer = 0, _
      Optional ByRef pqX As Integer = 0, _
      Optional ByRef pqHi As Integer = 0, _
      Optional ByRef pqLo As Integer = 0, _
      Optional ByRef pqTrueInputValue As Integer = 0, _
      Optional ByRef pqFalseInputValue As Integer = 0) _
      As T_Error
```

4.5.10 PQ_Clip Within.



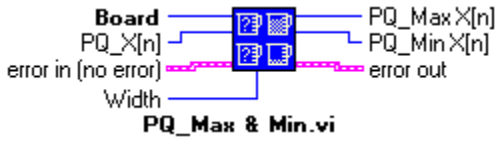
PQ_X[n] = PQ input.
Upper Limit a = LabVIEW constant used to determine the upper limit to clip X[n].
Lower Limit b = LabVIEW constant used to determine the lower limit to clip X[n].
Constant Update? = LabVIEW boolean input that determines if the LabVIEW variables are to be constantly updated or not. When left in the 'false' position, the LabVIEW constants are only updated once during the queueing phase thereby avoiding DSP-Host communications overhead. When left in the default 'true' position, LabVIEW constants are continuously updated thereby implementing real time variable updates for maximum flexibility.

PQ_Clipped Output Y[n] = PQ output reflecting the input X[n] bound by the upper and lower limits.

QuBASE Equivalent.

```
'************************************************************************
'       Function PQ_ClipWithin
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqX, Integer
'               upperLimitA, Double
'               lowerLimitB, Double
'
'       Ouput Parameters:
'               oPQClip, Integer
'
'************************************************************************

  Function PQ_ClipWithin( _
      ByRef board As Integer, _
      ByRef errIn As T_Error, _
      Optional ByRef oPQClip As Integer = 0, _
      Optional ByRef pqX As Integer = 0, _
      Optional ByRef upperLimitA As Double = 5, _
      Optional ByRef lowerLimitB As Double = -5) _
      As T_Error
```

4.5.11 PQ_Max & Min.



PQ_X[n] = PQ input.
Width N = LabVIEW constant to determine the number of points inside a moving block from which to extract the high and low peak values.

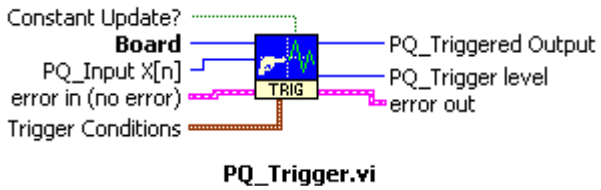
PQ_Max X[n] = PQ output of the maximum value from a block of "N" points.
PQ_Min X[n] = PQ output of the minimum value from a block of "N" points.

QuBASE Equivalent.

```
'*************************************************************************
'       Function PQ_MaxMin
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqX, Integer
'               iWidth, Integer
'
'       Ouput Parameters:
'               oPQMin, Integer
'               oPQMax, Integer
'
'*************************************************************************

  Function PQ_MaxMin( _
      ByRef board As Integer, _
      ByRef errIn As T_Error, _
      Optional ByVal pqX As Integer = 0, _
      Optional ByVal iWidth As Integer = 100, _
      Optional ByRef oPQMin As Integer = 0, _
      Optional ByRef oPQMax As Integer = 0) _
      As T_Error
```

4.5.12 PQ_Trigger.

The PQ_Trigger function is the classical implementation of an analog trigger detector. When the trigger conditions are met, a PQ "zero" value or TRUE is returned, while a PQ non-zero value indicates that the conditions are not met. The resultant of the trigger function may be used to synchronize any signal to any time dependent event. It can be fed into the “PQ_Sync In”, “PQ_Enable” or "PQ_Trig_In" input terminals of any PQ VI.



PQ_X[n] = PQ input signal to be examined.

Trigger Conditions = LabVIEW cluster used to determine the conditions upon which a trigger condition is met. Please refer to the next section for more details.

Constant Update? = LabVIEW boolean input that determines if the LabVIEW variables are to be constantly updated or not. When left in the 'false' position, the LabVIEW constants are only updated once during the queueing phase thereby avoiding DSP-Host communications overhead. When left in the default 'true' position, LabVIEW constants are continuously updated thereby implementing real time variable updates for maximum flexibility.

PQ_Triggered Output = PQ output that returns a low asserted value or a PQ "zero" for a true or trigger condition met, or a high value or PQ non-zero value for a false or no trigger.

PQ_Trigger Level = PQ output value representing the value of the input when the trigger condition occurred.

QuBASE Equivalent.

```
'**********************************************************************
'       Function PQ_Trigger
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               TriggerConditions, T_PQTrigger
'               PQInput, Integer
'
'       Ouput Parameters:
'               oPQTriggered, Integer
'               oPQTriggerLevel, Integer
'
'**********************************************************************

    Function PQ_Trigger( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef TriggerConditions As T_PQTrigger, _
        Optional ByRef oPQTriggered As Integer = 0, _
        Optional ByRef oPQTriggerLevel As Integer = 0, _
        Optional ByRef PQInput As Integer = 0) _
        As T_Error
```

4.5.12.1 Trigger Conditions.

1) Trigger Level: The TRIGLEVEL value determines the voltage level at which the specified trigger channel must intersect. The trigger voltage is specified within the maximum range of the analog input of one of the daughter modules. Triggering is ignored if the slope is disabled or set to none.

2) Trigger Slope: The TRIGGER SLOPE setting determines the direction or slope at which the specified trigger channel must cross the trigger level. Triggering can be set on a positive slope, negative slope, or be disabled if set to none.

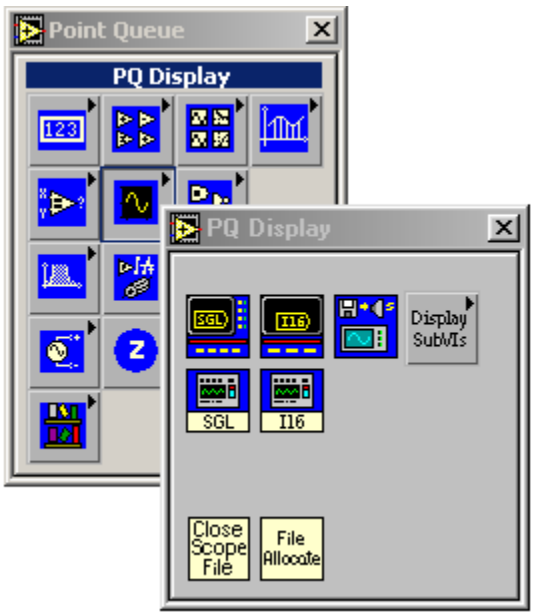
3) Hysteresis: The HYSTERESIS value determines a voltage window which the trigger channel must exceed before intersecting the trigger level. The hysteresis is an absolute value, whereby false triggers can be avoided.

For example, a hysteresis value of 1V means the signal must drop 1V below the trigger level, in order for a trigger to occur on a positive slope; or the signal must rise 1V above the trigger level for the negative slope case.

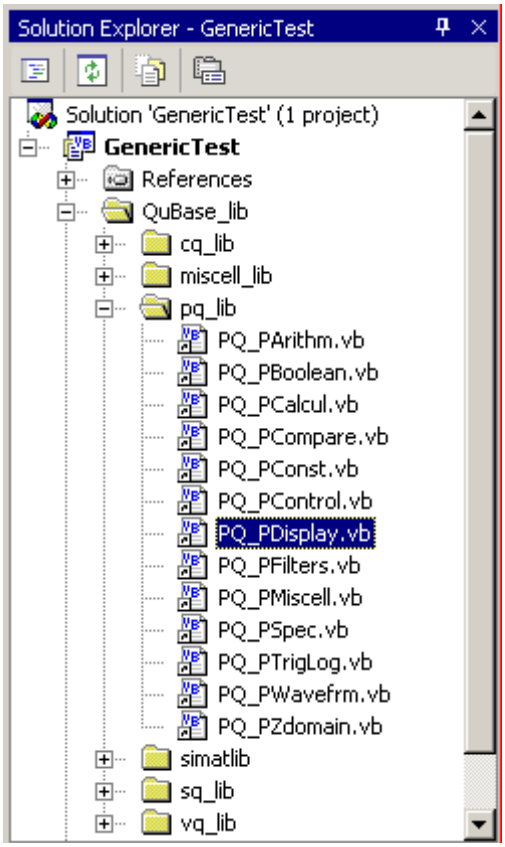
4) Pretrigger/Delay: The PRETRIG/DELAY specifies the number of samples to keep track of before or after a trigger event. A negative entry for Pretrig/Delay specifies the number of data points to keep track of before a trigger event, thereby defining a starting point to collect and display any data specified in the Samples entry. A "0" entry or a positive entry for Pretrig/Delay specifies the number of data points to wait after a trigger event before collecting and displaying any data specified in the Samples entry.

Note: The Pretrigger/Delay is variable is not defined within the PQ_Trigger function, but rather in the PQ_Scope functions. Please refer to the next section for more details.

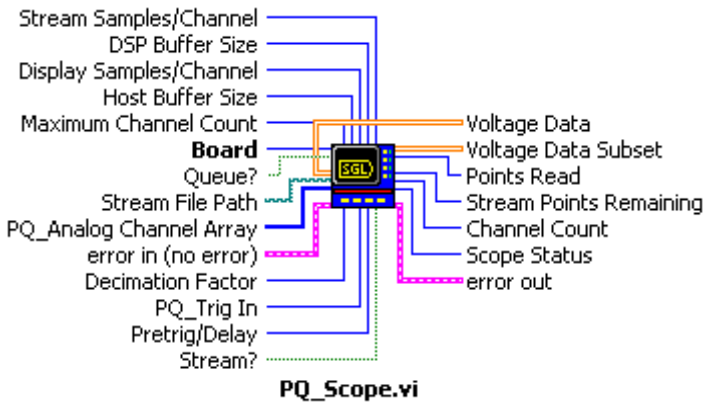
4.6 PQ Display & Log to Disk.



QuBASE Equivalent.



4.6.1 PQ_Scope.



Stream Samples/Channel = LabVIEW input that specifies the number of points to be recorded to disk.

Display Samples/Channel = LabVIEW input that specifies the number of points to be displayed per channel.

Host Buffer Size = LabVIEW input that specifies the maximum host buffer allocated for display or disk recording. It is recommended to be at least 2-4 times larger than the 'DSP Buffer Size' variable, typically 128k.

DSP Buffer Size = LabVIEW input that specifies the maximum buffer set aside within the DSP's memory space. A recommended size of 32k is appropriate for most applications, when more SRAM memory is available, increasing the buffer size to 64k will improve data transfer speeds.

Maximum Channel Count = LabVIEW input used to limit the maximum number of elements in the 'PQ_Analog Channel Array' input. Allows the number of elements inside of the 'PQ_Analog Channel Array' to be altered in real time while a VI is running.

Queue? = LabVIEW boolean input that specifies if the VI is to be executed in a queuing sequence or not.

Stream File Path = LabVIEW filepath input that specifies the target file to record data during stream to disk operations. The format of the resultant data inside the file is 32 bit single precision floating point per channel, and may represent voltages or any other 'PQ' source.
NOTE: *Do not leave the filepath fields blank – any filename must be present.*

PQ_Analog Channel Array = PQ input array that specifies the PQ input sources to be displayed or recorded to disk.
NOTE: *The format of the PQ data wired to the PQ_Analog Channel Array terminal may represent scaled or floating point values only.*

Decimation Factor = LabVIEW input used to specify the number of points to drop for display or disk recording.

PQ_Trig In = PQ input that specifies a PQ trigger source.

Pretrig/Delay = LabVIEW input that takes in a negative value for pretriggering and positive values for postriggering, this value is limited by the amount allotted in Scope Buffer Size input.

Stream? = LabVIEW boolean input that specifies if the outgoing data is being written to

a target file or not.

Voltage Data Subset = LabVIEW two dimensional output array, that contains the voltage scaled data to be used by LabVIEW.

Voltage Data = LabVIEW one dimensional output array, very similar to Voltage Output, except all channel data is interleaved.

Points Read = LabVIEW output that indicates the number of points read through a queuing iteration.

Stream Points Remaining = LabVIEW output that indicates the number of points remaining during disk recording.

Channel Count = LabVIEW output that indicates the number of channels used for display or disk recording.

The PQ_Scope VI allows LabVIEW to receive data from the DSP. This data can then be displayed and/or streamed to disk. The minimal inputs to the PQ_Scope VI, whether streaming or displaying, is an array wired into the PQ_Analog Channel Array. This array can be built by wiring in any PQ variable within the block diagram. The data is returned to LabVIEW in the form of a 32 bit single precision floating point voltage formatted as a two dimensional array. With PQ_Analog Channel Array being the only input wired, PQ_Scope defaults to a display of 256 points per channel. The input to Display Samples/chan can be wired to any desired value.

***NOTE**: Unlike the rest of the queuing functions, the PQ_Scope function can only be used once in a queuing sequence. However, a duplicate may be made in a sequence that follows the PQ_Finish subVI with the 'Queuing?' input wired to a 'false', in order to be used in conjunction with LabVIEW function icons. Conversely, if a second icon is not used, its output array may passed to a later sequence by using either a LabVIEW Sequence Local or a LabVIEW Local Variable*.

The 'PQ_Scope' VI can also stream data to disk, with the format of the resultant data inside the file represented as 32 bit single precision floating point voltage per channel. To do so, a file path must be wired into 'Stream File Path' and a LabVIEW value must be wired to the 'Stream Samples/Channel' input. Lastly, to set PQ_Scope in stream mode, one must wire the 'Stream?' boolean input to a LabVIEW True, since by default this input is set to a LabVIEW False.

The PQ_Scope VI can simultaneously display and stream to disk by simply wiring each appropriate input for both display and stream as previously outlined. The display may slow the streamer down enough that erroneous data may be logged to disk. If this occurs, PQ_Scope VI will return an overflow error. This may be remedied by the following:

1) Reducing the number of samples or channels.
2) Closing all subVIs during execution.
3) Closing any Windows applications which may be stealing time.
4) Changing graph attributes such as size, autoscaling, smooth updates, etc.

If overflow errors continue, disconnect the output graph and try to stream without simultaneously running the display.

The PQ_Trigger VI can be used to trigger either the streamer or display by wiring its output to the 'PQ_Trig In' input. The trigger functions are the same as described in a previous section covering dedicated Oscilloscope VIs. While in display mode, each display buffer will be based on the trigger event. If streaming, the trigger will only occur during the first buffer. The 'Pretrig/Delay' input can be wired if desired. This input can be changed during execution to slide the trigger event while displaying data.

Besides the two dimensional array outputs, other outputs include; an indicator showing number of points left/channel; another show the number of points in the current two dimensional array/channel; as well as an error indicator with the following codes:

0) No Error
1) Underflow
2) Trigger Conditions not Met
3) board disconnected
-1) Overflow

The decimation input allows for only a portion of the acquired data to be passed over to the host display or disk.

Decimation has the effect of the acquired data dropping every nth point. Therefore, a decimation factor of "0" will not drop points, but a "1" will drop every other point. This will reduce your effective stream rate by a factor of 1/(n+1). Additionally, excessive underflows may occur during the scope mode of operation. During an underflow, the last valid buffer will be redisplayed. Reducing the number of display points will account for a more real-time acquisition.

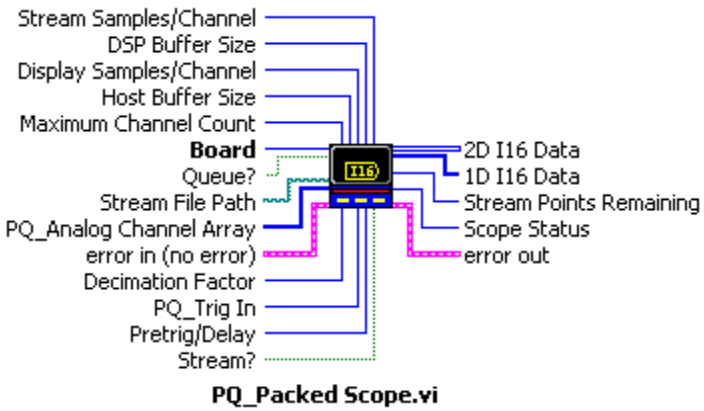
The PQ_Scope VI can also be executed outside of the queuing sequence. To do so, only the 'PQ_Analog Channel Array' input must be wired within the queuing sequence, and no graphs should be connected to the output arrays. As a result, another instance of the PQ_Scope VI can be implemented with a different VI altogether. Be sure to include the same inputs to the 'PQ_Analog Channel Array' as was wired in the original queuing sequence; additionally, wire the boolean 'Queue?' input to a LabVIEW False.

QuBASE Equivalent.

```
'*********************************************************************
'           Function PQ_Scope
'
'  Input Parameters with default values:
'     PQAnalogChannelArray, Integer:
'        Array of input channel numbers
'     pqTrigIn, Integer:
'        Trigger channel source
'     isaHandshake:
'        ISA bus handshake mode, not used
'     streamModeTF, Integer:
'        T: stream to disk, F: no stream
'     streamFilePath, String:
'        file path for streaming
'     scopeBufferSize, Integer:
'        Buffer size allocated in DSP, default 32k
'     streamSamplesPerChannel, Integer:
'        number of points to stream per channel
'     queueTF, Integer:
'        T: queueing phase, F: non-queueing, standalone operation
'     decimationFactor, Integer:
'        number of points to decimate or drop
'     pretrig_Delay, Integer:
'        number of points to display realtive to trigger point, "+" or "-"
'     displaySamplesPerChannel, Integer:
'        number of points to display per channel
'
'  Output Parameters with default values:
'     oChannelCount, Integer:
'        number of channels
'     oPointsRead, Integer:
'        stream points read
'     oPointsRemaining, Integer:
'        stream point remaining
'     oVoltage, Double:
'        output voltage array for indexed for display
'     oVoltageAll, Double:
'        output voltage array for entire output
'
'*********************************************************************

   Function PQ_Scope( _
            ByRef board As Integer, _
            ByRef errIn As T_Error, _
            ByRef oError As Integer, _
            ByRef pqAnalogChannelArray() As Integer, _
            ByRef oVoltage(,) As Double, _
            ByRef oVoltageAll(,) As Double, _
            Optional ByRef maxChannelCount As Integer = 8, _
            Optional ByRef oChannelCount As Integer = 1, _
            Optional ByRef oPointsRead As Integer = 0, _
            Optional ByRef oPointsRemaining As Integer = 50000, _
            Optional ByRef pqTrigIn As Integer = 0, _
```

```
Optional ByRef streamModeTF As Integer = 0, _
Optional ByRef streamFilePath As String = "", _
Optional ByRef hostBufferSize As Integer = 32768, _
Optional ByRef dspBufferSize As Integer = 15000, _
Optional ByRef streamSamplesPerChannel As Integer = 50000, _
Optional ByRef queueTF As Integer = 1, _
Optional ByRef decimationFactor As Integer = 0, _
Optional ByRef pretrig_Delay As Integer = 0, _
Optional ByRef displaySamplesPerChannel As Integer = 256) _
As T_Error
```

4.6.2 PQ_Packed Scope.



The PQ_Packed Scope VI acts in most respects like the PQ_Scope, except that data is transferred from the DSP as two "packed" 16-bit integers, where two 16 bit integer words are concatenated to form a single 32 bit word, thereby maximizing throughput across the PCI bus.

Stream Samples/Channel = LabVIEW input that specifies the number of points to be recorded to disk.

Display Samples/Channel = LabVIEW input that specifies the number of points to be displayed per channel.

Host Buffer Size = LabVIEW input that specifies the maximum host buffer allocated for display or disk recording. It is recommended to be at least 2-4 times larger than the 'DSP Buffer Size' variable, typically 128k.

DSP Buffer Size = LabVIEW input that specifies the maximum buffer set aside within the DSP's memory space. A recommended size of 32k is appropriate for most applications, when more SRAM memory is available, increasing the buffer size to 64k will improve data transfer speeds.

Maximum Channel Count = LabVIEW input used to limit the maximum number of elements in the 'PQ_Analog Channel Array' input. Allows the number of elements inside of the 'PQ_Analog Channel Array' to be altered in real time while a VI is running.

Queue? = LabVIEW boolean input that specifies if the VI is to be executed in a queuing sequence or not.

Stream File Path = LabVIEW filepath input that specifies the target file to record data during stream to disk operations. The format of the resultant data inside the file is 16 bit signed integer per channel, representing raw ADC data.
NOTE: *Do not leave the filepath fields blank – any filename must be present.*

PQ_Analog Channel Array = PQ input array that specifies the PQ input sources to be displayed or recorded to disk.
NOTE*: The format of the PQ data wired to the PQ_Analog Channel Array terminal cannot represent scaled or floating point values, but rather integer values not exceeding 16 bits in size. Therefore, only the values returned by the 'Unscaled Analog Input' VIs may be used.*

Decimation Factor = LabVIEW input used to specify the number of points to drop for

display or disk recording.

PQ_Trig In = PQ input that specifies a PQ trigger source.

Pretrig/Delay = LabVIEW input that takes in a negative value for pretriggering and positive values for postriggering this value is limited by the amount allotted in Scope Buffer Size input.

Stream? = LabVIEW boolean input that specifies if the outgoing data is being written to a target file or not.

2D I16 Data = LabVIEW 2D output array that contains the scaled voltage data.

1D I16 Data = LabVIEW 1D output array of 16 bit integers, with all channel data interleaved.

Points Read = LabVIEW output that indicates the number of points read through a queuing iteration.

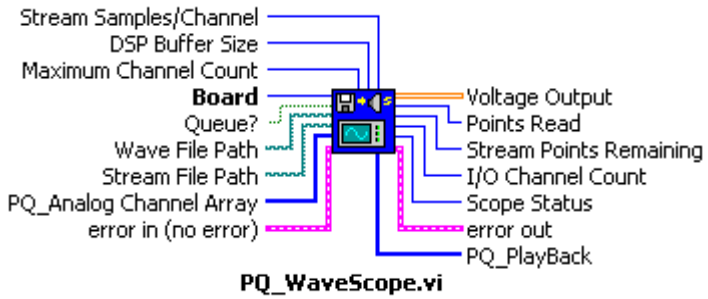
Stream Points Remaining = LabVIEW output that indicates the number of points remaining during disk recording.

QuBASE Equivalent.

```
'*********************************************************************
'           Function PQ_PackedScope
'
'  Input Parameters with default values:
'     PQAnalogChannelArray, :
'        Array of input channel numbers
'     maxChannelCount, Integer:
'        defaulted to a maximum of 8
'     pqTrigIn, Integer:
'        Trigger channel source
'     streamModeTF, Integer:
'        T: stream to disk, F: no stream
'     streamFilePath, String:
'        file path for streaming
'     hostBufferSize, Integer:
'        Buffer size allocated in DSP, default 32k
'     dspBufferSize, Integer:
'        Buffer size allocated in DSP, default 15k
'     streamSamplesPerChannel, Integer:
'        number of points to stream per channel
'     queueTF, Integer:
'        T: queueing phase, F: non-queueing, standalone operation
'     decimationFactor, Integer:
'        number of points to decimate or drop
'     pretrig_Delay, Integer:
'        number of points to display realtive to trigger point, "+" or "-"
'     displaySamplesPerChannel, Integer:
'        number of points to display per channel
'
'  Output Parameters with default values:
'     oChannelCount, Integer:
'        number of channels
'     oPointsRead, Integer:
'        stream points read
'     oPointsRemaining, Integer:
'        stream point remaining
'     oVoltage, Double:
'        output voltage array for indexed for display
'     oPackedData, Integer:
'        output binary array for entire output
'     oError, Integer:
'        error output
'
*********************************************************************

   Function PQ_PackedScope( _
            ByRef board As Integer, _
            ByRef errIn As T_Error, _
            ByRef oError As Integer, _
            ByRef pqAnalogChannelArray() As Integer, _
            ByRef oPackedData() As Integer, _
            ByRef oVoltage(,) As Double, _
            Optional ByRef maxChannelCount As Integer = 8, _
```

```
Optional ByRef oChannelCount As Integer = 1, _
Optional ByRef oPointsRead As Integer = 0, _
Optional ByRef oPointsRemaining As Integer = 5000, _
Optional ByRef pqTrigIn As Integer = 0, _
Optional ByRef streamModeTF As Integer = 0, _
Optional ByRef streamFilePath As String = "", _
Optional ByRef hostBufferSize As Integer = 32768, _
Optional ByRef dspBufferSize As Integer = 15000, _
Optional ByRef streamSamplesPerChannel As Integer = 5000, _
Optional ByRef queueTF As Integer = 1, _
Optional ByRef decimationFactor As Integer = 0, _
Optional ByRef pretrig_Delay As Integer = 0, _
Optional ByRef displaySamplesPerChannel As Integer = 256) _
As T_Error
```

4.6.3 PQ_WaveScope.



The PQ_WaveSope VI is used for simultaneous stream to disk and playback from disk. This same functionality can be achieved using several discrete VIs. However, for convenience this special purpose VI will allow both operations to be performed more efficiently.

The inputs to this VI are analogous to those used on the PQ_Scope VIs reviewed above.

***NOTE**: The number of channels to record must be equal to the number of channels to playback.*

Stream Samples/Channel = LabVIEW input that specifies the number of points to be recorded to disk.

DSP Buffer Size = LabVIEW input that specifies the maximum buffer set aside within the DSP's memory space. A recommended size of 32k is appropriate for most applications, when more SRAM memory is available, increasing the buffer size to 64k will improve data transfer speeds.

Maximum Channel Count = LabVIEW input used to limit the maximum number of elements in the 'PQ_Analog Channel Array' input. Allows the number of elements inside of the 'PQ_Analog Channel Array' to be altered in real time while a VI is running.

Queue? = LabVIEW boolean input that specifies if the VI is to be executed in a queuing sequence or not.

Wave File Path = LabVIEW Path and filename of the playback file. The format of the resident data inside the file is 32 bit single precision floating point per channel, and may represent voltages or any other 'PQ' source.

Stream File Path = LabVIEW Path and filename of the file to be recorded. The format of the resultant data inside the file is 32 bit single precision floating point per channel, and may represent voltages or any other 'PQ' source.

***NOTE:** Do not leave the filepath fields blank – any filename must be present.*

PQ_Analog Channel Array = PQ input array that specifies the PQ input sources to be displayed or recorded to disk.

***NOTE:** The format of the PQ data wired to the PQ_Analog Channel Array terminal may represent scaled or floating point values only.*

Voltage Ouput = LabVIEW 2D output array that contains the scaled voltage data.

Points Read = LabVIEW output that indicates the number of points read through a queuing iteration.

Stream Points Remaining = LabVIEW output that indicates the number of points

remaining during disk recording.

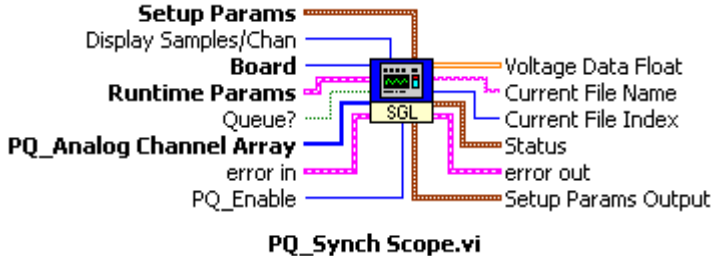
I/O Channel Count = LabVIEW output that indicates the number of channels used for display or disk recording.

PQ_Playback = PQ 1D output array containing the playback data.

QuBASE Equivalent.

```
'*********************************************************************
'           Function PQ_WaveScope
'
'  Input Parameters with default values:
'     PQAnalogChannelArray, Integer:
'        Array of input channel numbers
'     iMaxChannels, Integer:
'        defaulted to a maximum of 8
'     streamFilePath, String:
'        file path for streaming data
'     waveFilePath, String:
'        file path for playback file
'     scopeBufferSize, Integer:
'        Buffer size allocated in DSP, default 325k
'     streamSamplesPerChannel, Integer:
'        number of points to stream per channel
'     queueTF, Integer:
'        T: queueing phase, F: non-queueing, standalone operation
'
'  Output Parameters with default values:
'     oChannelCount, Integer:
'        number of channels
'     oPointsRead, Integer:
'        stream points read
'     oPointsRemaining, Integer:
'        stream point remaining
'     oVoltage, Double:
'        output voltage array for indexed for display
'     oPQ_PlayBack, Integer
'*********************************************************************

   Function PQ_WaveScope( _
               ByVal board As Integer, _
               ByRef errIn As T_Error, _
               ByRef oError As Integer, _
               ByRef pqAnalogChannelArray() As Integer, _
               ByRef oVoltage(,) As Double, _
               ByRef oPQ_PlayBack() As Integer, _
               Optional ByRef oChannelCount As Integer = 1, _
               Optional ByRef oPointsRead As Integer = 0, _
               Optional ByRef oPointsRemaining As Integer = 5000, _
               Optional ByVal iMaxChannels As Integer = 8, _
               Optional ByRef streamFilePath As String = "", _
               Optional ByRef waveFilePath As String = "", _
               Optional ByRef scopeBufferSize As Integer = 32000, _
               Optional ByRef streamSamplesPerChannel As Integer = 5000, _
               Optional ByRef queueTF As Integer = 1) As T_Error
```

4.6.4 PQ_Synch Scope.

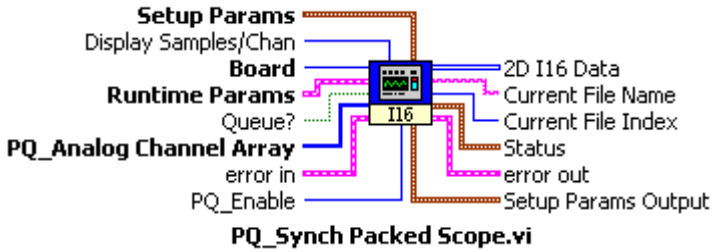


This VI synchronously transfers data from a DSP buffer to host memory. The data may be displayed on the screen as an oscilloscope as well as be logged to disk. Each data point transferred is represented as a 32 bit, single precision floating point value.

QuBASE Equivalent.

```
'*********************************************************************
'           Function PQ_SynchScope
'
'  Input Parameters with default values:
'     board, Integer.
'     errIn, T_Error.
'     pqChannels, Integer:
'        Array of input channel numbers
'     setupParams, syncScopeParamsIn:
'        HostBufferSize defaulted 128k
'        DSPBufferSize defaulted 32k
'        DSPBlockSize defaulted 256
'        MaxChannels defaulted 8
'     runtimeParams, stcRuntimeParams:
'        filepaths for streaming data
'        trigger conditions
'        save to disk defaulted to OFF
'     displaySamplesPerChannel, Integer:
'        samples to display on screen defaulted to 1k
'     pqEnable, Integer:
'        0: enable, non0: wait for enable
'
'  Output Parameters with default values:
'     voltageData, Double:
'        output voltage array for indexed for display
'     currentFileName, String:
'        current filename being used
'     currentFileIndex, Integer:
'        File index
'     status, stcStatus:
'        DSP buffer Overflow/underflow
'        Host buffer Overflow/underflow
'        trigger feedback
'        points remaining
'     setupParamsOut, syncScopeParamsOut
'        DSP buffer limit
'        DSP block count per channel
'
'*********************************************************************
   Function PQ_SynchScope( _
            ByRef board As Integer, _
            ByRef errIn As T_Error, _
            ByRef voltageData(,) As Double, _
            ByRef currentFileName As String, _
            ByRef currentFileIndex As Integer, _
            ByRef status As stcStatus, _
            ByRef setupParamsOut As syncScopeParamsOut, _
            ByRef pqChannels() As Integer, _
            ByRef setupParams As syncScopeParamsIn, _
            ByRef runtimeParams As stcRuntimeParams, _
            Optional ByRef displaySamplesPerChannel As Integer = 1000, _
            Optional ByRef pqEnable As Integer = 0) _
            As T_Error
```

4.6.5 PQ_Synch Packed Scope.



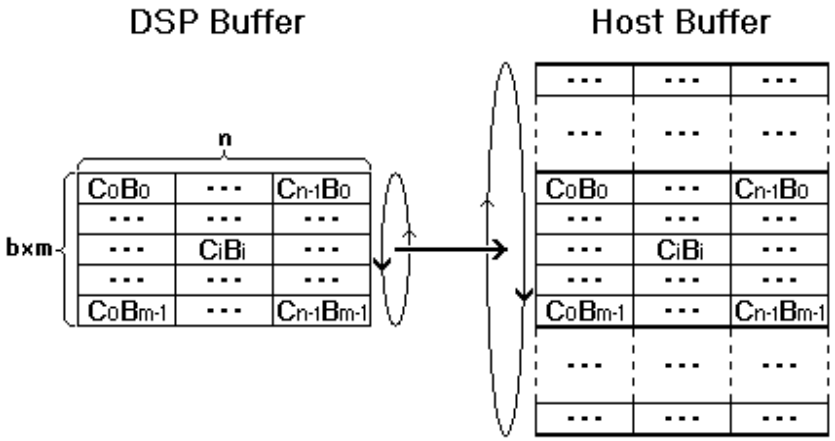
This VI synchronously transfers data from a DSP buffer to host memory. The data may be displayed on the screen as an oscilloscope as well as be logged to disk. Each data point transferred is represented as two 16 bit integers, concatenated to form a single 32 bit fixed point value. Concatenating or packing two 16 bit words into a single 32 bit word effectively increases data transfer bandwidth, as well as doubling the amount of data points within a given block of requested memory in both the DSP and the host domains.

QuBASE Equivalent.

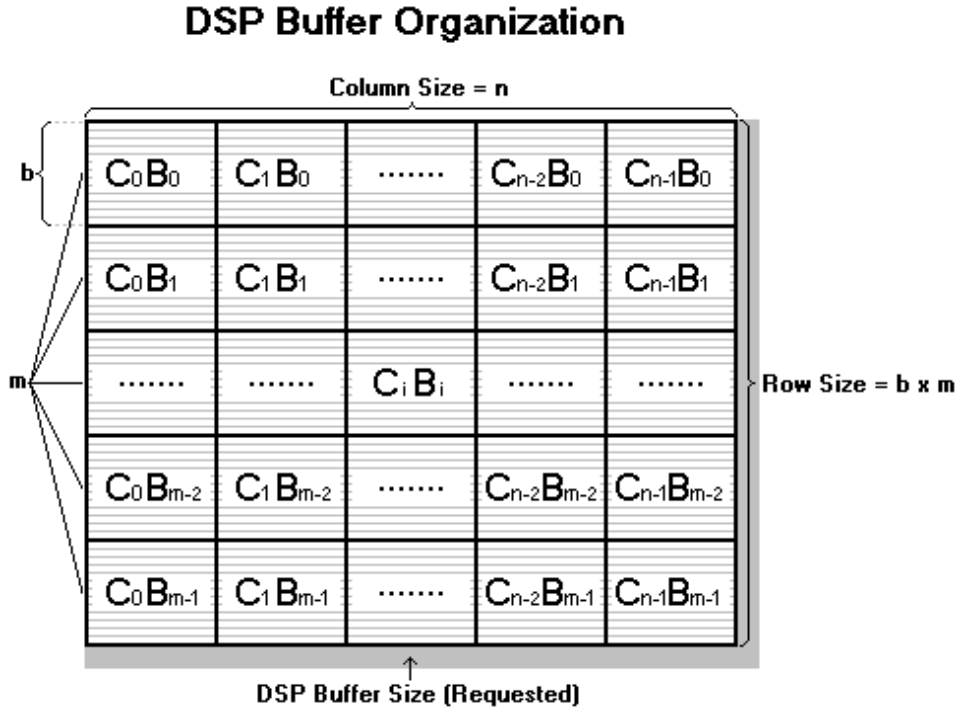
```
'*********************************************************************
'           Function PQ_SynchScope
'
'  Input Parameters with default values:
'     board, Integer.
'     errIn, T_Error.
'     pqChannels, Integer:
'        Array of input channel numbers
'     setupParams, syncScopeParamsIn:
'        HostBufferSize defaulted 128k
'        DSPBufferSize defaulted 32k
'        DSPBlockSize defaulted 256
'        MaxChannels defaulted 8
'     runtimeParams, stcRuntimeParams:
'        filepaths for streaming data
'        trigger conditions
'        save to disk defaulted to OFF
'     displaySamplesPerChannel, Integer:
'        samples to display on screen defaulted to 1k
'     pqEnable, Integer:
'        0: enable, non0: wait for enable
'
'  Output Parameters with default values:
'     voltageData, Double:
'        output voltage array for indexed for display
'     currentFileName, String:
'        current filename being used
'     currentFileIndex, Integer:
'        File index
'     status, stcStatus:
'        DSP buffer Overflow/underflow
'        Host buffer Overflow/underflow
'        trigger feedback
'        points remaining
'     setupParamsOut, syncScopeParamsOut
'        DSP buffer limit
'        DSP block count per channel
'*********************************************************************

   Function PQ_SynchScope( _
            ByRef board As Integer, _
            ByRef errIn As T_Error, _
            ByRef voltageData(,) As Double, _
            ByRef currentFileName As String, _
            ByRef currentFileIndex As Integer, _
            ByRef status As stcStatus, _
            ByRef setupParamsOut As syncScopeParamsOut, _
            ByRef pqChannels() As Integer, _
            ByRef setupParams As syncScopeParamsIn, _
            ByRef runtimeParams As stcRuntimeParams, _
            Optional ByRef displaySamplesPerChannel As Integer = 1000, _
            Optional ByRef pqEnable As Integer = 0) _
            As T_Error
```

4.6.6 Common Synchronous Scope Parameters.

Both the DSP and the host exchange large amounts of data using a circular buffer scheme to account for their speed differences.



The DSP buffer is configured as a matrix since it's a convenient method for implementing a circular buffer, and is organized as follows:



Row Size = b x m
Column Size = n

where

b = Block Size/Channel

Sets the number of points per channel contained in each block to be transferred from the DSP. Each time the host reads points from the DSP, it is implemented by block multiples. Each block is represented in the illustration as 'Bi', where i = 0 to (m-1).

m = Number of Blocks/Channel
Computed by taking the integer portion of the result from the following formula:

*Number of Blocks/Channel = Integer[('DSP Buffer Size (Requested)' - Max_Channels)]/[b * n].*

The integer part of the resultant must be equal or larger than '3'.

n = Channel Count
The 'Channel Count' is total number of channels to be recorded, and is derived from the size of the input array 'PQ_Analog Channel Array'. Each channel is represented in the illustration as 'Ci', where i = 0 to (n-1).

DSP Buffer Size (Requested)
Used to approximate the requested DSP's total buffer size to be allocated to support the appropriate amount of data to be transferred. It is always slightly larger than the actual allocated DSP buffer size. A minimum is typically set between 32K to 64K, but larger values of 128k to 512k may be used when more DSP memory is available.

DSP Buffer Size (Actual)
The actually allocated DSP buffer size, which is computed using the following formula:

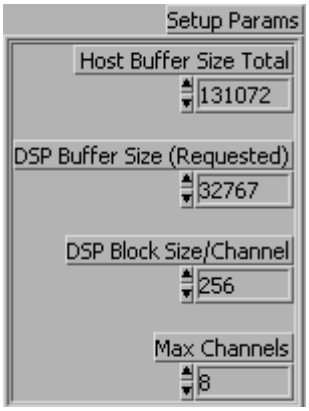
*DSP Buffer Size (Actual) = b * n * m*
or
*DSP Buffer Size (Actual) = (Block Size/Channel) * (Channel Count) * (Block Count/Channel)*

Max Channels
Variable used to limit the maximum number of channels.

Host Buffer Size Total
The host buffer serves as a temporary data buffer until it is logged to disk or displayed on the scope. It should be at least 1.5 to 4 larger than the 'DSP Buffer Size (Requested)' in order to account for PC latencies.

Setup Parameters = LabVIEW cluster representing parameters to configure the circular buffers in the DSP and the host. These circular buffers behave as temporary repositories when data is being transferred from the DSP to the host, particularly important when data is being logged to disk. They include:

Note: These parameters are to be set before running the application VI, and cannot be altered during runtime or after the DSP is active running an application.



Host Buffer Size Total = LabVIEW input that determines the host's total buffer size to be allocated to support the appropriate amount of data to be transferred. It is recommended to be at least 1.5 to 4 times larger than the 'DSP Buffer Size (Requested)' variable.

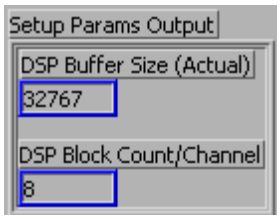
DSP Buffer Size (Requested) = LabVIEW input that approximates the requested DSP's total buffer size to be allocated to support the appropriate amount of data to be transferred. It is always slightly larger than the actual allocated DSP buffer size displayed in the 'DSP Buffer Size (Actual)' field inside of the 'Setup Params Output' cluster. A minimum is typically set between 32K to 64K, but larger values of up to 128k to 256k may be used when more DSP memory is available.

DSP Block Size/Channel = LabVIEW input that determines the number of points per channel contained in each block to be allocated within the 'DSP Buffer Size (Actual)'. Each time the host reads points from DSP, it is implemented by block multiples. Typically set to a power of 2 value of 256 or 512.

Note: The total amount of memory allocated within DSP is calculated as:
DSP Buffer Size (Actual) =
*(DSP Block Size/Channel) * (Number of PQ Input Channels)*(DSP Block Size/Channel).*

Max Channels = LabVIEW input used to limit the maximum number of elements in the 'PQ_Analog Channel Array' input array. Only allows the number of elements inside of the 'PQ_Analog Channel Array' to be altered below the limit in real time while the scope display is running, but NOT alter these number of elements during data streaming to disk.

Setup Parameters Output = LabVIEW output cluster that contains the DSP buffer sizes actually allocated for the transfer; they include:



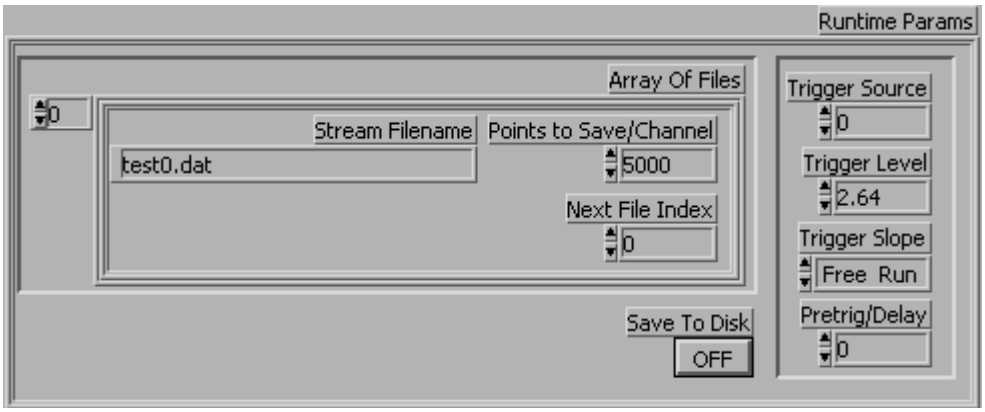
DSP Buffer Size (Actual) = LabVIEW output that indicates the DSP's total buffer size actually allocated. A computed value that always be slightly smaller than the input variable 'DSP Buffer Size (Requested)'.

DSP Block Count/Channel = DSP Block Count/Channel = LabVIEW output that indicates the actual number of blocks that were allocated per channel.

Note: The total amount of memory allocated within DSP is calculated as:
DSP Buffer Size (Actual) =
*(DSP Block Size/Channel) * (Number of PQ Input Channels)*(DSP Block Size/Channel).*

Display Samples/Chan = LabVIEW input that specifies the number of points to be displayed per channel.

Runtime Parameters = LabVIEW input cluster representing parameters that may be altered during runtime; they include:



Stream Filename = LabVIEW file path input that specifies the name of the file to contain the recorded data.
NOTE: *Do not leave the filepath fields blank – any filename must be present.*

Points to Save/Channel = LabVIEW input that determines the number of points per channel to be saved to file. The size of the resulting file is obtained multiplying this number by the number of channels. The maximum number of points saved to disk is only dependent on the drive's free space.

Next File Index = LabVIEW input that indexes the file in question. Indexing is a useful method to keep track of the order in which the files were logged.

Save to Disk? = LabVIEW boolean input that enables data to be saved to disk.

Trigger Source = LabVIEW input used to specify the channel number used as a trigger source.

Trigger Level = LabVIEW input that determines the desired threshold voltage level to cause a trigger.

Trigger Slope = LabVIEW input that determines the slope of the trigger.

Pretrig/Delay = LabVIEW input that takes in a negative value for pretriggering and positive values for post-triggering, this value is limited by the amount allotted in 'Scope Buffer Size' input variable.

Queue? = LabVIEW boolean input that specifies if the VI is to be executed in a queuing sequence or not.

PQ_Analog Channel Array = PQ input array that specifies the PQ input channels to be displayed or recorded to disk.

PQ_Enable = PQ input that enables the collection of data within the DSP's buffer, similar to a 'gate' input. The DSP's buffer is enabled when the queueing or PQ value is "1" or "True", and disabled when the PQ value is "0" or "False.

Note: Be aware that a queueing or PQ value is NOT a LabVIEW constant, but rather a constant within the DSP's domain which can only be sourced from a PQ variable.

Voltage Data Float = LabVIEW 2D output array representing all channel data as 32 bit, single precision, floating point values.
***Note**: Only applicable to the PQ_Synch Scope VI.*

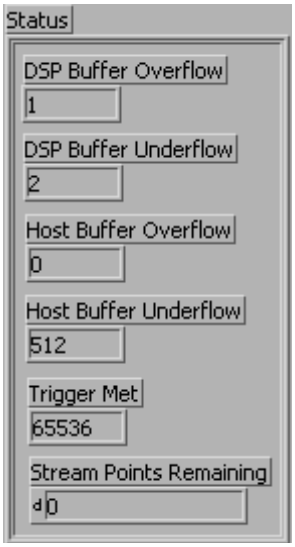
2D I16 Data = LabVIEW 2D output array representing all channel data as 16 bit, integer values, without any scaling performed within the DSP domain. This array can be easily converted to voltage values by simply applying the appropriate scaling factors.
***Note**: Only applicable to the PQ_Synch Packed Scope VI.*

Current File Name = LabVIEW output string that indicates the current file being accessed.

Current File Index = LabVIEW output that indicates the index of the current file being accessed. Indexing is a useful method to keep track of the order in which the files were logged.

***NOTE:** Do not leave the filepath fields blank – any filename must be present.*

Status = LabVIEW output cluster that contains various status variables of interest; they include:



DSP Buffer Overflow = LabVIEW output that indicates when the DSP buffer was overflowed, which occurs when the DSP collects data at a higher rate than that transferred to the host, thus causing the host to drop data or become out of synchronization. In many cases, increasing the DSP buffer size will help solve this problem.

DSP Buffer Underflow = LabVIEW output that indicates when the DSP has underflowed, which occurs when the DSP collects data at a slower rate than that requested to be transferred to the host. It occurs more frequently when selected blocks are relatively large or if the sample rate is slow relative to the transfer rate. This status indicator is benign as no data is lost.

Host Buffer Overflow = LabVIEW output that indicates that more points have been transferred to the host than it is able to display. In many cases, increasing 'Host Buffer Size' (in Setup Params) variable will help solve this problem.

Host Buffer Underflow = LabVIEW output that indicates when the amount of transferred data is insufficient relative to the requested amount of data to be displayed (selected by Display Samples/Chan). It occurs more frequently when the sample rate is relatively low, as data is shown as it is collected. This status indicator is benign as no data is lost, it is merely intended to indicate that there are insufficient points to be displayed on the screen.

Trigger Met = LabVIEW output that indicates when all trigger conditions are met.

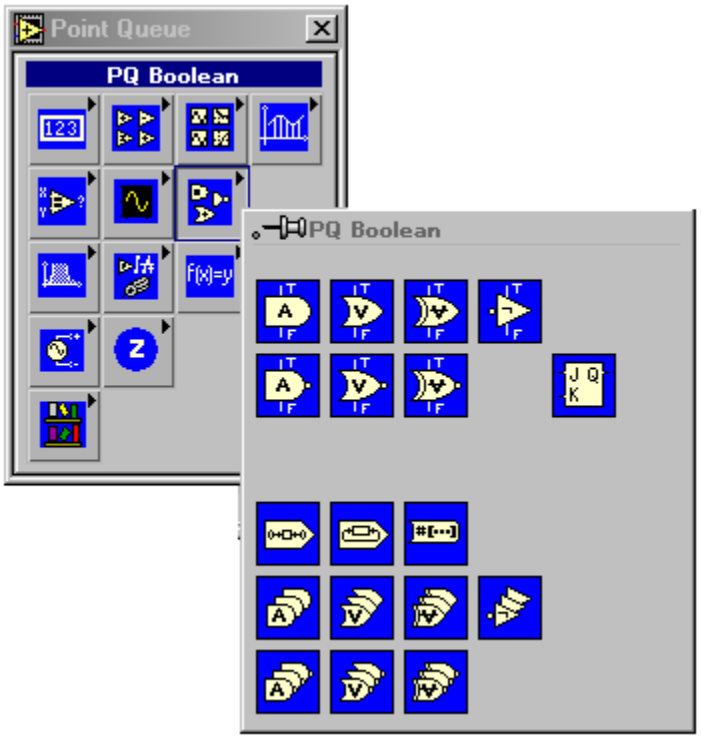
Stream Points Remaining = LabVIEW output that indicates the remaining number of points per channel to be saved to disk.

4.7 PQ Boolean.

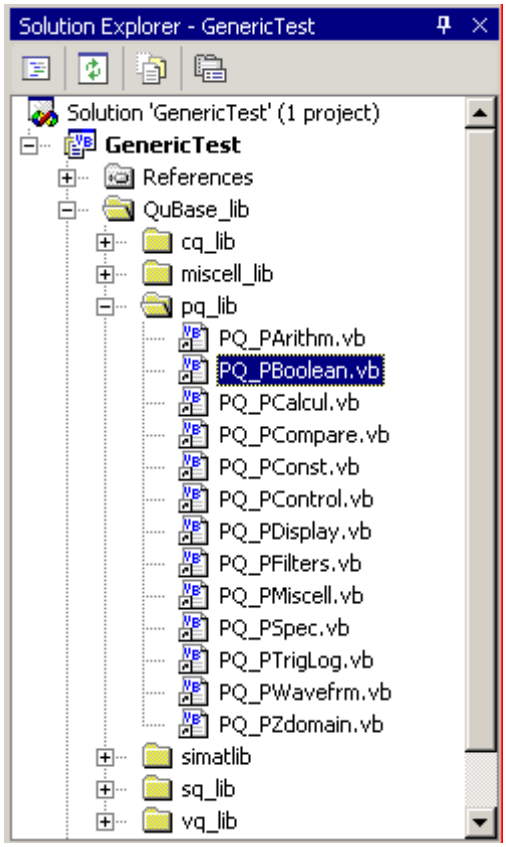
These VIs perform traditional boolean logic in two different forms:

1) Logic gates where each PQ input terminal is treated as a single logical value, and whose resultant, as in the PQ comparison VIs, is the PQ TRUE/FALSE as defined by a separate pair of queueing inputs. The output returns the values wired from the PQ TRUE/FALSE terminals, where the TRUE and FALSE states may be defined with unique PQ values.

2) Logic gates where each PQ input terminal is treated as a 24 bit word with each bit representing a unique logical value, and whose resultant is also reflected as a 24 bit word. The TRUE/FALSE states are defined as traditional "0" or "1" values respectively. Because of the DSP's floating point format, single PQ terminals are limited to an effective 24 bits maximum. For operations requiring wider widths, data must be split into smaller portions with each portion not exceeding the 24 bit limit.

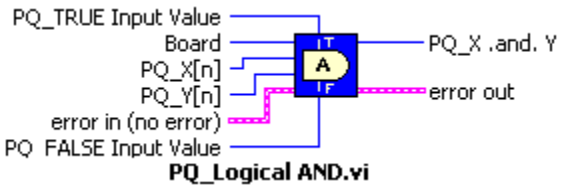


QuBASE Equivalent.



4.7.1 PQ_Logical AND.

Performs a Logical AND on two PQ inputs, and return PQ True/False values.



PQ_X[n] = PQ input.
PQ_Y[n] = PQ input.

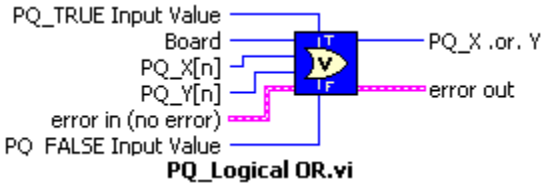
PQ_X .and. Y = PQ output, logical AND of X[n] and Y[n].

QuBASE Equivalent.

```
'**********************************************************************
'       Function PQ_LogicalAnd
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqX, Integer
'               pqY, Integer
'               pqTrueInputValue, Integer
'               pqFalseInputValue, Integer
'
'       Output Parameters:
'               oPQAnd, Integer
'
'**********************************************************************

  Function PQ_LogicalAnd( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef oPQAnd As Integer = 0, _
        Optional ByRef pqX As Integer = 0, _
        Optional ByRef pqY As Integer = 0, _
        Optional ByRef pqTrueInputValue As Integer = 0, _
        Optional ByRef pqFalseInputValue As Integer = 0) _
        As T_Error
```

4.7.2 PQ_Logical OR.

Performs a Logical OR on two PQ inputs, and return PQ True/False values.



PQ_X[n] = PQ input.
PQ_Y[n] = PQ input.

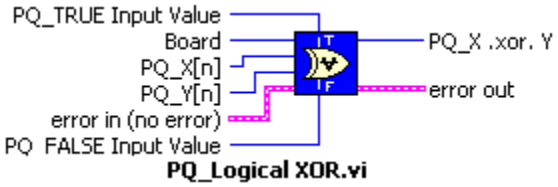
PQ_X .or. Y = PQ output, logical OR of X[n] and Y[n].

QuBASE Equivalent.

```
'*************************************************************************
'       Function PQ_LogicalOr
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqX, Integer
'               pqY, Integer
'               pqTrueInputValue, Integer
'               pqFalseInputValue, Integer
'
'       Output Parameters:
'               oPQOr, Integer
'
'*************************************************************************

  Function PQ_LogicalOr( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef oPQOr As Integer = 0, _
        Optional ByRef pqX As Integer = 0, _
        Optional ByRef pqY As Integer = 0, _
        Optional ByRef pqTrueInputValue As Integer = 0, _
        Optional ByRef pqFalseInputValue As Integer = 0) _
        As T_Error
```

4.7.3 PQ_Logical XOR.

Performs a Logical Exclusive OR on two PQ inputs, and return PQ True/False values.



PQ_X[n] = PQ input.
PQ_Y[n] = PQ input.

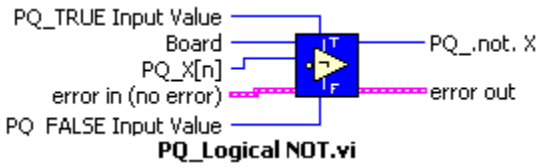
PQ_X .xor. Y = PQ output, logical XOR of X[n] and Y[n].

QuBASE Equivalent.

```
'************************************************************************
'       Function PQ_LogicalXor
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqX, Integer
'               pqY, Integer
'               pqTrueInputValue, Integer
'               pqFalseInputValue, Integer
'
'       Output Parameters:
'               oPQXor, Integer
'
'************************************************************************

  Function PQ_LogicalXor( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef oPQXor As Integer = 0, _
        Optional ByRef pqX As Integer = 0, _
        Optional ByRef pqY As Integer = 0, _
        Optional ByRef pqTrueInputValue As Integer = 0, _
        Optional ByRef pqFalseInputValue As Integer = 0) _
        As T_Error
```

4.7.4 PQ_Logical NOT.

Performs a Logical NOT or inverts the assertion level on a single PQ input.



PQ_X[n] = PQ input.

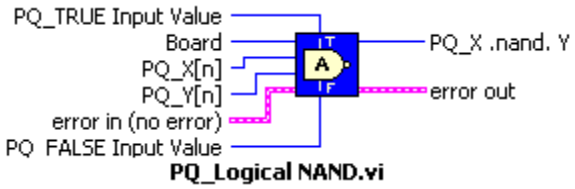
PQ_.not. X = PQ output, logical NOT of X[n].

QuBASE Equivalent.

```
'*********************************************************************
'       Function PQ_LogicalNot
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqX, Integer
'               pqY, Integer
'               pqTrueInputValue, Integer
'               pqFalseInputValue, Integer
'
'       Output Parameters:
'               oPQNot, Integer
'
'*********************************************************************

  Function PQ_LogicalNot( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef oPQNot As Integer = 0, _
        Optional ByRef pqX As Integer = 0, _
        Optional ByRef pqTrueInputValue As Integer = 0, _
        Optional ByRef pqFalseInputValue As Integer = 0) _
        As T_Error
```

4.7.5 PQ_Logical NAND.

Performs a Logical NAND on two PQ inputs, and return PQ True/False values.



PQ_X[n] = PQ input.
PQ_Y[n] = PQ input.

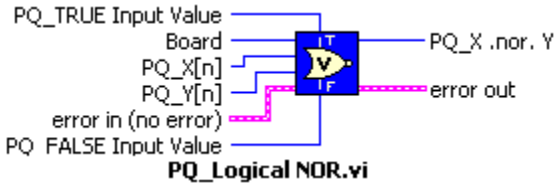
PQ_X .nand. Y = PQ output, logical NAND of X[n] and Y[n].

QuBASE Equivalent.

```
'*********************************************************************
'       Function PQ_LogicalAnd
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqX, Integer
'               pqY, Integer
'               pqTrueInputValue, Integer
'               pqFalseInputValue, Integer
'
'       Output Parameters:
'               oPQAnd, Integer
'
'*********************************************************************
```

4.7.6 PQ_Logical NOR.

Performs a Logical NOR on two PQ inputs, and return PQ True/False values.



PQ_X[n] = PQ input.
PQ_Y[n] = PQ input.

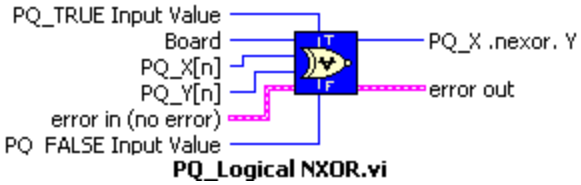
PQ_X .nor. Y = PQ output, logical NOR of X[n] and Y[n].

QuBASE Equivalent.

```
'*********************************************************************
'       Function PQ_LogicalNor
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqX, Integer
'               pqY, Integer
'               pqTrueInputValue, Integer
'               pqFalseInputValue, Integer
'
'       Output Parameters:
'               oPQNor, Integer
'
'*********************************************************************
```

4.7.7 PQ_Logical NOT XOR.

Performs a Logical NeXOR or NOT Exclusive OR on two PQ inputs, and return PQ True/False values.



PQ_X[n] = PQ input.
PQ_Y[n] = PQ input.

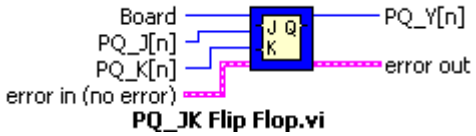
PQ_X .nexor. Y = PQ output, logical NEXOR of X[n] and Y[n].

QuBASE Equivalent.

```
'*************************************************************************
'       Function PQ_LogicalNotXor
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqX, Integer
'               pqY, Integer
'               pqTrueInputValue, Integer
"               pqFalseInputValue, Integer
'
'       Output Parameters:
'               oPQNotXor, Integer
'
'*************************************************************************
```

4.7.8 PQ_JK Flip Flop.

This function models the behavior of a JK flip-flop.



PQ_J[n] = PQ input, J terminal of JK flip-flop
PQ_K[n] = PQ input, K terminal of JK flip-flop

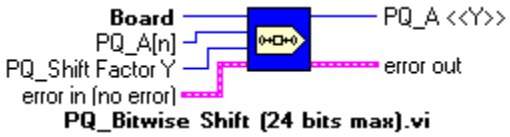
PQ_Y[n] = PQ output of flip-flop.

QuBASE Equivalent.

```
'*********************************************************************
'       Function PQ_JKFlipFlop
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqJn, Integer
'               pqKn, Integer
'
'       Output Parameters:
'               oPQYn, Integer
'

'*********************************************************************
```

4.7.9 PQ_Bitwise Shift (24 bits max).

Performs a Bitwise Shift on the single PQ input, with the Shift factor determining both the direction and the number of positions to shift the PQ input. The PQ variables represent a 24 bit integer.



PQ_A[n] = PQ input representing a 24 bit value.
PQ_Shift Factor Y = PQ input, whose value determines the number of positions to shift, while the sign represents the direction.

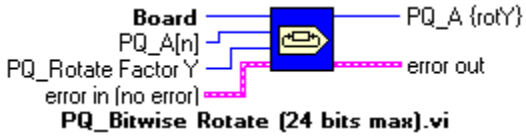
PQ_A <<Y>> = PQ output, representing the shifted 24 bit value.

QuBASE Equivalent.

```
'*************************************************************************
'       Function PQ_BitwiseShift
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqAn, Integer
'               pqBn, Integer
'               pqShiftFactor, Integer
'
'       Output Parameters:
'               oPQYn, Integer
'
'*************************************************************************

  Function PQ_BitwiseShift( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef oPQYn As Integer = 0, _
        Optional ByRef pqAn As Integer = 0, _
        Optional ByRef pqShiftFactor As Integer = 0) _
        As T_Error
```

4.7.10 PQ_Bitwise Rotate (24 bits max).

Performs a Bitwise Rotation on the single PQ input, with the Rotate factor determining both the direction and the number of positions to rotate the PQ input. The PQ variables represent a 24 bit integer.



PQ_A[n] = PQ input representing a 24 bit value.
PQ_Rotate Factor Y = PQ input, whose value determines the number of positions to rotate, while the sign represents the direction.

PQ_A {rotY} = PQ output, representing the rotated 24 bit value.

QuBASE Equivalent.

```
'*************************************************************************
'       Function PQ_BitwiseRotate
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqAn, Integer
'               pqBn, Integer
'               pqRotateFactor, Integer
'
'       Output Parameters:
'               oPQYn, Integer
'
'*************************************************************************

  Function PQ_BitwiseRotate( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef oPQYn As Integer = 0, _
        Optional ByRef pqAn As Integer = 0, _
        Optional ByRef pqRotateFactor As Integer = 0) _
        As T_Error
```

4.7.11 PQ_Bitwise Extraction (24 bits max)

Performs a Bitwise Extraction on the single PQ input, and returns a PQ output formatted as a 1D array. Each element of the PQ output array represents the value of each bit, with the nth position of the array corresponding to the nth position of the bit it represents. The PQ variables represent a 24 bit integer.

PQ_A[n] = PQ input representing a 24 bit value.

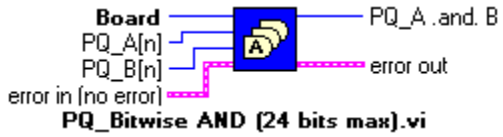
PQ_Extracted Bits Array = PQ output array, where each element of the PQ output array represents the value of each bit, with the nth position of the array corresponding to the nth position of the bit. Each array element is a 24 bit value.

QuBASE Equivalent.

```
'*************************************************************************
'       Function PQ_BitwiseExtraction
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               iPQAn, Integer
'
'       Output Parameters:
'               oPQExtractedBitsArray, Integer
'
'*************************************************************************
```

4.7.12 PQ_Bitwise AND (24 bits max).

Performs a Bitwise AND on each bit of the two corresponding PQ inputs, and returns a single PQ output with each bit representing the corresponding result. Each PQ variable represents a 24 bit integer.



PQ_A[n] = PQ input representing a 24 bit value.
PQ_B[n] = PQ input representing a 24 bit value.

PQ_A .and. B = PQ output of a bitwise A .and. B, a 24 bit value.

QuBASE Equivalent.

```
'*********************************************************************
'       Function PQ_BitwiseAnd
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqAn, Integer
'               pqBn, Integer
'
'       Output Parameters:
'               oPQYn, Integer
'
'*********************************************************************
```

4.7.13 PQ_Bitwise OR (24 bits max).

Performs a Bitwise OR on each bit of the two corresponding PQ inputs, and returns a single PQ output with each bit representing the corresponding result. Each PQ variable represents a 24 bit integer.

PQ_A[n] = PQ input representing a 24 bit value.
PQ_B[n] = PQ input representing a 24 bit value.

PQ_A .or. B = PQ output of a bitwise A .or. B, a 24 bit value.

QuBASE Equivalent.

```
'*********************************************************************
'       Function PQ_BitwiseOr
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqAn, Integer
'               pqBn, Integer
'
'       Output Parameters:
'               oPQYn, Integer
'
'*********************************************************************
```

4.7.14 PQ_Bitwise XOR (24 bits max).

Performs a Bitwise Exclusive OR on each bit of the two corresponding PQ inputs, and returns a single PQ output with each bit representing the corresponding result. Each PQ variable represents a 24 bit integer.

PQ_A[n] = PQ input representing a 24 bit value.
PQ_B[n] = PQ input representing a 24 bit value.

PQ_A .xor. B = PQ output of a bitwise A .xor. B, a 24 bit value.

QuBASE Equivalent.

```
'*************************************************************************
'       Function PQ_BitwiseXor
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqAn, Integer
'               pqBn, Integer
'
'       Output Parameters:
'               oPQYn, Integer
'
'*************************************************************************
```

4.7.15 PQ_Bitwise NOT (24 bits max).

Performs a Bitwise NOT or inverts the assertion level of each bit of the single PQ input, and returns a single PQ output with each bit representing the corresponding result.

PQ_A[n] = PQ input representing a 24 bit value.

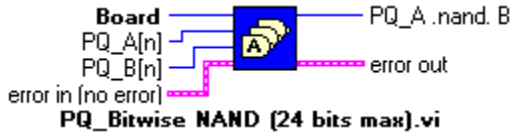
PQ_.not. A = PQ output of a bitwise .not A, a 24 bit value.

QuBASE Equivalent.

```
'*************************************************************************
'       Function PQ_BitwiseNot
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqAn, Integer
'               pqBn, Integer
'
'       Output Parameters:
'               oPQYn, Integer
'
'*************************************************************************
```

4.7.16 PQ_Bitwise NAND (24 bits max).

Performs a Bitwise NAND on each bit of the two corresponding PQ inputs, and returns a single PQ output with each bit representing the corresponding result. Each PQ variable represents a 24 bit integer.



PQ_A[n] = PQ input representing a 24 bit value.
PQ_B[n] = PQ input representing a 24 bit value.

PQ_A .nand. B = PQ output of a bitwise A .nand. B, a 24 bit value.

QuBASE Equivalent.

```
'*********************************************************************
'       Function PQ_BitwiseNand
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqAn, Integer
'               pqBn, Integer
'
'       Output Parameters:
'               oPQYn, Integer
'
'*********************************************************************
```

4.7.17 PQ_Bitwise NOR (24 bits max).

Performs a Bitwise NOR on each bit of the two corresponding PQ inputs, and returns a single PQ output with each bit representing the corresponding result. Each PQ variable represents a 24 bit integer.

PQ_A[n] = PQ input representing a 24 bit value.
PQ_B[n] = PQ input representing a 24 bit value.

PQ_A .nor. B = PQ output of a bitwise A .nor. B, a 24 bit value.

QuBASE Equivalent.

```
'*************************************************************************
'       Function PQ_BitwiseNor
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqAn, Integer
'               pqBn, Integer
'
'       Output Parameters:
'               oPQYn, Integer
'
'*************************************************************************
```

4.7.18 PQ_Bitwise NOT XOR (24 bits max).

Performs a Bitwise NOT Exclusive OR on each bit of the two corresponding PQ inputs, and returns a single PQ output with each bit representing the corresponding result. Each PQ variable represents a 24 bit integer.

PQ_A[n] = PQ input representing a 24 bit value.
PQ_B[n] = PQ input representing a 24 bit value.

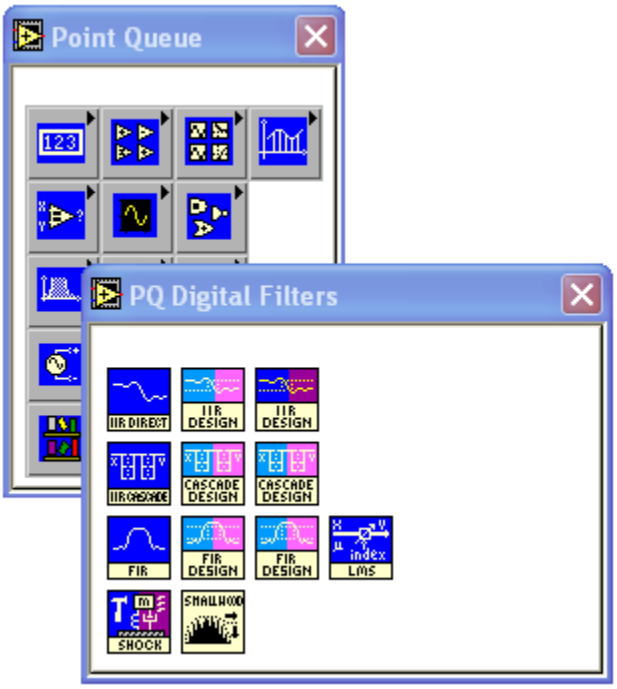
PQ_.not. {A .xor. B} = PQ output of a bitwise A .nexor. B, a 24 bit value.

QuBASE Equivalent.

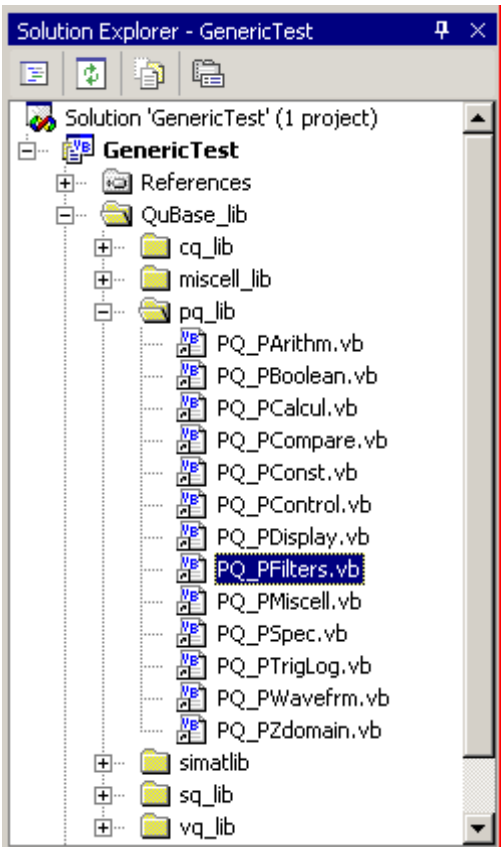
```
'*********************************************************************
'       Function PQ_BitwiseNotXor
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqAn, Integer
'               pqBn, Integer
'
'       Output Parameters:
'               oPQYn, Integer
'
'*********************************************************************
```

4.8 PQ Digital Filters.

These filters derive their coefficients from a separate conventional LabVIEW VI that is fed into a queuing VI. The LabVIEW VI is especially useful for finding coefficients because processing speeds are not as critical since coefficients are usually computed only once, and a large variety of filter functions already exist. It is very convenient to be able to define filter characteristics with a visual display of an impulse response in its time, frequency and phase graphs.

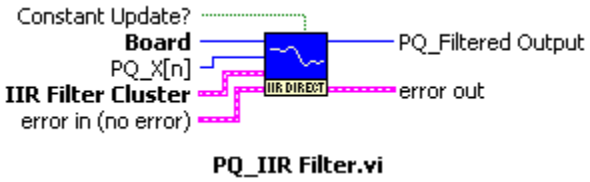


QuBASE Equivalent.



4.8.1 PQ_IIR Filter.

The IIR filters are setup as two separate VIs: a conventional LabVIEW based filter design VI and a DSP PQ_IIR Filter VI. The PQ_IIR Filter VI is implemented on the DSP using the well known summation of two terms or recursive formula. Please refer to the Analysis Library Reference Manual.



PQ_X[n] = PQ input to be filtered.

IIR Filter Cluster = LabVIEW input cluster containing the forward and reverse coefficients. The cluster may be derived by using one of the outputs of the LabVIEW based design VIs.

Constant Update? = LabVIEW boolean input that determines if the LabVIEW variables are to be constantly updated or not. When left in the 'false' position, the LabVIEW constants are only updated once during the queueing phase thereby avoiding DSP-Host communications overhead. When left in the default 'true' position, LabVIEW constants are continuously updated thereby implementing real time variable updates for maximum flexibility.

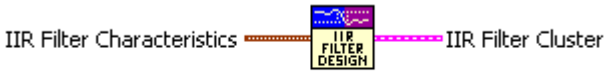
PQ_Filtered Output = PQ output that returns a continuous filtered value without the need for initial conditions, analogous to a conventional analog filter.

QuBASE Equivalent.

```
'*********************************************************************
'       Function PQ_IIRDirectFilter
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               filterCluster, IIRFilterCluster
'               pqXn, Integer
'               constantUpdate, Integer
'
'       Output Parameters:
'               oPQFilteredOutput, Integer
'
'*********************************************************************

   Function PQ_IIRDirectFilter( _
            ByRef board As Integer, _
            ByRef errIn As T_Error, _
            ByRef filterCluster As IIRFilterCluster, _
            Optional ByRef oPQFilteredOutput As Integer = 0, _
            Optional ByRef pqXn As Integer = 0, _
            Optional ByRef constantUpdate As Integer = 1) _
            As T_Error
```

4.8.1.1 SI_IIR Direct Filter Coefficients Design.

To be used as a companion VI with the PQ_IIR Filter VI.



IIR Direct Filter Coefficients Design.vi

IIR Filter Characteristics = LabVIEW input cluster to define the IIR filter characteristics.

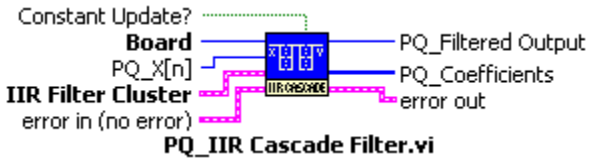
IIR Filter Cluster = LabVIEW cluster containing the forward and reverse coefficients.

QuBASE Equivalent.

```
'*************************************************************************
'       Function SI_IIRDirectFilterCoefficientsDesign
'
'       Description:
'
'       Input Parameters:
'               iFilterCharacteristics, IIRFilterCharacteristics
'
'       Output Parameters:
'               oFilterCluster, IIRFilterCluster
'
'*************************************************************************

  Function SI_IIRDirectFilterCoefficientsDesign( _
    ByVal iFilterCharacteristics As IIRFilterCharacteristics, _
    ByRef oFilterCluster As IIRFilterCluster) _
    As Integer
```

4.8.2 PQ_IIR Cascade Filter.

The IIR filters are setup as two separate VIs: a conventional LabVIEW based filter design VI and a DSP PQ_IIR Cascade Filter VI. The PQ_IIR Cascade Filter VI is implemented on the DSP using the well known summation of two terms or recursive formula. Please refer to the Analysis Library Reference Manual.



PQ_X[n] = PQ input.

IIR Filter Cluster = LabVIEW input cluster containing the forward and reverse coefficients. The cluster may be derived by using one of the outputs of the LabVIEW based design VIs.

Constant Update? = LabVIEW boolean input that determines if the LabVIEW variables are to be constantly updated or not. When left in the 'false' position, the LabVIEW constants are only updated once during the queueing phase thereby avoiding DSP-Host communications overhead. When left in the default 'true' position, LabVIEW constants are continuously updated thereby implementing real time variable updates for maximum flexibility.

PQ_Filtered Output = PQ output that returns a continuous filtered value without the need for initial conditions, analogous to a conventional analog filter.

PQ_Coefficients = PQ output that simply mirrors in the PQ domain the actual coefficients used by the cascade algorithm.

QuBASE Equivalent.

```
'*********************************************************************
'       Function PQ_IIRCascadeFilter
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               iirCluster, IIRFilterCluster
'               pqXn, Integer
'               constantUpdate, Integer
'
'       Output Parameters:
'               oPQFilteredOutput, Integer
'               oPQCoeffs, Integer
'
'*********************************************************************

  Function PQ_IIRCascadeFilter( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef oPQCoeffs() As Integer, _
        ByRef iirCluster As IIRFilterCluster, _
        Optional ByRef oPQFilteredOutput As Integer = 0, _
        Optional ByRef pqXn As Integer = 0, _
        Optional ByVal constantUpdate As Integer = 1) _
        As T_Error
```

4.8.2.1 SI_IIR Cascade Filter Coefficients Design.

To be used as a companion VI with the PQ_IIR Cascade Filter VI.

IIR Cascade Filter Coefficients Design.vi

IIR Filter Characteristics = LabVIEW input cluster to define the IIR filter characteristics.

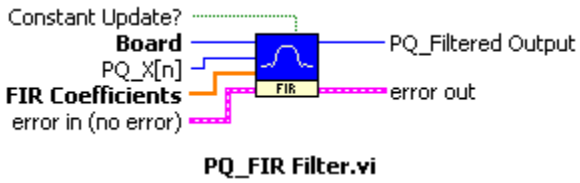
IIR Filter Cluster = LabVIEW cluster containing the forward and reverse coefficients.

QuBASE Equivalent.

```
'*********************************************************************
'       Function SI_IIRCascadeFilterCoefficientsDesign
'
'       Description:
'
'       Input Parameters:
'               iFilterCharacteristics, IIRFilterCharacteristics
'
'       Output Parameters:
'               oFilterCluster, IIRFilterCluster
'
'*********************************************************************

  Function SI_SI_IIRCascadeFilterCoefficientsDesign( _
    ByVal iFilterCharacteristics As IIRFilterCharacteristics, _
    ByRef oFilterCluster As IIRFilterCluster) _
    As Integer
```

4.8.3 PQ_FIR Filter.

The FIR filters are setup as two separate VIs: a conventional LabVIEW based filter design VI and a PQ_FIR Filter VI.
The PQ_FIR Filter VI is implemented on the DSP using the well known finite convolution or non-recursive formula. Please
refer to the Analysis Library Reference Manual.



PQ_X[n] = PQ input.

FIR Filter Coefficients = LabVIEW array containing the FIR coefficients. The array may be derived by using one of the outputs of the LabVIEW based design VIs.

Constant Update? = LabVIEW boolean input that determines if the LabVIEW variables are to be constantly updated or not. When left in the 'false' position, the LabVIEW constants are only updated once during the queueing phase thereby avoiding DSP-Host communications overhead. When left in the default 'true' position, LabVIEW constants are continuously updated thereby implementing real time variable updates for maximum flexibility.

PQ_Filtered Output = PQ output that returns a continuous filtered value without the need for initial conditions, analogous to a conventional analog filter.

QuBASE Equivalent.

```
'*********************************************************************
'       Function PQ_FIRFilter
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               firCoeffs, Double
'               pqXn, Integer
'               constantUpdate, Integer
'
'       Output Parameters:
'               oPQFilteredOutput, Integer
'
'*********************************************************************

   Function PQ_FIRFilter( _
          ByRef board As Integer, _
          ByRef errIn As T_Error, _
          ByRef firCoeffs() As Double, _
          Optional ByRef oPQFilteredOutput As Integer = 0, _
          Optional ByRef pqXn As Integer = 0, _
          Optional ByVal constantUpdate As Integer = 1) _
          As T_Error
```

4.8.3.1 SI_FIR Filter Coefficients Design.

The LabVIEW based PQ_FIR Filter Design VI is merely an example of a way to design and build an array of FIR coefficients. An array of FIR coefficients can be built alternatively by using the FIR filter VIs supplied with the LabVIEW Analysis Library. However, the diagram is much cleaner when used with the supplied single icon PQ_FIR Filter Design VI.

SI_FIR Filter Coefficients Design.vi

FIR Filter Characteristics = LabVIEW input cluster to define the FIR filter characteristics.

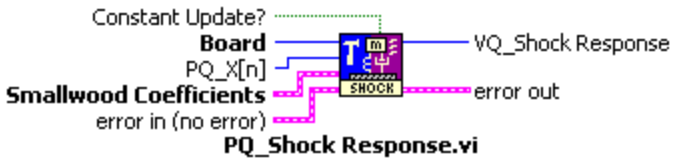
FIR Filter Cluster = LabVIEW output cluster containing the forward coefficients.

QuBASE Equivalent.

```
'**************************************************************************
'       Function SI_FIRFilterCoefficientsDesign
'
'       Description:
'
'       Input Parameters:
'               firFilterCharacteristics, FIRFilterCharacteristics,
'
'       Output Parameters:
'               oFirCoefficients, Double
'
'**************************************************************************

  Function SI_FIRFilterCoefficientsDesign( _
    ByVal firFilterCharacteristics As FIRFilterCharacteristics, _
    ByRef oFirCoefficients() As Double) _
    As Integer
```

4.8.4 PQ_Shock Response.

The Shock Response takes in an array of LabVIEW coefficients, performs a Smallwood filter on many frequencies, and returns the Maxi-Max response for each frequency.



PQ_X[n] = PQ input

Smallwood Coefficients = LabVIEW bundle of two arrays. These arrays are the a and b coefficients for the Smallwood filter. The bundle of coefficients is taken from the Smallwood coefficient generator.

Constant Update? = LabVIEW boolean input that determines if the LabVIEW variables are to be constantly updated or not. When left in the 'false' position, the LabVIEW constants are only updated once during the queueing phase thereby avoiding DSP-Host communications overhead. When left in the default 'true' position, LabVIEW constants are continuously updated thereby implementing real time variable updates for maximum flexibility.

VQ_Y[n] = VQ output. Each point in the outgoing vector corresponds with the array of frequencies wired into the Smallwood Coefficients terminal.

QuBASE Equivalent.

```
'*************************************************************************
'       Function PQ_ShockResponse
'
'       Description:
'
'
'       Input Parameters:
"               board, Integer
'               errIn, T_Error
'               smallwoodAprime, Double
'               smallwoodB, Double
'               pqXn, Integer
'
'       Output Parameters:
'               oVQShockResponse, Integer
'
'*************************************************************************
  Function PQ_ShockResponse( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef smallwoodAprime() As Double, _
        ByRef smallwoodB() As Double, _
        Optional ByRef oVQShockResponse As Integer = 0, _
        Optional ByRef pqXn As Integer = 0) _
        As T_Error
```

4.8.4.1 Smallwood Coefficients.

Used as a companion VI to the PQ_Shock Response VI.

Frequencies = A LabVIEW array of desired frequencies.
Q = LabVIEW input that defines the quantization level.
dT = LabVIEW input defining the sample period of the input in question.

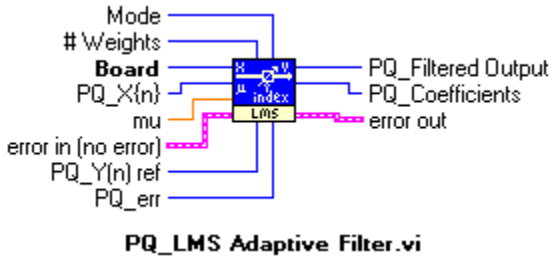
Smallwood Coefficients = A LabVIEW output cluster of the "a" and "b" coefficients for a Smallwood filter, to be wired into the Shock Response VI.

QuBASE Equivalent.

```
'*********************************************************************
'       Function LIB_SmallwoodCoefficients
'
'       Description:
'
'
'       Input Parameters:
'               freq(), Double
'               Q, Double:
'               defaulted to 1
'               dt, Double
'
'               Output Parameters:
'               oSmallwoodAprime, Double
'               oSmallwoodB, Double
'
'*********************************************************************
  Function LIB_SmallwoodCoefficients( _
        ByRef oSmallwoodAprime() As Double, _
        ByRef oSmallwoodB() As Double, _
        ByRef freq() As Double, _
        Optional ByRef Q As Double = 1, _
        Optional ByRef dt As Double = 0) _
        As T_Error
```

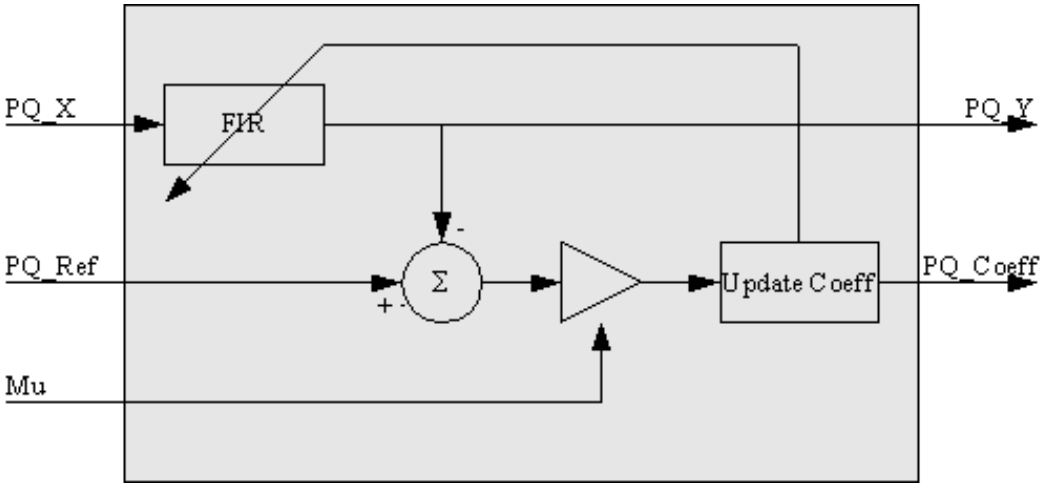
4.8.5 PQ_LMS_Adaptive Filter.

This VI performs an adaptive filtering algorithm using the Least Mean Squared method. The data that is input at PQ_X(n) is passed through the filter created by PQ_Y(n)(ref) and PQ_err. The PQ_coefficients output is an array of values that must be read using the 'DSP READ' function described above.

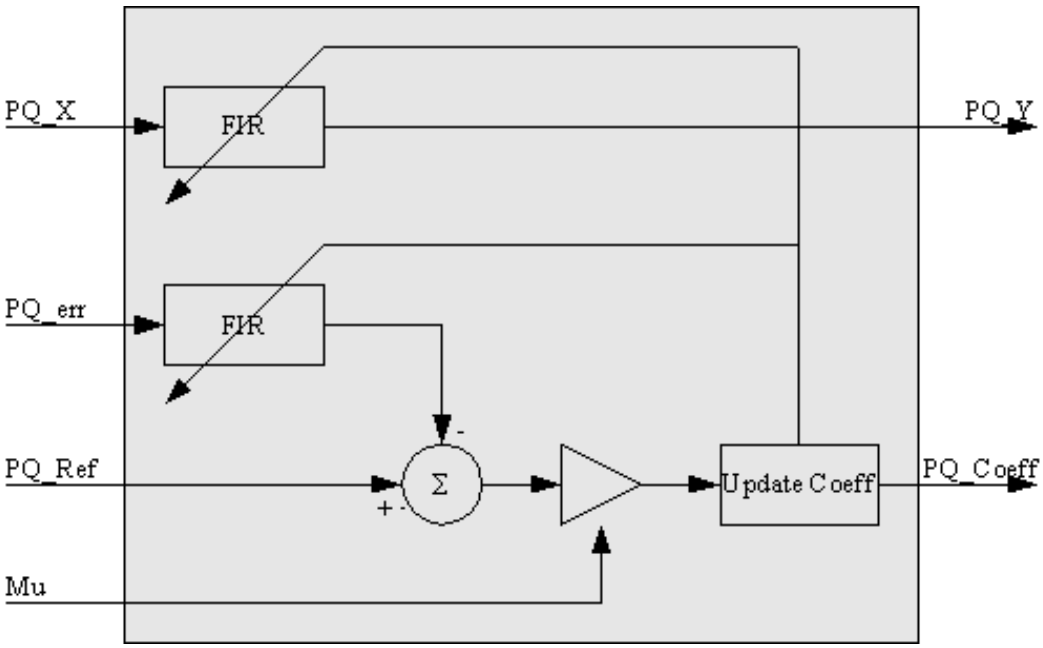


Mode = LabVIEW input, selects the mode of operation.

0 = Single Convolution Ignore PQ_Err: Ignores the PQ_Error input. Below is a block diagram:



1 = Double Convolution Use PQ_Err: Uses the PQ Error input. Below is a block diagram:



 # Weights = LabVIEW input that defines the number of weighting values or coefficients to be used.

PQ_X[n] = PQ Input signal source.

mu = LabVIEW input to define the 'mu' weighting variable.

PQ_Y[n] ref = PQ input, reference source signal.

PQ_err = PQ input, error input.

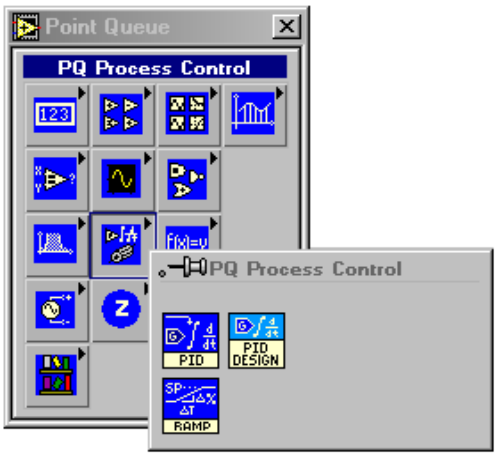
PQ_Filtered Output = PQ output that returns a continuous filtered value without the need for initial conditions, analogous to a conventional analog filter.

PQ_Coefficients = PQ output that simply mirrors in the PQ domain the actual coefficients used by the LMS algorithm.

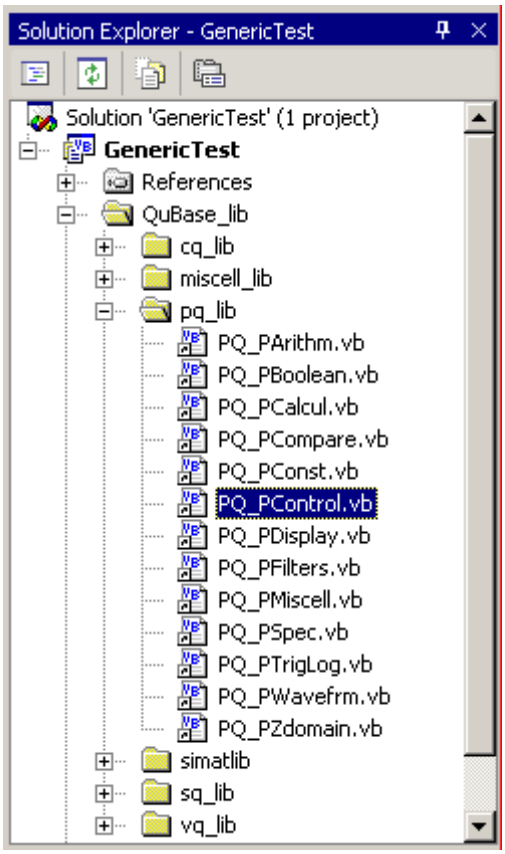
QuBASE Equivalent.

```
'*********************************************************************
'       Function PQ_LMSAdaptiveFilter
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               mode, Integer
'               numberOfWeights, Integer
'               mu, Double
'               pqXn, Integer
'               pqErr, Integer
'               pqYnRef, Integer
'
'       Output Parameters:
'               oPQFilteredOutput, Integer
'               oPQCoefficients, Integer
'
'*********************************************************************

  Function PQ_LMSAdaptiveFilter( _
        ByVal board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef oPQFilteredOutput As Integer = 0, _
        Optional ByRef oPQCoefficients As Integer = 0, _
        Optional ByRef mode As Integer = 0, _
        Optional ByRef numberOfWeights As Integer = 0, _
        Optional ByRef mu As Double = 0, _
        Optional ByRef pqXn As Integer = 0, _
        Optional ByRef pqErr As Integer = 0, _
        Optional ByRef pqYnRef As Integer = 0) _
        As T_Error
```

4.9 PQ Process Control.

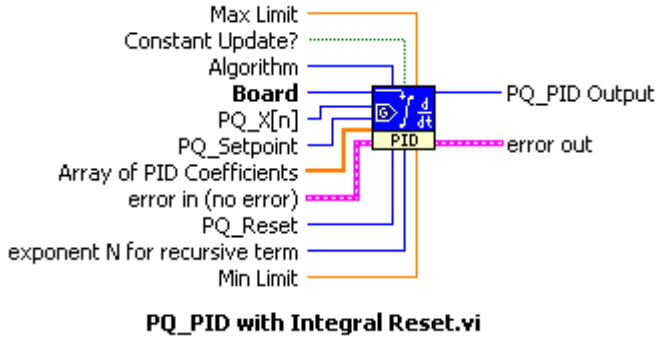


QuBASE Equivalent.



4.9.1 PQ_PID with Integral Reset.

The PQ_PID Controller VI is implemented on the DSP using the well known Proportional Integral Derivative algorithm.



PQ_X[n] = PQ input derived from the controlled system output.

PQ_Setpoint = PQ input used to pass on the desired setpoint of the controlled system in question.

PQ_Reset = PQ input used to clear out all outputs so that computations start from a known point. Useful for restarting a saturated or an uncontrollable loop.

Max Limit = LabVIEW input that limits the maximum output value, used to avoid output saturation if system becomes unstable.

Min Limit = LabVIEW input that limits the minimum output value, used to avoid output saturation if system becomes unstable.

Array of PID Coefficients = LabVIEW array of constants containing the coefficients used to weigh each 'Z' term. These coefficients may be derived by using a LabVIEW based design VI, such as the supplied PID Design VI whose inputs are the constants Kp, Ki, and Kd. ***NOTE:*** *The PID coefficients C1, C2, and C3 are distinct resultant values that are computed from the separate PID constants Kp, Ki, and Kd.*

Algorithm = LabVIEW input that determines if the algorithm is direct or derived from the Z transform.

Direct: If set for the direct algorithm, each term is computed discretely, rendering the same effect as if one were to use discrete function icons to represent each term (Subtraction, Integral, Derivative). This method is recommended for most applications.

Z Transform: If set for the Z transform algorithm, the function uses the equation derived by first taking the classical continuous time domain expression, then transforming it to the continuous frequency domain via the Laplace transform, then to the discrete frequency domain via the Z transform, and finally to the discrete time domain, or f(t) -> f(s) =>> F(z) -> F(n) yields:

$$u[n] = u[n-N] + C1\{e[n]\} + C2\ \{e[n-1]\} + C3\{e[n-2]\}$$

where "N", either a '1' or a '2', represents the exponent of the non weighted 'z' term, which in turn is manifest as the amount of delay in the non weighted recursive term of the discrete time domain equation. This exponent will range from '1' or '2' depending upon the 'z' approximation used when substituting for the 's' variable.

Finally, note that the Kp, Ki, and Kd constants are reevaluated as determined by the sample period to yield the new C[3:1] coefficients.

***NOTE**:*
1) The 'Z transform' algorithm is recommended for advanced users only, where the companion 'PID Design' VI must be used.
2) Please note that there are several approximations for the 'z' domain variable when substituting for Laplace's 's' variable. Please consult the 'PID Design' VI for more details.

Exponent N for recursive term = LabVIEW constant that determines the amount of delay of the non weighted recursive term. The value of '1' or '2' represents the exponent of the non weighted 'z' term, which in turn is manifest as the amount of delay in the non weighted recursive term of the discrete time domain equation. This exponent will range from '1' or '2' depending upon the 'z' approximation used when substituting for the 's' variable.
***NOTE:** Not applicable for the 'Direct' algorithm.*

PQ_PID Output = PQ command output to be sent to the controlled system input.

QuBASE Equivalent.

```
'*************************************************************************
'       Function PQ_LMSAdaptiveFilter
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               mode, Integer
'               numberOfWeights, Integer
'               mu, Double
'               pqXn, Integer
'               pqErr, Integer
'               pqYnRef, Integer
'
'       Output Parameters:
'               oPQFilteredOutput, Integer
'               oPQCoefficients, Integer
'
'*************************************************************************
  Function PQ_PIDWithIntegralReset( _
        ByRef Board As Integer, _
        ByRef errIn As T_Error, _
        ByRef arrayOfPIDConstants() As Double, _
        Optional ByRef oPQPIDOutput As Integer = 0, _
        Optional ByRef pqXn As Integer = 0, _
        Optional ByRef pqSetpoint As Integer = 0, _
        Optional ByRef exponentNforRecursiveTerm As Integer = 0, _
        Optional ByRef pqReset As Integer = 0, _
        Optional ByRef algorithm As Integer = 1, _
        Optional ByRef maxLimit As Double = 0, _
        Optional ByRef minLimit As Double = 0) _
        As T_Error
```

4.9.1.1 PID Design.

Used as a companion VI to the 'PQ_PID with Integral Reset' VI.

PID Design.vi

PID Parameters = LabVIEW input cluster to define the PID characteristics.
Sample Rate: LabVIEW input, required to compute the sample period.
Kp: LabVIEW input, constant that determines the weight of the proportional term.
Ki: LabVIEW input, constant that determines the weight of the integral term.
Kd: LabVIEW input, constant that determines the weight of the derivative term.
Approximation Method: LabVIEW input, selects the 'z' domain approximation when substituting for Laplace's 's' variable. The choices are:
1) None: best used for the direct algorithm
2) Backward Difference: One of three choices to compute the PID constants when using the 'Z Transform' algorithm.
3) Trapezoidal/Tustin/Bilinear: One of three choices to compute the PID constants when using the 'Z Transform' algorithm.
4) Rectangular: One of three choices to compute the PID constants when using the 'Z Transform' algorithm.

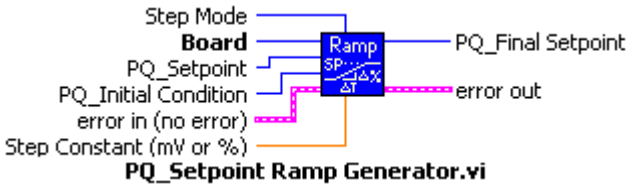
Array of PID Coefficients = LabVIEW output array containing the PID coefficients to be used by the DSP.

QuBASE Equivalent.

```
'*********************************************************************
'       Function LIB_PIDDesign
'
'       Description:
'
'       Input Parameters:
'               pidParameters, T_PIDParams
'
'       Output Parameters:
'               oPidConstantsArray, Double
'
'*********************************************************************

  Function LIB_PIDDesign( _
        ByRef pidParameters As T_PIDParams, _
        ByRef oPidConstantsArray() As Double) _
        As Object
```

4.9.2 PQ_Setpoint Ramp Generator.

The PQ_Setpoint Ramp Generator function converts a step signal to a ramp whose gradient is determined by the Step Constant. When a new setpoint is entered, an undesired transient may take place during the transition with the new setpoint value. The smaller the Step Constant value, the smaller the gradient will be in approaching the final setpoint value.



PQ_Setpoint = PQ input used to pass on the desired setpoint of the controlled system in question.

PQ_Initial Condition = PQ input used to set the initial value from which to start the ramp.

Step Mode = LabVIEW input that determines if the Step Constant is to be expressed as units of mV or a percentage between the initial and final points.

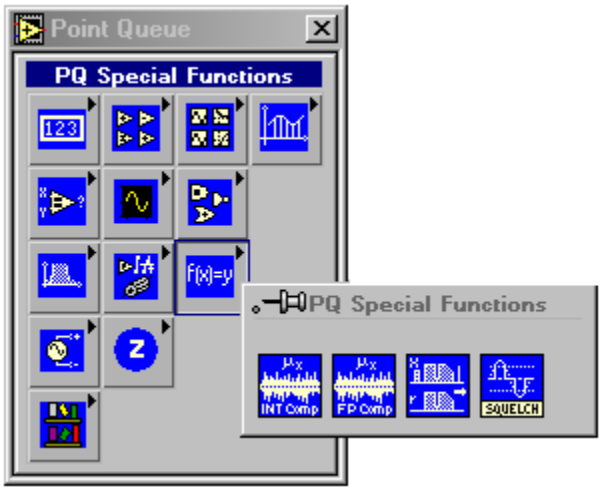
Step Constant = LabVIEW input used to determine the gradient of the slope of the PQ_Final Setpoint output. If the Step Mode is set for mV, each sample will increment or decrement by this mV value until the final setpoint is reached. If the Step Mode is set for percentage, each sample will be incremented to a proportion equal to the percent of the difference between the current or initial value and the final setpoint value.

PQ_Final Setpoint = PQ output used to pass on the ramped setpoint.

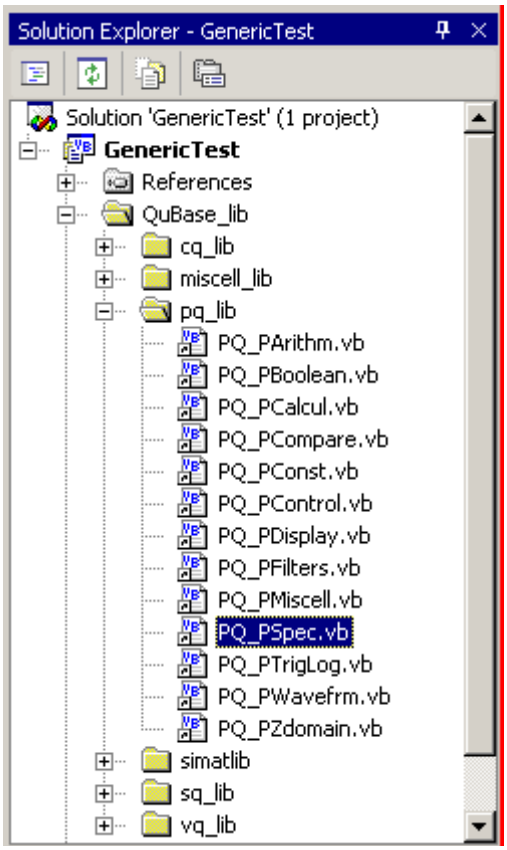
QuBASE Equivalent.

```
'**********************************************************************
'       Function PQ_SetpointRampGenerator
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               stepMode, Integer
'               stepConstant, Double
'               pqSetpoint, Integer
'               pqInitialCondition, Integer
'
'       Output Parameters:
'               oPQFinalSetpoint, Integer
'
'**********************************************************************

  Function PQ_SetpointRampGenerator( _
        ByRef Board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef oPQFinalSetpoint As Integer = 0, _
        Optional ByRef stepMode As Integer = 0, _
        Optional ByRef stepConstant As Double = 0, _
        Optional ByRef pqSetpoint As Integer = 0, _
        Optional ByRef pqInitialCondition As Integer = 0) _
        As T_Error
```

4.10 PQ_Special Functions.

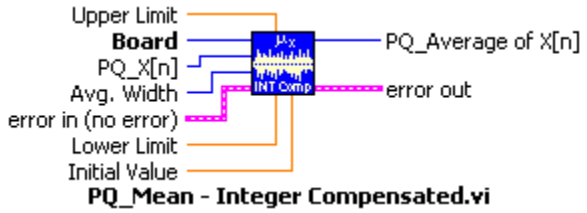


QuBASE Equivalent.



4.10.1 PQ_Mean - Integer Compensated.

The 'PQ_Mean - Integer Compensated' function computes and returns a moving average of the input data points based on the Width variable. Therefore, the moving mean or average output will behave similar to a lowpass filter.



PQ_X[n] = PQ input.

Avg. Width = LabVIEW constant to determine the number of points from which to compute the moving average.

Upper Limit = LabVIEW input, defines the upper limit/clipping value if the 'wind-up' phenomenon is detected.

Lower Limit = LabVIEW input, defines the lower limit/clipping value if the 'wind-up' phenomenon is detected.

Initial Value = LabVIEW input, initial condition.

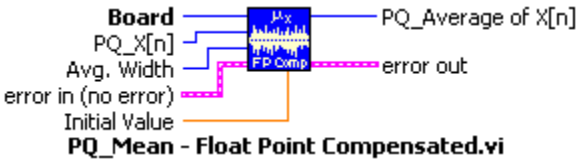
PQ_Average of X[n] = PQ output reflecting the moving average on a block of "N" points.

QuBASE Equivalent.

```
'**********************************************************************
'       Function PQ_MeanIntCompensated
'
'       Description:
'               Moving Average or Mean function, with no long term
'               compensation for wind-up phenomenon except for limit clipping
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqXn, Integer
'               width, Integer
'               initialValue, Double
'               upperLimit, Double
'               lowerLimit, Double
'
'       Output Parameters:
'               oPQAverage, Integer
'
'**********************************************************************

  Function PQ_MeanIntCompensated( _
      ByRef Board As Integer, _
      ByRef errIn As T_Error, _
      Optional ByRef oPQAverage As Integer = 0, _
      Optional ByRef pqXn As Integer = 0, _
      Optional ByRef width As Integer = 0, _
      Optional ByRef initialValue As Double = 0, _
      Optional ByRef upperLimit As Double = 10, _
      Optional ByRef lowerLimit As Double = -10) _
      As T_Error
```

4.10.2 PQ_Mean Float-Point Compensated.

The 'PQ_Mean - Float Point Compensated' VI function computes and returns a moving average of the input data points based on the Width variable. The mantissa portion of the floating point result is compensated so as to nullify the 'wind-up' phenomenon, caused by the limited floating point resolution of the DSP. The moving mean or average output will behave similar to a lowpass filter.



PQ_X[n] = PQ input.
Avg. Width = LabVIEW constant to determine the number of points from which to compute the moving average.
Initial Value = LabVIEW input, initial condition.

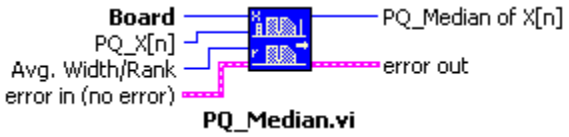
PQ_Average of X[n] = PQ output reflecting the moving average on a block of "N" points.

QuBASE Equivalent.

```
'**********************************************************************
'       Function PQ_MeanFloatCompensated
'
'       Description:
'               Moving Average or Mean function, with compensation imposed
'               on limiting the mantissa against wind-up phenomenon.
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqXn, Integer
'               width, Integer
'               initialValue, Double
'
'       Output Parameters:
'               oPQAverage, Integer
'
'**********************************************************************

Function PQ_MeanFloatCompensated( _
        ByVal Board As Integer, _
        ByVal errIn As T_Error, _
        Optional ByRef oPQAverage As Integer = 0, _
        Optional ByVal pqXn As Integer = 0, _
        Optional ByVal width As Integer = 0, _
        Optional ByVal initialValue As Integer = 0) _
        As T_Error
```

4.10.3 PQ_Median.

The 'PQ_Median' VI function computes and returns a moving median of the input data points based on the Width/Rank variable. The moving median output will behave similar to a lowpass filter.



PQ_X[n] = PQ input.
Avg. Width/Rank = LabVIEW constant to determine the number of points from which to compute the moving median.

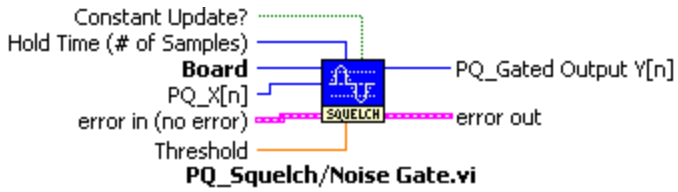
PQ_Median of X[n] = PQ output reflecting the moving average on a block of "N" points.

QuBASE Equivalent.

```
'**********************************************************************
'       Function PQ_Median
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               ipqXn, Integer
'               iWidth, Integer
'
'       Output Parameters:
'               oPQMedian, Integer
'
'**********************************************************************
  Function PQ_Median( _
            ByVal Board As Integer, _
            ByVal errIn As T_Error, _
            ByRef oPQMedian As Integer, _
            ByVal ipqXn As Integer, _
            ByVal iWidth As Integer) _
            As T_Error
```

4.10.4 PQ_Squelch/Noise Gate.

This VI behaves much like a CB radio squelch control. It reduces the output level to 'zero' when the input signal falls below a predefined threshold.



PQ_X(n) = PQ input.

Hold Time = LabVIEW input representing the number of samples to hold the gate open after the input signal falls below the threshold.

Threshold = LabVIEW input, minimum signal level required to open the gate.

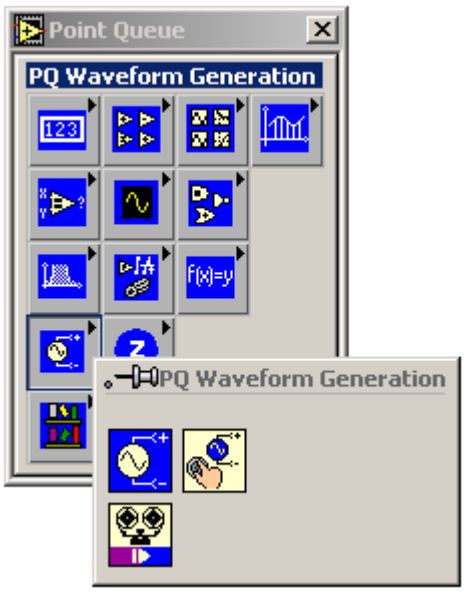
Constant Update? = LabVIEW boolean input that determines if the LabVIEW variables are to be constantly updated or not. When left in the 'false' position, the LabVIEW constants are only updated once during the queueing phase thereby avoiding DSP-Host communications overhead. When left in the default 'true' position, LabVIEW constants are continuously updated thereby implementing real time variable updates for maximum flexibility.

PQ_Gated Output = PQ output.

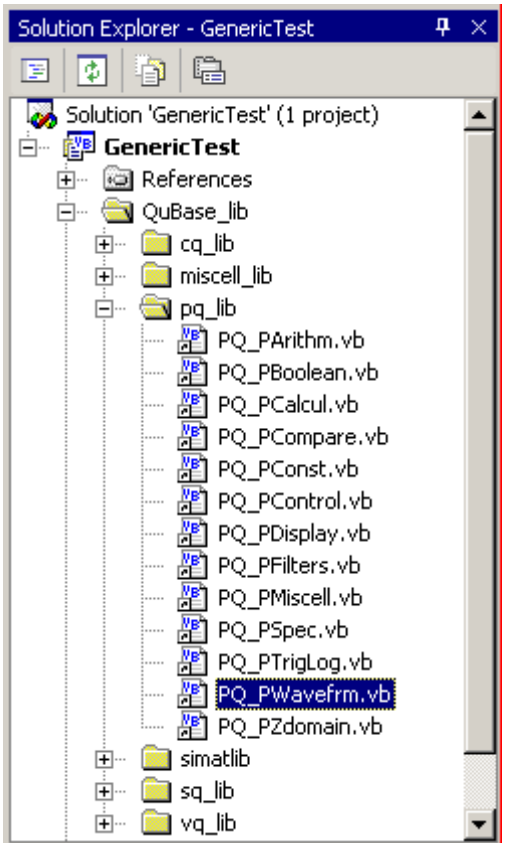
QuBASE Equivalent.

```
'**************************************************************************
'           Function PQ_SquelchNoiseGate
'
'           Description:
'
'           Input Parameters:
'                       board, Integer
'                       errIn, T_Error
'                       pqXn, Integer
'                       holdTime, Integer
'                       threshold, Double
'
'           Output Parameters:
'                       oPQGatedOutput, Integer
'
'**************************************************************************

   Function PQ_SquelchNoiseGate( _
            ByRef Board As Integer, _
            ByRef errIn As T_Error, _
            Optional ByRef oPQGatedOutput As Integer = 0, _
            Optional ByRef pqXn As Integer = 0, _
            Optional ByRef holdTime As Integer = 5, _
            Optional ByRef threshold As Double = -5) _
            As T_Error
```

4.11 Waveform Generation.

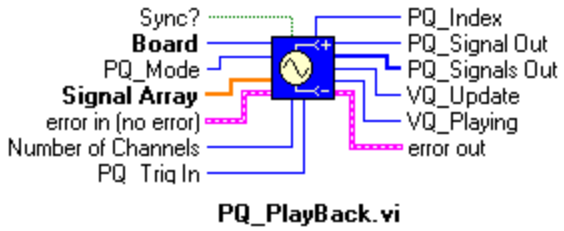


QuBASE Equivalent.



4.11.1 PQ_PlayBack.

The PQ_Playback VI allows the DSP to constantly play back an array of values, useful for the creation of function generators.



Signal Array = LabVIEW array input used to generate a signal derived from a conventional LabVIEW VI, such as the SI_Waveform Design VI.
***NOTE**: As with filter design VIs, the LabVIEW based 'SI_Waveform Design' VI is merely an example of a way to design and build an array containing a conventional signal. An array for any signal can be built alternatively by using the function pattern VIs supplied with the LabVIEW Analysis Library. However, the diagram is much cleaner when used with the supplied single icon 'SI_Waveform Design' VI.*

PQ_Mode = PQ input which selects the output waveform to be continuous or single shot.
Sync? = When this boolean is set to F, the Signal Array will be updated Asynchronously. When TRUE, the VI will wait until the current array has finished playing before updating the waveform. If the output waveform is static, i.e. a sine table, then this should be set to FALSE, which will provide better performance. If the waveform is arbitrary, and constantly changing, i.e. a voice recording, then Sync should be set to TRUE. This can slow down the VI since it must wait until the previous array has finished playing.
PQ_Trig In = PQ input when set to a PQ zero, enables the output to be updated. Any PQ nonzero values disable updating of outputs. Default to PQ zero if not wired.
Number of Channels = Allows for an array of duplicate output waveforms to be made available on the PQ_Signal Out array terminal. Useful in the case where multiple outputs need to be simultaneously reproduced.

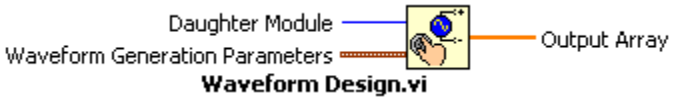
PQ_Index = PQ output that returns the current pointer position of the waveform buffer.
PQ_Signal Out = PQ output used to wire to a source needing a waveform, such as the PQ_Analog Output VI.
PQ_Signals Out = PQ output, similar to PQ_Signal Out, but has has multiple interleaved signals. Useful in designed needing various function generator outputs.
VQ_Update = VQ output used by the PQ_Waveform Playback VI to synchronize the buffer pointer currently being updated, useful for playback of data from a file.
VQ_Playing = VQ output used by the PQ_Waveform Playback VI to synchronize the buffer pointer currently being played back, useful for playback of data from a file.

QuBASE Equivalent.

```
'*************************************************************************
'       Function PQ_Playback
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               signalArray, Double:
'               input/wave pattern
'               sync, Integer
'               pqTrig, Integer
'               pqMode, Integer
'               numberOfChannels, Integer
'
'       Output Parameters:
'               oVQUpdate, Integer
'               oVQPlaying, Integer
'               oPQSignalOut, Integer
'               oPQIndex, Integer
'
'*************************************************************************

   Function PQ_Playback( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef oPQSignalsOut() As Integer, _
        ByRef signalArray() As Double, _
        Optional ByRef oVQUpdate As Integer = 0, _
        Optional ByRef oVQPlaying As Integer = 0, _
        Optional ByRef oPQSignalOut As Integer = 0, _
        Optional ByRef oPQIndex As Integer = 0, _
        Optional ByRef sync As Integer = 0, _
        Optional ByRef pqTrig As Integer = 0, _
        Optional ByRef pqMode As Integer = 0, _
        Optional ByRef numberOfChannels As Integer = 1) _
        As T_Error
```

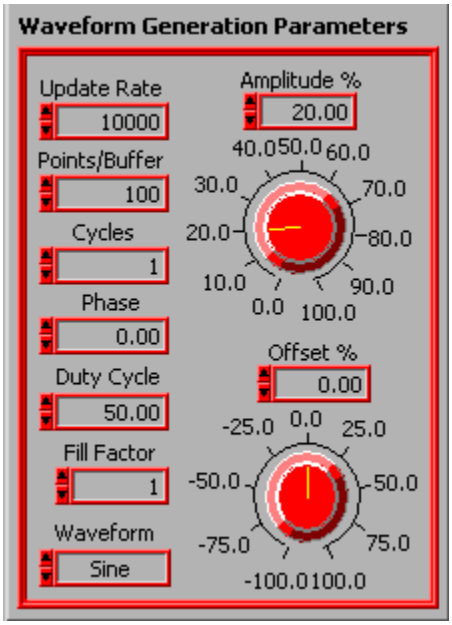
4.11.2 SI_Waveform Design.

To be used as a companion to the PQ_Playback VI.



Output Array = LabVIEW output array containing all the values to build the desired waveform.

Waveform Generation Parameters = A LabVIEW input cluster of Controls Parameters to generate the desired Waveform Array, typical control is shown below:



Parameters included to define a waveform are as follows:

Update Rate = DAC update rate, only used for computational purposes.

Points / Buffer = Number of points to include in the resulting buffer (array) containing the desired waveform.

Cycles (Default 1) = Number of cycles per buffer (repetitions) of a waveform.

Phase (Default 0) = Phase in radians that defines the waveform starting point. Defaulted to a 'zero' value.

Duty Cycle (Default 50) = Ratio of the positive portion of a waveform; only applicable to the SQUARE wave.

Fill Factor (Default 1) = The fill factor inserts "dummy" points or multiples of the same points that define a waveform. Useful when an output is fed back to an input, where it is desired to measure th response of signal of several multiple of output samples. The fill factor effectively increases the buffer size by multiplying it times the buffer size.

Waveform (Default Sine) = Waveform selection, user can select a standard waveform pattern: Sine, Square, Triangle, Sawtooth, Exponent, Sync, Pulse, Noise.

Amplitud % = Amplitude of desired signal.
Offset = Constant biasing value to be added to resulting signal.

QuBASE Equivalent.

```
'*********************************************************************
'       Function SILIB_WaveformDesign
'
'       Description:
'
'       Input Parameters:
'               intModule, Integer
'               updateRate, Integer
'               pointsPerBuffer, Integer
'               cycles, Integer
'               phase, Double
'               dutyCyclePercent, Double
'               waveform, E_WaveformType
'               amplitudePercent, Double
'               offsetPercent, Double
'               fillFactor, Integer
'
'       Output Parameters:
'               oOutputArray(), Double
'               oActualFreq, Double
'
'       Notes:
'               This function is same as SILIB_WaveformDesignNoLocalVars
'
'*********************************************************************

  Function SILIB_WaveformDesign( _
      ByRef oOutputArray() As Double, _
      Optional ByRef intModule As Integer = 2, _
      Optional ByRef updateRate As Integer = 10000, _
      Optional ByRef pointsPerBuffer As Integer = 100, _
      Optional ByRef cycles As Integer = 1, _
      Optional ByRef phase As Double = 0, _
      Optional ByRef dutyCyclePercent As Double = 50, _
      Optional ByRef waveform As E_WaveformType = _
            E_WaveformType.E_Sine, _
      Optional ByRef amplitudePercent As Double = 20, _
      Optional ByRef offsetPercent As Double = 0, _
      Optional ByRef fillFactor As Integer = 1, _
      Optional ByRef oActualFreq As Double = 0) _
      As T_Error
```

4.11.3 PQ_Waveform Playback. (Advanced Users)

This VI is used when creating a non-requeueing VI that employs a PQ_PlayBack function. This VI would normally reside in the LabVIEW While Loop in order to continually update the output waveform. Particularly useful for playback of data from a file in real time without gaps.

VQ_Update = VQ input derived from the PQ_Playback VI, used to synchronize the buffer pointer currently being updated.

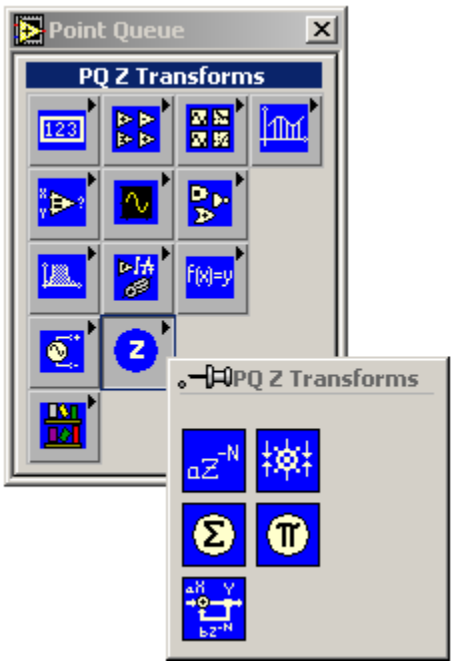
VQ_Playing = VQ input derived from the PQ_Playback VI, used to synchronize the buffer pointer currently being played.

Signal Array = LabVIEW array input used to generate a signal derived from a binary data source such as a conventional LabVIEW VI, or the PQ_Waveform Design VI.

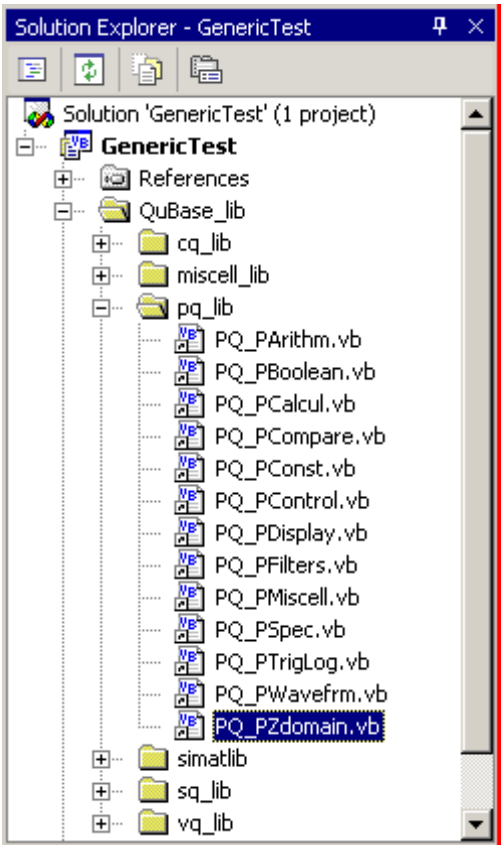
QuBASE Equivalent.

```
'*************************************************************************
'       Function PQ_WaveformPlayback
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               signalArray, Double:
'                       input/wave pattern
'               vqPlaying, Integer
'               vqUpdate, Integer
'
'*************************************************************************

  Function PQ_WaveformPlayback( _
      ByRef board As Integer, _
      ByRef errIn As T_Error, _
      ByRef signalArray() As Double, _
      Optional ByRef vqPlaying As Integer = 0, _
      Optional ByRef vqUpdate As Integer = 0) _
      As T_Error
```

4.12 Z Transforms.

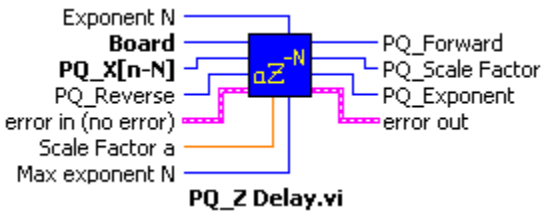


QuBASE Equivalent.



4.12.1 PQ_Z Delay.

The PQ_Z Delay will delay a queuing input by the LabVIEW constant defined by the "N" delay variable. The PQ_Feedback Node VI is used in conjunction with the PQ_Z Delay in order to satisfy LabVIEW's requirement of data dependency. The PQ_Z Delay and PQ_Feedback Node subVIs can be used to form feedback loops when creating a lattice structure. This flexibility is especially useful for performing unconventional transfer functions.



PQ_X[n-N] = PQ input to be delayed by "exponent N" sample periods, which is exclusively derived from the PQ_Feedback Node VI output of the same name. Because LabVIEW does not allow for recursive wiring, this delayed queuing input variable cannot be fed backward and directly into another queuing input.

PQ_Reverse = PQ input to be delayed by "exponent N" sample periods, and used only in the case that the delayed sample is fed forward. In this case, data dependency is maintained and no intermediate VIs are necessary.

Exponent N = LabVIEW input to determine the number of sample periods to delay a PQ input variable.

Scale Factor a = LabVIEW input that directly multiplies the PQ input variable in question.

Max exponent N = LabVIEW input that defines maximum number of delays.

PQ_Forward = PQ output, reflecting the delayed sample input.

PQ_Scale Factor = LabVIEW constant that is multiplied by the "Z" value.

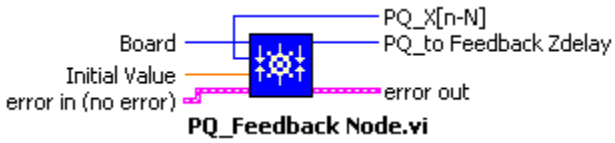
PQ_Exponent = PQ returned exponent. Theoretically it must be same as Exponent N, but because we have a limit (Max exponent N), this output is used as an indicator to the subsequent VIs following as to the N value actually used.

QuBASE Equivalent.

```
'**********************************************************************
'       Function PQ_ZDelay
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               scaleFactorA, Double
'               pqXnMinusN, Integer
'               exponentN, Integer
'               maxExponentN, Integer
'               pqReverse, Integer
'
'       Output Parameters:
'               oPQScaleFactor, Integer
'               oPQExponent, Integer
'               oPQForward, Integer
'
'**********************************************************************

  Function PQ_ZDelay( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef oPQScaleFactor As Integer = 0, _
        Optional ByRef oPQExponent As Integer = 0, _
        Optional ByRef oPQForward As Integer = 0, _
        Optional ByRef pqXnMinusN As Integer = 0, _
        Optional ByRef exponentN As Integer = 0, _
        Optional ByRef maxExponentN As Integer = 0, _
        Optional ByRef pqReverse As Integer = 0, _
        Optional ByRef scaleFactorA As Double = 1) _
        As T_Error
```

4.12.2 PQ_Feedback Node.

Most Z Transform implementations require feedback to represent recursive terms. LabVIEW's graphical compiler does not support direct feedback. In order to overcome this problem, the PQ_Feedback Node VI was created as a companion VI to the PQ_Z Delay VI in backward mode. Therefore, when using the PQ_Z Delay VI in backward mode, a feedback loop can be implemented in conjunction with the companion PQ_Feedback Node VI. To do this, the output terminal of the PQ_Z Delay (PQ_X[n-N]) is wired to the same terminal of the PQ_Feedback Node VI. The PQ_X[n-N] terminal is then wired to any queuing input of a QuVIEW function; typically, the input of the PQ_Summing Node VI.



Initial Value = LabVIEW input that is added to the delayed queuing variable in question.

PQ_X[n-N] = PQ input to be delayed by "exponent N" sample periods, which is exclusively fed into the PQ_Z Delay VIs queuing input of the same name. Because LabVIEW does not allow for recursive wiring, this delayed queuing input variable cannot be fed backward and directly into another queuing input.

PQ_to Feedback Zdelay = PQ output that must be connected to the 'PQ_Reverse' terminal of the 'PQ_Z Delay VI' when normal feedback is needed. This is used as a method to circumvent LabVIEW's restriction on recursive feedback wires.

QuBASE Equivalent.

```
'*********************************************************************
'       Function PQ_FeedbackNode
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               initialValue, Double
'
'       Output Parameters:
'               oPQXnMinusN, Integer
'               oPQFeedbackZDelay, Integer
'
'*********************************************************************

  Function PQ_FeedbackNode( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef oPQXnMinusN As Integer = 0, _
        Optional ByRef oPQFeedbackZDelay As Integer = 0, _
        Optional ByRef initialValue As Double = 0) _
        As T_Error
```

4.12.3 PQ_Summing Node.

PQ_Array of X[i] = PQ input array that is added and returned as a single value. The array format is derived by using a conventional LabVIEW build array icon, whose inputs are wired from one or more PQ output variables.

PQ_Sum of Array = PQ output whose value is the sum of the array of PQ inputs.

QuBASE Equivalent.

```
'**************************************************************************
'       Function PQ_SummingNode
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqArray, Integer
'
'       Output Parameters:
'               oPQSumOfArray, Integer
'
'**************************************************************************

  Function PQ_SummingNode( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef pqArray() As Integer, _
        Optional ByRef oPQSumOfArray As Integer = 0) _
        As T_Error
```

4.12.4 PQ_Multiplying Node.

PQ_Array of X[i] = PQ input array whose elements are multiplied and returned as a single value, similar in concept to a factorial. The array format is derived by using a conventional LabVIEW build array icon, whose inputs are wired from one or more PQ output variables.

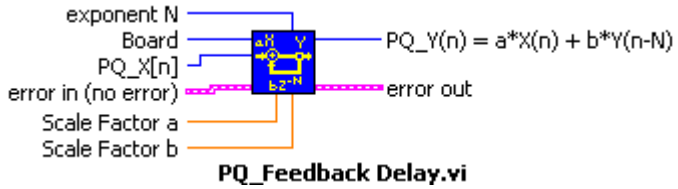
PQ_Product of Array = PQ output whose value is the product of the array of PQ inputs.

QuBASE Equivalent.

```
'************************************************************************
'       Function PQ_MultiplyingNode
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqArray, Integer
'
'       Output Parameters:
'               oPQProductOfArray, Integer
'
'************************************************************************

  Function PQ_MultiplyingNode( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef pqArray() As Integer, _
        Optional ByRef oPQProductOfArray As Integer = 0) _
        As T_Error
```

4.12.5 PQ_Feedback Delay.

The PQ_Feedback Delay VI will delay a queuing output by the LabVIEW constant defined by the "N" delay variable. The reason for having an extra VI perform this function is because LabVIEW forces data dependency on the program structure. Therefore, a delayed output cannot be added onto itself unless this unique VI is used.



PQ_X[n] = PQ input to be delayed by "N" sample periods.

Exponent N = LabVIEW input to determine the number of sample periods to delay a PQ input variable.

Scale Factor a = LabVIEW input that directly multiplies the PQ input variable in question.

Scale Factor b = LabVIEW input that is added to the result of the product of 'a * PQ_X[n]'.

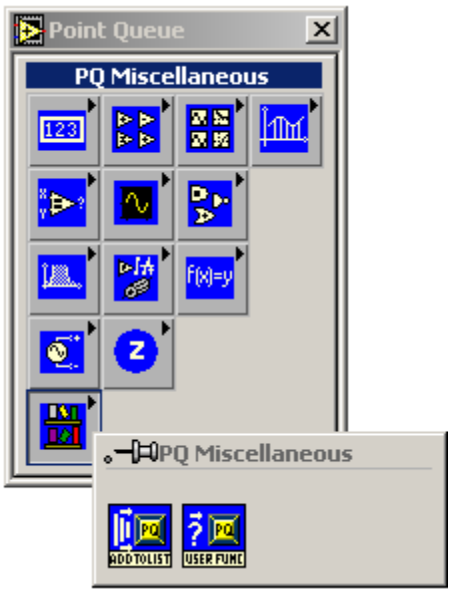
PQ_Y[n] = a*X[n] + b*Y[n-N] = PQ output.

QuBASE Equivalent.

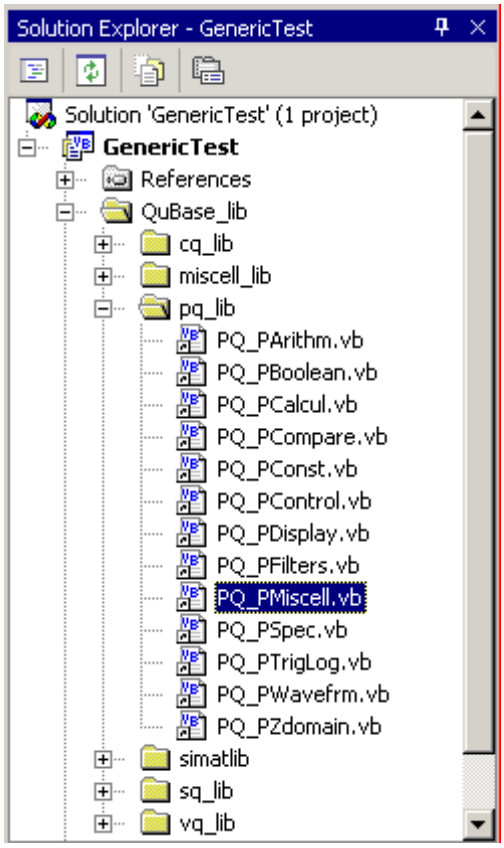
```
'*********************************************************************
'       Function PQ_FeedbackDelay
'
'       Description:
'               Y(n) = a*X(n) + b*Y(n-N)
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqXn, Integer
'               exponentN, Integer
'               scaleFactorA, Double
'               scaleFactorB, Double
'
'       Output Parameters:
'               oPQYn, Integer
'
'*********************************************************************

  Function PQ_FeedbackDelay( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef oPQYn As Integer = 0, _
        Optional ByRef pqXn As Integer = 0, _
        Optional ByRef exponentN As Integer = 0, _
        Optional ByRef scaleFactorA As Double = 1, _
        Optional ByRef scaleFactorB As Double = 0.1) _
        As T_Error
```

4.13 PQ Miscellaneous

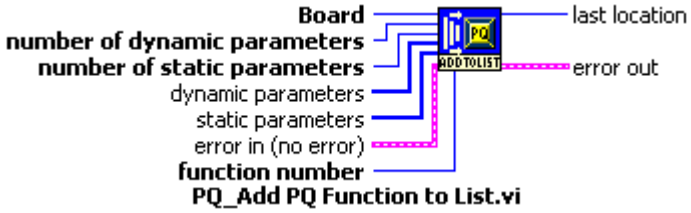
These VIs are not intended for users, and are nothing more than building block functions for QuVIEW.



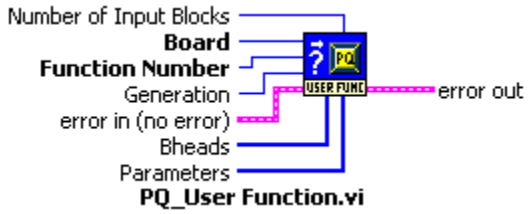
QuBASE Equivalent.



4.13.1 PQ_Add PQ Function to List

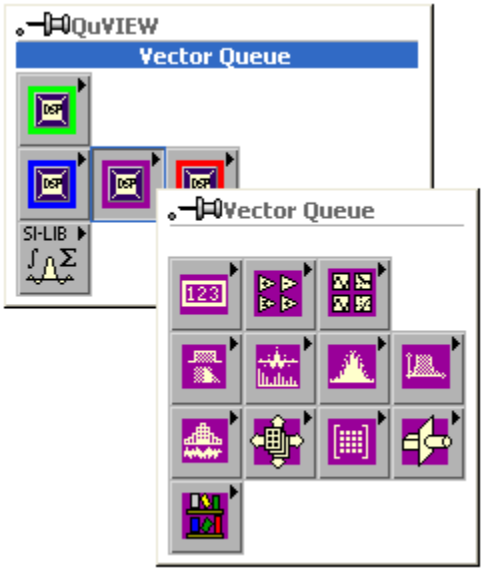


4.13.2 PQ_User Function.

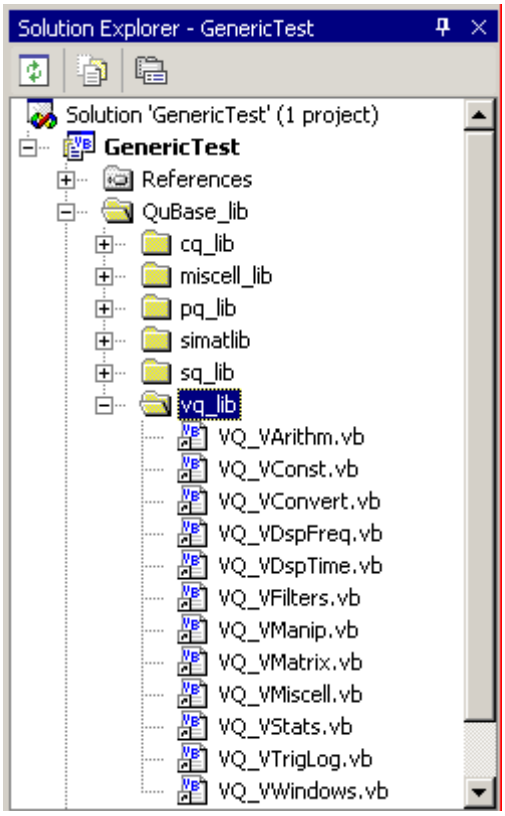


5.0 Vector Queuing VIs.

These VIs incorporate DSP functionality when a block of data conversions/update cycles occurs for one or more channels of inputs and/or outputs. They are to be used when DSP computation is required on a block by block basis. Each block is formed by accumulating a set of pre-determined points,



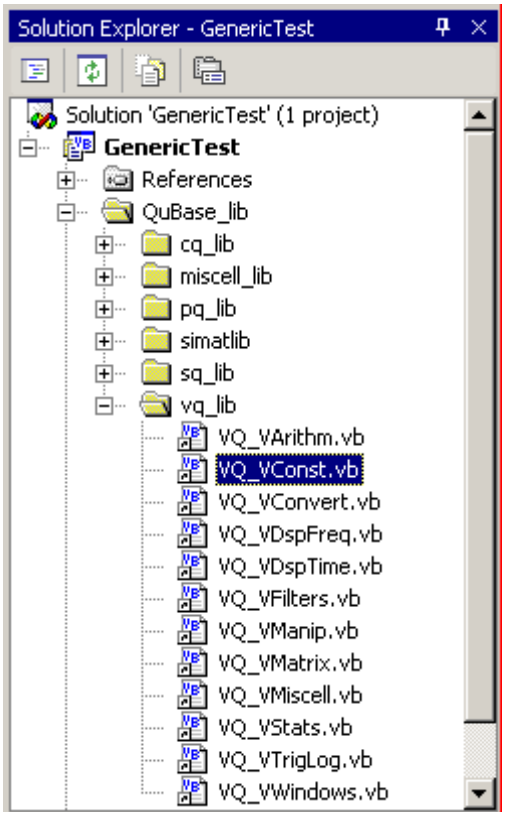
QuBASE Equivalent.



5.1 VQ Constants.

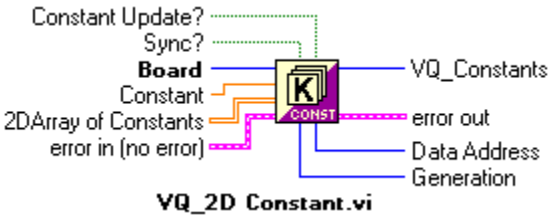
These VIs are very useful because the interconnecting wires in some of the queuing VIs pass over "queuing variables" instead of direct LabVIEW numbers. With this in mind, the VQ_Constants will take any any regular LabVIEW numeric constant array, translate it to a vector queuing array, and download it to the DSP so it may be used within the DSP domain. For example, some queuing VIs have two "VQ" inputs, where on eof them may require a constant array.

QuBASE Equivalent.



5.1.1 VQ_2D Constants.

Transfers a 2D LabVIEW array of constants to the DSP.



Constant Update? = LabVIEW boolean input that determines if the LabVIEW variables are to be constantly updated or not. When left in the 'false' position, the LabVIEW constants are only updated once during the queueing phase thereby avoiding DSP-Host communications overhead. When left in the default 'true' position, LabVIEW constants are continuously updated thereby implementing real time variable updates for maximum flexibility.

Sync? = LabVIEW boolean input, indicates if execution is to be carried out Asynchronously, regardless if data is updated from the host.

Constant = LabVIEW input, a single constant value. The value wired to this terminal is used by default if the '2D Array of Constants' is left unwired

2D Array of Constants = LabVIEW input, a 2D array of constants. The array wired to this terminal overrides the single value wired into the 'Constant' terminal.

VQ_Constants = VQ output containing the constants wired in from LabVIEW.

Data Address = LabVIEW output, a DSP pointer indicating where the input data was allocated in the DSP. Intended for advanced users only.

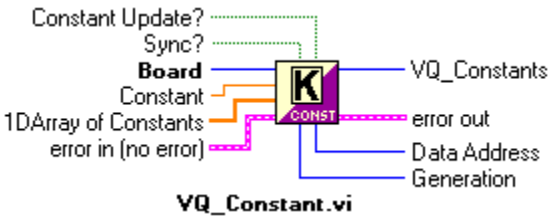
Generation = LabVIEW output, a generation number of the data block allocated, used by other VQ's for synchronization. Intended for advanced users only.

QuBASE Equivalent.

```
'*********************************************************************
'       Function VQ_2DConstant
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               consts, Double
'               sync, Integer
'
'       Output Parameters:
'               oDataAddress, Integer
'               oGeneration, Integer
'               constantUpdate, Integer
'
'*********************************************************************

   Function VQ_2DConstant( _
          ByRef board As Integer, _
          ByRef errIn As T_Error, _
          ByRef consts(,) As Double, _
          Optional ByRef oVQConstants As Integer = 0, _
          Optional ByRef oDataAddress As Integer = 0, _
          Optional ByRef oGeneration As Integer = 0, _
          Optional ByRef sync As Integer = 1, _
          Optional ByRef constantUpdate As Integer = 0) _
          As T_Error
```

6.1.2 VQ_Constant.

Transfers a 1D LabVIEW array of constants to the DSP.



Constant Update? = LabVIEW boolean input that determines if the LabVIEW variables are to be constantly updated or not. When left in the 'false' position, the LabVIEW constants are only updated once during the queueing phase thereby avoiding DSP-Host communications overhead. When left in the default 'true' position, LabVIEW constants are continuously updated thereby implementing real time variable updates for maximum flexibility.

Sync? = LabVIEW boolean input, indicates if execution is to be carried out Asynchronously, regardless if data is updated from the host.

Constant = LabVIEW input, a single constant value. The value wired to this terminal is used by default if the '1D Array of Constants' is left unwired

1D Array of Constants = LabVIEW input, a 1D array of constants. The array wired to this terminal overrides the single value wired into the 'Constant' terminal.

VQ_Constants = VQ output containing the constants wired in from LabVIEW.

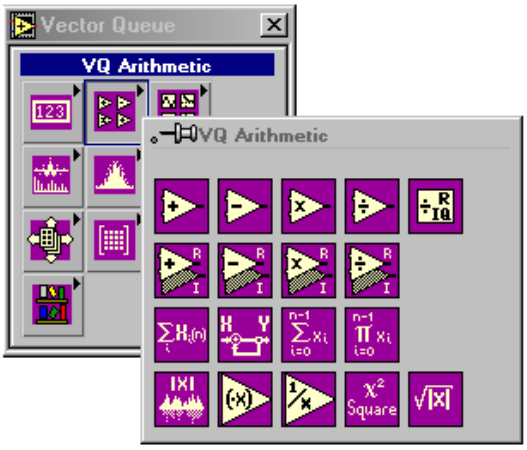
Data Address = LabVIEW output, a DSP pointer indicating where the input data was allocated in the DSP. Intended for advanced users only.

Generation = LabVIEW output, a generation number of the data block allocated, used by other VQ's for synchronization. Intended for advanced users only.

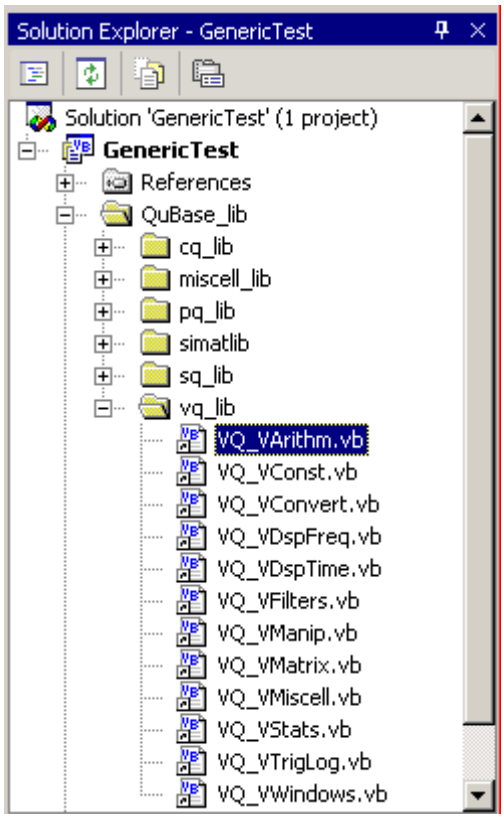
QuBASE Equivalent.

```
'*********************************************************************
'       Function VQ_Constant
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               ieeeConstants, Double
'               sync, Integer
'
'       Output Parameters:
'               oDataAddress, Integer
'               oGeneration, Integer
'               constantUpdate, Integer
'
'*********************************************************************
  Function VQ_Constant( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef ieeeConstants() As Double, _
        Optional ByRef oVQConstants As Integer = 0, _
        Optional ByRef oDataAddress As Integer = 0, _
        Optional ByRef oGeneration As Integer = 0, _
        Optional ByRef constantUpdate As Integer = 1, _
        Optional ByRef sync As Integer = 0) _
        As T_Error
```

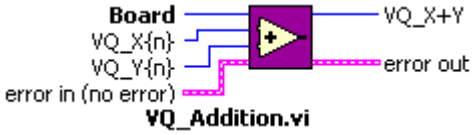
5.2 VQ Arithmetic Functions.



QuBASE Equivalent.



5.2.1 VQ_Addition.



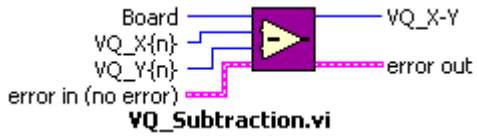
VQ_X{n} = VQ input.
VQ_Y{n} = VQ input.

VQ_X+Y = VQ output, sum of X{n} and Y{n}.

QuBASE Equivalent.

```
'**********************************************************************
'       Function VQ_Addition
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'               vqYn, Integer
'
'       Output Parameters:
'               oVQXPlusY, Integer
'
'**********************************************************************

  Function VQ_Addition( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef vqXn As Integer = 0, _
        Optional ByRef vqYn As Integer = 0, _
        Optional ByRef oVQXPlusY As Integer = 0) _
        As T_Error
```

5.2.2 VQ_Subtraction.



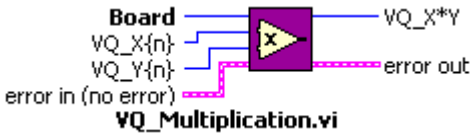
VQ_X{n} = VQ input.
VQ_Y{n} = VQ input.

VQ_X-Y = VQ output, difference of X{n} and Y{n}.

QuBASE Equivalent.

```
'**********************************************************************
'       Function VQ_Subtraction
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'               vqYn, Integer
'
'       Output Parameters:
'               oVQXMinusY, Integer
'
'**********************************************************************

  Function VQ_Subtraction( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef vqXn As Integer = 0, _
        Optional ByRef vqYn As Integer = 0, _
        Optional ByRef oVQXMinusY As Integer = 0) _
        As T_Error
```

5.2.3 VQ_Multiplication.



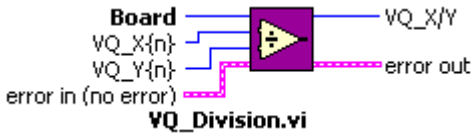
VQ_X{n} = VQ input.
VQ_Y{n} = VQ input.

VQ_X*Y = VQ output, product of X{n} and Y{n}.

QuBASE Equivalent.

```
'**********************************************************************
'       Function VQ_Multiplication
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'               vqYn, Integer
'
'       Output Parameters:
'               oVQXTimesY, Integer
'
'**********************************************************************
  Function VQ_Multiplication( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef vqXn As Integer = 0, _
        Optional ByRef vqYn As Integer = 0, _
        Optional ByRef oVQXTimesY As Integer = 0) _
        As T_Error
```

5.2.4 VQ_Division.



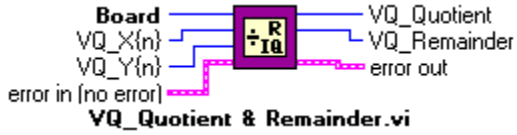
VQ_X{n} = VQ input.
VQ_Y{n} = VQ input.

VQ_X/Y = VQ output, ratio of X{n} over Y{n}.

QuBASE Equivalent.

```
'**********************************************************************
'       Function VQ_Division
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'               vqYn, Integer
'
'       Output Parameters:
'               oVQXDivideByY, Integer
'
'**********************************************************************

  Function VQ_Division( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef vqXn As Integer = 0, _
        Optional ByRef vqYn As Integer = 0, _
        Optional ByRef oVQXDivideByY As Integer = 0) _
        As T_Error
```

5.2.5 VQ_Quotient & Remainder.



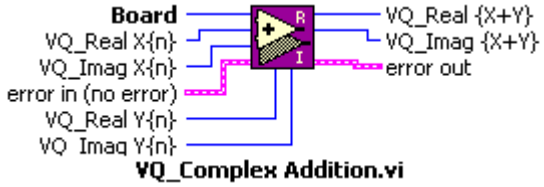
VQ_X{n} = VQ input.
VQ_Y{n} = VQ input.

VQ_Quotient = VQ output, quotient from the ratio of X{n} over Y{n}.
VQ_Remainder = VQ output, remainder from the ratio of X{n} over Y{n}.

QuBASE Equivalent.

```
'**************************************************************************
'       Function VQ_QuotientRemainder
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqX, Integer
'               vqY, Integer
'
'       Output Parameters:
'               oVQQuotient, Integer
'               oVQRemainder, Integer
'
'**************************************************************************

  Function VQ_QuotientRemainder( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef vqX As Integer = 0, _
        Optional ByRef vqY As Integer = 0) _
        Optional ByRef oVQQuotient As Integer = 0, _
        Optional ByRef oVQRemainder As Integer = 0, _
        As T_Error
```

5.2.6 VQ_Complex Addition.



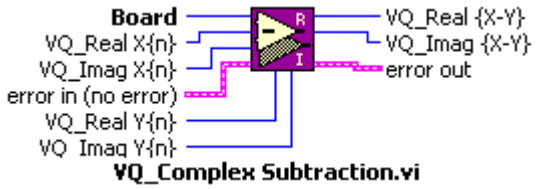
VQ_Real X{n} = VQ input.
VQ_Imag X{n} = VQ input.
VQ_Real Y{n} = VQ input.
VQ_Imag Y{n} = VQ input.

VQ_Real X+Y = VQ output, sum of real parts of X{n} and Y{n}.
VQ_Imag X+Y = VQ output, sum of imaginary parts of X{n} and Y{n}.

QuBASE Equivalent.

```
'************************************************************************
'       Function VQ_ComplexAddition
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXnReal, Integer
'               vqXnImag, Integer
'               vqYnReal, Integer
'               vqYnImag, Integer
'
'       Output Parameters:
'               oVQReal, Integer
'               oVQImag, Integer
'
'************************************************************************

  Function VQ_ComplexAddition( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef vqXnReal As Integer = 0, _
        Optional ByRef vqXnImag As Integer = 0, _
        Optional ByRef vqYnReal As Integer = 0, _
        Optional ByRef vqYnImag As Integer = 0, _
        Optional ByRef oVQReal As Integer = 0, _
        Optional ByRef oVQImag As Integer = 0) _
        As T_Error
```

5.2.7 VQ_Complex Subtraction.



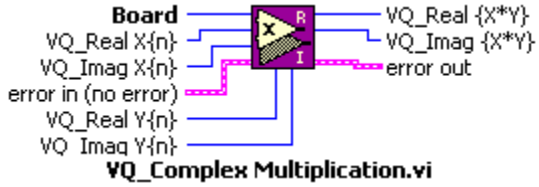
VQ_Real X{n} = VQ input.
VQ_Imag X{n} = VQ input.
VQ_Real Y{n} = VQ input.
VQ_Imag Y{n} = VQ input.

VQ_Real X-Y = VQ output, difference of real parts of X{n} and Y{[n}.
VQ_Imag X-Y = VQ output, difference of imaginary parts of X{n} and Y{n}.

QuBASE Equivalent.

```
'**********************************************************************
'       Function VQ_ComplexSubtraction
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXnReal, Integer
'               vqXnImag, Integer
'               vqYnReal, Integer
'               vqYnImag, Integer
'
'       Output Parameters:
'               oVQReal, Integer
'               oVQImag, Integer
'
'**********************************************************************

  Function VQ_ComplexSubtraction( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef vqXnReal As Integer = 0, _
        Optional ByRef vqXnImag As Integer = 0, _
        Optional ByRef vqYnReal As Integer = 0, _
        Optional ByRef vqYnImag As Integer = 0, _
        Optional ByRef oVQReal As Integer = 0, _
        Optional ByRef oVQImag As Integer = 0) _
        As T_Error
```

5.2.8 VQ_Complex Multiplication.



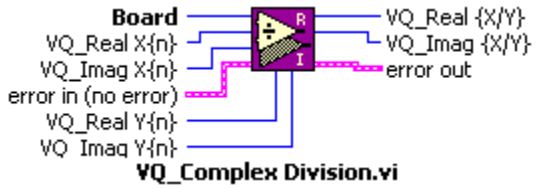
VQ_Real X{n} = VQ input.
VQ_Imag X{n} = VQ input.
VQ_Real Y{n} = VQ input.
VQ_Imag Y{n} = VQ input.

VQ_Real X*Y = VQ output, real part of complex multiplication of X{n} and Y{n}.
VQ_Imag X*Y = VQ output, imaginary part of complex multiplication of X{n} and Y{n}.

QuBASE Equivalent.

```
'**********************************************************************
'       Function VQ_ComplexMultiplication
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXnReal, Integer
'               vqXnImag, Integer
'               vqYnReal, Integer
'               vqYnImag, Integer
'
'       Output Parameters:
'               oVQReal, Integer
'               oVQImag, Integer
'
'**********************************************************************

  Function VQ_ComplexMultiplication( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef vqXnReal As Integer = 0, _
        Optional ByRef vqXnImag As Integer = 0, _
        Optional ByRef vqYnReal As Integer = 0, _
        Optional ByRef vqYnImag As Integer = 0, _
        Optional ByRef oVQReal As Integer = 0, _
        Optional ByRef oVQImag As Integer = 0) _
        As T_Error
```

5.2.9 VQ_Complex Division.



VQ_Real X{n} = VQ input.
VQ_Imag X{n} = VQ input.
VQ_Real Y{n} = VQ input.
VQ_Imag Y{n} = VQ input.

VQ_Real X/Y = VQ output, real part of complex division of X{n} and Y{n}.
VQ_Imag X/Y = VQ output, imaginary part of complex division of X{n} and Y{n}.

QuBASE Equivalent.

```
'**********************************************************************
'       Function VQ_ComplexDivision
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXnReal, Integer
'               vqXnImag, Integer
'               vqYnReal, Integer
'               vqYnImag, Integer
'
'       Output Parameters:
'               oVQReal, Integer
'               oVQImag, Integer
'
'**********************************************************************

  Function VQ_ComplexDivision( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef vqXnReal As Integer = 0, _
        Optional ByRef vqXnImag As Integer = 0, _
        Optional ByRef vqYnReal As Integer = 0, _
        Optional ByRef vqYnImag As Integer = 0, _
        Optional ByRef oVQReal As Integer = 0, _
        Optional ByRef oVQImag As Integer = 0) _
        As T_Error
```

5.2.10 VQ_Vector Summation.

This VI returns a vector, VQ_Sum Xi{n}, equal in size to each of the vectors in the input array, VQ_Array of X{n}, which are also of the same size. Each i'th element of the result is the sum of the respective i'th elements of the input vectors; i.e., the 3rd element of VQ_Sum Xi{n} is the sum of the 3rd elements of all of the input VQ_X{n} vectors.

VQ_Array of X{n} = VQ input array.

VQ_Sum Xi{n} = VQ output whose values are the sum of each corresponding input vector element.

QuBASE Equivalent.

```
'************************************************************************
'       Function VQ_VectorSummation
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqArrayOfXn, Integer
'
'       Output Parameters:
'               oVQSum, Integer
'
'************************************************************************

  Function VQ_VectorSummation( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef vqArrayOfXn() As Integer, _
        Optional ByRef oVQSum As Integer = 0) _
        As T_Error
```

5.2.11 VQ_Recursive Vector Summation.

This VI returns a vector, VQ_Yi{n}, equal in size to the input vector , VQ_Xi{n}. Each i'th element of the result is the sum of the respective i'th elements of the input vector and the previous result; i.e., the 3rd element of VQ_Yi{n} is the sum of the 3rd element of the input vector VQ_Xi{n} and the 3rd element of the previous output vector VQ_Y(i-1){n}. This algorithm is especially useful when trying to find the result of recursive averages over a set of data blocks.

VQ_Xi{n} = VQ input.

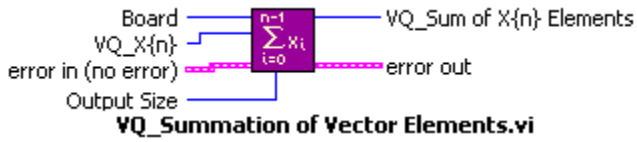
VQ_Yi{n} = Xi{n} + Yi-1{n} = VQ output whose value is the compounding sum of the current input vector elements and the sum of the previous input vector elements.

QuBASE Equivalent.

```
'*************************************************************************
'       Function VQ_RecursiveVectorSummation
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'
'       Output Parameters:
'               oVQSum, Integer
'
'*************************************************************************

  Function VQ_RecursiveVectorSummation( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef vqXn As Integer = 0, _
        Optional ByRef oVQSum As Integer = 0) _
        As T_Error
```

5.2.12 VQ_Summation of Vector Elements.

This VI returns a vector, VQ_Sum of X{n} Elements, equal in size to the LabVIEW input, Output Size. Each i'th element of the result is the sum of all of the elements of a corresponding input vector; i.e., the 3rd element of VQ_Sum of X{n} Elements is the sum of all of the elements of the 3rd input vector VQ_X{n}. This algorithm is especially useful when trying to find a trend or behavior over a long period of time or large set of data blocks.



VQ_X{n} = VQ input.

Output Size = LabVIEW input. Specifies the output vector size such that each element of the VQ output is the resultant sum of each input vector element.

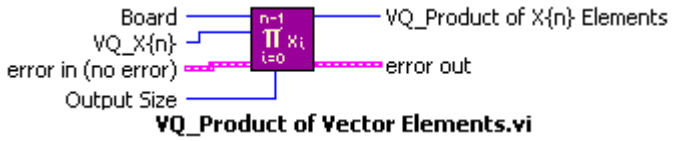
VQ_Sum of X{n} Elements = VQ output whose individual values are the sum of each input vector's elements. The overall length of the output vector array, specified by the LabVIEW input Output Size, represents the overall number of vectors used for the computation.

QuBASE Equivalent.

```
'**************************************************************************
'       Function VQ_SummationOfVectorElements
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'               outputSize, Integer
'
'       Output Parameters:
'               oVQSum, Integer
'
'**************************************************************************

  Function VQ_SummationOfVectorElements( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef vqXn As Integer = 0, _
        Optional ByRef outputSize As Integer = 1, _
        Optional ByRef oVQSum As Integer = 0) _
        As T_Error
```

5.2.13 VQ_Product of Vector Elements.

This VI returns a vector, VQ_Product of X{n} Elements, equal in size to the LabVIEW input, Output Size. Each i'th element of the result is the product of all of the elements of a corresponding input vector; i.e., the 3rd element of VQ_Product of X{n} Elements is the product of all of the elements of the 3rd input vector VQ_X{n}. This algorithm is especially useful when trying to find a trend or behavior over a long period of time or large set of data blocks.



VQ_X{n} = VQ input.

Output Size = LabVIEW input. Specifies the output vector size such that each element of the VQ output is the resultant product of each input vector element.

VQ_Product of X{n} Elements = VQ output whose individual values are the product of each input vector's elements. The overall length of the output vector array, specified by the LabVIEW input Output Size, represents the overall number of vectors used for the computation.

QuBASE Equivalent.

```
'**************************************************************************
'       Function VQ_ProductOfVectorElements
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'               outputSize, Integer
'
'       Output Parameters:
'               oVQProduct, Integer
'
'**************************************************************************

  Function VQ_ProductOfVectorElements( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef vqXn As Integer = 0, _
        Optional ByRef outputSize As Integer = 1, _
        Optional ByRef oVQProduct As Integer = 0) _
        As T_Error
```

5.2.14 VQ_Absolute Value.

VQ_X{n} = VQ input.

VQ_|X{n}| = VQ output, absolute value of X{n}.

QuBASE Equivalent.

```
'*********************************************************************
'       Function VQ_AbsoluteValue
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'
'       Output Parameters:
'               oVQXn, Integer
'
'*********************************************************************

  Function VQ_AbsoluteValue( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef vqXn As Integer = 0, _
        Optional ByRef oVQXn As Integer = 0) _
        As T_Error
```

5.2.15 VQ_Negate.

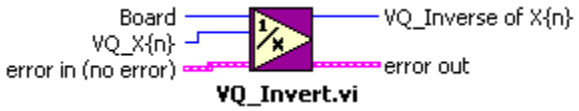
VQ_X{n} = VQ input.

VQ_-(X{n}) = VQ output, negative of X{n}.

QuBASE Equivalent.

```
'**********************************************************************
'       Function VQ_Negate
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'
'       Output Parameters:
'               oVQXn, Integer
'
'**********************************************************************

  Function VQ_Negate( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef vqXn As Integer = 0, _
        Optional ByRef oVQXn As Integer = 0) _
        As T_Error
```

5.2.16 VQ_Invert.



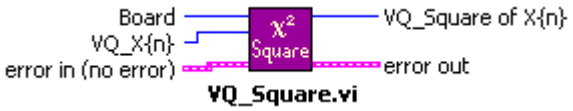
VQ_X{n} = VQ input.

VQ_Inverse of X{n} = VQ output, inverse of X{n}.

QuBASE Equivalent.

```
'*************************************************************************
'       Function VQ_Inverse
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'
'       Output Parameters:
'               oVQXn, Integer
'
'*************************************************************************

   Function VQ_Inverse( _
          ByRef board As Integer, _
          ByRef errIn As T_Error, _
          Optional ByRef vqXn As Integer = 0, _
          Optional ByRef oVQXn As Integer = 0) _
          As T_Error
```

5.2.17 VQ_Square.



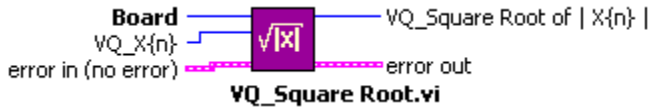
VQ_X{n} = VQ input.

VQ_Square of X{n} = VQ output, square of input vector X{n}.

QuBASE Equivalent.

```
'*************************************************************************
'       Function VQ_Square
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'
'       Output Parameters:
'               oVQXn, Integer
'
'*************************************************************************

   Function VQ_Square( _
          ByRef board As Integer, _
          ByRef errIn As T_Error, _
          Optional ByRef vqXn As Integer = 0, _
          Optional ByRef oVQXn As Integer = 0) _
          As T_Error
```

5.2.18 VQ_Square Root.



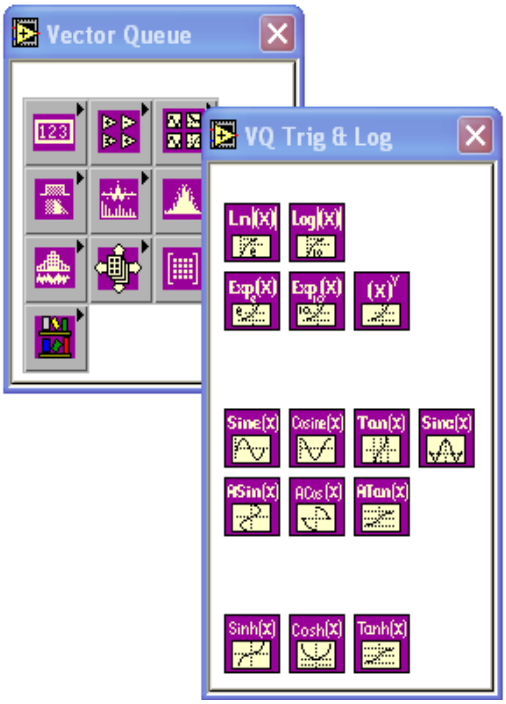
VQ_X{n} = VQ input.

VQ_Square Root of |X{n}| = VQ output, square root of absolute value of X{n}.

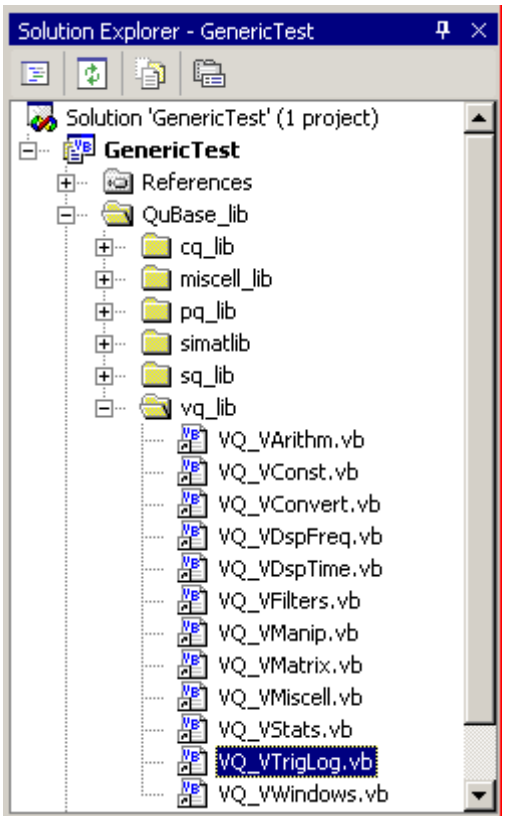
QuBASE Equivalent.

```
'************************************************************************
'       Function VQ_Squareroot
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'
'       Output Parameters:
'               oVQXn, Integer
'
'************************************************************************

  Function VQ_Squareroot( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef vqXn As Integer = 0, _
        Optional ByRef oVQXn As Integer = 0) _
        As T_Error
```

5.3 VQ Trig and Log.



QuBASE Equivalent.



5.3.1 VQ_Natural Logarithm.

VQ_X{n} = VQ input.

VQ_Natural Log of |X{n}| = VQ output, computes natural logarithm base "e" with argument as the absolute value of X{n}.

QuBASE Equivalent.

```
'*************************************************************************
'       Function VQ_NaturalLog
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'
'       Output Parameters:
'               oVQXn, Integer
'
'*************************************************************************

   Function VQ_NaturalLog( _
          ByRef Board As Integer, _
          ByRef errIn As T_Error, _
          Optional ByRef vqXn As Integer = 0, _
          Optional ByRef oVQXn As Integer = 0) _
          As T_Error
```

5.3.2 VQ_Log Base 10.

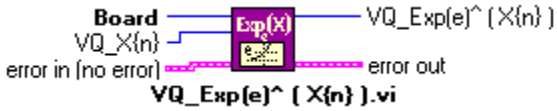
VQ_X{n} = VQ input.

VQ_Log|X{n}| = VQ output, computes logarithm base "10" with argument as the absolute value of X{n}.

QuBASE Equivalent.

```
'**********************************************************************
'       Function VQ_LogBase10
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'
'       Output Parameters:
'               oVQXn, Integer
'
'**********************************************************************

   Function VQ_LogBase10( _
          ByRef Board As Integer, _
          ByRef errIn As T_Error, _
          Optional ByRef vqXn As Integer = 0, _
          Optional ByRef oVQXn As Integer = 0) _
          As T_Error
```

5.3.3 VQ_Exp(e)^(X{n}).



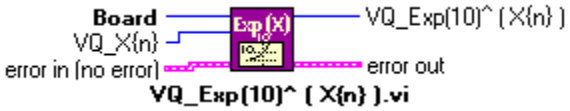
VQ_X{n} = VQ input.

VQ_Exp(e)^(X{n}) = VQ output, computes exponential base "e" with power of X{n}.

QuBASE Equivalent.

```
'*************************************************************************
'       Function VQ_Exponential
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'
'       Output Parameters:
'               oVQXn, Integer
'
'*************************************************************************

  Function VQ_Exponential( _
        ByRef Board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef vqXn As Integer = 0, _
        Optional ByRef oVQXn As Integer = 0) _
        As T_Error
```

5.3.4 VQ_Exp(10)^(X{n}).



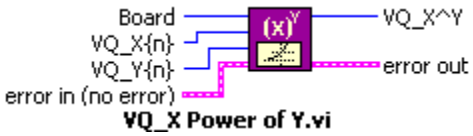
VQ_X{n} = VQ input.

VQ_Exp(10)^(X{n}) = VQ output, computes exponential base "10" with power of X{n}.

QuBASE Equivalent.

```
'*************************************************************************
'       Function VQ_Exp10
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'
'       Output Parameters:
'               oVQXn, Integer
'
'*************************************************************************

  Function VQ_Exp10( _
        ByRef Board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef vqXn As Integer = 0, _
        Optional ByRef oVQXn As Integer = 0) _
        As T_Error
```

5.3.5 VQ_X to Power of Y.



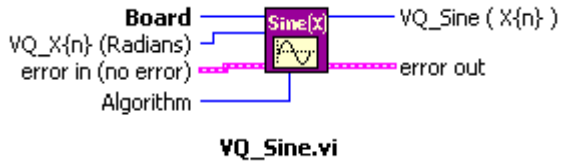
VQ_X{n} = VQ input.
VQ_Y{n} = VQ input.

VQ_X^Y = VQ output, computes exponential base X{n} with power of Y{n}.

QuBASE Equivalent.

```
'*************************************************************************
'       Function VQ_XPowerOfY
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'               vqYn, Integer
'
'       Output Parameters:
'               oVQXY, Integer
'
'*************************************************************************

  Function VQ_XPowerOfY( _
        ByRef Board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef vqXn As Integer = 0, _
        Optional ByRef vqYn As Integer = 0, _
        Optional ByRef oVQXY As Integer = 0) _
        As T_Error
```

5.3.6 VQ_Sine.



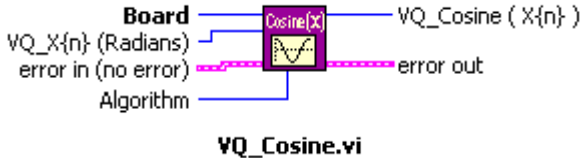
VQ_X{n} = VQ input in radians.
Algorithm = LabVIEW input, selects the method for results as follows:
1) *Generate* = values are computed by the DSP using an accurate but relatively time consuming iterative process.
2) *Table* = values are sourced from a look up table for higher speed and interpolated for acceptable accuracy.
Table Resolution = LabVIEW input that determines the size of the look up table; defaulted to 1024.

VQ_Sine(X{n}) = VQ output, computes the Sine of radians expressed in the X{n} vector.

QuBASE Equivalent.

```
'**************************************************************************
'       Function VQ_Sine
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'
'       Output Parameters:
'               oVQXn, Integer
'
'**************************************************************************

  Function VQ_Sine( _
        ByRef Board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef vqXn As Integer = 0, _
        Optional ByRef oVQXn As Integer = 0) _
        As T_Error
```

5.3.7 VQ_Cosine.



VQ_X{n} = VQ input in radians.
Algorithm = LabVIEW input, selects the method for results as follows:
1) *Generate* = values are computed by the DSP using an accurate but relatively time consuming iterative process.
2) *Table* = values are sourced from a look up table for higher speed and interpolated for acceptable accuracy.
Table Resolution = LabVIEW input that determines the size of the look up table; defaulted to 1024.

VQ_Cosine(X{n}) = VQ output, computes the Cosine of radians expressed in the X{n} vector.

QuBASE Equivalent.

```
'*************************************************************************
'       Function VQ_Cosine
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'
'       Output Parameters:
'               oVQXn, Integer
'
'*************************************************************************

   Function VQ_Cosine( _
          ByRef Board As Integer, _
          ByRef errIn As T_Error, _
          Optional ByRef vqXn As Integer = 0, _
          Optional ByRef oVQXn As Integer = 0) _
          As T_Error
```

5.3.8 VQ_Tangent.

VQ_X{n} = VQ input in radians.

VQ_Tangent(X{n}) = VQ output, computes the Tangent of radians expressed in the X{n} vector.

QuBASE Equivalent.

```
'**********************************************************************
'       Function VQ_Tangent
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'
'       Output Parameters:
'               oVQTangent, Integer
'
'**********************************************************************

  Function VQ_Tangent( _
        ByRef Board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef vqXn As Integer = 0, _
        Optional ByRef oVQTangent As Integer = 0) _
        As T_Error
```

5.3.9 VQ_Sinc.

VQ_X{n} = VQ input in radians.

VQ_Sinc(X{n}) = VQ output, computes the Sinc of the X{n} vector.

QuBASE Equivalent.

```
'**********************************************************************
'       Function VQ_Sinc
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'
'       Output Parameters:
'               oVQSinc, Integer
'
'**********************************************************************

   Function VQ_Sinc( _
          ByRef Board As Integer, _
          ByRef errIn As T_Error, _
          Optional ByRef vqXn As Integer = 0, _
          Optional ByRef oVQSinc As Integer = 0) _
          As T_Error
```

5.3.10 VQ_ArcSine.

VQ_X{n} = VQ input, represented as any floating point value where -1 =< X{n} =<1. Values that fall out of this range are replaced with a '-1' for negative values beyond '-1', and similarly replaced with a '+1' for positive values beyond '+1'.

VQ_ArcSine(X{n}) = VQ output in radians, computes the Inverse Sine of the X{n} vector.

QuBASE Equivalent.

```
'************************************************************************
'       Function VQ_ArcSine
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'
'       Output Parameters:
'               oVQArcSine, Integer
'
'************************************************************************

  Function VQ_ArcSine( _
        ByRef Board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef vqXn As Integer = 0, _
        Optional ByRef oVQArcSine As Integer = 0) _
        As T_Error
```

5.3.11 VQ_ArcCosine.

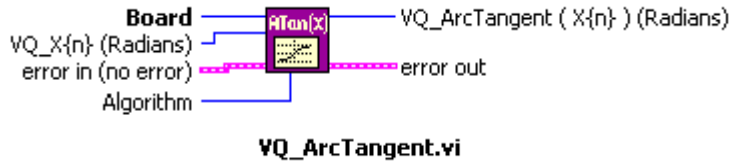
VQ_X{n} = VQ input, represented as any floating point value where -1 =< X{n} =<1. Values that fall out of this range are replaced with a '-1' for negative values beyond '-1', and similarly replaced with a '+1' for positive values beyond '+1'.

VQ_ArcCosine(X{n}) = VQ output in radians, computes the Inverse Cosine of the X{n} vector.

QuBASE Equivalent.

```
'**********************************************************************
'       Function VQ_ArcCosine
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'
'       Output Parameters:
'               oVQArcCosine, Integer
'
'**********************************************************************

  Function VQ_ArcCosine( _
        ByRef Board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef vqXn As Integer = 0, _
        Optional ByRef oVQArcCosine As Integer = 0) _
        As T_Error
```

5.3.12 VQ_ArcTangent.



VQ_X{n} = VQ input, represented as any floating point value.
Algorithm = LabVIEW input, selects the method for results as follows:
1) *Generate* = values are computed by the DSP using an accurate but relatively time consuming iterative process.
2) *Table* = values are sourced from a look up table for higher speed and interpolated for acceptable accuracy.
Table Resolution = LabVIEW input that determines the size of the look up table; defaulted to 1024.

VQ_ArcTangent(X{n}) = VQ output in radians, computes the Inverse Tangent of the X{n} vector.

QuBASE Equivalent.

```
'*************************************************************************
'       Function VQ_ArcTangent
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'
'       Output Parameters:
'               oVQArcTangent, Integer
'
'*************************************************************************

  Function VQ_ArcTangent( _
        ByRef Board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef vqXn As Integer = 0, _
        Optional ByRef oVQArcTangent As Integer = 0) _
        As T_Error
```

5.3.13 VQ_Hyperbolic Sine.

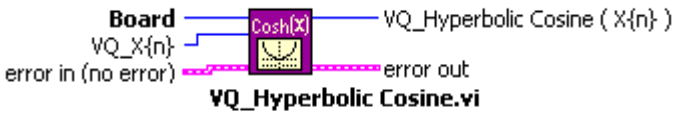
VQ_X{n} = VQ input, represented as any floating point value.

VQ_Hyperbolic Sine(X{n}) = VQ output, computes the Hyperbolic Sine of the X{n} vector.

QuBASE Equivalent.

```
'**************************************************************************
'       Function VQ_HyperbolicSine
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'
'       Output Parameters:
'               oVQHyperbolicSine, Integer
'
'**************************************************************************

  Function VQ_HyperbolicSine( _
        ByRef Board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef vqXn As Integer = 0, _
        Optional ByRef oVQHyperbolicSine As Integer = 0) _
        As T_Error
```

5.3.14 VQ_Hyperbolic Cosine.



VQ_X{n} = VQ input, represented as any floating point value.

VQ_Hyperbolic Cosine(X{n}) = VQ output, computes the Hyperbolic Cosine of the X{n} vector.

QuBASE Equivalent.

```
'**********************************************************************
'       Function VQ_HyperbolicCosine
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'
'       Output Parameters:
'               oVQHyperbolicCosine, Integer
'
'**********************************************************************

  Function VQ_HyperbolicCosine( _
        ByRef Board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef vqXn As Integer = 0, _
        Optional ByRef oVQHyperbolicCosine As Integer = 0) _
        As T_Error
```

5.3.13 VQ_Hyperbolic Tangent.

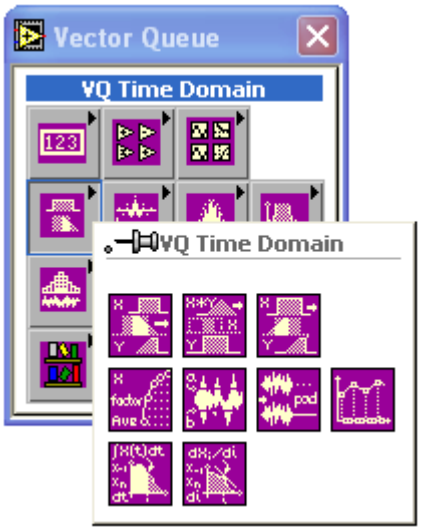
VQ_X{n} = VQ input, represented as any floating point value.

VQ_Hyperbolic Tangent(X{n}) = VQ output, computes the Hyperbolic Tangent of the X{n} vector.

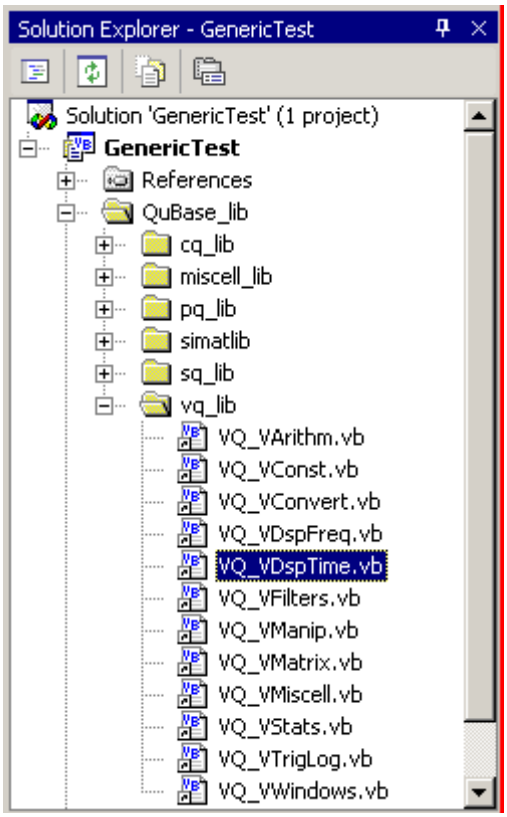
QuBASE Equivalent.

```
'**********************************************************************
'       Function VQ_HyperbolicTangent
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'
'       Output Parameters:
'               oVQHyperbolicTangent, Integer
'
'**********************************************************************

  Function VQ_HyperbolicTangent( _
        ByRef Board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef vqXn As Integer = 0, _
        Optional ByRef oVQHyperbolicTangent As Integer = 0) _
        As T_Error
```

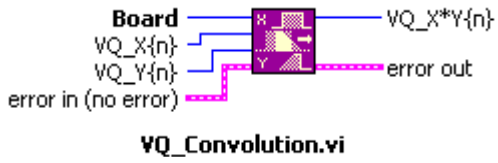
5.4 VQ Time Domain Signal Processing.



QuBASE Equivalent.



5.4.1 VQ_Convolution.



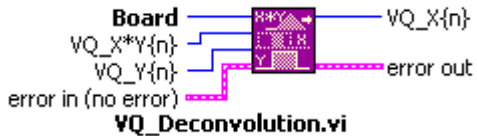
VQ_X{n} = VQ input.
VQ_Y{n} = VQ input.

VQ_Convolution H{n} = VQ output.

QuBASE Equivalent.

```
'*************************************************************************
'       Function VQ_Convolution
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'               vqYn, Integer
'
'       Output Parameters:
'               oVQConvolution, Integer
'
'*************************************************************************

  Function VQ_Convolution( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef vqXn As Integer = 0, _
        Optional ByRef vqYn As Integer = 0, _
        Optional ByRef oVQConvolution As Integer = 0) _
        As T_Error
```

5.4.2 VQ_Deconvolution.



VQ_X*Y{n} = VQ input representing the convolved function.
VQ_Y{n} = VQ input.

VQ_Deconvolved X{n} = VQ output.

QuBASE Equivalent.

```
'*************************************************************************
'       Function VQ_DeConvolution
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXYn, Integer
'               vqYn, Integer
'
'       Output Parameters:
'               oVQDeConvolution, Integer
'
'*************************************************************************

    Function VQ_DeConvolution( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef vqXYn As Integer = 0, _
        Optional ByRef vqYn As Integer = 0, _
        Optional ByRef oVQDeConvolution As Integer = 0) _
        As T_Error
```

5.4.3 VQ_Correlation.

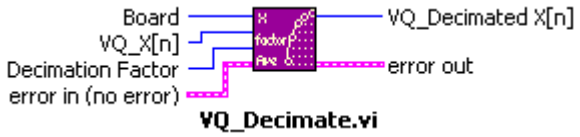
VQ_X{n} = VQ input.
VQ_Y{n} = VQ input.

VQ_Correlation Rxy{n} = VQ output.

QuBASE Equivalent.

```
'*************************************************************************
'       Function VQ_Correlation
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'               vqYn, Integer
'
'       Output Parameters:
'               oVQCorrelation, Integer
'
'*************************************************************************

  Function VQ_Correlation( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef vqXn As Integer = 0, _
        Optional ByRef vqYn As Integer = 0, _
        Optional ByRef oVQCorrelation As Integer = 0) _
        As T_Error
```

5.4.4 VQ_Decimate.



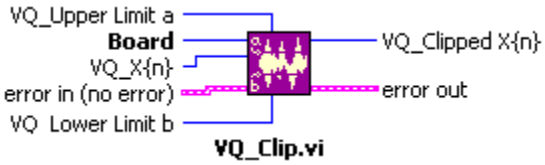
VQ_X{n} = VQ input.
Decimation Factor = LabVIEW input that determines every nth point to keep in the newly sized vector.

VQ_Decimated X{n} = VQ output.

QuBASE Equivalent.

```
'*************************************************************************
'       Function VQ_Decimate
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'               decimationFactor, Integer
'
'       Output Parameters:
'               oVQDecimateXn, Integer
'
'*************************************************************************

  Function VQ_Decimate( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef vqXn As Integer = 0, _
        Optional ByRef decimationFactor As Integer = 0, _
        Optional ByRef oVQDecimateXn As Integer = 0) _
        As T_Error
```

5.4.5 VQ_Clip.



VQ_X{n} = VQ input.

VQ_Upper Limit a = VQ input that determines the upper value to clip incoming points in the array that exceed it.

VQ_Lower Limit a = VQ input that determines the lower value to clip incoming points in the array that exceed it.

VQ_Clipped X{n} = VQ output with all values clipped.

QuBASE Equivalent.

```
'**************************************************************************
'       Function VQ_Clip
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'               vqUpperLimit, Integer
'               vqLowerLimit, Integer
'
'       Output Parameters:
'               vqClipped, Integer
'
'**************************************************************************

  Function VQ_Clip( _
        ByVal board As Integer, _
        ByRef errIn As T_Error, _
        ByRef vqXn As Integer, _
        ByRef vqClipped As Integer, _
        Optional ByRef vqUpperLimit As Integer = 0, _
        Optional ByRef vqLowerLimit As Integer = 0) _
        As T_Error
```

5.4.6 VQ_Zero Padder.

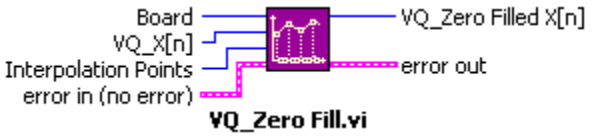
VQ_X{n} = VQ input.

VQ_Zero Padded X{n} = VQ output that places 'zero' values into an array until it reaches the next power of '2' in size. Mainly used before an FFT.

QuBASE Equivalent.

```
'**********************************************************************
'       Function VQ_ZeroPadder
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'
'       Output Parameters:
'               vqZeroPadder, Integer
'
'**********************************************************************

   Function VQ_ZeroPadder( _
            ByVal board As Integer, _
            ByRef errIn As T_Error, _
            ByVal vqXn As Integer, _
            ByRef vqZeroPadder As Integer) _
            As T_Error
```

5.4.7 VQ_Zero Fill.



VQ_X{n} = VQ input.

Interpolation Points = LabVIEW input that determines the number of 'zeros' to insert between every sample.

VQ_Zero Filled X{n} = VQ output.

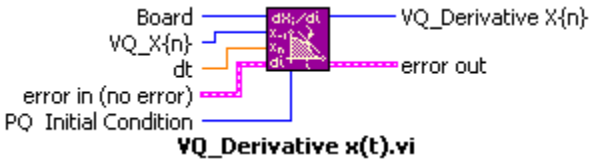
QuBASE Equivalent.

```
'**************************************************************************
'       Function VQ_ZeroFill
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'               interpolationPoints, Integer
'
'       Output Parameters:
'               oVQZeroFillXn, Integer
'
'**************************************************************************

  Function VQ_ZeroFill( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef vqXn As Integer = 0, _
        Optional ByRef interpolationPoints As Integer = 0, _
        Optional ByRef oVQZeroFillXn As Integer = 0) _
        As T_Error
```

5.4.8 VQ_Derivative x(t).

The VQ_Derivative VI performs a discrete differentiation of the sampled signal vector represented by a VQ Input. The differentiation of VQ_X[n], d/dt X[n], is computed using the following equation:

d/dtX[n][= (1/2*dt) (X[n+1] – X[n-1]).



VQ_X{n} = VQ input.

dt = LabVIEW input that determines width at which to divide the current and previous values of the input vector.

PQ_Initial Condition = PQ input, used to fill in an absent point when computing the first slope value.

VQ_Derivative X{n} = VQ output.

QuBASE Equivalent.

```
'*********************************************************************
'       Function VQ_Derivative
'
'       Description:
'               Used as an subfunction
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'               pqInitialCondition, Integer
'               dt, Double
'
'       Output Parameters:
'               oVQDerivativeXn, Integer
'
'*********************************************************************

  Function VQ_Derivative( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef vqXn As Integer = 0, _
        Optional ByRef dt As Double = 1, _
        Optional ByRef pqInitialCondition As Integer = 0, _
        Optional ByRef oVQDerivativeXn As Integer = 0) _
        As T_Error
```

5.4.9 VQ_Integral x(t).

The VQ_Integral x(t) VI performs the discrete integration of the sampled signal vector represented by a VQ Input. The integration is obtained using the following equation:

Y[n] = Y[n-1] + (X[n] * dt).

where
dt = 1
0=<n<=k-1.

VQ_X{n} = VQ input.

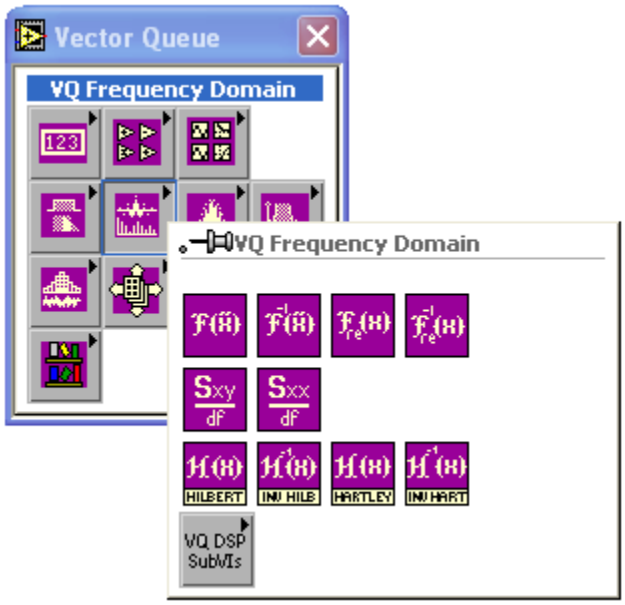
VQ_Integral X{n} = VQ output.

***NOTE**: It is possible to observe a wrap around effect (also known as wind up) when using the Integration function. This is not an error in the DSP code, but rather the compounded effect of the input signal's offset added back onto itself. The Calibrate function within the miscellaneous operations subset may be used to digitally correct this initial input signal offset.*

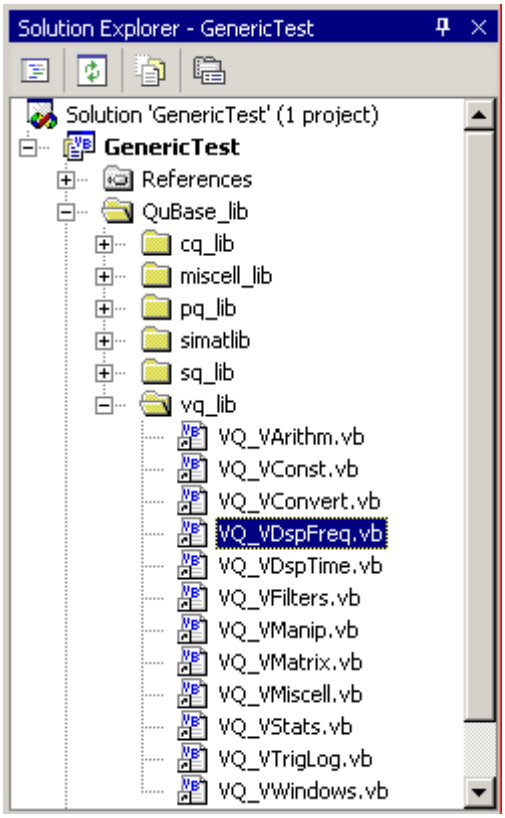
QuBASE Equivalent.

```
'************************************************************************
'       Function VQ_Integral
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'
'       Output Parameters:
'               oVQIntegralXn, Integer
'
'************************************************************************

  Function VQ_Integral( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef vqXn As Integer = 0, _
        Optional ByRef oVQIntegralXn As Integer = 0) _
        As T_Error
```

5.5 VQ Frequency Domain Signal Processing.

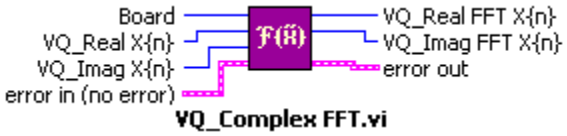


QuBASE Equivalent.



5.5.1 VQ_Complex FFT.

The VQ_Complex FFT VI computes the Complex Fast Fourier Transform (cFFT) on a pair of VQ inputs with a power of '2' size. All vector pairs represent complex values in rectangular form.



VQ_Real X{n} = VQ input.
VQ_Imag X{n} = VQ input.

VQ_Real FFT X{n} = VQ output.
VQ_Imag FFT X{n} = VQ output.

QuBASE Equivalent.

```
'**********************************************************************
'       Function VQ_ComplexFFT
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               ivqReal, Integer
'               ivqImag, Integer
'
'       Output Parameters:
'               oVQReal, Integer
'               oVQImag, Integer
'
'**********************************************************************

  Function VQ_ComplexFFT( _
            ByRef board As Integer, _
            ByRef errIn As T_Error, _
            Optional ByRef ivqReal As Integer = 0, _
            Optional ByRef ivqImag As Integer = 0, _
            Optional ByRef oVQReal As Integer = 0, _
            Optional ByRef oVQImag As Integer = 0) _
            As T_Error
```

5.5.2 VQ_Inverse Complex FFT.

The VQ_Inverse Complex FFT VI computes the Inverse Complex Fast Fourier Transform (ciFFT) on a pair of VQ inputs with a power of '2' size. All vector pairs represent complex values in rectangular form.

VQ_Real FFT X{n} = VQ input.
VQ_Imag FFT X{n} = VQ input.

VQ_Real X{n} = VQ output.
VQ_Imag X{n} = VQ output.

QuBASE Equivalent.

```
'*************************************************************************
'       Function VQ_InverseComplexFFT
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               ivqReal, Integer
'               ivqImag, Integer
'
'       Output Parameters:
'               oVQReal, Integer
'               oVQImag, Integer
'
'*************************************************************************

  Function VQ_InverseComplexFFT( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef ivqReal As Integer = 0, _
        Optional ByRef ivqImag As Integer = 0, _
        Optional ByRef oVQReal As Integer = 0, _
        Optional ByRef oVQImag As Integer = 0) _
        As T_Error
```

5.5.3 VQ_Real FFT.

The VQ_Real FFT VI computes the Real Fast Fourier Transform (FFT) on a single real VQ input with a power of '2' size. The resultant vector pair represent complex values in rectangular form.

VQ_Real X{n} = VQ input.

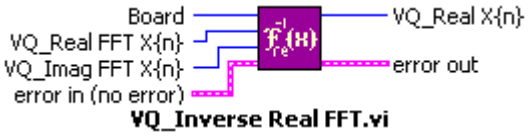
VQ_Real FFT X{n} = VQ output.
VQ_Imag FFT X{n} = VQ output.

QuBASE Equivalent.

```
'**************************************************************************
'       Function VQ_RealFFT
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               iIntVQRealXn, Integer
'
'       Output Parameters:
'               oIntVQRealFFTXn, Integer
'               oIntVQImagFFTXn, Integer
'
'**************************************************************************

  Function VQ_RealFFT( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef iIntVQRealXn As Integer, _
        Optional ByRef oIntVQRealFFTXn As Integer = 0, _
        Optional ByRef oIntVQImagFFTXn As Integer = 0) _
        As T_Error
```

5.5.4 VQ_Inverse Real FFT.

The VQ_Inverse Real FFT VI computes the Inverse Real Fast Fourier Transform (iFFT) on a pair of VQ inputs with a power of '2' size. The input vector pairs represent complex values while the resultant vector represents single real values, all in rectangular form.



VQ_Real FFT X{n} = VQ output input.
VQ_Imag FFT X{n} = VQ output input.

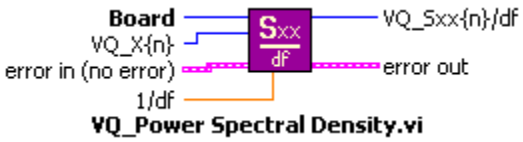
VQ_Real X{n} = VQ output.

QuBASE Equivalent.

```
'**************************************************************************
'       Function VQ_InverseRealFFT
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               VQRealFFTXn, Integer
'               VQImagFFTXn, Integer
'
'       Output Parameters:
'               oVQRealXn, Integer
'
'**************************************************************************

  Function VQ_InverseRealFFT( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef VQRealFFTXn As Integer = 0, _
        Optional ByRef VQImagFFTXn As Integer = 0, _
        Optional ByRef oVQRealXn As Integer = 0) _
        As T_Error
```

5.5.5 VQ_Power Spectral Density.

The VQ_Power Spectral Density VI computes the Power Spectral Density (Sxx), weighted against frequency, on a single real VQ input with a power of '2' size. All vectors represent real values in rectangular form.



VQ_X{n} = VQ input, real values.
1/df = LabVIEW input. If left unwired this defaults to '1' and returns a Power Spectrum. Otherwise, this value can be computed by the equation = Sample Rate/n.

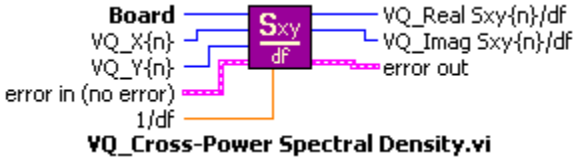
VQ_Sxx{n}/df = VQ output that reflects the Power Spectral Density weighted against frequency.

QuBASE Equivalent.

```
'*************************************************************************
'       Function VQ_PowerSpectralDensity
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqRealXn, Integer
'               vqImagXn, Integer
'               df_inverse, Double
'
'       Output Parameters:
'               oVQSxx, Integer
'
'*************************************************************************

  Function VQ_PowerSpectralDensity( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef vqRealXn As Integer = 0, _
        Optional ByRef vqImagXn As Integer = 0, _
        Optional ByRef df_inverse As Double = 0, _
        Optional ByRef oVQSxx As Integer = 0) _
        As T_Error
```

5.5.6 VQ_Cross-Power Spectral Density.

The VQ_Cross-Power Spectral Density VI computes the Cross-Power Spectral Density (Sxy), weighted against frequency, on a pair of VQ inputs with a power of '2' size. Each input vector represents real values while the resultant vector represents complex values, all in rectangular form.



VQ_Real X{n} = VQ input for the first real vector.
VQ_Real Y{n} = VQ input of the second real vector.
1/df = LabVIEW input. If left unwired this defaults to a '1' and returns a Cross-Power Spectrum. Otherwise, this value can be computed by the equation = Sample Rate/n.

VQ_Real Sxy{n}/df = VQ output of the real portion of the Cross-Spectral Density.
VQ_Imag Sxy{n}/df = VQ output of the imaginary portion of the Cross-Spectral Density.

QuBASE Equivalent.

```
'**********************************************************************
'       Function VQ_CrossPowerSpectralDensity
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqRealXn, Integer
'               vqImagXn, Integer
'               vqRealYn, Integer
'               vqImagYn, Integer
'               df_inverse, Double
'
'       Output Parameters:
'               oVQRealSxx, Integer
'               oVQImagSxx, Integer
'
'**********************************************************************

  Function VQ_CrossPowerSpectralDensity( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef vqRealXn As Integer = 0, _
        Optional ByRef vqImagXn As Integer = 0, _
        Optional ByRef vqRealYn As Integer = 0, _
        Optional ByRef vqImagYn As Integer = 0, _
        Optional ByRef df_inverse As Double = 1, _
        Optional ByRef oVQRealSxx As Integer = 0, _
        Optional ByRef oVQImagSxx As Integer = 0) _
        As T_Error
```

5.5.7 VQ_Fast Hilbert Transform.

The VQ_Fast Hilbert Transform VI computes the Fast Hilbert Transform, using Fourier Transform (FFT) identities, on a single real vector with a power of '2' size. All vectors represent real values in rectangular form.

VQ_X{n} = VQ input, real vector.

VQ_Hilbert{n} = VQ output.

QuBASE Equivalent.

```
'*************************************************************************
'       Function VQ_FastHilbertTransform
'
'       Description:
'               Used as an subfunction
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               iVQXn, Integer
'
'       Output Parameters:
'               oVQYn, Integer
'
'*************************************************************************

  Function VQ_FastHilbertTransform( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef iVQXn As Integer = 0, _
        Optional ByRef oVQYn As Integer = 0) _
        As T_Error
```

5.5.8 VQ_Inverse Fast Hilbert Transform.

The VQ_Inverse Fast Hilbert Transform VI computes the Inverse Fast Hilbert Transform, using Fourier Transform (FFT) identities, on a single real vector with a power of '2' size. All vectors represent real values in rectangular form.

VQ_X{n} = VQ input, real vector.

VQ_Inv. Hilbert{n} = VQ output.

QuBASE Equivalent.

```
'*************************************************************************
'       Function VQ_InverseFastHilbertTransform
'
'       Description:
'               Used as an subfunction
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               iVQXn, Integer
'
'       Output Parameters:
'               oVQYn, Integer
'
'*************************************************************************

   Function VQ_InverseFastHilbertTransform( _
          ByRef board As Integer, _
          ByRef errIn As T_Error, _
          Optional ByRef iVQXn As Integer = 0, _
          Optional ByRef oVQYn As Integer = 0) _
          As T_Error
```

5.5.9 VQ_Fast Hartley Transform.

The VQ_Fast Hartley Transform VI computes the Fast Hartley Transform (FHT) on a single real vector with a power of '2' size. All vectors represent real values in rectangular form.

VQ_X{n} = VQ input, real vector.

VQ_FHT{n} = VQ output.

QuBASE Equivalent.

```
'**************************************************************************
'       Function VQ_FastHartleyTransform
'
'       Description:
'               Used as an subfunction
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               iVQXn, Integer
'
'       Output Parameters:
'               oVQYn, Integer
'
'**************************************************************************

  Function VQ_FastHartleyTransform( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef iVQXn As Integer = 0, _
        Optional ByRef oVQYn As Integer = 0) _
        As T_Error
```

5.5.10 VQ_Inverse Fast Hartley Transform.

The VQ_Inverse Fast Hartley Transform VI computes the Inverse Fast Hartley Transform (iFHT) on a single real vector with a power of '2' size. All vectors represent real values in rectangular form.

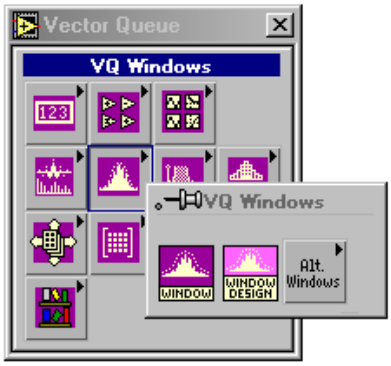
VQ_X{n} = VQ input, real vector.

VQ_Inv. FHT{n} = VQ output.

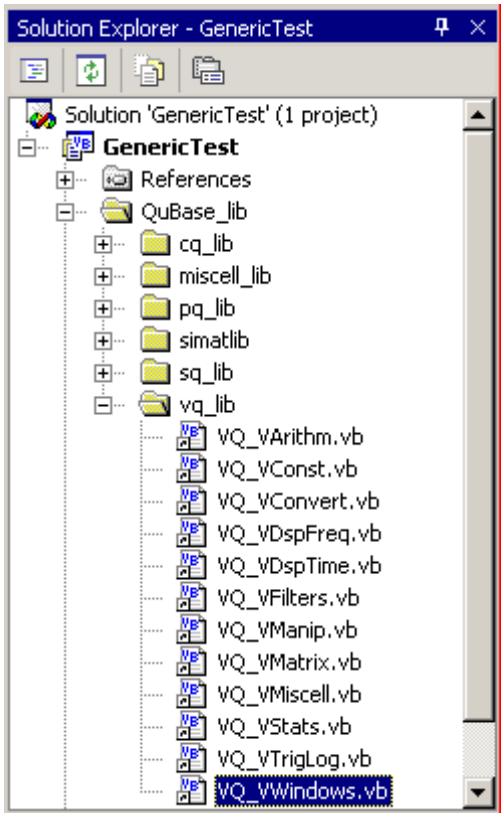
QuBASE Equivalent.

```
'*************************************************************************
'       Function VQ_InverseFastHartleyTransform
'
'       Description:
'               Used as an subfunction
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               iVQXn, Integer
'
'       Output Parameters:
'               oVQYn, Integer
'
'*************************************************************************

  Function VQ_InverseFastHartleyTransform( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef iVQXn As Integer = 0, _
        Optional ByRef oVQYn As Integer = 0) _
        As T_Error
```

5.6 VQ Windows.

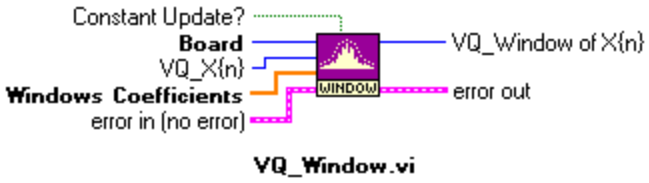


QuBASE Equivalent.



5.6.1 VQ_Window.

The 'VQ_Window' VI function applies a windowing function to any input vector; may be used in conjunction with the 'Windows Coefficients Design' VI.



Constant Update? = LabVIEW boolean input that determines if the LabVIEW variables are to be constantly updated or not. When left in the 'false' position, the LabVIEW constants are only updated once during the queueing phase thereby avoiding DSP-Host communications overhead. When left in the default 'true' position, LabVIEW constants are continuously updated thereby implementing real time variable updates for maximum flexibility.

VQ_X{n} = VQ input, data to be subjected to a windowing function.
Windows Coefficients = LabVIEW input array of the coefficients that represent the windowing function of interest; may be derived from the Windows Coefficients Design VI.

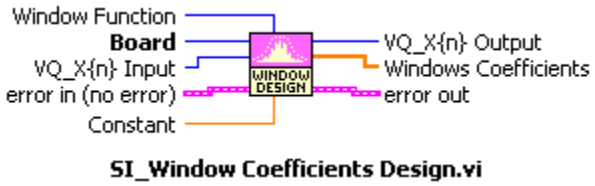
VQ_Window of X{n} = VQ output, windowed function.

QuBASE Equivalent.

```
'*************************************************************************
'       Function VQ_Window
'
'       Description:
'               The Window function is nothing more than the VQ_Convolution of an
'               X{n} input source and a set of coefficiants derived from the
'               LIB_Window Coefficients Design function.
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               iintVQXn, Integer
'               idblWindowCoefficients, Double
'               iintConstantUpdate, Integer
'
'       Output Parameters:
'               ointVQWindowXn, Integer
'
'*************************************************************************

   Function VQ_Window( _
          ByRef board As Integer, _
          ByRef errIn As T_Error, _
          ByRef iintVQXn As Integer, _
          ByRef idblWindowCoefficients() As Double, _
          ByRef ointVQWindowXn As Integer, _
          Optional ByRef iintConstantUpdate As Integer = 0) _
          As T_Error
```

5.6.2 SI_Window Coefficients Design.

The SI Windows Coefficients Design VI computes the coefficients for a specific windowing function in the LabVIEW/host domain, very convenient to be used used in conjunction with the 'VQ_Window' VI function.



Window Function = LabVIEW input to select the algorithm with which to compute the windowing coefficients of interest.

VQ_X{n} Input = VQ input, only used to derive the size of the vector since it must coincide with the number of windowing coefficients.

Constant = LabVIEW input, ancillary variable necessary to compute coefficients for some variety of windowing functions.

VQ_X{n} Output = VQ output that only mirrors the VQ_Input terminal as a convenience, may be optionally used to feed the VQ input terminal to the VQ_Window.vi function.

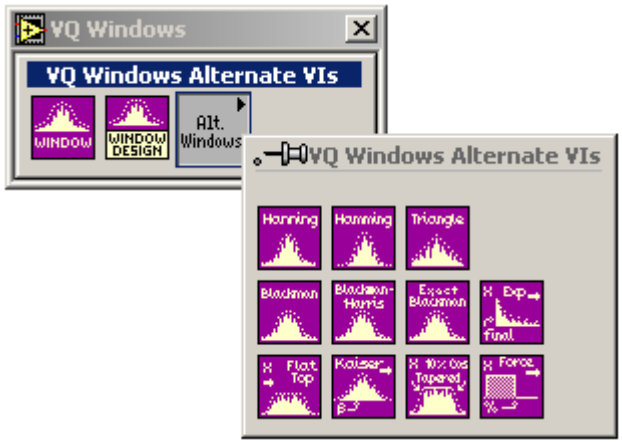
Windows Coefficients = LabVIEW output array, actual windowing coefficients to be fed into the same terminal on the 'VQ_Window' VI function.

QuBASE Equivalent.

```
'*********************************************************************
'       Function SI_WindowCoefficientsDesign
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn As Integer, _
'               vqWindowCoefficients, Integer
'               constant, Double
'
'       Output Parameters:
'               windowCoefficients(), Double
'               windowFunction, Integer
'
'*********************************************************************

  Function SI_WindowCoefficientsDesign( _
    ByVal board As Integer, _
    ByRef errIn As T_Error, _
    ByVal vqXn As Integer, _
    ByRef vqWindowCoefficients As Integer, _
    ByRef windowCoefficients() As Double, _
    Optional ByVal windowFunction As Integer = 0, _
    Optional ByVal constant As Double = 0) _
    As T_Error
```

5.6.3 VQ Windows Alternate VIs.

These functions are redundant as they duplicate the functionality of the Window Coefficients Design VI and the VQ_Window VI functions, and are not necessary. Only provided as a convenience if only one function icon is desired to represent a single windowing algorithm.



5.6.3.1 VQ_Hanning.

VQ_X{n} = VQ input.

VQ_Hanning of X{n} = VQ output.

QuBASE Equivalent.

```
'*************************************************************************
'       Function VQ_HanningWindow
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'
'       Output Parameters:
'               oVQXn, Integer
'
'*************************************************************************

  Function VQ_HanningWindow( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef oVQXn As Integer = 0, _
        Optional ByRef vqXn As Integer = 0) _
        As T_Error
```

5.6.3.2 VQ_Hamming.

VQ_X{n} = VQ input.

VQ_Hamming of X{n} = VQ output.

QuBASE Equivalent.

```
'************************************************************************
'       Function VQ_HammingWindow
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'
'       Output Parameters:
'               oVQXn, Integer
'
'************************************************************************

   Function VQ_HammingWindow( _
          ByRef board As Integer, _
          ByRef errIn As T_Error, _
          Optional ByRef oVQXn As Integer = 0, _
          Optional ByRef vqXn As Integer = 0) _
          As T_Error
```

5.6.3.3 VQ_Triangle.

VQ_X{n} = VQ input.

VQ_Triangle of X{n} = VQ output.

QuBASE Equivalent.

```
'*************************************************************************
'       Function VQ_TriangleWindow
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'
'       Output Parameters:
'               oVQXn, Integer
'
'*************************************************************************

  Function VQ_TriangleWindow( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef oVQXn As Integer = 0, _
        Optional ByRef vqXn As Integer = 0) _
        As T_Error
```

5.6.3.4 VQ_Blackman.

VQ_X{n} = VQ input.

VQ_Blackman of X{n} = VQ output.

QuBASE Equivalent.

```
'*************************************************************************
'       Function VQ_BlackmanWindow
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'
'       Output Parameters:
'               oVQXn, Integer
'
'*************************************************************************

  Function VQ_BlackmanWindow( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef oVQXn As Integer = 0, _
        Optional ByRef vqXn As Integer = 0) _
        As T_Error
```

5.6.3.5 VQ_Exact Blackman.

VQ_X{n} = VQ input.

VQ_Exact Blackman of X{n} = VQ output.

QuBASE Equivalent.

```
'**************************************************************************
'       Function VQ_ExactBlackmanWindow
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'
'       Output Parameters:
'               oVQXn, Integer
'
'**************************************************************************

  Function VQ_ExactBlackmanWindow( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef oVQXn As Integer = 0, _
        Optional ByRef vqXn As Integer = 0) _
        As T_Error
```

5.6.3.6 VQ_Blackman Harris.

VQ_X{n} = VQ input.

VQ_Blackman Harris of X{n} = VQ output.

QuBASE Equivalent.

```
'*************************************************************************
'       Function VQ_BlackmanHarrisWindow
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'
'       Output Parameters:
'               oVQXn, Integer
'
'*************************************************************************

   Function VQ_BlackmanHarrisWindow( _
            ByRef board As Integer, _
            ByRef errIn As T_Error, _
            Optional ByRef oVQXn As Integer = 0, _
            Optional ByRef vqXn As Integer = 0) _
            As T_Error
```

5.6.3.7 VQ_Flat Top.

VQ_X{n} = VQ input.

VQ_Flat Top of X{n} = VQ output.

QuBASE Equivalent.

```
'*********************************************************************
'       Function VQ_FlatTopWindow
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'
'       Output Parameters:
'               oVQXn, Integer
'
'*********************************************************************

  Function VQ_FlatTopWindow( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef oVQXn As Integer = 0, _
        Optional ByRef vqXn As Integer = 0) _
        As T_Error
```

5.6.3.8 VQ_Kaiser.

VQ_X{n} = VQ input.
Beta = LabVIEW input that defines the Beta weighting variable.

VQ_Kaiser of X{n} = VQ output.

QuBASE Equivalent.

```
'*************************************************************************
'       Function VQ_KaiserWindow
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'
'       Output Parameters:
'               oVQXn, Integer
'
'*************************************************************************

   Function VQ_KaiserWindow( _
            ByRef board As Integer, _
            ByRef errIn As T_Error, _
            Optional ByRef oVQXn As Integer = 0, _
            Optional ByRef vqXn As Integer = 0, _
            Optional ByRef beta As Double = 1) _
            As T_Error
```

5.6.3.9 VQ_Cosine Tapered.

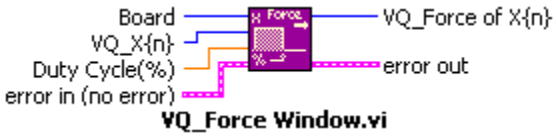
VQ_X{n} = VQ input.

VQ_Cosine Tapered of X{n} = VQ output.

QuBASE Equivalent.

```
'*********************************************************************
'       Function VQ_CosineTapered
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'
'       Output Parameters:
'               oVQXn, Integer
'
'*********************************************************************

  Function VQ_CosineTapered( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef oVQXn As Integer = 0, _
        Optional ByRef vqXn As Integer = 0) _
        As T_Error
```

5.6.3.10 VQ_Force.



VQ_X{n} = VQ input.
Duty Cycle(%) = LabVIEW input to determine the duty cycle.

VQ_Force of X{n} = VQ output.

QuBASE Equivalent.

```
'*************************************************************************
'       Function VQ_ForceWindow
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'
'       Output Parameters:
'               oVQXn, Integer
'
'*************************************************************************

  Function VQ_ForceWindow( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef oVQXn As Integer = 0, _
        Optional ByRef vqXn As Integer = 0, _
        Optional ByRef dutycycle As Double = 0) _
        As T_Error
```

5.6.3.11 VQ_Exponential.

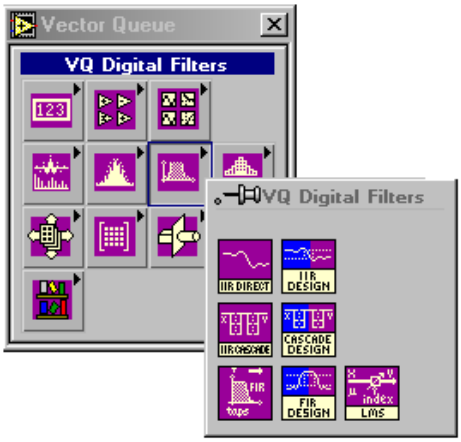
VQ_X{n} = VQ input.
Final Value = LabVIEW input to determine final output value of the input signal

VQ_Exponential of X{n} = VQ output.

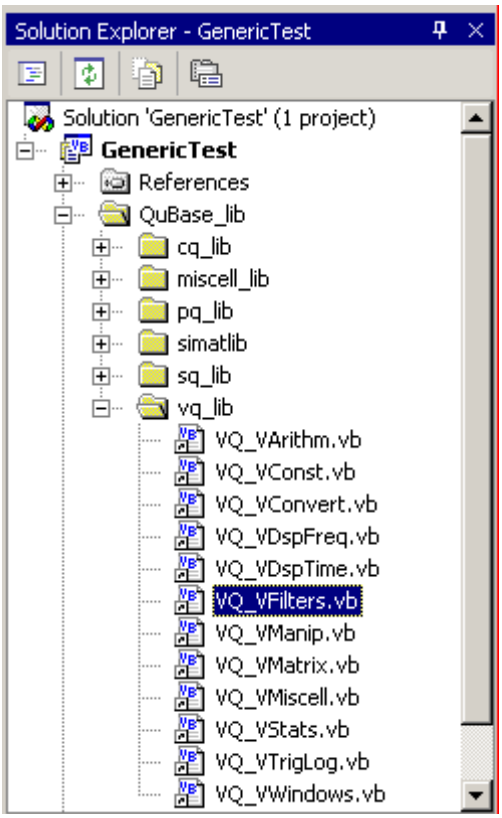
QuBASE Equivalent.

```
'*********************************************************************
'       Function VQ_ExponentialWindow
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'
'       Output Parameters:
'               oVQXn, Integer
'
'*********************************************************************

   Function VQ_ExponentialWindow( _
            ByRef board As Integer, _
            ByRef errIn As T_Error, _
            Optional ByRef oVQXn As Integer = 0, _
            Optional ByRef vqXn As Integer = 0, _
            Optional ByRef finalValue As Double = 0) _
            As T_Error
```

5.7 VQ Digital Filters.

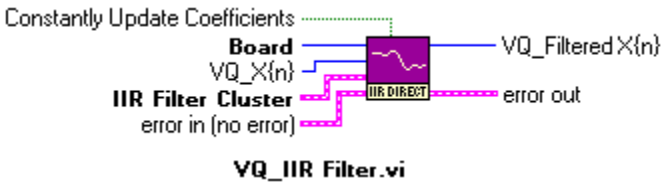


QuBASE Equivalent.



5.7.1 VQ_IIR Filter.

The IIR filters are setup as 2 separate VIs: a conventional LabVIEW based filter design VI and a DSP VQ_IIR Coefficient Filter Design VI. The VQ_IIR Filter VI is implemented on the DSP using the well known summation of two terms or recursive formula. Please refer to the Analysis Library Reference Manual.



Constant Update? = LabVIEW boolean input that determines if the LabVIEW variables are to be constantly updated or not. When left in the 'false' position, the LabVIEW constants are only updated once during the queueing phase thereby avoiding DSP-Host communications overhead. When left in the default 'true' position, LabVIEW constants are continuously updated thereby implementing real time variable updates for maximum flexibility.

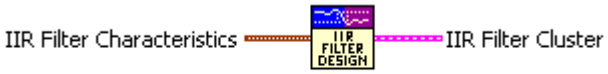
VQ_X{n} = VQ input.
IIR Filter Cluster = LabVIEW cluster containing the forward and reverse coefficients. The cluster may be derived by using one of the outputs of the LabVIEW based design VIs.

VQ_Filtered X{n} = VQ output that returns a filtered vector only on the input vector in question.

QuBASE Equivalent.

```
'**********************************************************************
'       Function VQ_IIRDirectFilter
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'               iirCluster, IIRFilterCluster
'               update, Integer
'
'       Output Parameters:
'               oVQFiltered, Integer
'               oVQCoefficients, Integer
'
'**********************************************************************

   Function VQ_IIRDirectFilter( _
          ByRef board As Integer, _
          ByRef errIn As T_Error, _
          ByRef iirCluster As IIRFilterCluster, _
          Optional ByRef oVQFiltered As Integer = 0, _
          Optional ByRef oVQCoefficients As Integer = 0, _
          Optional ByRef vqXn As Integer = 0, _
          Optional ByRef update As Integer = 0) _
          As T_Error
```

5.7.1.1 SI_IIR Direct Filter Coefficients Design.



IIR Direct Filter Coefficients Design.vi

IIR Filter Characteristics = LabVIEW input cluster to define the IIR filter characteristics.

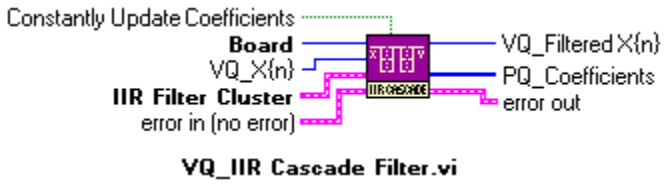
IIR Filter Cluster = LabVIEW cluster containing the forward and reverse coefficients.

QuBASE Equivalent.

```
'**********************************************************************
'       Function SI_IIRDirectFilterCoefficientsDesign
'
'       Description:
'
'       Input Parameters:
'               iFilterCharacteristics, IIRFilterCharacteristics
'
'       Output Parameters:
'               oFilterCluster, IIRFilterCluster
'
'**********************************************************************

  Function SI_IIRDirectFilterCoefficientsDesign( _
    ByVal iFilterCharacteristics As IIRFilterCharacteristics, _
    ByRef oFilterCluster As IIRFilterCluster) _
    As Integer
```

5.7.2 VQ_IIR Cascade Filter.

The IIR filters are setup as 2 separate VIs: a conventional LabVIEW based filter design VI and a DSP VQ_IIR Cascade Coefficient Design VI. The VQ_IIR Cascade Filter VI is implemented on the DSP using the well known cascade formula. Please refer to the Analysis Library Reference Manual.



Constant Update? = LabVIEW boolean input that determines if the LabVIEW variables are to be constantly updated or not. When left in the 'false' position, the LabVIEW constants are only updated once during the queueing phase thereby avoiding DSP-Host communications overhead. When left in the default 'true' position, LabVIEW constants are continuously updated thereby implementing real time variable updates for maximum flexibility.

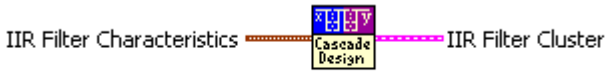
VQ_X{n} = VQ input.
IIR Filter Cluster = LabVIEW cluster containing the forward and reverse coefficients. The cluster may be derived by using one of the outputs of the LabVIEW based design VIs.

VQ_Filtered X{n} = VQ output that returns a filtered vector only on the input vector in question.

QuBASE Equivalent.

```
'*************************************************************************
'       Function VQ_IIRCascadeFilter
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'               iirCluster, IIRFilterCluster
'               update, Integer
'
'       Output Parameters:
'               oVQFiltered, Integer
'               oOutputOffset, Integer
'               oPQCoeffs, Integer
'
'*************************************************************************

  Function VQ_IIRCascadeFilter( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef oPQCoeffs() As Integer, _
        ByRef iirCluster As IIRFilterCluster, _
        Optional ByRef vqXn As Integer = 0, _
        Optional ByRef oOutputOffset As Integer = 0, _
        Optional ByRef oVQFiltered As Integer = 0, _
        Optional ByRef update As Integer = 0) _
        As T_Error
```

5.7.2.1 SI_IIR Cascade Filter Coefficients Design.



IIR Cascade Filter Coefficients Design.vi

IIR Filter Characteristics = LabVIEW input cluster to define the IIR filter characteristics.

IIR Filter Cluster = LabVIEW cluster containing the forward and reverse coefficients.

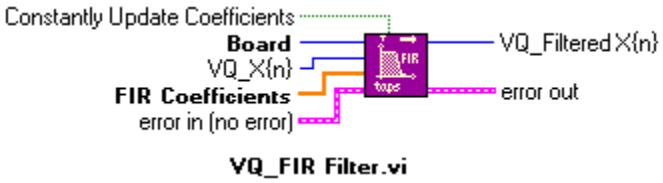
QuBASE Equivalent.

```
'**************************************************************************
'       Function SI_IIRCascadeFilterCoefficientsDesign
'
'       Description:
'
'       Input Parameters:
'               iFilterCharacteristics, IIRFilterCharacteristics
'
'       Output Parameters:
'               oFilterCluster, IIRFilterCluster
'
'**************************************************************************

  Function SI_SI_IIRCascadeFilterCoefficientsDesign( _
    ByVal iFilterCharacteristics As IIRFilterCharacteristics, _
    ByRef oFilterCluster As IIRFilterCluster) _
    As Integer
```

5.7.3 VQ_FIR Filter.

The FIR filters are setup as 2 separate VIs: a conventional LabVIEW based filter design VI and a VQ_FIR Filter VI. The VQ_FIR Filter VI is implemented on the DSP using the well known finite convolution or recursive formula. Please refer to the Analysis Library Reference Manual.

The LabVIEW based VQ_FIR Filter Design VI is merely an example of a way to design and build an array of FIR coefficients. An array of FIR coefficients can be built alternatively by using the FIR filter VIs supplied with the LabVIEW Analysis Library. However, the diagram is much cleaner when used with the supplied single icon VQ_FIR Filter Design VI.



Constant Update? = LabVIEW boolean input that determines if the LabVIEW variables are to be constantly updated or not. When left in the 'false' position, the LabVIEW constants are only updated once during the queueing phase thereby avoiding DSP-Host communications overhead. When left in the default 'true' position, LabVIEW constants are continuously updated thereby implementing real time variable updates for maximum flexibility.

VQ_X{n} = VQ input.
FIR Coefficients = LabVIEW array containing the FIR coefficients. The array may be derived by using one of the outputs of the LabVIEW based design VIs.

VQ_Filtered X{n} = VQ output that returns a filtered vector only on the input vector in question.

QuBASE Equivalent.

```
'*********************************************************************
'       Function VQ_FIRFilter
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'               firCoeffs, Double
'               update, Integer
'
'       Output Parameters:
'               oVQFilteredOutput, Integer
'
'*********************************************************************

  Function VQ_FIRFilter( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef firCoeffs() As Double, _
        Optional ByRef oVQFilteredOutput As Integer = 0, _
        Optional ByRef vqXn As Integer = 0, _
        Optional ByRef update As Integer = 0) _
        As T_Error
```

5.7.3.1 SI_FIR Filter Coefficients Design.

SI_FIR Filter Coefficients Design.vi

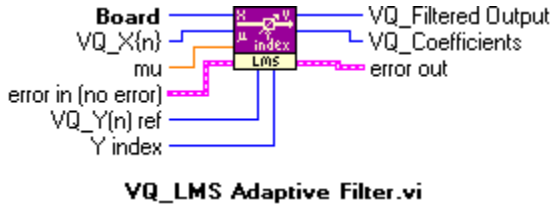
FIR Filter Characteristics = LabVIEW input cluster to define the FIR filter characteristics.

FIR Filter Cluster = LabVIEW output cluster containing the forward coefficients.

QuBASE Equivalent.

```
'**************************************************************************
'       Function SI_FIRFilterCoefficientsDesign
'
'       Description:
'
'       Input Parameters:
'               firFilterCharacteristics, FIRFilterCharacteristics,
'
'       Output Parameters:
'               oFirCoefficients, Double
'
'**************************************************************************

  Function SI_FIRFilterCoefficientsDesign( _
    ByVal firFilterCharacteristics As FIRFilterCharacteristics, _
    ByRef oFirCoefficients() As Double) _
    As Integer
```

5.7.4 VQ_LMS Adaptive Filter.



VQ_LMS Adaptive Filter.vi

VQ_X{n} = VQ input.

Scale Factor (mu) = LabVIEW constant, representing the compensation factor that is multiplied by the error between the actual signal and the desired signal.

VQ_Y{n} ref = VQ input, reference source signal.

Y Index = LabVIEW input, determines the index point of the input Y[n} vector.

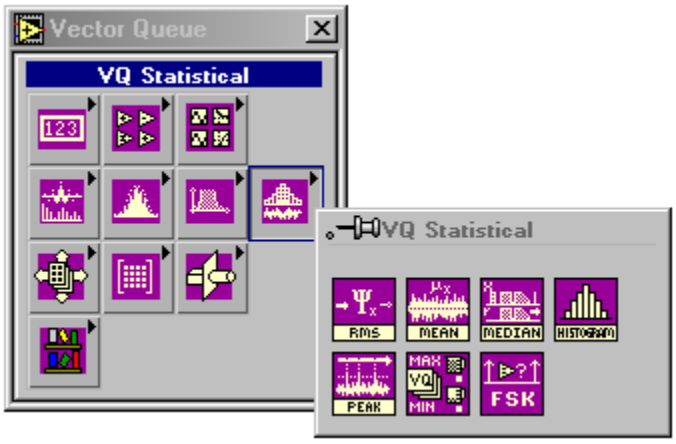
VQ_Filtered Output = VQ output that returns a continuous filtered value without the need for initial conditions, analogous to a conventional analog filter.

VQ_Coefficients = VQ output that simply mirrors in the VQ domain the actual coefficients used by the LMS algorithm.

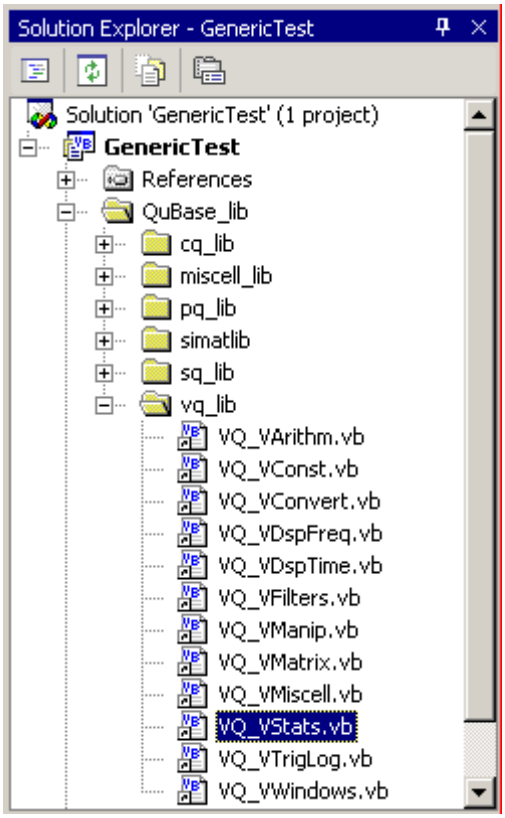
QuBASE Equivalent.

```
'*********************************************************************
'       Function VQ_LMSAdaptiveFilter
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               mode, Integer
'               numberOfWeights, Integer
'               vqXn, Integer
'               mu, Double
'               vqYnRef, Integer
'               yIndex, Integer
'               clearCoefficientsTF, Integer
'
'       Output Parameters:
'               oVQFilteredOutput, Integer
'               oVQCoefficients, Integer
'
'*********************************************************************

  Function VQ_LMSAdaptiveFilter( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef oVQFilteredOutput As Integer = 0, _
        Optional ByRef oVQCoefficients As Integer = 0, _
        Optional ByRef vqXn As Integer = 0, _
        Optional ByRef mu As Double = 0, _
        Optional ByRef vqYnRef As Integer = 0, _
        Optional ByRef yIndex As Integer = 0, _
        Optional ByRef clearCoefficientsTF As Integer = 0) _
        As T_Error
```

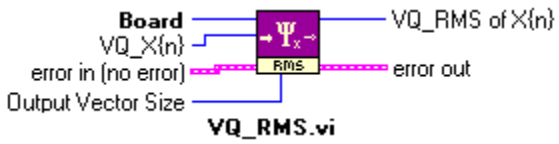
5.8 VQ Statistical.



QuBASE Equivalent.



5.8.1 VQ_RMS.



VQ_X{n} = VQ input.

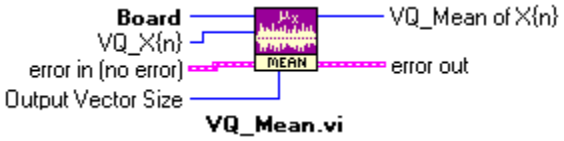
Output Size = LabVIEW input. Specifies the output vector size such that each element is the RMS of a Vector.

VQ_RMS of X{n} = VQ output.

QuBASE Equivalent.

```
'*********************************************************************
'       Function VQ_RMS
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'               outputSize, Integer
'
'       Output Parameters:
'               oVQYn, Integer
'
'*********************************************************************

  Function VQ_RMS( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef vqXn As Integer = 0, _
        Optional ByRef outputSize As Integer = 0, _
        Optional ByRef oVQYn As Integer = 0) _
        As T_Error
```

5.8.2 VQ_Mean.



VQ_X{n} = VQ input.

Output Vector Size = LabVIEW input. Specifies the output vector size such that each element is the Mean of a Vector.

VQ_Mean of X{n} = VQ output.

QuBASE Equivalent.

```
'*************************************************************************
'       Function VQ_Mean
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'               outputSize, Integer
'
'       Output Parameters:
'               oVQYn, Integer
'
'*************************************************************************

  Function VQ_Mean( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef oVQYn As Integer = 0, _
        Optional ByRef vqXn As Integer = 0, _
        Optional ByRef outputSize As Integer = 0) _
        As T_Error
```

5.8.3 VQ_Median.

VQ_X{n} = VQ input.

Rank: Output Vector Size = LabVIEW input. Specifies the output vector size such that each element is the Median of a Vector.

VQ_Median of X{n} = VQ output.

QuBASE Equivalent.

```
'*************************************************************************
'       Function VQ_Median
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'               width, Integer
'
'       Output Parameters:
'               oVQMedian, Integer
'
'*************************************************************************

  Function VQ_Median( _
            ByVal Board As Integer, _
            ByVal errIn As T_Error, _
            ByRef oVQMedian As Integer, _
            ByVal vqXn As Integer, _
            Optional ByVal width As Integer = 100) _
            As T_Error
```

5.8.4 VQ_Histogram.

This VI will produce an output array that contains a histogram of the input array. The Range input determines what range of values will be included in the histogram. The lowest value that the Histogram will use is 0. This means that any zero centered data must be offset so that it is centered in the specified range.

The Histogram function places data in bins only on integer boundaries. Therefore, if higher precision is needed, the data should be scaled appropriately.

EXAMPLE: If the data to be used ranges from -5 to +5, and we wish to see intervals of 0.5 the data would have to be scaled as follows:

(input array + 5) * 2

Adding 5 centers our data in the array, and multiplying by two allows the 0.5 intervals. The RANGE input for this example would need to be set to 20.

VQ_X{n} = VQ input.
Range = LabVIEW input, range of values which Histogram will be included in the histogram.

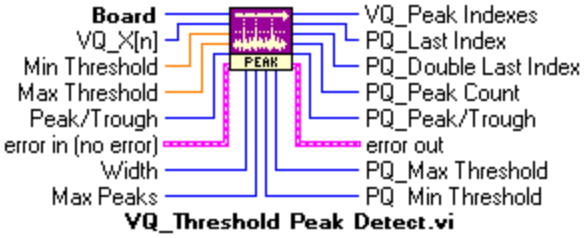
VQ_Histogram = VQ output.

QuBASE Equivalent.

```
'*************************************************************************
'       Function VQ_Histogram
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'               range, Integer
'
'       Output Parameters:
'               oVQYn, Integer
'
'*************************************************************************

   Function VQ_Histogram( _
            ByRef board As Integer, _
            ByRef errIn As T_Error, _
            Optional ByRef vqXn As Integer = 0, _
            Optional ByRef range As Integer = 0, _
            Optional ByRef oVQYn As Integer = 0) _
            As T_Error
```

5.8.5 VQ_Threshold Peak Detect.

This VI outputs an array of indexes detailing the location of peaks inside a vector, with detailed criteria.



VQ_X{n} = VQ input to be evaluated.

Min Threshold = LabVIEW input, minimum value for an element to be considered a valid peak. If the element falls out of the range set by the min/max thresholds, no count is made and the element is ignored.

Max Threshold = LabVIEW input, maximum value for an element to be considered a valid peak. If the element falls out of the range set by the min/max thresholds, no count is made and the element is ignored.

NOTE: A peak is determined when the input element exceeds the min threshold, then goes below it again, without exceeding the max threshold. If the max threshold is exceeded, the peak is ignored.

Peak/Trough = LabVIEW input, determines the detection of a peak or trough.

0 = Detect positive going peaks.

1 = negative going troughs.

Width = LabVIEW input, indicates the minimum number of points above the minimum threshold in order for the peak to be counted. If this parameter is set to '0', then all peaks will be detected regardless of width.

Max Peaks = LaVIEW input, used to determine the maximum number of peaks to detect and count. If this number is exceeded, subsequent peaks are ignored.

VQ_Peak Indexes = VQ output, index numbers that indicate the location of the peaks and troughs.

PQ_Last Index = PQ output, index that indicates the location of the last peak inside of the input vector.

PQ_Double Last Index = PQ output, index that indicates the location of the last peak in the previous array.

NOTE: This index will always be negative since the first point in the current array is at index '0'.

PQ_Peak Count = PQ output, count of peaks/troughs detected.

PQ_Peak/Trough = PQ output, highest Peak/Trough value detected in the vector.

PQ_Max Threshold = PQ output, value of Maximum Threshold.

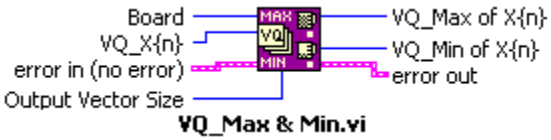
PQ_Min Threshold = PQ output, value of Minimum Threshold.

QuBASE Equivalent.

```
'*************************************************************************
'       Function VQ_ThresholdPeakDetect
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'               width, Integer
'               maxPeaks, Integer
'               peakTrough, Integer
'               minThreshold, Integer
'               maxThreshold, Integer
'
'       Output Parameters:
'               oVQPeakIndexes, Integer
'               oPQPeakTrough, Integer
'               oPQMinThreshold, Integer
'               oPQMaxThreshold, Integer
'               oPQLastIndex, Integer
'               oPQDoubleLastIndex, Integer
'               oPQPeakCount, Integer
'
'*************************************************************************

  Function VQ_ThresholdPeakDetect( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef maxPeaks As Integer = 200, _
        Optional ByRef vqXn As Integer = 0, _
        Optional ByRef width As Integer = 0, _
        Optional ByRef minThreshold As Double = 0, _
        Optional ByRef maxThreshold As Double = 6, _
        Optional ByRef peakTrough As Integer = 0, _
        Optional ByRef oVQPeakIndexes As Integer = 0, _
        Optional ByRef oPQPeakTrough As Integer = 0, _
        Optional ByRef oPQMinThreshold As Integer = 0, _
        Optional ByRef oPQMaxThreshold As Integer = 0, _
        Optional ByRef oPQLastIndex As Integer = 0, _
        Optional ByRef oPQDoubleLastIndex As Integer = 0, _
        Optional ByRef oPQPeakCount As Integer = 0) _
        As T_Error
```

5.8.6 VQ_Max & Min.

Obtains the maximum and minimum values of a VQ vector.



VQ_X{n} = VQ input.
Output Vector Size = LabVIEW input. Specifies the output vector size such that each element is the Max or Min of a Vector.

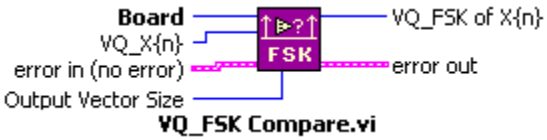
VQ_Max of X{n} = VQ output, vector with the maximum values detected in a vector.
VQ_Min of X{n} = VQ output, vector with the minimum values detected in a vector.

QuBASE Equivalent.

```
'*************************************************************************
'       Function VQ_MaxAndMin
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'               outputSize, Integer
'
'       Output Parameters:
'               oVQMax, Integer
'               oVQMin, Integer
'
'*************************************************************************

  Function VQ_MaxAndMin( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef oVQMax As Integer = 0, _
        Optional ByRef oVQMin As Integer = 0, _
        Optional ByRef vqXn As Integer = 0, _
        Optional ByRef outputSize As Integer = 0) _
        As T_Error
```

5.8.7 VQ_FSK Compare.

Not used.



VQ_X{n} = VQ input ? ? ?

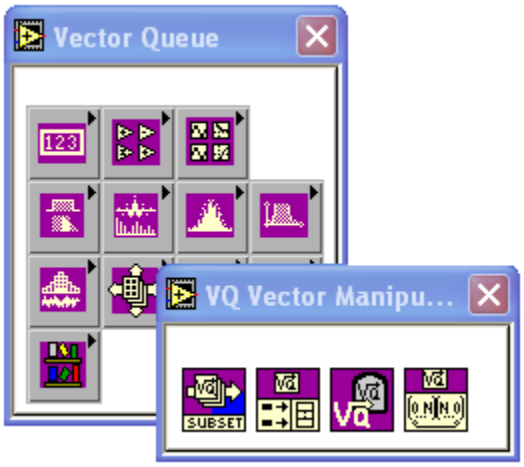
OutputVector Size = LabVIEW input, defines the size of VQ_FSK of X{n}? ? ?

VQ_FSK of X{n} = VQ output ? ? ?

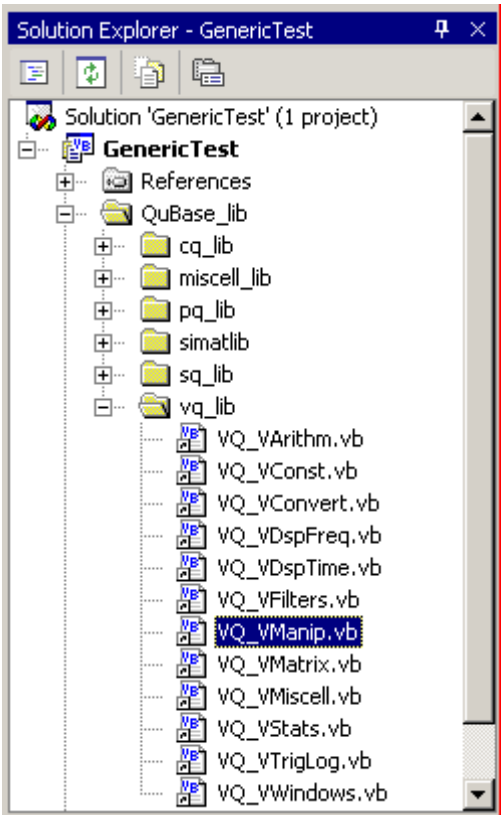
QuBASE Equivalent.

```
'**********************************************************************
'       Function VQ_FSKCompare
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'               outputSize, Integer
'
'       Output Parameters:
'               oVQYn, Integer
'
'**********************************************************************

   Function VQ_FSKCompare( _
          ByRef board As Integer, _
          ByRef errIn As T_Error, _
          Optional ByRef oVQYn As Integer = 0, _
          Optional ByRef vqXn As Integer = 0, _
          Optional ByRef outputSize As Integer = 0) _
          As T_Error
```

5.9 VQ Vector Manipulation.

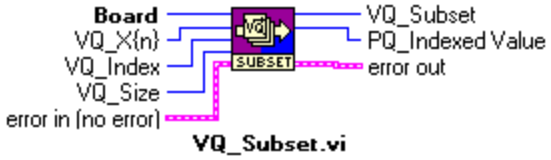


QuBASE Equivalent.



5.9.1 VQ_Subset.

The VQ_Subset VI returns a portion of a VQ input starting at an index and of a predefinable size.



VQ_X{n} = VQ input.
VQ_Index = VQ input, used to determine the location of the first point within the current vector that starts the new vector block.
VQ_Size = VQ input, used to determine the size of the new vector block.

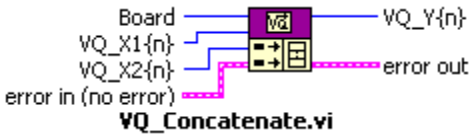
VQ_Subset = VQ output, new vector subset.
PQ_Indexed Value = PQ output, value extracted from the vector as indexed by the 'VQ_Index' input.

QuBASE Equivalent.

```
'**********************************************************************
'       Function VQ_Subset
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'               vqIndex, Integer
'               vqSize, Integer
'
'       Output Parameters:
'               oVQSubset, Integer
'               opqIndexedValue, Integer
'
'**********************************************************************

  Function VQ_Subset( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef oVQSubset As Integer = 0, _
        Optional ByRef opqIndexedValue As Integer = 0, _
        Optional ByRef vqXn As Integer = 0, _
        Optional ByRef vqIndex As Integer = 0, _
        Optional ByRef vqSize As Integer = 0) _
        As T_Error
```

5.9.2 VQ_Concatenate Vectors.

Concatenates the two input vectors into a single output vector.



VQ_X1{n} = VQ input vector.
VQ_X2{n} = VQ input vector.

VQ_Y{n} = Resulting single VQ output, with VQ_X2{n} concatenated after VQ_X1{n}.

QuBASE Equivalent.

```
'*********************************************************************
'       Function VQ_Concatenate
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn1, Integer
'               vqXn2, Integer
'
'       Output Parameters:
'               oVQYn, Integer
'
'*********************************************************************

  Function VQ_Concatenate( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef oVQYn As Integer = 0, _
        Optional ByRef vqXn1 As Integer = 0, _
        Optional ByRef vqXn2 As Integer = 0) _
        As T_Error
```

5.9.3 VQ_Mirror.

This VI is useful for turning the output of a single sided Real FFT into a full sided Real FFT.

VQ_X{n} = VQ input.

VQ_Y{n} = VQ output that is doubled in size, with the new half having its elements reversed.

QuBASE Equivalent.

```
'*************************************************************************
'       Function VQ_Mirror
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'
'       Output Parameters:
'               oVQYn, Integer
'
'*************************************************************************

  Function VQ_Mirror( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef oVQYn As Integer = 0, _
        Optional ByRef vqXn As Integer = 0) _
        As T_Error
```

5.9.4 VQ_Reverse.

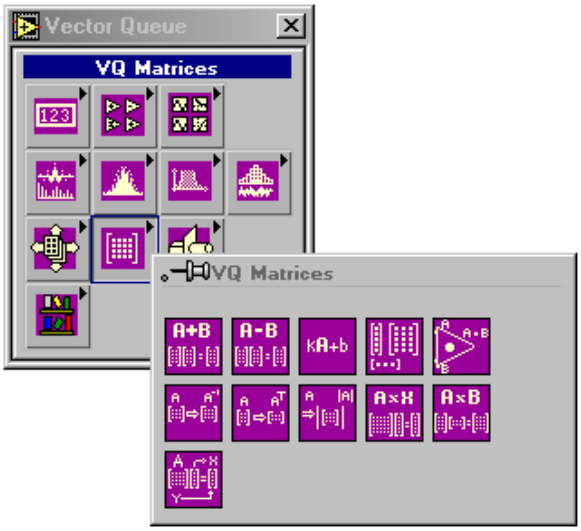
VQ_X{n} = VQ input.

VQ_Y{n} = VQ output with its elements reversed.

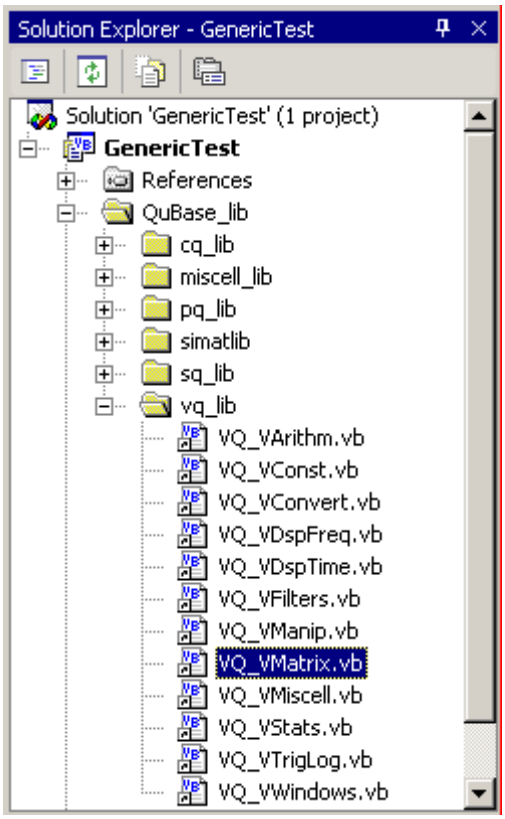
QuBASE Equivalent.

```
'*************************************************************************
'       Function VQ_Reverse
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'
'       Output Parameters:
'               oVQYn, Integer
'
'*************************************************************************

   Function VQ_Reverse( _
          ByRef board As Integer, _
          ByRef errIn As T_Error, _
          Optional ByRef oVQYn As Integer = 0, _
          Optional ByRef vqXn As Integer = 0) _
          As T_Error
```

5.10 VQ Matrices.



QuBASE Equivalent.



5.10.1 VQ_Matrix Addition.

Computes the addition of two NxM matrices.

VQ_[A] = VQ Input vector.
VQ_[B] = VQ Input vector.

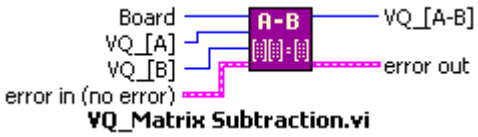
VQ_[A+B] = VQ output, elements are the addition of each corresponding A and B element.

QuBASE Equivalent.

```
'************************************************************************
'       Function VQ_MatrixAddition
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'               vqYn, Integer
'
'       Output Parameters:
'               oVQXPlusY, Integer
'
'************************************************************************

  Function VQ_MatrixAddition( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef vqXn As Integer = 0, _
        Optional ByRef vqYn As Integer = 0, _
        Optional ByRef oVQXPlusY As Integer = 0) _
        As T_Error
```

5.10.2 VQ_Matrix Subtraction.

Computes the subtraction of one NxM matrix from another.



VQ_[A] = VQ Input vector
VQ_[B] = VQ Input vector

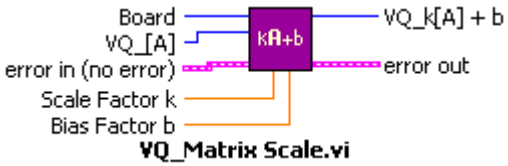
VQ_[A-B] = VQ output, elements are the difference of each corresponding A and B element.

QuBASE Equivalent.

```
'**************************************************************************
'       Function VQ_MatrixSubtraction
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'               vqYn, Integer
'
'       Output Parameters:
'               oVQXMinusY, Integer
'
'**************************************************************************

  Function VQ_MatrixSubtraction( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef vqXn As Integer = 0, _
        Optional ByRef vqYn As Integer = 0, _
        Optional ByRef oVQXMinusY As Integer = 0) _
        As T_Error
```

5.10.3 VQ_Matrix Scale.

Scales each element of an NxM matrix with a gain factor "k" and a bias factor "b."



VQ_[A] = VQ Input vector.
Scale Factor k = LabVIEW input, Gain factor of scale.
Scale Factor b = LabVIEW input, Bias factor of scale.

VQ_k[A] + b = VQ output which elements are obtained multiplying each element of VQ_A by the gain factor "k", then added the bias factor "b".

QuBASE Equivalent.

```
'************************************************************************
'       Function VQ_MatrixScale
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               ilngVQX, Integer
'               isngk, Double
'               isngb, Double
'
'       Output Parameters:
'               olngVQMatrixScale, Integer
'
'************************************************************************

  Function VQ_MatrixScale( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef ilngVQX As Integer, _
        Optional ByRef isngk As Double = 1, _
        Optional ByRef isngb As Double = 0, _
        Optional ByRef olngVQMatrixScale As Integer = 0) _
        As T_Error
```

5.10.4 VQ_Matrix Outer Product.

Computes the Outer Product of two NxM matrices.

VQ_[A] = VQ input vector
VQ_[B] = VQ input vector

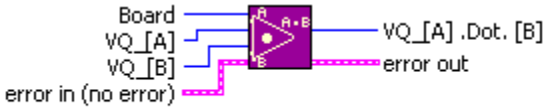
VQ_[Ai][Bj] = VQ Outer Product of VQ_A by VQ_B

QuBASE Equivalent.

```
'*************************************************************************
'       Function VQ_MatrixOuterProduct
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'               vqYn, Integer
'
'       Output Parameters:
'               oVQOuterProduct, Integer
'
'*************************************************************************

  Function VQ_MatrixOuterProduct( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef vqXn As Integer = 0, _
        Optional ByRef vqYn As Integer = 0, _
        Optional ByRef oVQOuterProduct As Integer = 0) _
        As T_Error
```

5.10.5 VQ_A .Dot. B.

Computes the "Scalar" or "Dot" product of two Nx1 matrices.



VQ_A .Dot. B.vi

VQ_[A] = VQ input vector
VQ_[B] = VQ input vector

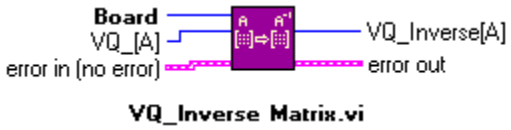
VQ_[A].Dot.[B] = VQ Dot Product of VQ_A by VQ_B

QuBASE Equivalent.

```
'**************************************************************************
'       Function VQ_AdotB
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               ilngVQA, Integer
'               ilngVQB, Integer
'
'       Output Parameters:
'               olngVQAdotB, Integer
'
'**************************************************************************

  Function VQ_AdotB( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef ilngVQA As Integer, _
        ByRef ilngVQB As Integer, _
        Optional ByRef olngVQAdotB As Integer = 0) _
        As T_Error
```

5.10.6 VQ_Inverse Matrix.

Computes the Inverse of an NxM matrix.



VQ_[A] = VQ input matrix.

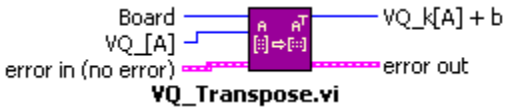
VQ_Inverse[A] = The resultant Inverse of VQ_[A] matrix.

QuBASE Equivalent.

```
'************************************************************************
'       Function VQ_InverseMatrix
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               ilngVQMatrixX, Integer
'
'       Output Parameters:
'               olngVQInverseMatrixX, Integer
'
'************************************************************************

  Function VQ_InverseMatrix( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef ilngVQMatrixX As Integer, _
        Optional ByRef olngVQInverseMatrixX As Integer = 0) _
        As T_Error
```

5.10.7 VQ_Transpose.

Transposes an NxM matrix.



VQ_[A] = VQ input matrix.

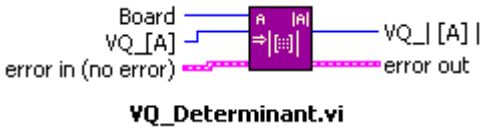
VQ_Transpose[A] = The resultant Transposed of VQ_[A] matrix.

QuBASE Equivalent.

```
'*************************************************************************
'       Function VQ_Transpose
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               ilngVQA, Integer
'
'       Output Parameters:
'               olngVQTransposeofA, Integer
'
'*************************************************************************

  Function VQ_Transpose( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef ilngVQA As Integer, _
        Optional ByRef olngVQTransposeofA As Integer = 0) _
        As T_Error
```

5.10.8 VQ_Determinant.

Computes the Determinant of a square NxN matrix.



VQ_[A] = VQ input matrix.

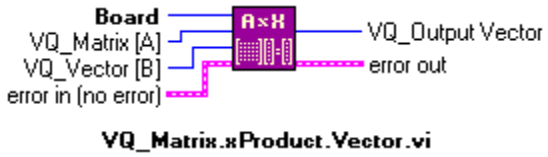
VQ_Determinant[A] = The resultant Determinant matrix of VQ_[A].

QuBASE Equivalent.

```
'**************************************************************************
'       Function VQ_Determinant
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               ilngVQA, Integer
'
'       Output Parameters:
'               olngVQDeterminantofA, Integer
'
'**************************************************************************

  Function VQ_Determinant( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef ilngVQA As Integer, _
        Optional ByRef olngVQDeterminantofA As Integer = 0) _
        As T_Error
```

5.10.9 VQ_AxX{n}.

Computes the "Cross" product of an [NxR] matrix times an [Rx1] matrix or vector, and returns a single Nx1 matrix or vector.



VQ_[A] = VQ input vectors, represents an [NxR] matrix.
VQ_X{n} = VQ input vectors, represents an [Rx1] vector.

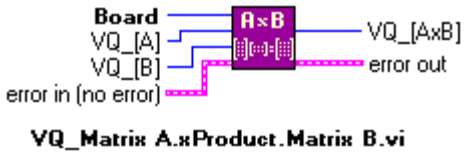
VQ_[A x X{n}] = VQ Cross Product of a matrix VQ_A by vector VQ_X{n}.

QuBASE Equivalent.

```
'**************************************************************************
'       Function VQ_AxXVector
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               ilngVQA, Integer
'               ilngVQX, Integer
'
'       Output Parameters:
'               olngVQAxXVector, Integer
'
'**************************************************************************

  Function VQ_AxXVector( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef ilngVQA As Integer, _
        ByRef ilngVQB As Integer, _
        Optional ByRef olngVQAxXVector As Integer = 0) _
        As T_Error
```

5.10.10 VQ_AxB.

Computes the "Cross" product of an [NxR] matrix times an [RxM] matrix, and returns a single NxM matrix.



VQ_Matrix A.xProduct.Matrix B.vi

VQ_[A] = VQ input vectors, represents an [NxR] matrix.
VQ_[B] = VQ input vectors, represents an [RxM] matrix.

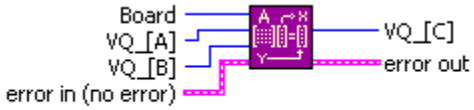
VQ_[A x B] = VQ Cross Product of matrix VQ_A by matrix VQ_B.

QuBASE Equivalent.

```
'************************************************************************
'       Function VQ_AxB
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               ilngVQA, Integer
'               ilngVQB, Integer
'
'       Output Parameters:
'               olngVQAxB, Integer
'
'************************************************************************

  Function VQ_AxB( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef ilngVQA As Integer, _
        ByRef ilngVQB As Integer, _
        Optional ByRef olngVQAxB As Integer = 0) _
        As T_Error
```

5.10.11 VQ_Solve Linear Equations.

Solves for a linear system, with variables represented by a square NxN matrix.



VQ_Solve Linear Equations .vi

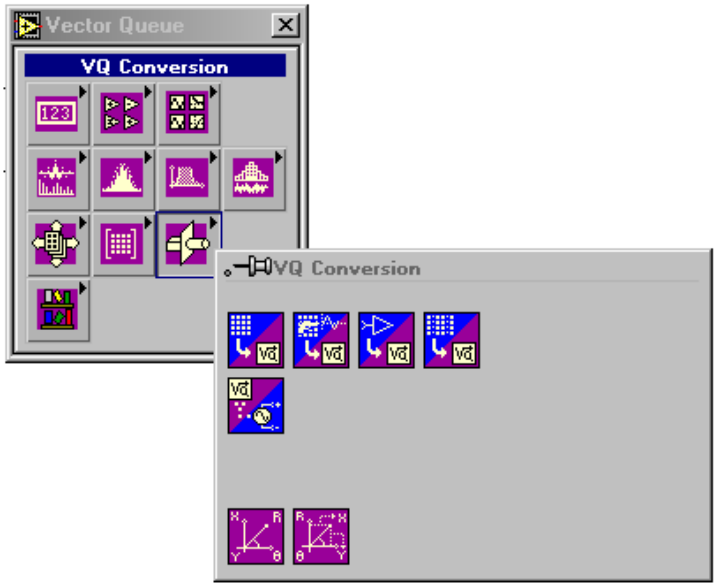
VQ_[A] = VQ input matrix.
VQ_[B] = VQ input matrix.

VQ_[C] = Resultant VQ matrix containing the linear parameters for the input matrices.

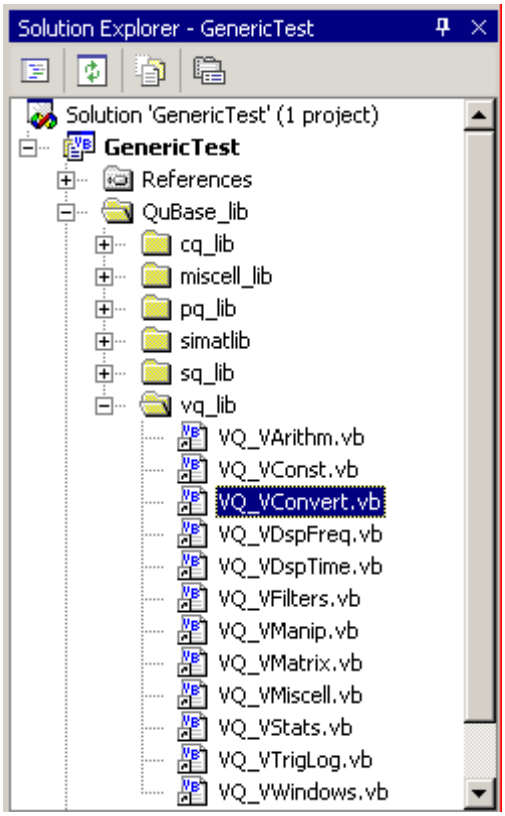
QuBASE Equivalent.

```
'**************************************************************************
'       Function VQ_SolveLinearEquations
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               ilngVQA, Integer
'               ilngVQB, Integer
'
'       Output Parameters:
'               olngVQSolveLinearEquations, Integer
'
'**************************************************************************

  Function VQ_SolveLinearEquations( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef ilngVQA As Integer, _
        ByRef ilngVQB As Integer, _
        Optional ByRef olngVQSolveLinearEquations As Integer = 0) _
        As T_Error
```

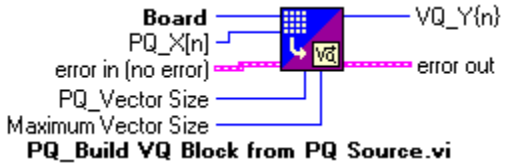
5.11 VQ Conversion.



QuBASE Equivalent.



5.11.1 VQ_Build VQ Block From PQ Source.



PQ_X[n] = PQ input used to determine the input source from which to build a vector block.

PQ Vector Size = PQ input, used to determine the number of points from which to build the vector block.

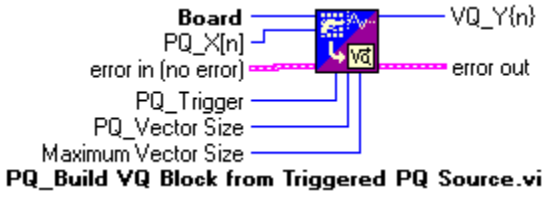
Maximum Vector Size = LabVIEW input, used to set a limit for a maximum vector size. Useful for reserving a maximum fixed VQ buffer, thereby allowing the dynamic resize of the resultant vector within the DSP domain.

VQ_Y{n} = VQ output, containing the number of points as defined by the 'PQ_Vector Size' input.

QuBASE Equivalent.

```
'**********************************************************************
'       Function VQ_Build VQ Block From PQ Source
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqXn, Integer
'               pqVectorSize, Integer
'               maxVectorSize, Integer:
'                       defaulted to 512
'
'       Output Parameters:
'               oVQYn, Integer
'
'**********************************************************************

  Function PQ_BuildVQBlockFromPQSource( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef pqXn As Integer = 0, _
        Optional ByRef oVQYn As Integer = 0, _
        Optional ByRef maxVectorSize As Integer = 500, _
        Optional ByRef pqVectorSize As Integer = 0) _
        As T_Error
```

5.11.2 VQ_Build VQ Block From Triggered PQ Source.



PQ_X[n] = PQ input used to determine the triggered input source from which to build a vector block.

PQ_Trigger = PQ input, used as an enable or trigger marker to begin collecting data to fill the vector.

PQ Vector Size = PQ input, used to determine the number of points from which to build the vector block.

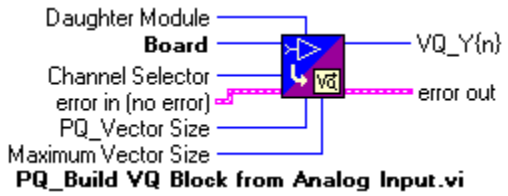
Maximum Vector Size = LabVIEW input, used to set a limit for a maximum vector size. Useful for reserving a maximum fixed VQ buffer, thereby allowing the dynamic resize of the resultant vector within the DSP domain.

VQ_Y{n} = VQ output, containing the number of points as defined by the 'PQ_Vector Size' input.

QuBASE Equivalent.

```
'*********************************************************************
'       Function VQ_Build VQ Block From Triggered PQ Source
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqXn, Integer
'               pqVectorSize, Integer
'               maxVectorSize, Integer:
'               defaulted to 500
'
'       Output Parameters:
'               oVQYn, Integer
'
'*********************************************************************

  Function PQ_BuildVQBlockFromTriggeredPQSource( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef oVQYn As Integer = 0, _
        Optional ByRef pqXn As Integer = 0, _
        Optional ByRef pqTrigger As Integer = 0, _
        Optional ByRef maxVectorSize As Integer = 500, _
        Optional ByRef pqVectorSize As Integer = 0) _
        As T_Error
```

5.11.3 VQ_Build VQ Block From Analog Input.



Channel Selector = LabVIEW input used to select the analog input channel of interest.

PQ Vector Size = PQ input, used to determine the number of points from which to build the vector block.

Maximum Vector Size = LabVIEW input, used to set a limit for a maximum vector size. Useful for reserving a maximum fixed VQ buffer, thereby allowing the dynamic resize of the resultant vector within the DSP domain.

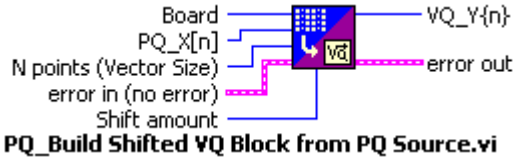
VQ_Y{n} = VQ output, containing the number of points as defined by the 'PQ_Vector Size' input.

QuBASE Equivalent.

```
'*********************************************************************
'       Function VQ_Build VQ Block From Analog Input
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               channelSelector, Integer
'               pqVectorSize, Integer
'               maxVectorSize, Integer:
'               defaulted to 512
'
'       Output Parameters:
'               oVQYn, Integer
'
'*********************************************************************

  Function PQ_BuildVQBlockFromAnalogInput( _
        ByRef board As Integer, _
        ByRef intModule As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef oVQYn As Integer = 0, _
        Optional ByRef channelSelector As Integer = 0, _
        Optional ByRef maxVectorSize As Integer = 512, _
        Optional ByRef pqVectorSize As Integer = 0) _
        As T_Error
```

5.11.4 VQ_Build Shifted VQ Block from PQ Source.

This VI will build a block much like the VQ_Build VQ Block From PQ Source VI. The difference is that the VI maintains the previous N-1 points, and appends the new point on to the end of the array. This provides a constantly shifting array that is useful for convolutions and filtering applications.



PQ_X[n] = PQ values.

N points (Vector Size) = LabVIEW input, used to determine the number of points to group into a VQ vector.

Shift amount = LabVIEW input, the amount with which to shift the incoming PQ data set.

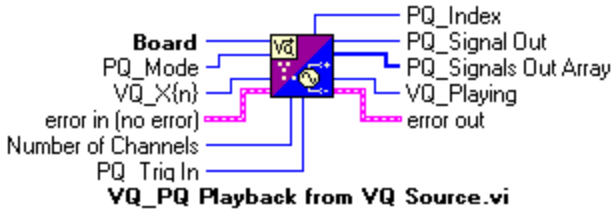
VQ_Y{n} = VQ vector output.

QuBASE Equivalent.

```
'*************************************************************************
'       Function VQ_Build Shifted VQ Block From PQ Source
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqXn, Integer
'               shiftAmount, Integer
'               vectorSizeN, Integer:
'               defaulted to 1k
'
'       Output Parameters:
'               oVQYn, Integer
'
'*************************************************************************

  Function PQ_BuildShiftedVQBlockFromPQSource( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef oVQYn As Integer = 0, _
        Optional ByRef pqXn As Integer = 0, _
        Optional ByRef shiftAmount As Integer = 1, _
        Optional ByRef vectorSizeN As Integer = 1024) _
        As T_Error
```

5.11.5 PQ Playback from VQ Source.

The VQ_Playback from VQ Source VI constantly plays back an array of values as sourced from a VQ input, thereby treating a vector as a pattern for waveform generation.



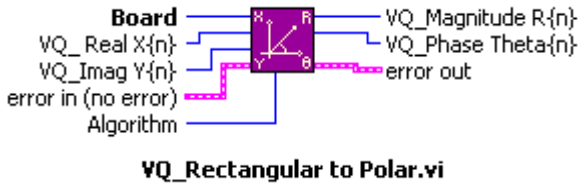
VQ_X{n} = VQ input, in a vector (array) form.
PQ_Mode = PQ input that selects between single shot or continous signal generation.
0 = Continous signal generation; default.
nonzero = Single shot generation.
PQ_Trig In = PQ input that enables or disables signal generation.
0 = Constantly enables signal generation; default.
nonzero = External mode control, disables signal generation.
Number of Channels = LabVIEW input to determine the number of output channels to reproduce the PQ output.

PQ_Index = PQ output of the index number indicating the relative position of the current output point within the waveform buffer; starts at '0' and ends at 'n-1'.
PQ_Signal Out = PQ output of signal being played back (point form).
PQ_Signals Out Array = PQ output array of multiple signals being played back (point form), number of signals played back depends on the 'Number of Channels' variable.
VQ_Playing = VQ output of buffer/signal being played back (vector form); used by PQ_Waveform Playback VI.

QuBASE Equivalent.

```
'*************************************************************************
'       Function VQ_PlaybackFromVQSource
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'               shiftAmount, Integer
'               vectorSizeN, Integer:
'               defaulted to 1k
'
'       Output Parameters:
'               oPQSignalOut, Integer
'               oVQPlaying, Integer
'
'*************************************************************************

   Function VQ_PlaybackFromVQSource( _
          ByRef board As Integer, _
          ByRef errIn As T_Error, _
          Optional ByRef oPQSignalOut As Integer = 0, _
          Optional ByRef oVQPlaying As Integer = 0, _
          Optional ByRef vqXn As Integer = 0, _
          Optional ByRef shiftAmount As Integer = 0, _
          Optional ByRef vectorSizeN As Integer = 1024) _
          As T_Error
```

5.11.6 VQ_Rectangular to Polar.



VQ_Rectangular to Polar.vi

VQ_Real X{n} = VQ input.
VQ_Imag Y{n} = VQ input.
Algorithm = LabVIEW input, selects the method for results as follows:

1) *Generate* = values are computed by the DSP using an accurate but relatively time consuming iterative process.

2) *Table* = values are sourced from a look up table for higher speed and interpolated for acceptable accuracy.

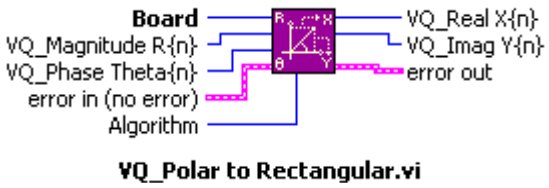
Table Resolution = LabVIEW input that determines the size of the look up table; defaulted to 1024.

VQ_Magnitude R{n} = VQ output.
VQ_Phase Theta{n} = VQ output.

QuBASE Equivalent.

```
'*********************************************************************
'       Function VQ_RectangularToPolar
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               ilngVQRealXn, Integer
'               ilngVQImagXn, Integer
'
'       Output Parameters:
'               olngVQMagnitude, Integer
'               olngVQPhase, Integer
'
'*********************************************************************

  Function VQ_RectangularToPolar( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef ilngVQRealXn As Integer, _
        ByRef ilngVQImagXn As Integer, _
        Optional ByRef olngVQMagnitude As Integer = 0, _
        Optional ByRef olngVQPhase As Integer = 0) _
        As T_Error
```

5.11.7 VQ_Polar to Rectangular.



VQ_Magnitude R{n} = VQ input.
VQ_Phase Theta{n} = VQ input.
Algorithm = LabVIEW input, selects the method for results as follows:

1) *Generate* = values are computed by the DSP using an accurate but relatively time consuming iterative process.

2) *Table* = values are sourced from a look up table for higher speed and interpolated for acceptable accuracy.

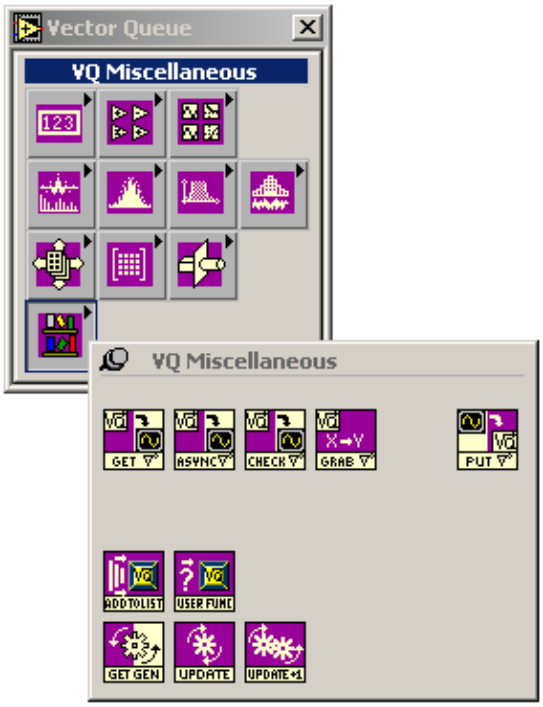
Table Resolution = LabVIEW input that determines the size of the look up table; defaulted to 1024.

VQ_Real X{n} = VQ output.
VQ_Imag Y{n} = VQ output.

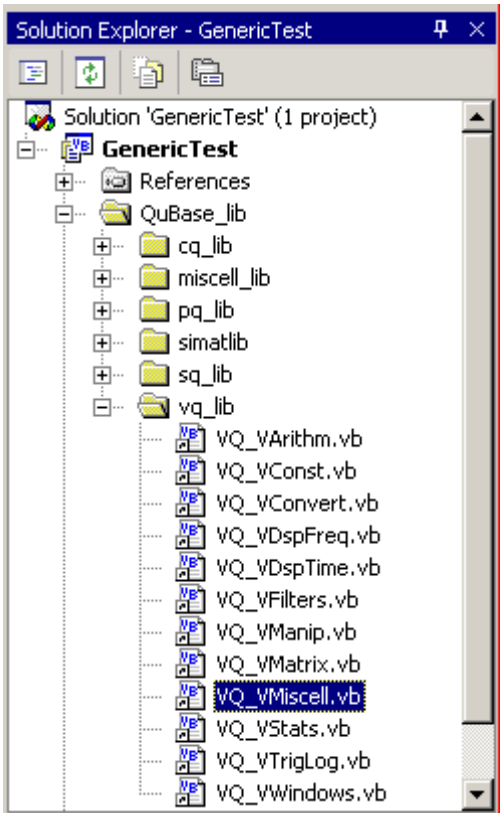
QuBASE Equivalent.

```
'*********************************************************************
'       Function VQ_PolarToRectangular
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqMagnitude, Integer
'               vqPhase, Integer
'
'       Output Parameters:
'               oVQReal, Integer
'               oVQImag, Integer
'
'*********************************************************************

  Function VQ_PolarToRectangular( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef vqMagnitude As Integer = 0, _
        Optional ByRef vqPhase As Integer = 0, _
        Optional ByRef oVQReal As Integer = 0, _
        Optional ByRef oVQImag As Integer = 0) _
        As T_Error
```

5.12 VQ Miscellaneous.

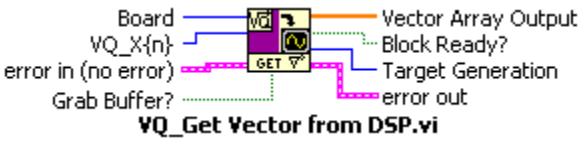


QuBASE Equivalent.



5.12.1 VQ_Get Vector from DSP.

This VI, when running in 'Wait' mode, assumes the host will be fast enough to pickup every processed vector. This assumption requires a very fast host or a slow set of vector operations. The 'False' case will pick up a buffer regardless of whether it is ready. The 'Block ready' output displays if this has occurred. Especialy useful for quickly displaying data from a vector.



VQ_X{n} = VQ input designating the address where the incoming vector is to be placed.

Grab Buffer? = LabVIEW boolean input, used to indicate whether or not to accept an upcoming vector input.

F = Grab: When False, the currently available input vector is used.

T = Wait: Forces the processing to wait for a valid synchronized input vector, useful in applications where no points can be dropped.

Vector Array Output = LabVIEW array of binary data read from DSP.

Block Ready? = LabVIEW boolean that indicates whether or not the current VQ input is available or not.

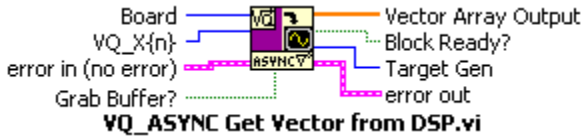
Target Generation = LabVIEW output that indicates the current generation number for the current VQ input; used mainly for debugging.

QuBASE Equivalent.

```
'**********************************************************************
'       Function VQ_GetVectorFromDSP
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'               grabWait, Boolean = False
'
'       Output Parameters:
'               oTargetGen, Integer
'               oVectorArray, Double
'               oBlockReady, Boolean = False
'
'**********************************************************************

  Function VQ_GetVectorFromDSP( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef oVectorArray() As Double, _
        Optional ByRef oBlockReady As Boolean = False, _
        Optional ByRef oTargetGen As Integer = 0, _
        Optional ByRef vqXn As Integer = 0, _
        Optional ByRef grabWait As Boolean = False) _
        As T_Error
```

5.12.2 VQ_ASYNC Get Vector from DSP.

This VI is similar to Get Vector from DSP. It is a littler slower but allows the host to pickup data when the host will lag behind the DSP. The Grab/Wait and ready terminals perform as mentioned in Get Vector from DSP.



VQ_X{n} = VQ input designating the address where the incoming vector is to be placed.
Grab Buffer? = LabVIEW boolean input, used to indicate whether or not to accept an upcoming vector input.

F = Grab: When False, the currently available input vector is used.

T = Wait: Forces the processing to wait for a valid synchronized input vector, useful in applications where no points can be dropped.

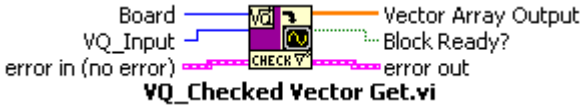
Vector Array Output = LabVIEW array of binary data read from DSP.
Block Ready? = LabVIEW boolean that indicates whether or not the current VQ input is available or not.
Target Generation = LabVIEW output that indicates the current generation number for the current VQ input; used mainly for debugging.

QuBASE Equivalent.

```
'*********************************************************************
'       Function VQ_AsyncGetVectorFromDSP
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'               grabWait, Boolean = False
'
'       Output Parameters:
'               oTargetGen, Integer
'               oVectorArray, Double
'               oBlockReady, Boolean = False
'               oVectorArraySize, Integer
'               oBlockPresentGen, Integer
'               oOutputBlockAddress, Integer
'
'*********************************************************************

  Function VQ_AsyncGetVectorFromDSP( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef oVectorArray() As Double, _
        Optional ByRef oVectorArraySize As Integer = 0, _
        Optional ByRef oTargetGen As Integer = 0, _
        Optional ByRef oBlockPresentGen As Integer = 0, _
        Optional ByRef oOutputBlockAddress As Integer = 0, _
        Optional ByRef oBlockReady As Boolean = False, _
        Optional ByRef vqXn As Integer = 0, _
        Optional ByRef grabWait As Boolean = False) _
        As T_Error
```

5.12.3 VQ_Checked Vector Get.

The VQ_Checked Vector Get VI allows the host to retrieve data after checking for validity.



VQ_Input = VQ input.

Vector Array Output = LabVIEW array of binary data read from DSP.
Block Ready? = LabVIEW boolean that indicates whether or not the current VQ input is available or not.

QuBASE Equivalent.

```
'**************************************************************************
'       Function VQ_CheckedVectorGet
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'
'       Output Parameters:
'               oVectorArray, Double
'               oBlockReady, Boolean = False
'
'**************************************************************************

  Function VQ_CheckedVectorGet( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef oVectorArray() As Double, _
        Optional ByRef oBlockReady As Boolean = False, _
        Optional ByRef vqXn As Integer = 0) _
        As T_Error
```

5.12.4 VQ_Grab Vector.

This function just copies the VQ input vector and places it into the output vector. Only copies only when generation is 'zero'.

VQ_X{n} = VQ input.

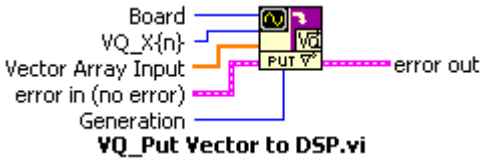
VQ_Y{n} = VQ output.

QuBASE Equivalent.

```
'*************************************************************************
'       Function VQ_GrabVector
'
'       Description:
'
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'
'       Output Parameters:
'               oVQYn, Integer
'
'*************************************************************************

   Function VQ_GrabVector( _
          ByRef board As Integer, _
          ByRef errIn As T_Error, _
          Optional ByRef oVQYn As Integer = 0, _
          Optional ByRef vqXn As Integer = 0) _
          As T_Error
```

5.12.5 VQ_Put Vector to DSP.

This VI performs a similar function as VQ_Constant, except that this one doesn't allocate memory space in the DSP; instead, it just writes to an allocated VQ block.



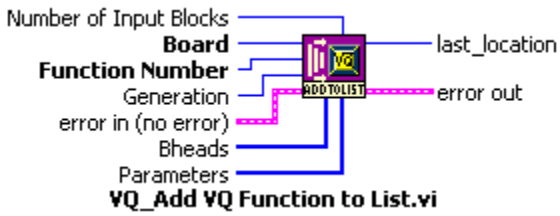
VQ_X{n} = VQ input.
Vector Array Input = LabVIEW array of binary data to be sent to the DSP.
Generation = LabVIEW input.

QuBASE Equivalent.

```
'*************************************************************************
'       Function VQ_OverwriteVectorToDSP
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               vqXn, Integer
'               vectorArrayInput, Double
'               generation, Integer
'
'*************************************************************************

  Function VQ_OverwriteVectorToDSP( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef vectorArrayInput() As Double, _
        Optional ByRef vqXn As Integer = 0, _
        Optional ByRef generation As Integer = 0) _
        As T_Error
```

5.12.6 VQ_Add VQ Function to List.

Adds a VQ Function to VQ List, inside DSP board, not intended for users. Used as a general building block for VQ functions.

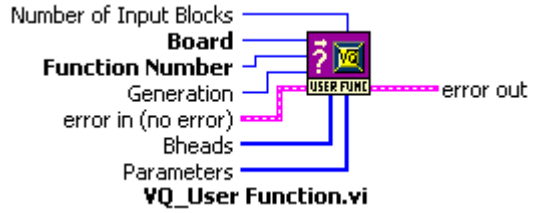


Number of Input Blocks = Number of input VQ vector blocks
Function Number = A number which identifies the funcion, with what DSP knows what to do
Generation = An incremental value to know when data is updated and ready to be processed.
Bheads = An array of pointers to the several vectors needed for current function
Parameters = An array of parameters or constant values needed by the function

last_location = Last DSP address, generally used by output of function.

5.12.7 VQ_User Function.

Available for customer to add custom VQ Functions



Number of Input Blocks = Number of input VQ vector blocks
Function Number = A number which identifies the funcion, with what DSP knows what to do
Generation = An incremental value to know when data is updated and ready to be processed.
Bheads = An array of pointers to the several vectors needed for current function
Parameters = An array of parameters or constant values needed by the function

5.12.8 VQ_Get Generation.

Gets Current Generation number

Generation = Current generation

5.12.9 VQ_Update Generation.

Updates Generation of a VQ vector, with generation value of another VQ. Not intneded for general use.

VQ_X{org} = VQ vector, from where generation must be taken from.
VQ_Y{copy} = VQ vector, where generation must be copied/updated to.

VQ_Y{out} = same as VQ_Y{copy}, except after execution, generation of vector is same as of VQ_X{org}..

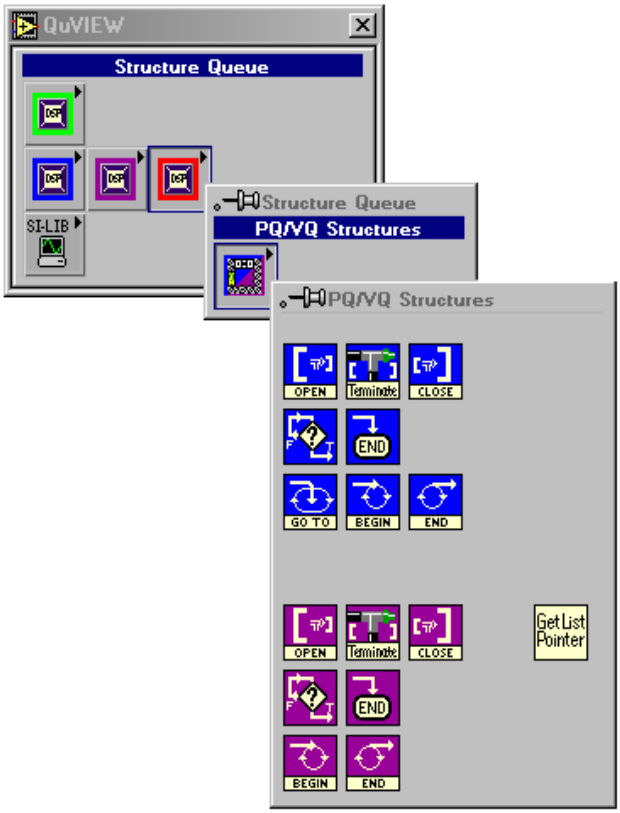
5.12.10 VQ_Update +1 Generation.

Increments a Generation numer

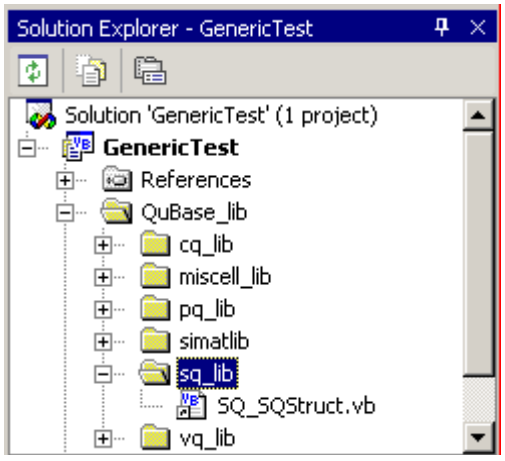
VQ_X{orig} = VQ vector, from where generation must be taken from.
VQ_Y{copy} = VQ vector, where generation must be copied/updated to.

VQ_Y{out} = same as VQ_Y{copy}, except after execution, generation of vector is also incremented by 1.

6.0 Queue Structure VIs.

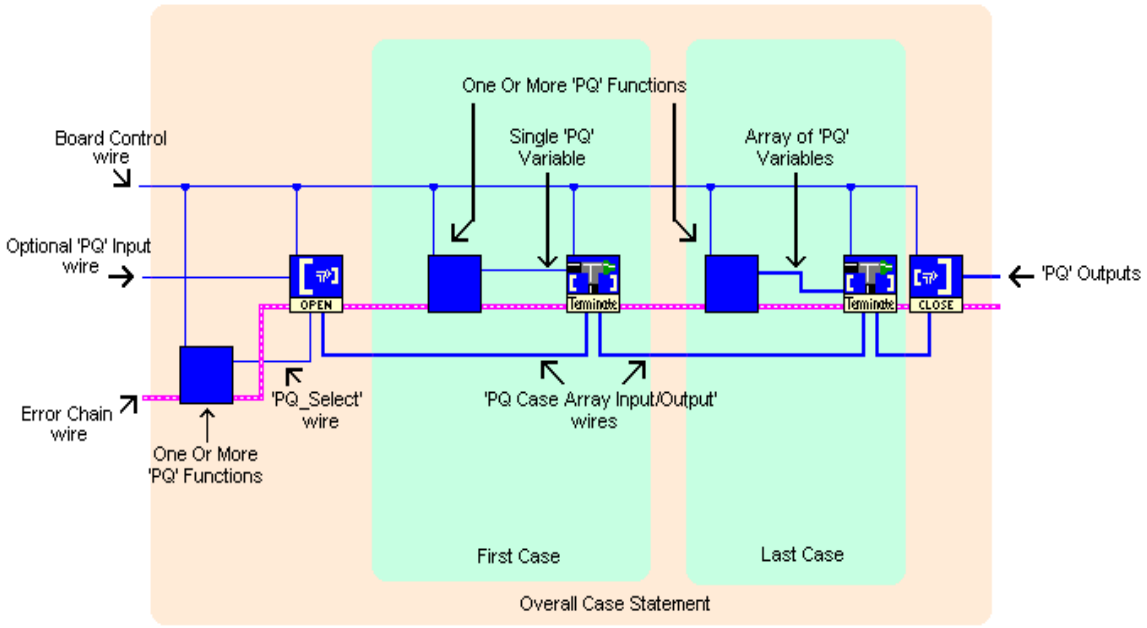


QuBASE Equivalent.



6.1 PQ Case Structures.

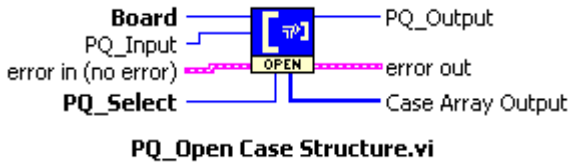
Below is a diagram illustrating the general form that one must follow in order to implement case structures for PQ functions.



NOTE: *In order to create Case Structures on PQ values, four sets of VIs are necessary:*
1) PQ Open Case Structure, indicates that a case structure is to be opened. Only a single instance can be present per case structure.
2) PQ_Terminate This Case, which serves as a marker to close a single subcase. Multiple instances may be present as multiple cases are taken into account.
3) PQ_Close Case Structure, indicates that a case structure is to be closed. Only a single instance can be present per case structure.
4) Any PQ evaluation VI, whose output will select a case structure to be invoked.

6.1.1 PQ_Open Case Structure.

The PQ_Open Case Structure VI marks the beginning or opening point of a single case structure. Only a single instance can be present per case structure.



PQ_Input = PQ input, optionally used to force precedence.
PQ_Select = PQ value for selecting the case to be executed, ranges from 0 to N-1 where N is the number of cases. If case selection is invalid, none of the cases are executed and the results of the previous execution are preserved.

PQ_Output = PQ output, optionally used to force precedence.
Case Array Output = LabVIEW output array, connects to the 'Case Array Input' input terminal of the next 'PQ_Terminate This Case' VI, thereby forming a daisy chain of PQ case arrays.

QuBASE Equivalent.

```
'*********************************************************************
'       Function PQ_OpenCaseStructure
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               iintPQInput, Integer
'               iintPQSelect, Integer
'
'       Output Parameters:
'               ointPQtoTerminatePQCaseArrayInput, Integer
'               ointPQOutput, Integer
'
'*********************************************************************

  Function PQ_OpenCaseStructure( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef iintPQInput As Integer, _
        ByRef iintPQSelect As Integer, _
        ByRef ointPQtoTerminatePQCaseArrayInput() As Integer, _
        Optional ByRef ointPQOutput As Integer = 0) _
        As T_Error
```

6.1.2 PQ_Terminate This SubCase.

The PQ_Terminate This Case VI serves as a marker to close a single subcase within an overall case structure. Multiple instances may be present as multiple cases are taken into account.

PQ_Input = PQ input, used to pass in PQ values or results from the current case. If left unwired, the 'PQ input array' terminal must at least be wired.

PQ_Input Array = PQ input source array, used to pass in an array of PQ values or results from the current case. If left unwired, the single 'PQ input' terminal must at least be wired.

Case Array Input = LabVIEW input array, connects with the 'Case Array Output' terminal from the previous 'PQ_Terminate This Case' VI, thereby forming a daisy chain of PQ case arrays.

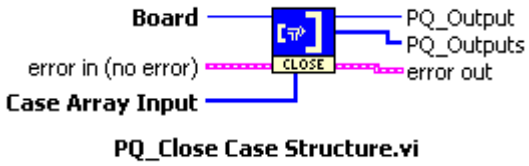
Case Array Output = LabVIEW output array, connects to the 'Case Array Input' terminal of the next 'PQ_Terminate This Case' VI, thereby forming a daisy chain of PQ case arrays.

QuBASE Equivalent.

```
'*********************************************************************
'       Function PQ_TerminateThisCase
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               iintPQInput, Integer
'               iintPQCaseArrayInput, Integer
'
'       Output Parameters:
'               ointPQCaseArrayOutput, Integer
'
'*********************************************************************

  Function PQ_TerminateThisCase( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef iintPQInput() As Integer, _
        ByRef iintPQCaseArrayInput() As Integer, _
        ByRef ointPQCaseArrayOutput() As Integer) _
        As T_Error
```

6.1.3 PQ_Close Case Structure.

The PQ_Close Case Structure VI indicates that a case structure is to be closed. Only a single instance can be present per case structure.



Case Array Input = LabVIEW input array, connects with the 'Case Array Output' terminal from the previous 'PQ_Terminate This Case' VI, thereby forming the last terminal of the daisy chain of PQ case arrays.

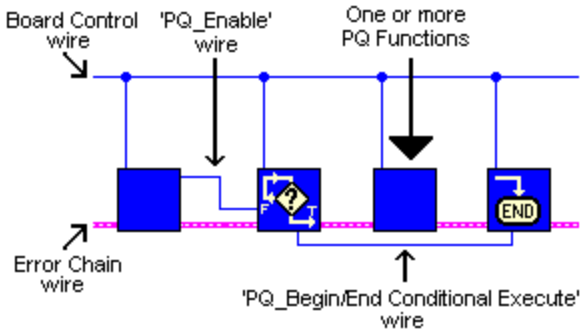
PQ_Output = PQ output, resultant of the first value from a selected subcase.
PQ_Outputs = PQ output array, resultant of all of the PQ values from a selected case if more than a single PQ output function is present inside a subcase.

QuBASE Equivalent.

```
'**************************************************************************
'       Function PQ_CloseCaseStructure
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               iintPQCaseArrayInput, Integer
'
'       Output Parameters:
'               ointPQOutput, Integer
'
'**************************************************************************

  Function PQ_CloseCaseStructure( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef iintPQCaseArrayInput() As Integer, _
        ByRef ointPQOutput() As Integer) _
        As T_Error
```

6.2 PQ Conditional Execute Structures.

Below is a diagram illustrating the general form that one must follow in order to implement conditional execute structures for PQ functions.



NOTE: *In order to create Conditional Execute Structures on PQ values, three sets of VIs are necessary:*
1) PQ_Begin Conditional Execute, indicates that the conditional execute structure is to begin. Only a single instance can be present per conditional structure.
2) PQ_End Conditional Execute, indicates that a conditional execute structure is to end. Only a single instance can be present per conditional structure.
3) Any PQ evaluation VI, whose output will activate a conditional execution to be invoked.

6.2.1 PQ_Begin Conditional Execute.

The PQ_Begin Conditional Execute, indicates that the conditional execute structure is to begin. Only a single instance can be present per conditional structure.

PQ_Enable (Execute) = PQ input, serves to conditionally activate the conditional execution of PQ functions that reside inside of the structure:

1) PQ_Enable = '0' or False, PQ functions inside structure are ignored.
2) PQ_Enable = '1' or True, PQ functions inside structure are executed.

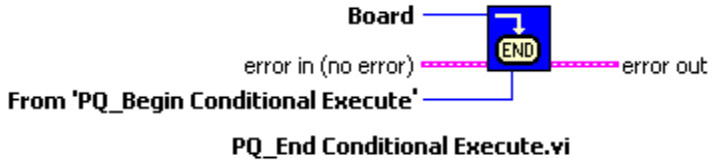
To "End Conditional Execute" = LabVIEW output, values to be passed on to the 'From Begin Conditional Execute' input terminal of the 'PQ_End Conditional Execute' VI, thereby forming a hard link to the structure's end.

QuBASE Equivalent.

```
'**************************************************************************
'       Function PQ_BeginConditionalExecute
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqExec, Integer
'
'       Output Parameters:
'               oToEndConditionalExecute, Integer
'
'**************************************************************************

  Function PQ_BeginConditionalExecute( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef pqExec As Integer = 0, _
        Optional ByRef oToEndConditionalExecute As Integer = 0) _
        As T_Error
```

6.2.2 PQ_End Conditional Execute.

The PQ_End Conditional Execute, indicates that the conditional execute structure is to end. Only a single instance can be present per conditional structure.



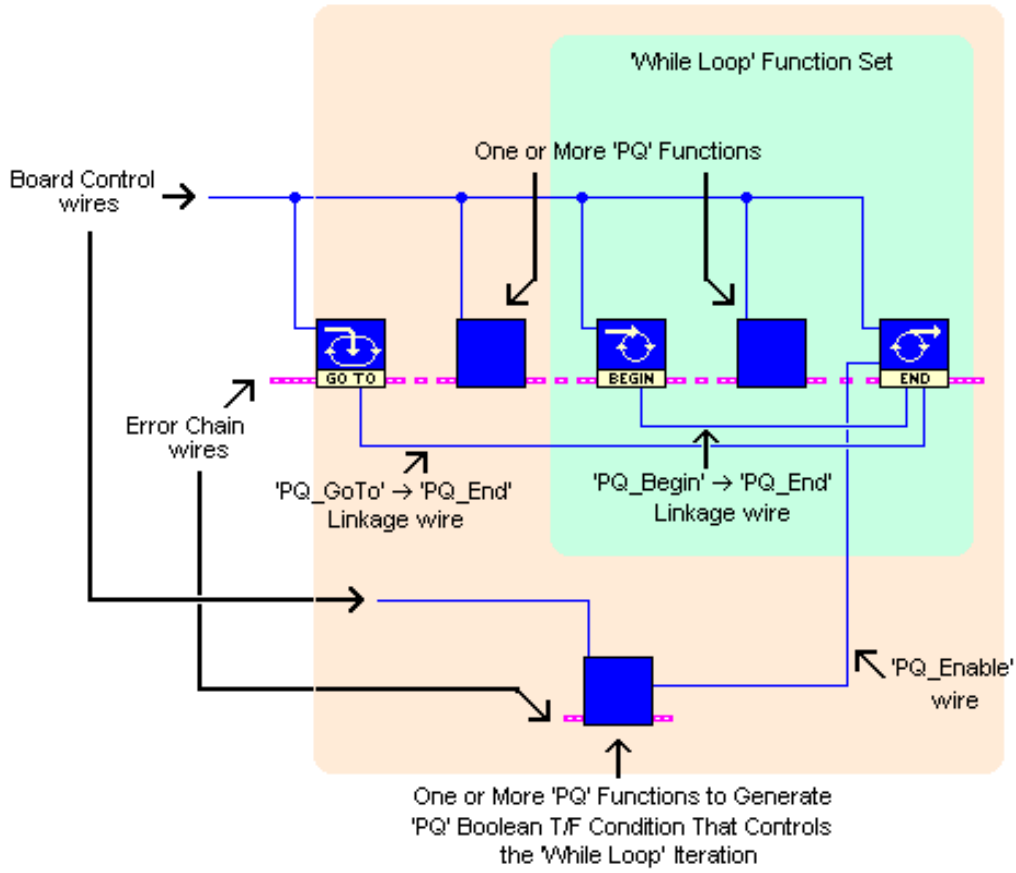
From 'Begin Conditional Execute' = LabVIEW input, connects with the 'To PQ End Conditional Execute' output terminal from the previous 'PQ_Begin Conditional Execute' VI, thereby completing the hard link to the structure.

QuBASE Equivalent.

```
'*************************************************************************
'       Function PQ_EndConditionalExecute
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               fromBeginConditionalExecute, Integer
'
'       Output Parameters:
'
'*************************************************************************

  Function PQ_EndConditionalExecute( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef fromBeginConditionalExecute As Integer = 0) _
        As T_Error
```

6.3 PQ While Loop Structures.

Below is a diagram illustrating the general form that one must follow in order to implement while loop structures for PQ functions.



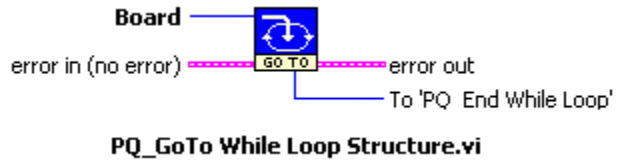
NOTE: In order to create While Loop Structures on PQ values, four sets of VIs are necessary:
1) PQ_GoTo While Loop Structure, marks the starting point of any while loop structure inside an application VI. Only a single instance can be present in each application VI, irrespective of the number of loop structures inside the application VI. However, there can be several nested subloops inside a single structure as well as multiple loop structures as required in the application.
2) PQ_Begin While Loop Structure, which serves as a marker to begin a new loop structure. Multiple instances may be present as multiple subloops may be nested for each structure.
3) PQ_End While Loop Structure, marks the end of a while loop structure. Only a single instance can be present per loop structure, however there can be several nested subloops inside a single structure as well as multiple loop structures as required in the application.
4) Any carefully placed PQ evaluation VI, whose output will render an evaluation of a loop structure to be invoked.

6.3.1 PQ_GoTo While Loop Structure.

The PQ_GoTo While Loop Structure VI marks the starting point of any while loop structure inside an application VI.

In essence, it is an ancillary function required as a marker for the PQ While Loop Structure, but itself is not involved nor is subject to the loop condition. It must be the first PQ function in the PQ list when a PQ While Loop is desired.

NOTE: *Only a single instance can be present in each application VI, irrespective of the number of loop structures inside the application VI. However, there can be several nested subloops inside a single structure as well as multiple loop structures as required in the application.*



To "PQ End While Loop" = LabVIEW output, must be connected to every instance of a 'PQ_End While Loop Structure' VI throughout the application.

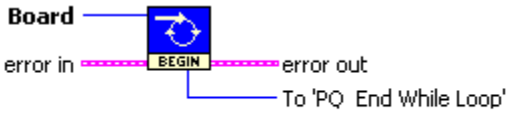
QuBASE Equivalent.

```
'*************************************************************************
'       Function PQ_GotoWhileLoopStructure
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               toPQEndWhileLoop, Integer
'
'       Output Parameters:
'
'*************************************************************************

  Function PQ_GotoWhileLoopStructure( _
            ByVal board As Integer, _
            ByRef errIn As T_Error, _
            ByRef toPQEndWhileLoop As Integer) _
            As T_Error
```

6.3.2 PQ_Begin While Loop Structure.

The PQ_Begin While Loop Structure serves as a marker to begin a new loop structure. Multiple instances may be present as multiple subloops may be nested for each structure.

NOTE: It is very important that the VI used to evaluate the 'TRUE' or 'FALSE' condition must be included inside of the loop structure, in other words somewhere in between the 'PQ_Begin While Loop Structure' VI and the 'PQ_End While Loop Structure' VI. Otherwise, an evaluation change in condition from 'True' to 'False' may never occur and an undesired infinite loop may result.



PQ_Begin While Loop Structure.vi

To "PQ End While Loop" = LabVIEW output, which connects to the 'From PQ_Begin While Loop' terminal of the 'PQ_End While Loop Structure' VI, thereby forming a hard link to the structure's end.

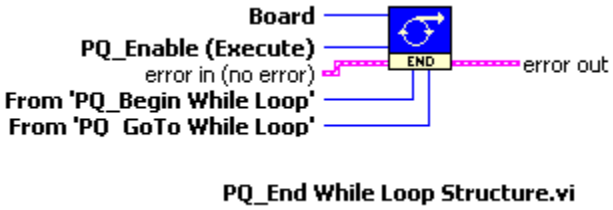
QuBASE Equivalent.

```
'**************************************************************************
'       Function PQ_BeginWhileLoopStructure
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               toPQEndWhileLoop, Integer
'
'       Output Parameters:
'
'**************************************************************************

  Function PQ_BeginWhileLoopStructure( _
            ByVal board As Integer, _
            ByRef errIn As T_Error, _
            ByRef toPQEndWhileLoop As Integer) _
            As T_Error
```

6.3.3 PQ_End While Loop Structure.

The PQ_End While Loop Structure serves as a marker to end the loop structure. Only a single instance can be present per loop structure, however there can be several nested subloops inside a single structure as well as multiple loop structures as required in the application.

NOTE: *It is very important that the VI used to evaluate the 'TRUE' or 'FALSE' condition must be included inside of the loop structure, in other words somewhere in between the 'PQ_Begin While Loop Structure' VI and the 'PQ_End While Loop Structure' VI. Otherwise, an evaluation change in condition from 'True' to 'False' may never occur and an undesired infinite loop may result.*



PQ_Enable (Execute) = PQ input, an evaluation that takes place at the end of the loop structure which serves as an indicator to either remain inside or proceed outside of the loop structure. As a rule, the set of one or more PQ functions used for evaluation whose result is connected to this terminal must reside inside the loop:

1) *PQ_Enable = '0' or False*, indicator used by the DSP to exit the loop structure and proceed to the rest of the application.

2) *PQ_Enable = '1' or True*, indicator used by the DSP to remain inside of the loop structure.

From 'PQ_Begin While Loop' = LabVIEW input, must be connected to the 'PQ_End While Loop' output terminal of the 'PQ_Begin While Loop' VI, thereby forming a hard link to the structure's beginning.

From 'PQ_GoTo While Loop' = LabVIEW input, must be connected to the 'PQ_End While Loop' output terminal of the single instance 'PQ_GoTo While Loop' VI, thereby forming a hard link to the structure's marker.

QuBASE Equivalent.

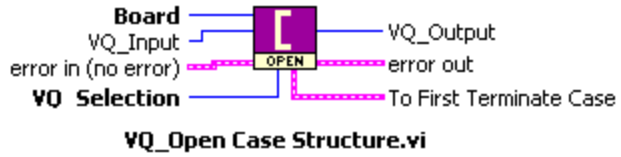
```
'*********************************************************************
'       Function PQ_EndWhileLoopStructure
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               pqEnable, Integer
'               fromPQGotoWhileLoop, Integer
'               fromPQBeginWhileLoop, Integer
'
'       Output Parameters:
'
'*********************************************************************

  Function PQ_EndWhileLoopStructure( _
          ByVal board As Integer, _
          ByRef errIn As T_Error, _
          ByRef pqEnable As Integer, _
          ByRef fromPQGotoWhileLoop As Integer, _
          ByRef fromPQBeginWhileLoop As Integer) _
          As T_Error
```

6.4 VQ Case Structures.

NOTE: *In order to create Case Structures on VQ values, four sets of VIs are necessary:*
1) VQ Open Case Structure, indicates that a case structure is to be opened. Only a single instance can be present per case structure.
2) VQ_Terminate This Case, which serves as a marker to close a single subcase. Multiple instances may be present as multiple cases are taken into account.
3) VQ_Close Case Structure, indicates that a case structure is to be closed. Only a single instance can be present per case structure.
4) Any VQ evaluation VI, whose output will select a case structure to be invoked.

6.4.1 VQ_Open Case Structure.

The VQ_Open Case Structure VI marks the beginning or opening point of a single case structure. Only a single instance can be present per case structure.



VQ_Input = VQ input, optionally used to force precedence.
VQ Selection = VQ input, sued for selecting the case to be executed, ranges from 0 to N-1 where N is the number of cases. If case selection is invalid, none of the cases are executed and the results of the previous execution are preserved.

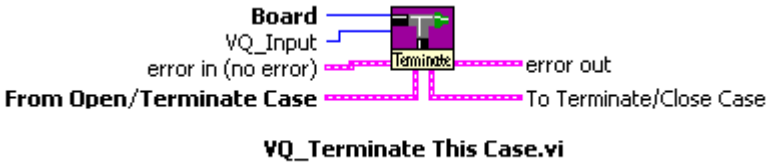
VQ_Output = VQ output, optionally used to force precedence.
To First Terminate Case = LabVIEW cluster output, must be connected to the 'From Open/Terminate Case' input cluster of the next 'VQ Terminate This Case' VI, thereby forming a daisy chain of VQ case clusters.

QuBASE Equivalent.

```
'*********************************************************************
'       Function VQ_OpenCaseStructure
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               iintVQInput, Integer
'               iintVQSelection, Integer
'               toTerminateCloseCase, Integer
'
'       Output Parameters:
'               ointVQOutput, Integer
'
'*********************************************************************

  Function VQ_OpenCaseStructure( _
            ByRef board As Integer, _
            ByRef errIn As T_Error, _
            ByRef iintVQInput As Integer, _
            ByRef iintVQSelection As Integer, _
            ByRef toTerminateCloseCase As CaseBundle, _
            Optional ByRef ointVQOutput As Integer = 0) _
            As T_Error
```

6.4.2 VQ_Terminate This SubCase.

The VQ_Terminate This SubCase VI serves as a marker to close a single subcase within an overall case structure. Multiple instances may be present as multiple cases are taken into account.



VQ_Input = VQ input, used to pass in VQ values or results from the current case.
From Open/Terminate Case = LabVIEW cluster input, must be connected to the 'To First Terminate Case' output cluster of the 'VQ Open Case' VI, or to the 'To Terminate/Close Case' output cluster of the previous 'VQ Terminate Case' VI.

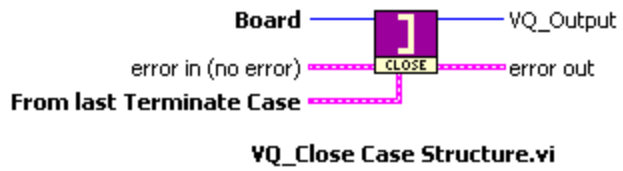
To Open/Terminate Case = LabVIEW cluster output, must be connected to the 'From Open/Terminate Case' input cluster of the next 'VQ Terminate This Case' VI, or to the 'From Last Terminate Case' input cluster of the final 'VQ Close Case' VI.

QuBASE Equivalent.

```
'************************************************************************
'       Function VQ_TerminateThisCase
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               iintVQInput, Integer
'               iFromOpenTerminateCase, CaseBundle
'
'       Output Parameters:
'               otoTerminateCloseCase, CaseBundle
'
'************************************************************************

  Function VQ_TerminateThisCase( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef iintVQInput As Integer, _
        ByRef iFromOpenTerminateCase As CaseBundle, _
        ByRef otoTerminateCloseCase As CaseBundle) _
        As T_Error
```

6.4.3 VQ_Close Case Structure.

The VQ_Close Case Structure VI indicates that a case structure is to be closed. Only a single instance can be present per case structure.



From last Terminate Case = LabVIEW cluster input, must be connected to the 'To Terminate/Close Case' output cluster of the last 'VQ Terminate Case' VI.

VQ_Output = VQ output.

QuBASE Equivalent.

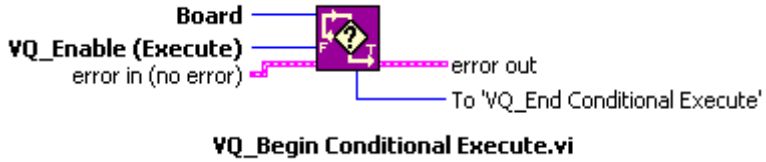
```
'**************************************************************************
'       Function VQ_CloseCaseStructure
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               iFromOpenTerminateCase, CaseBundle
'
'       Output Parameters:
'               ointVQOutput, Integer
'
'**************************************************************************

  Function VQ_CloseCaseStructure( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef iFromOpenTerminateCase As CaseBundle, _
        Optional ByRef ointVQOutput As Integer = 0) _
        As T_Error
```

6.5 VQ Conditional Execute Structures.

NOTE: *In order to create Conditional Execute Structures on VQ values, three sets of VIs are necessary:*
1) VQ_Begin Conditional Execute, indicates that the conditional execute structure is to begin. Only a single instance can be present per conditional structure.
2) VQ_End Conditional Execute, indicates that a conditional execute structure is to end. Only a single instance can be present per conditional structure.
3) Any VQ evaluation VI, whose output will activate a conditional execution to be invoked.

6.5.1 VQ_Begin Conditional Execute.

The VQ_Begin Conditional Execute VI indicates that the conditional execute structure is to begin. Only a single instance can be present per conditional structure.



VQ_Enable (Execute) = VQ input, serves to conditionally activate the conditional execution of PQ functions that reside inside of the structure:

1) *VQ_Enable = '0' or False*, VQ functions inside structure are ignored.
2) *VQ_Enable = '1' or True*, VQ functions inside structure are executed.

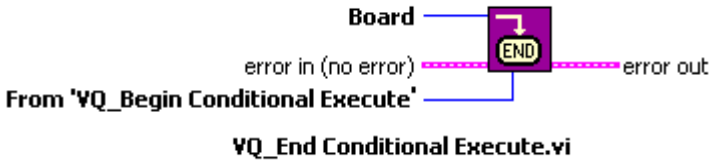
To 'VQ_End Conditional Execute' = LabVIEW output, wired to the 'From VQ_Begin Conditional Execute' input terminal of the final 'VQ_End Conditional Execute' VI, thereby forming a hard link to the structure's end.

QuBASE Equivalent.

```
'*********************************************************************
'       Function VQ_BeginConditionalExecute
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               intVQExecute, Integer
'               intVQtoEndConditionalExecute, Integer
'
'       Output Parameters:
'
'*********************************************************************

  Function VQ_BeginConditionalExecute( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef intVQExecute As Integer, _
        ByRef intVQtoEndConditionalExecute As Integer) _
        As T_Error
```

6.5.2 VQ_End Conditional Execute.

The VQ_End Conditional Execute VI indicates that the conditional execute structure is to end. Only a single instance can be present per conditional structure.



From 'VQ_Begin Conditional Execute' = LabIEW input, wired from the 'To VQ End Conditional Execute' output terminal of the 'VQ_Begin Conditional Execute' VI, thereby forming a hard link to the structure's beginning.

QuBASE Equivalent.

```
'*********************************************************************
'       Function VQ_EndConditionalExecute
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               intVQFromBeginConditionalExecute, Integer
'
'       Output Parameters:
'
'*********************************************************************

  Function VQ_EndConditionalExecute( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        ByRef intVQFromBeginConditionalExecute As Integer) _
        As T_Error
```

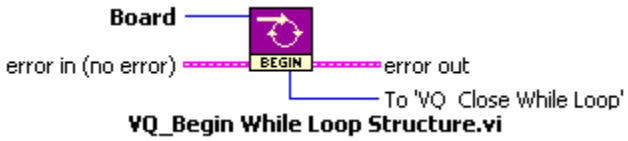
6.6 VQ_While Loop Structures.

NOTE: *In order to create While Loop Structures on VQ values, three sets of VIs are necessary:*
1) VQ_Begin While Loop Structure, which serves as a marker to begin a new loop structure. Only a single instance can be present per loop structure, however there can be several nested subloops inside a single structure as well as multiple loop structures as required in the application.
2) VQ_End While Loop Structure, marks the end of a while loop structure. Only a single instance can be present per loop structure, however there can be several nested subloops inside a single structure as well as multiple loop structures as required in the application.
3) Any carefully placed VQ evaluation VI, whose output will render an evaluation of a loop structure to be invoked.

6.6.1 VQ_Begin While Loop Structure.

The VQ_Begin While Loop Structure VI serves as a marker to begin a new loop structure. Only a single instance can be present per loop structure, however there can be several nested subloops inside a single structure as well as multiple loop structures as required in the application.

NOTE: *It is very important that the VI used to evaluate the 'TRUE' or 'FALSE' condition must be included inside of the loop structure, in other words somewhere in between the 'VQ_Begin While Loop Structure' VI and the 'VQ_End While Loop Structure' VI. Otherwise, an evaluation change in condition from 'True' to 'False' may never occur and an undesired infinite loop may result.*



To 'VQ Close While Loop' = LabVIEW output, must be connected to the 'From VQ_Begin While Loop' input terminal of the final 'VQ_End While Loop Structure' VI, thereby forming a hard link to the structure's end.

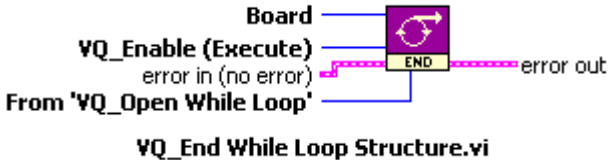
QuBASE Equivalent.

```
'**************************************************************************
'       Function VQ_BeginWhileLoopStructure
'
'       Description:
'
'       Input Parameters:
'               board, Integer
'               errIn, T_Error
'               intToCloseWhile, Integer
'
'       Output Parameters:
'
'**************************************************************************

  Function VQ_BeginWhileLoopStructure( _
        ByRef board As Integer, _
        ByRef errIn As T_Error, _
        Optional ByRef intToCloseWhile As Integer = 0) _
        As T_Error
```

6.6.2 VQ_End While Loop Structure.

The VQ_End While Loop Structure VI serves as a marker to end the loop structure. Only a single instance can be present per loop structure, however there can be several nested subloops inside a single structure as well as multiple loop structures as required in the application.

NOTE: *It is very important that the VI used to evaluate the 'TRUE' or 'FALSE' condition must be included inside of the loop structure, in other words somewhere in between the 'VQ_Begin While Loop Structure' VI and the 'VQ_End While Loop Structure' VI. Otherwise, an evaluation change in condition from 'True' to 'False' may never occur and an undesired infinite loop may result.*



VQ_Enable (Execute) = VQ input, an evaluation that takes place at the end of the loop structure which serves as an indicator to either remain inside or proceed outside of the loop structure. As a rule, the set of one or more VQ functions used for evaluation whose result is connected to this terminal must reside inside the loop:

1) *VQ_Enable = '0' or False*, indicator used by the DSP to exit the loop structure and proceed to the rest of the application.

2) *VQ_Enable = '1' or True*, indicator used by the DSP to remain inside of the loop structure.

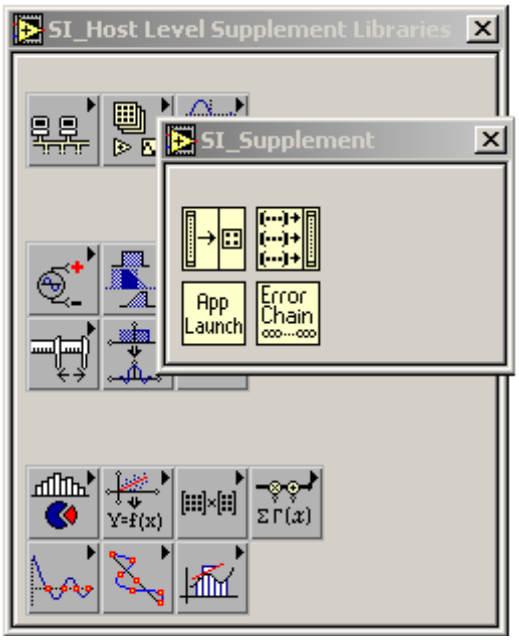
From 'VQ_Begin While Loop' = LabVIEW input, must be connected to the 'VQ_End While Loop' output terminal of the 'VQ_Begin While Loop' VI, thereby forming a hard link to the structure's beginning.

QuBASE Equivalent.

```
'**************************************************************************
'           Function VQ_EndWhileLoopStructure
'
'           Description:
'
'           Input Parameters:
'                       board, Integer
'                       errIn, T_Error
'                       intVQExecute, Integer
'                       intFromOpenWhile, Integer
'
'           Output Parameters:
'
'**************************************************************************

    Function VQ_EndWhileLoopStructure( _
            ByRef board As Integer, _
            ByRef errIn As T_Error, _
            ByRef intVQExecute As Integer, _
            ByRef intFromOpenWhile As Integer) _
            As T_Error
```

7.0 Supplemental SI Functions.

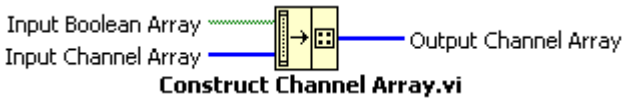
These functions only run on the host, but are convenient and serve as supplemental utilities not offered as standard LabVIEW library functions.



QuBASE Equivalent.

7.1 Construct Channel Array.

The Construct Channel Array VI converts an array of randomly selected boolean inputs to a numeric array whose values reflect the enabled boolean input's relative position in the array, and concatenates with a separate numeric array input. Useful for translating boolean buttons that are used to select one or more input channels to a numeric array whose values reflect the actual channel numbers, and then combining the results with a secondary numeric input array. Created because no native equivalent exists in the LabVIEW libraries.



Input Boolean Array = LabVIEW input of boolean array.
Input Channel Array = LabVIEW input of a numeric array.

Output Channel Array = LabVIEW output of a numeric array.

QuBASE Equivalent.

7.2 Boolean to Channel Array.

The Boolean to Channel Array VI converts an array of randomly selected boolean inputs to a numeric array whose values reflect the enabled boolean input's relative position in the array. Useful for translating boolean buttons that are used to select one or more input channels to a numeric array whose values reflect the actual channel numbers. Created because no native equivalent exists in the LabVIEW libraries.

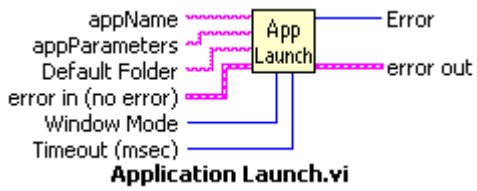
Input Boolean Array = LabVIEW input of boolean array.

Output Channel Array = LabVIEW output of a numeric array.

QuBASE Equivalent.

7.3 Application Launch.

Launches a DOS/Windows application from within LabVIEW.



appName = LabVIEW input string, name of application to be launched.

appParameters = LabVIEW input string, parameters/switches needed to launch the application.

Default Folder = LabVIEW input string, folder where the application resides.

Window Mode = LabVIEW input, specifies if the mode of operation of the command line window after program execution.

Timeout (msec) = LabVIEW input, the time (in ms) to delay the entry into the next function after the program has executed.