

Application Training 15

How to control and program the 81250



Agilent Technologies

Innovating the HP Way

"Theory is, if you know everything, but nothing works.

Practice is, if everything works, but nobody knows why.

Here Theory and Practice are joined at the optimum level, nothing works and nobody knows why."

Application Focus

- **Possible VXI connections**
- **Fundamentals of programming**
- **Programming Environments**
 - C/C++**
 - LabVIEW**
 - Agilent VEE**
- **Read out BER results**

This presentation will introduce first into the possible VXI connection to control the 81250.

-external internal PC

-How connected

To understand the fundamentals of the programming 81250

-programming possibilities

-Basic program flow

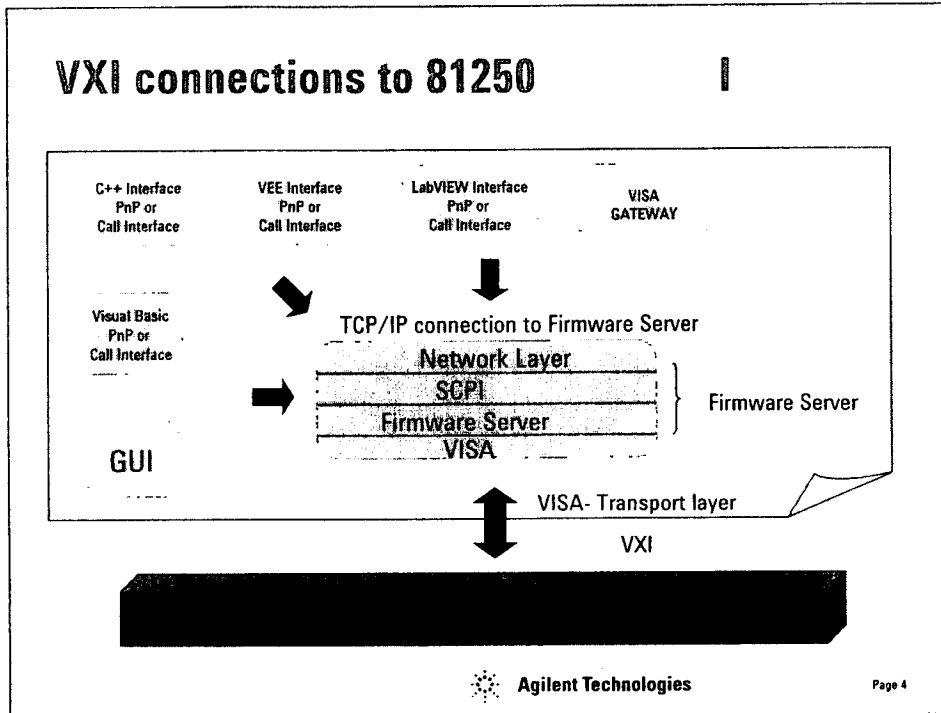
We will focus on programming environments of LabVIEW, Agilent VEE and C++

-programming environments are shown

-Their speciality

How to read out BER results

- measure BER without UI what to do



The picture shows the 81250 as an embedded PC with Hardware.

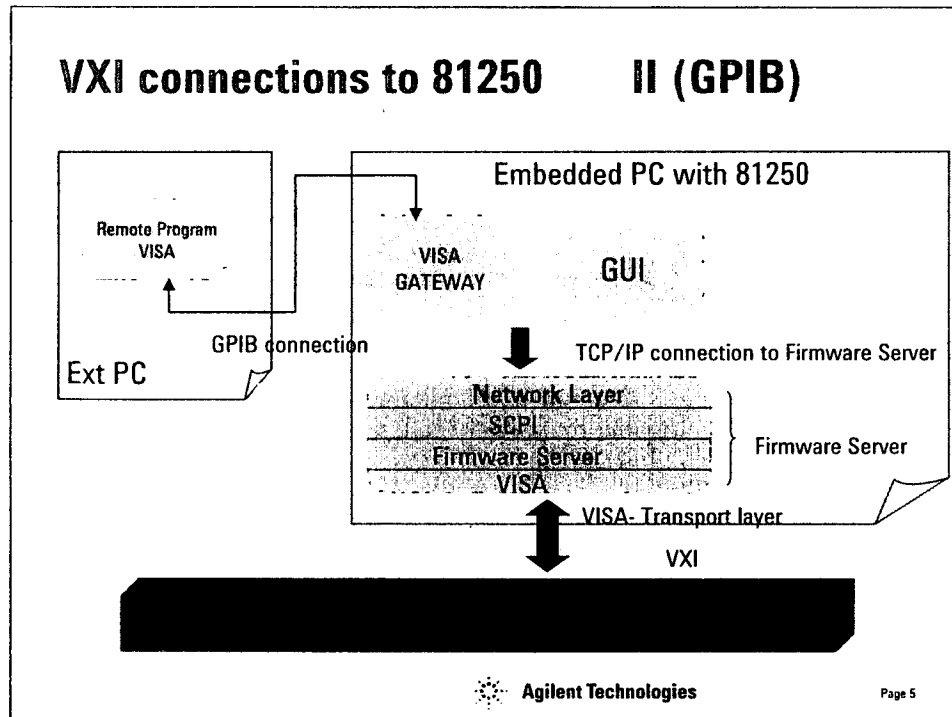
The 2-slot PC and the data modules are plugged in a VXI mainframe.

The Firmware Server consists of Network Layer, SCPI-command Parser that interprets the SCPI commands and the Firmware Server that realizes the commands. VISA is used for transportation.

Within the Mainframe is a VXI connection established to send data from the Firmware Server to the Hardware. This does require Agilent I/O Libraries to run the Firmware Server to establish the VXI connection.

Installations on your embedded PC:

- 81250 SW
- Agilent I/O Libraries



This picture shows how to control the 81250 with an embedded PC via GPIB from an external PC.

-Installations on your **embedded PC**:

- 81250 SW
- Agilent I/O Libraries

-Installations on your **external PC with Agilent GPIB Interface Card**:

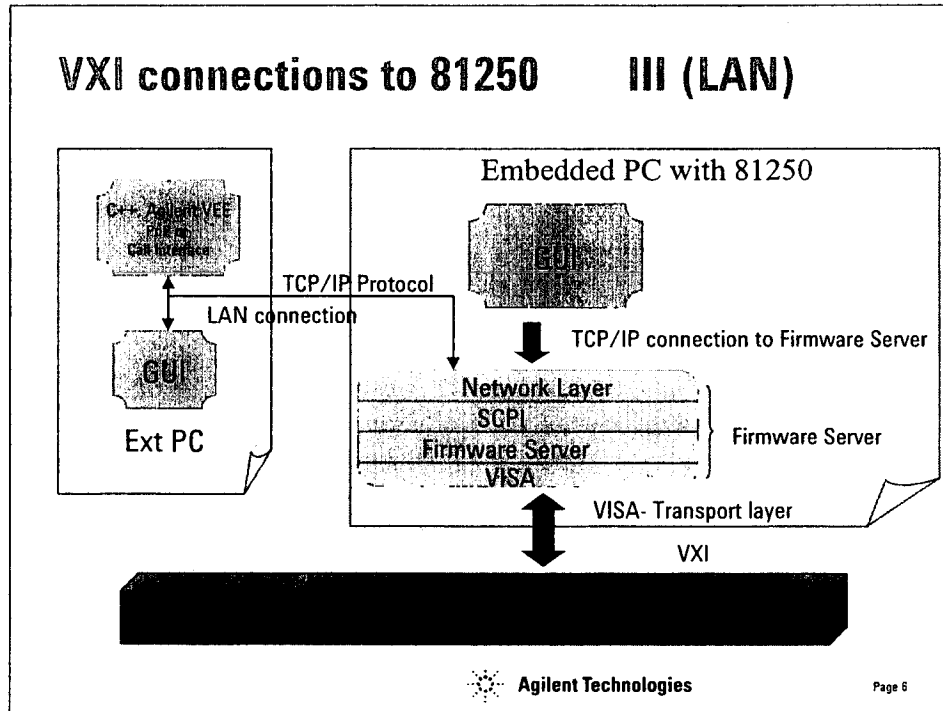
- Program Environment
- Agilent I/O Libraries

-Installations on your **external PC with NI GPIB Interface Card**:

- Program Environment
- NI I/O Libraries

Be aware, that you cannot use PnP-Drivers, only SCPI commands, to program the 81250.

No good solution more fault prone, bad error handling, takes much longer to develop



This picture shows how to control the 81250 with an embedded PC via LAN from an external PC.

-Installations on your **embedded PC**:

- 81250 SW
- Agilent I/O Libraries

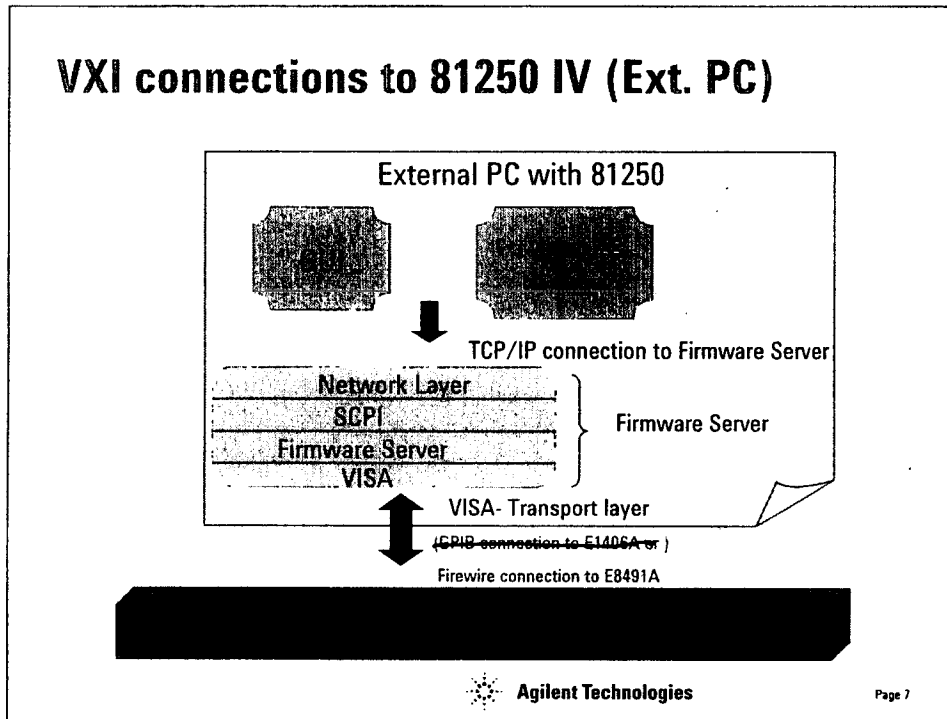
-Installations on your **external PC with any LAN Interface Card**:

- Program Environment
- 81250 SW
- Win NT SP 3 or higher
- *IO LIBS*

You can use PnP drivers but it is an expensive solution as you need the 2-slot controller.

Slow data transmission time

VXI connections to 81250 IV (Ext. PC)



This picture shows how to control the 81250 Hardware with an external PC via ~~GPIB~~ or Firewire.

-Installations on your external PC with **Firewire Interface Card**:

- Program Environment
- Agilent Firewire card
- Agilent I/O Libraries
- 81250 SW
- Win NT SP 3 or higher

-Installations on your external PC with any **GPIB Interface Card and E1406A Command Module**:

- We strongly recommend not to use this configuration

VXI Plug and Play driver versus SCPI commands

- **VXI Plug and Play driver:**
 - comfortable function calls to driver
 - good error handling
 - single function call can contain multiple SCPI commands
 - `hp81200_systemSelect(vi, "DSRA", "DSR")`
- **SCPI commands:**
 - low level programming language
 - worse error handling
 - `:DVT:INST:HAND:CRE? HANDLE, 'DSR', 'DSRA'`

Plug and Play drivers are C functions that are provided to make the programming more comfortable

SCPI commands are very low level commands that can be interpreted by measurement instruments, they contain mnemonics, pay attention to right spelling

Fundamental programming know-how

1. Init – returns vi session handle
2. Connect – to server / port
3. Select System – instrument name, system name

4. Program measurement tasks

5. Close

 Agilent Technologies

Page 9

With init I get a Vi session handle, depends e.g. on VXI connection and so the means of transportation

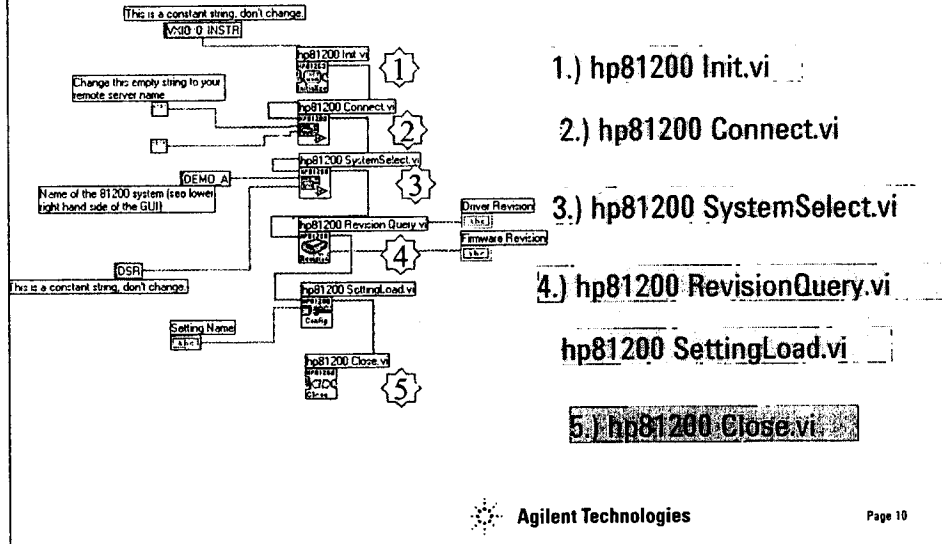
With PnP driver I can omit connect if I use localhost and port 2203 otherwise I have to specify the location of the firmware server and the port where it is listening

With Select System I choose the system from the instrument

You can program whatever you want, every PnP function gets the handle returned by init from its predecessor

Close the session to free resources

Using VXI PnP with LabVIEW

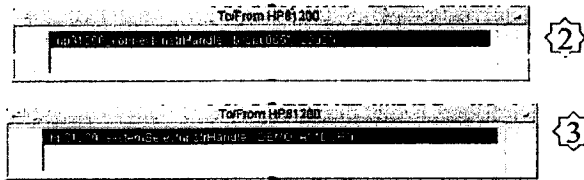


LabVIEW is a fourth generation programming language, that is graphical oriented, you choose icons that represent functions. They have inputs and output that need to be wired.

Using VXI PnP with Agilent VEE I

① "Init" is done by the Agilent VEE frame!

Start



2.) hp81200 Connect(instrHandle, "serverName", "portNumber")

3.) hp81200 SystemSelect(instrHandle, "systemName",
instrumentName")

Agilent Technologies

Page 11

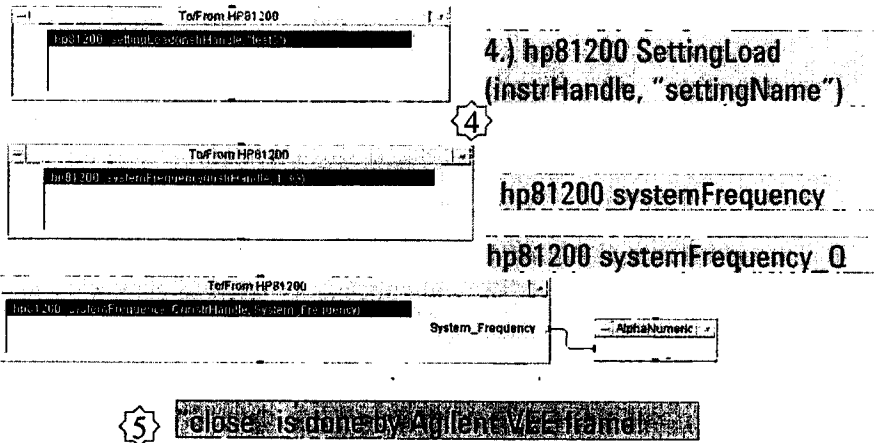
The principal is the same as with LabVIEW, you connect graphical representations.

Init is done by Agilent VEE and Close is done by Agilent VEE.

Instead of the Start you can use the Run icon from the menu bar.

Using VXI PnP with Agilent VEE II

Start



Agilent Technologies

Page 12

If your program doesn't run smoothly and hangs in the middle you have to close it on your own, do this by right clicking on a function and selecting Close session

Using VXI PnP with C/C++

```
*****
open session, VXI::0::INSTR is the address of the Agilent 81250
VI_FALSE is used to allow multiple access modes simultaneously.
vi is an session identifier that we use for the whole program
*****
if (vistatReturn = hp81200_init("VXI::0::INSTR",
    VI_FALSE, VI_FALSE, &vi)) != VI_SUCCESS)
{
    // connection failed
    printf("Connection Failed!\n");
}
return vistatReturn;

"connect" is not used, because this
program is running on a local PC on port 2203

*****
the hp81200_systemSelect gets as parameters the session identifier,
also called handle (vi) the system name (DSRA) and the instrument
name (DSR).
*****
if (vistatReturn = hp81200_systemSelect(vi, "DSRA", "DSR")) !=
VI_SUCCESS)
{
    // system select failed
    printf("System select failed!\n");
    hp81200_close(vi);
    return vistatReturn;
}

1.) hp81200_init
Returns you the instrHandle

2

3.) hp81200 SystemSelect
(instrHandle, "systemName",
"instrumentName")
```

Agilent Technologies

Page 13

C and C++ are text based programming languages.

C++ is the object oriented extension of C.

Plug and Play drivers imply a procedural programming style so it is basically C programming, anyway as C is a subset of C++ it's a good idea to use C++ files, this way one can use special features only provided by C++

Using VXI PnP with C/C++

II

```
.....  
load setting called EXSET with the session identifier and the name  
setting as parameters.  
.....  
if ((xistatReturn = hp81200_settingLoad(vi, "EXSET")) != VI_SUCCESS)  
{  
    // connection failed  
    printf("Connection Failed!\n");  
    return xistatReturn;  
}  
  
// close session  
xistatReturn = hp81200_close(vi);  
printf("Program complete.\n");  
return 0;
```

4

4.) hp81200 SettingLoad
(viHandle, "settingName")

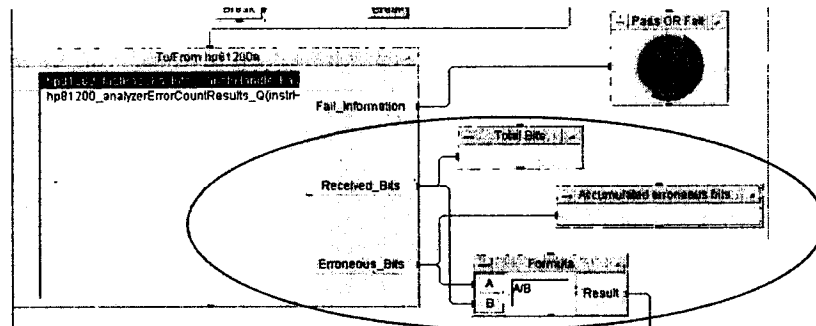
5

5.) hp81200_close(viHandle)

How to calculate BER

- To get the actual Bit Error Rate displayed on the screen programmatically, you need to calculate following:

Erroneous bits divided by Received bits



Agilent Technologies

Page 15

The PnP driver functions return you the number of received bits and the number of erroneous bits.

To get the BER you have to divide the erroneous bits by the received bits.

Pitfalls



Escape characters depend on programming language:

- " in a string and
- \ in a file path

Firmware server not ready (no setting, GUI is still running)

Wrong files included

- hp81200.h
- hp81200.lib

For example in C you have to use quotes if you define a String, now if there are again quotes within the String they would be interpreted as the end of the string that is why you use the escape character \ (backslash) in front of the quote.

Now the backslash has a special meaning so if you really want to use one like in a file path you have to use two.

Everything that works with your program should work with the GUI, too. It's sometimes a good idea first to use the GUI and if this works your program should be able to do the same.

If you're program crashed in the middle (for example during running a measurement) it will possibly still run in the GUI and you can't start your program again. Change to the GUI and stop the setting there.

The files hp81200.h and hp81200.lib exist two times on the PC depending on your program you need to choose one or the other file. There is a new and an old installation path.

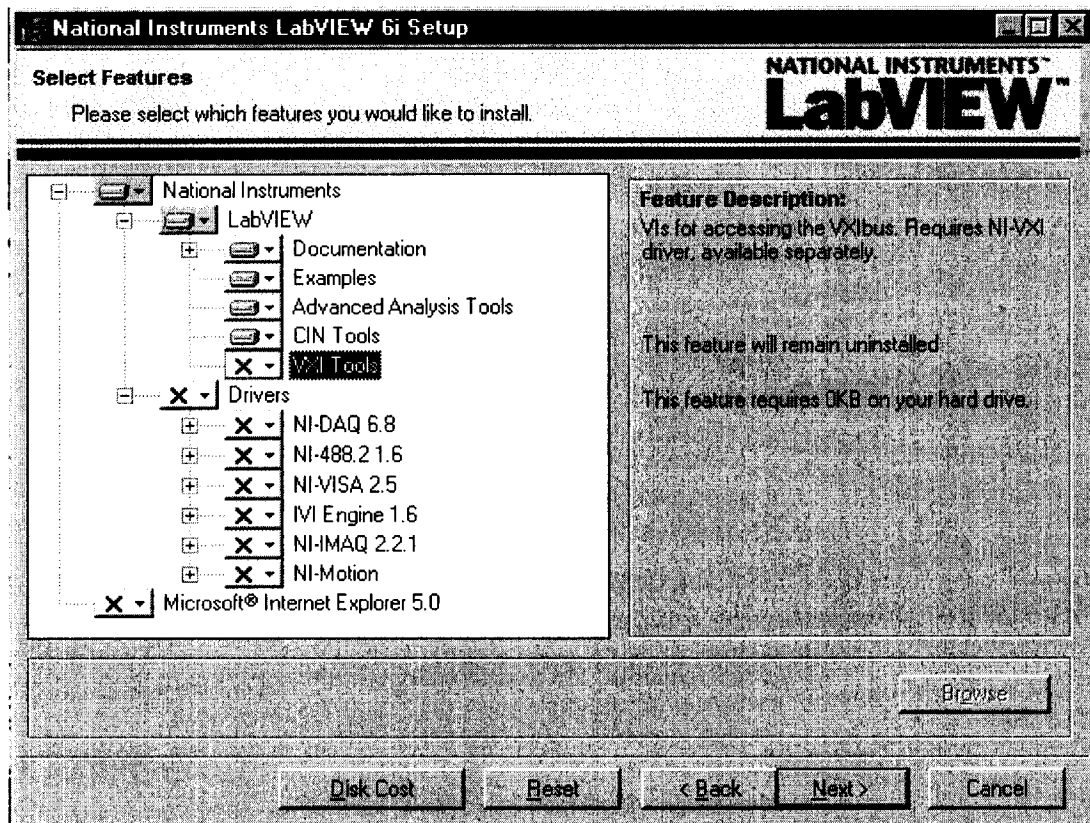
Getting started programming the Agilent 81250 using - LabVIEW -

If you already programmed the Agilent 81250 with PnP drivers in LabVIEW you can go to chapter 4.

1. Install LabVIEW without the drivers

Follow the installation instructions be careful to select at Installation Type **Custom** here deselect the Microsoft Internet Explorer and all the drivers, the VXI Tool can remain deselected (default).

Here is a picture that shows you the parts to install.

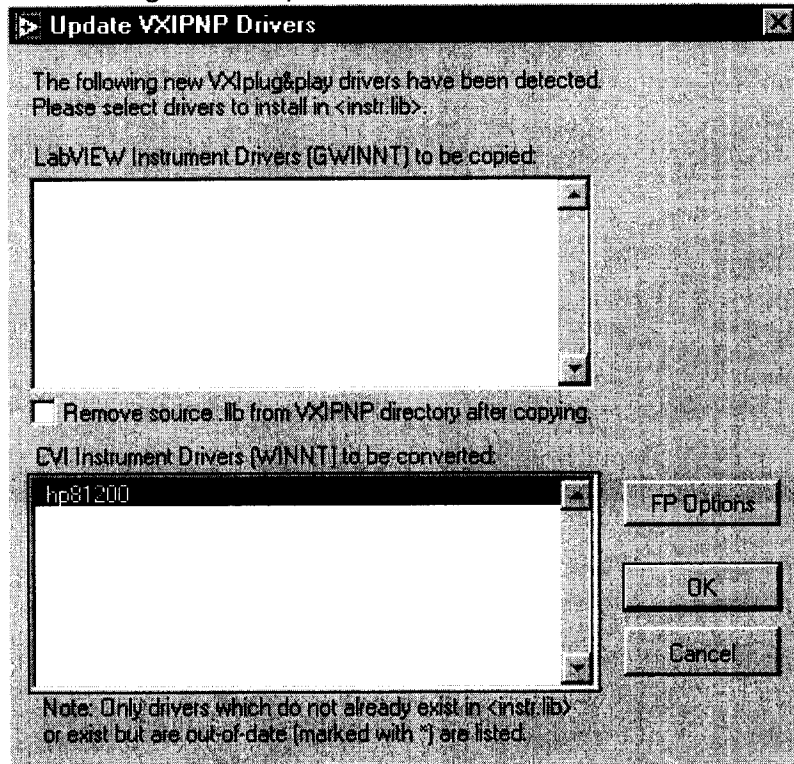


2. Include the PnP drivers

In LabVIEW do the following:

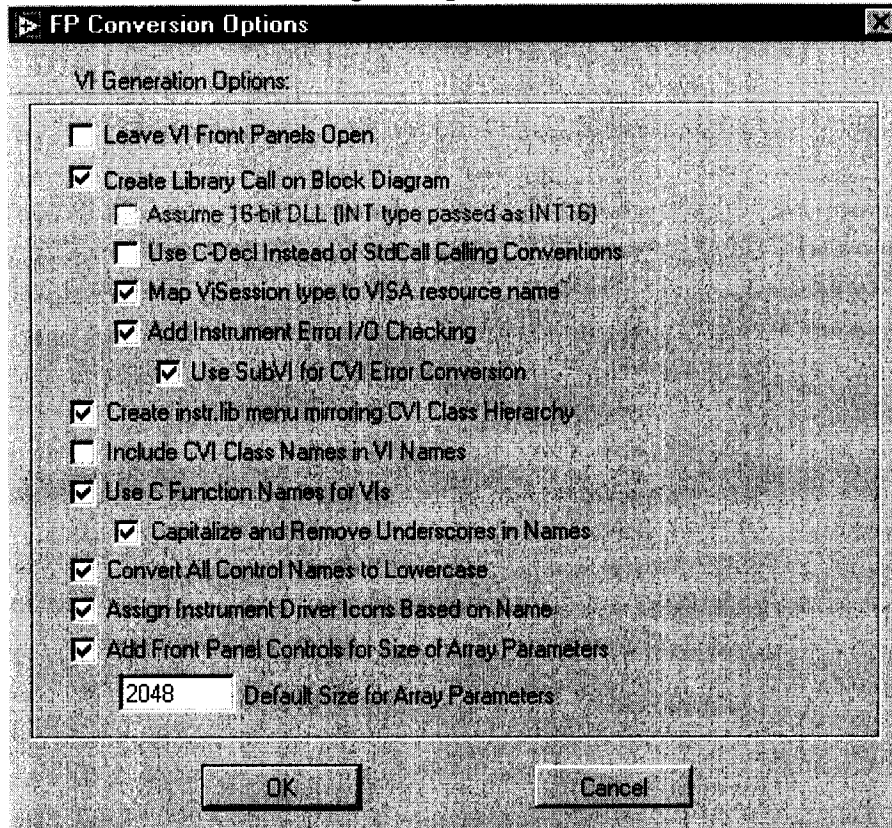
1. Choose Tools from the menu bar
2. Click on Instrumentation
3. Choose Update VXIplug&play drivers

4. This dialog will be opened



5. Click on hp81200
6. Click on FP Options

7. You will see the following dialog



8. Check the options like shown above
9. Click on OK
10. Click on OK
11. The conversion will take some time use it to read the introduction to LabVIEW

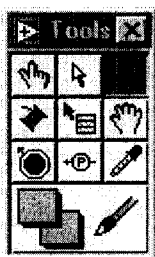
3. Introduction to LabVIEW

A program consists of two Views, the Panel and the Diagram.

Panel: Looks like the front of an instrument, can display results or a value can be entered as a control. The Controls Palette can be displayed in this view.

Diagram: Here you wire together the vis, controls, indicators and constants. The Functions Palette can be made visible in this view.

The Tools, Functions and Controls palette can be made visible by selecting them from the **Window** menu in the menu bar.



Tools Palette

Here you can choose which tool your mouse pointer represents it's functioning is similar to the one you might now from Microsoft Paint.



for positioning

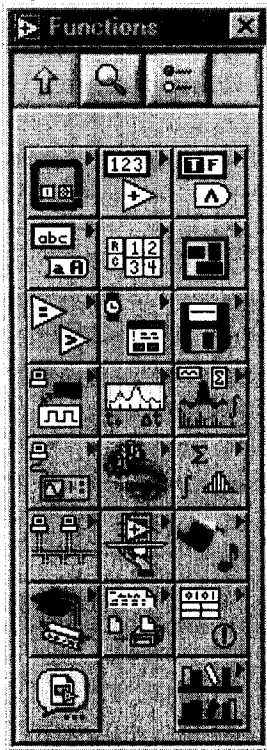


for writing/labeling



for wiring

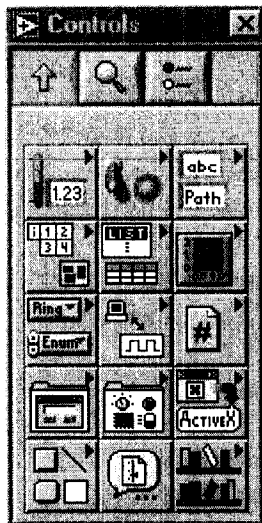
If you work with the Diagram view you have in addition the Functions palette.



Functions palette



This palette gives you access to all vis that are offered by LabVIEW.

The Controls palette that is displayed with the front panel allows you to create building blocks e.g. fields to display a value (Controls or Indicators).



Controls Palette

Basic Techniques:

- Open the help by clicking Ctrl+h.
- Wiring: Select the wiring tool  from the Tools palette, left click on the first terminal you want to connect and then left click on the second terminal. Note that the selected terminal blinks if you come in its vicinity with the wiring tool.
Trick: If you need to create an input or output to a terminal of a vi right click on it and select Create >> Constant (or Indicator or Control). If it doesn't work make sure you've selected the wiring tool from the Tools palette.
- Writing: Choose the writing tool  from the Tools palette and click where you want to write. Key in whatever you need.
- Selecting a vi: The Functions Palette is used to choose the vis. Click on the displayed icons to navigate one level deeper. The name of the group of the vis that are represented by the icon is displayed on top. Click on Instrument I/O >> Instrument Drivers >> hp81200 to find the PnP drivers.

4. Preparations

As our program should load a setting, go to the GUI and prepare any Setting under DSRA save it as EXSET. Click on the Setting New icon



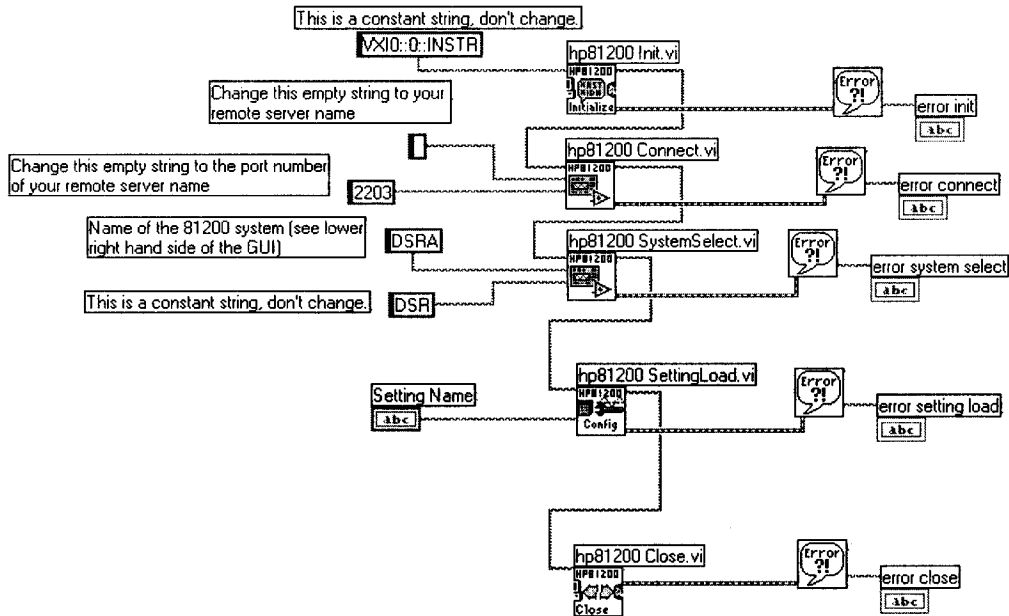
before returning to your LabVIEW program.

5. Writing your program

Your final program will look like the following, a step by step guide can be found afterwards:

The diagram:

This programming example uses the Agilent 81250 PnP drivers to load a setting.



Location of the vis in the Functions Palette:

hp81200 Init.vi: Instrument I/O >> Instrument Drivers >> hp81200

hp81200 Connect.vi: Instrument I/O >> Instrument Drivers >> hp81200 >> Administration

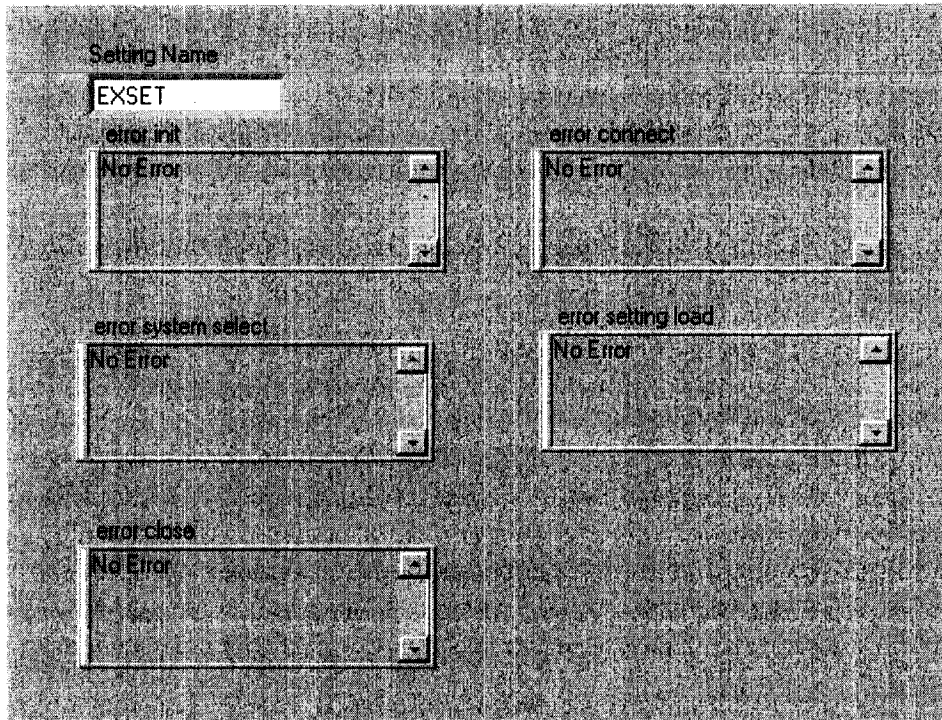
hp81200 SystemSelect.vi: Instrument I/O >> Instrument Drivers >> hp81200 >> Administration

hp81200 SettingLoad.vi: Instrument I/O >> Instrument Drivers >> hp81200 >> Mass Memory

hp81200 Close.vi: Instrument I/O >> Instrument Drivers >> hp81200

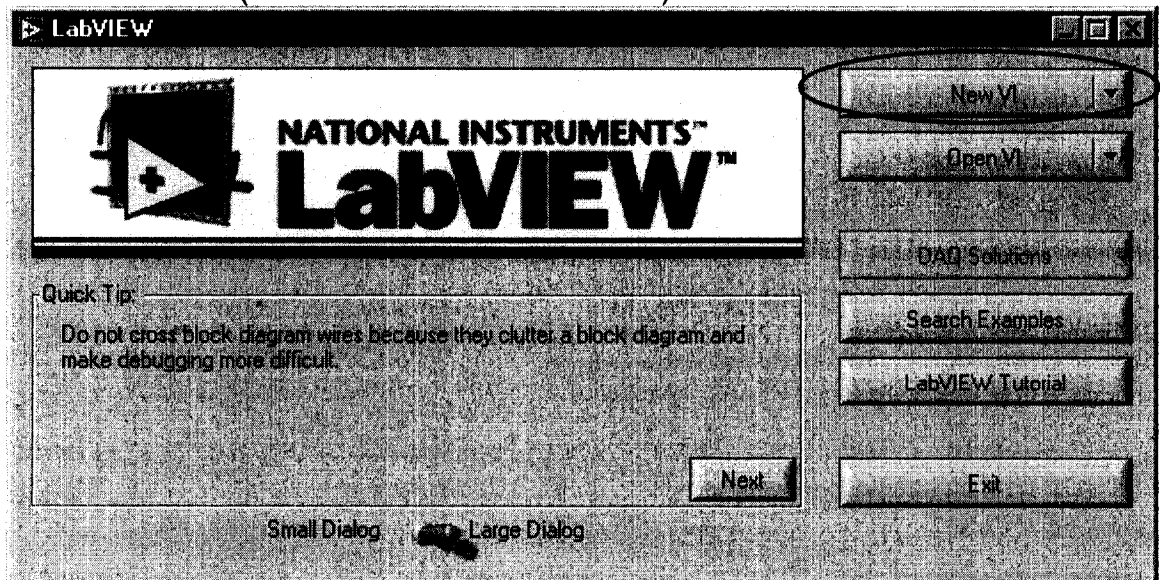
Simple Error Handler.vi: Time & Dialog

The Panel:




Select all the vis you need



1. Create a new vi (start LabVIEW select New VI)



2. Change to the diagram view (press Ctrl+E)
3. Display the Tools and Functions Palette (activate under **Window** in the menu bar)
4. On the Functions palette click on **Instrument I/O** followed by **Instrument Drivers** and **hp81200**.

5. Drag&Drop the hp81200 Init.vi to your Diagram (click on the text keep the mouse pressed and release it at the place in the Diagram where you want to have the vi)
6. In the Functions Palette double-click on the **Utility**, double-click on **Administration**. Here you find the hp81200 Connect.vi place it on your Diagram beneath the Init.vi as shown in the diagram view of the program listed above.
7. In the Functions Palette drag&drop the hp81200 System Select.vi to your diagram.
8. In the Functions Palette click on the upward arrow  to navigate one level higher. Double-click on **Mass Memory**. Find the hp81200 SettingLoad.vi and place it on your diagram.
9. Navigate again two levels higher by clicking two times on the upward arrow found in the Functions palette. Now you find the hp81200 Close.vi. Place it on your diagram.
10. Navigate to the highest level of the functions palette by clicking repeatedly on the upward arrow and then click on Time & Dialog and here select Simple Error Handler.vi. Place this icon 5 times on your diagram as shown in the program overview.
11. Now that your vis should be arranged as above we have to connect them. First right-click on each item, extend the Visible Items and select Label. Furthermore display the help by pressing Ctrl+h on your keyboard. If you point now on a vi, its description is displayed in the help window.

Create Inputs/Outputs

12. From the Tools Palette choose the wiring tool . Right click on each terminal that needs an input. E.g. take at the hp81200 Init.vi the terminal in the top left corner called resource name. As a help the name is displayed and in your help window you can see the connected terminal, too. After right-clicking choose **Create** and afterwards **Constant** from the appearing menu.
13. Write immediately the input VXI0::0::INSTR as shown in the program overview. If you did click anywhere after step 12, you need to select the writing tool  from the tools palette and click into your constant and start writing now.
14. Now here is a list that shows you the vi, which terminal needs to be connected and what input it needs. Right click on each terminal select **Create** and then **Constant** and start writing (sometimes there are even default values)

hp81200 Init.vi: Already done in step 12+13

hp81200 Connect.vi:
 server name: IP address or computer name of the computer where your
 firmware server is running
 port number: 2203

hp81200 SystemSelect.vi:
hp81200 system name: DSRA
hp81200 application name: DSR



hp81200 SettingLoad.vi:
setting name: EXSET (if you right click here and select **Create >> Control** instead of **Create >> Constant**, you'll find a control in the Panel view where you can enter the name of the setting)

Simple Error Handler.vi:
Right-click on the message terminal and select **Create** and then **Indicator**. The messages will now be displayed on the panel.

Connect the vis

15. Wire the **instrument handle out** of the predecessor vi to the **instrument handle** of its following vi. This is done by left clicking on the first terminal and again left clicking on the second terminal.
Note if you click somewhere on the free space in the diagram the wire will make this a point where it will make a turn.
16. Connect all the **error out** with the **error in (no error)** of the Simple Error Handler.vi.

Run your program

17. Save your program by selecting **File** and **Save as...**
18. First run your program in the highlighted mode:
click on the light bulb  and select the run button .
- Now you see the program flow of your program keep an eye on the error handler you can already see if it is okay or if it returns an error.
19. If this works click again on the light bulb so it isn't highlighted any more , press run and your program will run in normal mode.
If you encounter problems read the error messages in the Panel and try to solve it!

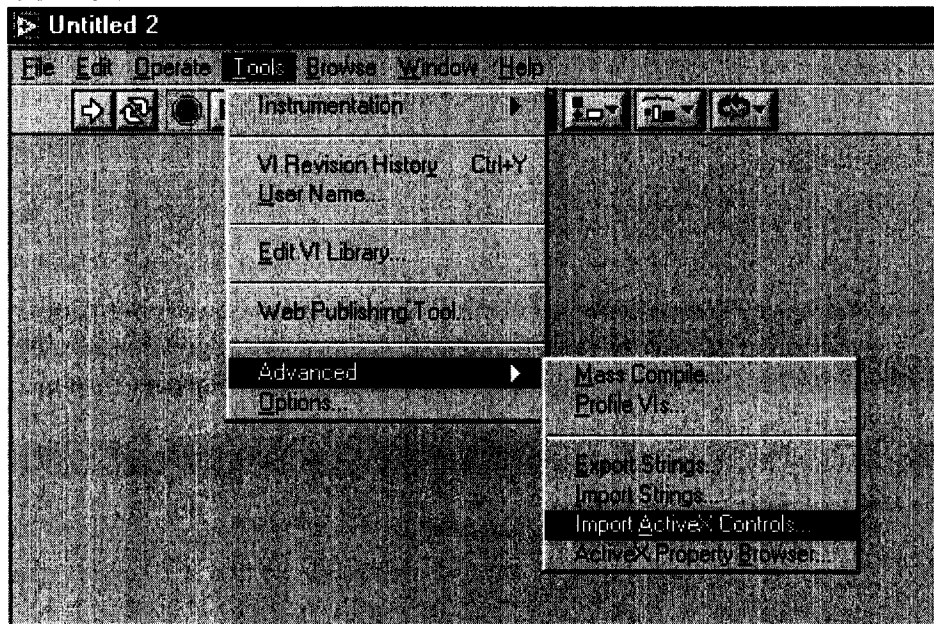
Tasks for advanced LabVIEW users

MUI – Eye Opening

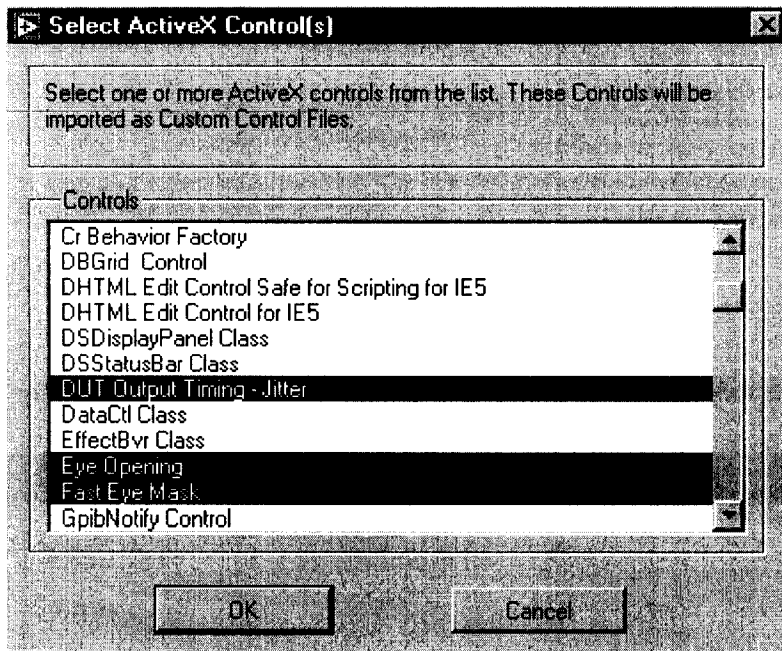
1. ActiveX

For writing a LabVIEW program that controls the MUI, ActiveX can be used. Do the following to import the ActiveX Control.

- 1) Select Tools from the menu bar click Advanced and Import ActiveX Controls

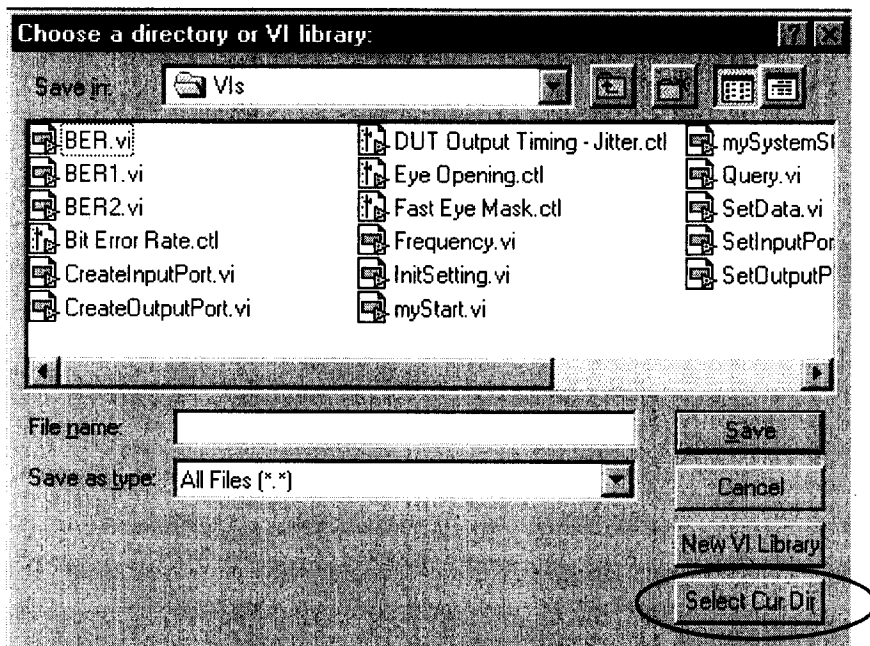


- 2) At the following dialog select your ActiveX controls that you plan to use (Eye Opening, for ambitious programmers DUT-Output Timing, Fast Eye Mask and Bit Error Ratio)



3) Click OK

4) Select a folder where you want to save the controls and click on Select Cur Dir



Now you are ready to use ActiveX controls in your LabVIEW programs.

2. The LabVIEW program

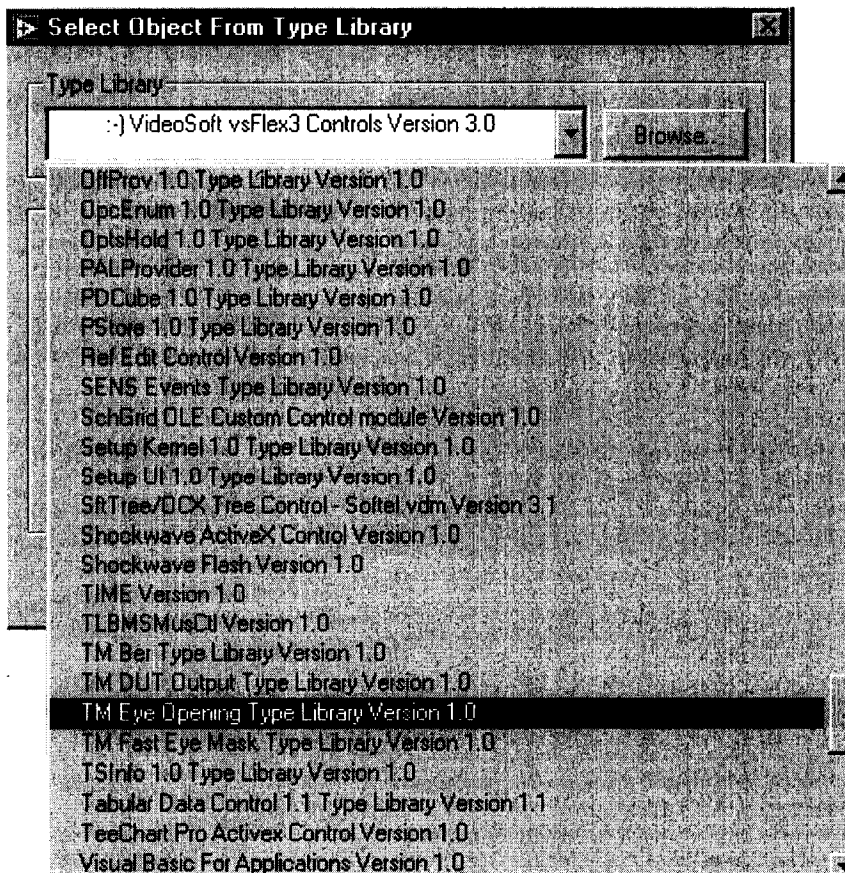
To use the MUI a BER measurement must be loaded in the GUI. Make sure the control inputs in your MUI program correspond to the setting of your GUI (server

name, port number,...). The settings in the GUI should correspond to Single-ended Normal and ECL to GND for the Analyzer and
For the MUI program you have first to include the ActiveX control into your Panel.

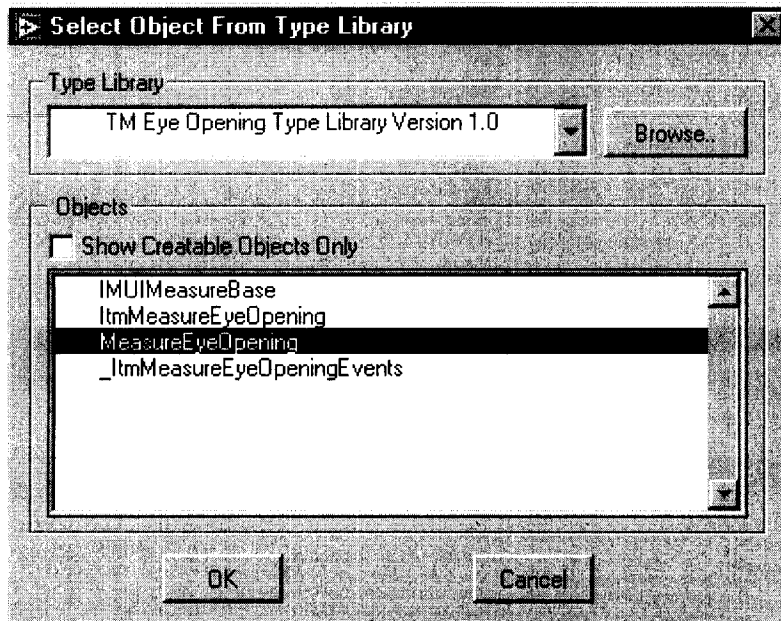
- 1) From the Control Palette choose Select a Control
- 2) Navigate to the location where you saved your imported controls and choose the Eye Opening.ctl.
- 3) Click on Open
- 4) Place the control wherever you want
- 5) Adjust its size.

Now you have the graphical ActiveX control in your Panel and in the Diagram View, change to the Diagram View (press Ctrl+E) and you'll find a TM_EYE_OPENING_Lib.ItmMeasureEyeOpening object.

The commands that control the ActiveX component are either Invoke nodes for calling Methods or Property nodes for setting Properties. In principal you do the following; from the Functions Palette choose Application Control and then Property Node or Invoke Node, place it into your diagram and right click on it. First expand the item Select ActiveX Class and choose the class TM_EYE_OPENING_Lib.ItmMeasureEyeOpening if you can't find it select Browse... from the Type Library choose TM Eye Opening Type Library Version 1.0 and as Object choose MeasureEyeOpening.



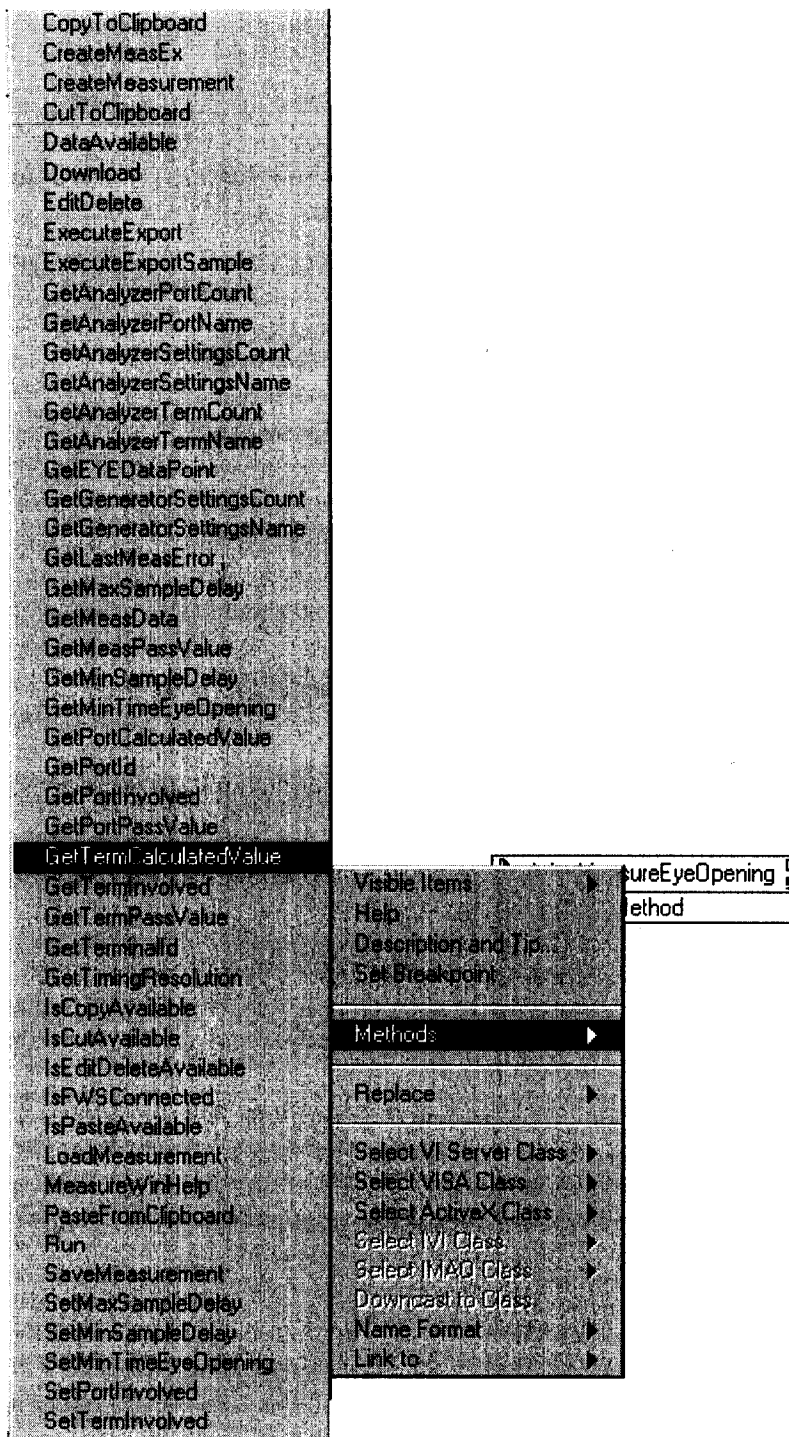
LabVIEW Type Library for ActiveX



LabVIEW Objects in Type Library

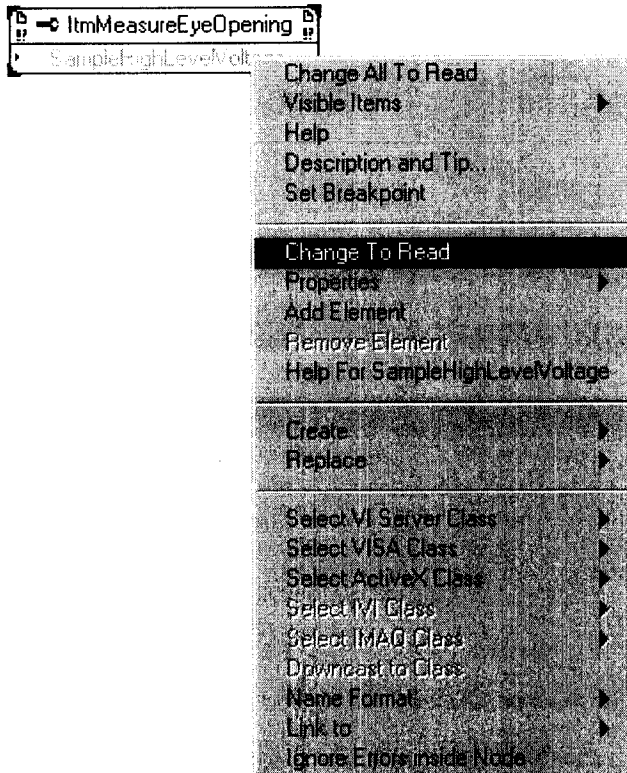
Now you have chosen the right class what remains is to choose the Property or Method that you need. Right click again on your Property or Invoke Node in the Diagram View go the Properties or Methods respectively and select what you need.

For example:



LabVIEW create Properties/Methods

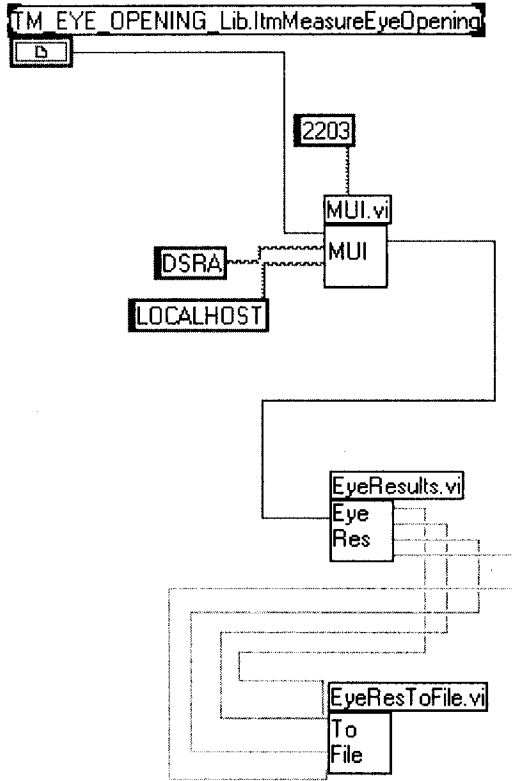
In case the parameters that should be outputs are inputs or vice versa right click and select Change to Read or Change to Write or Change all to Read/Write.



LabVIEW change Read/Write inputs

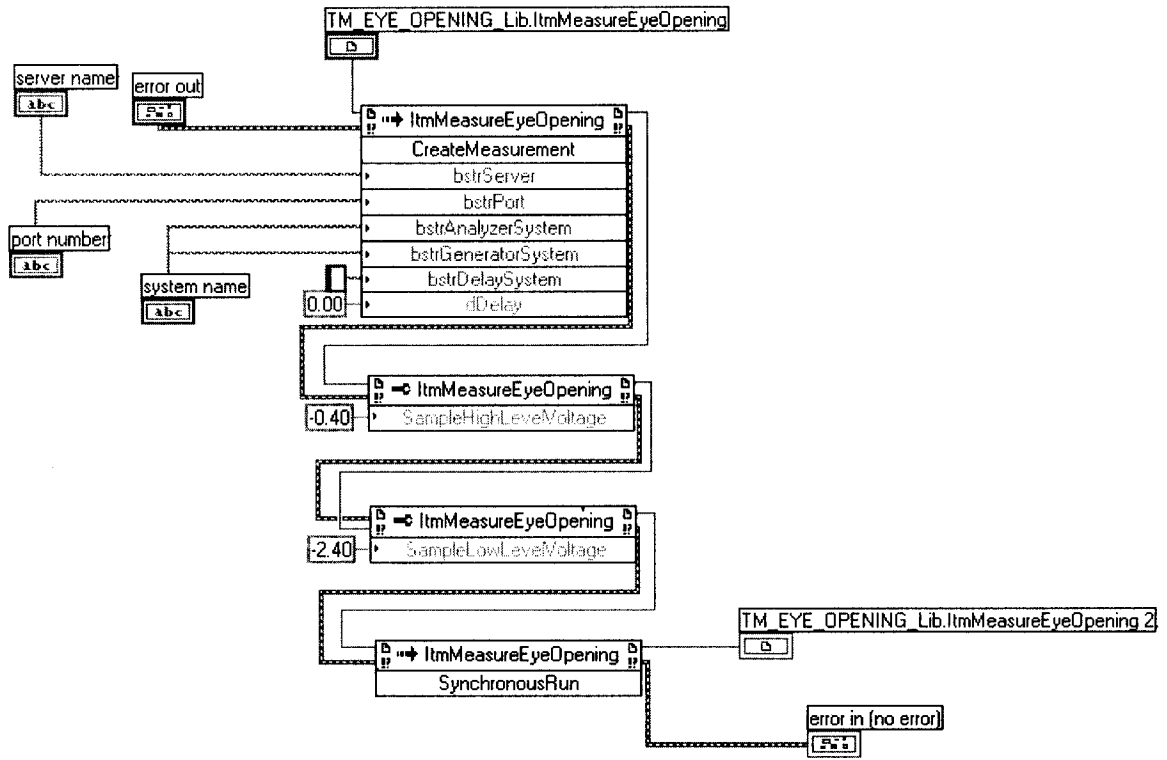
This is the general way to program in LabVIEW with the ActiveX controls from the MUI. Now here is the program that runs the Eye Opening displays the result and writes the values into a file. For sake of a better overview subvis are created and the following explanation will go from general to detail.

Here is the program:



LabVIEW MUI Eye Opening top level

The functions that need to be called to prepare and run the Eye Opening measurement are done in the MUI.vi.



LabVIEW MUI Eye Opening adjust settings

CreateMeasurement:

- 1) Functions >> Application Control >> Invoke Node put it on the diagram
- 2) Right click and select TM_EYE_OPENING_Lib.ItmMeasureEyeOpening
- 3) Right click and select Methods >> Create Measurement

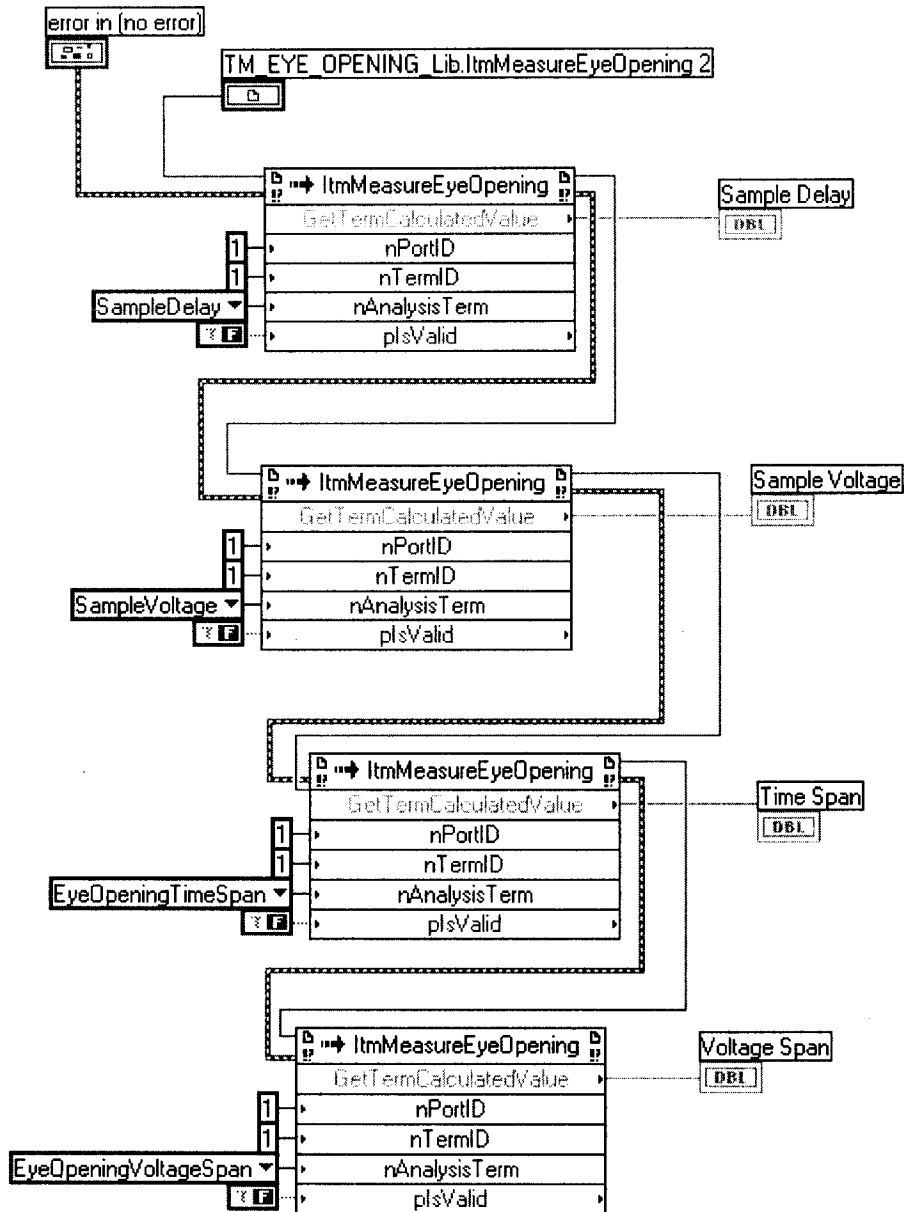
SampleHighLevelVoltage/ SampleLowLevelVoltage:

- 1) Functions >> Application Control >> Property Node put it on the diagram
- 2) Right click and select TM_EYE_OPENING_Lib.ItmMeasureEyeOpening
- 3) Right click and select Properties >> SampleHighLevelVoltage/
SampleLowLevelVoltage

SynchronousRun:

- 1) Functions >> Application Control >> Invoke Node put it on the diagram
- 2) Right click and select TM_EYE_OPENING_Lib.ItmMeasureEyeOpening
- 3) Right click and select Methods >> SynchronousRun


The next subvi, EyeResults.vi is used to read out the results that are displayed under the diagram.



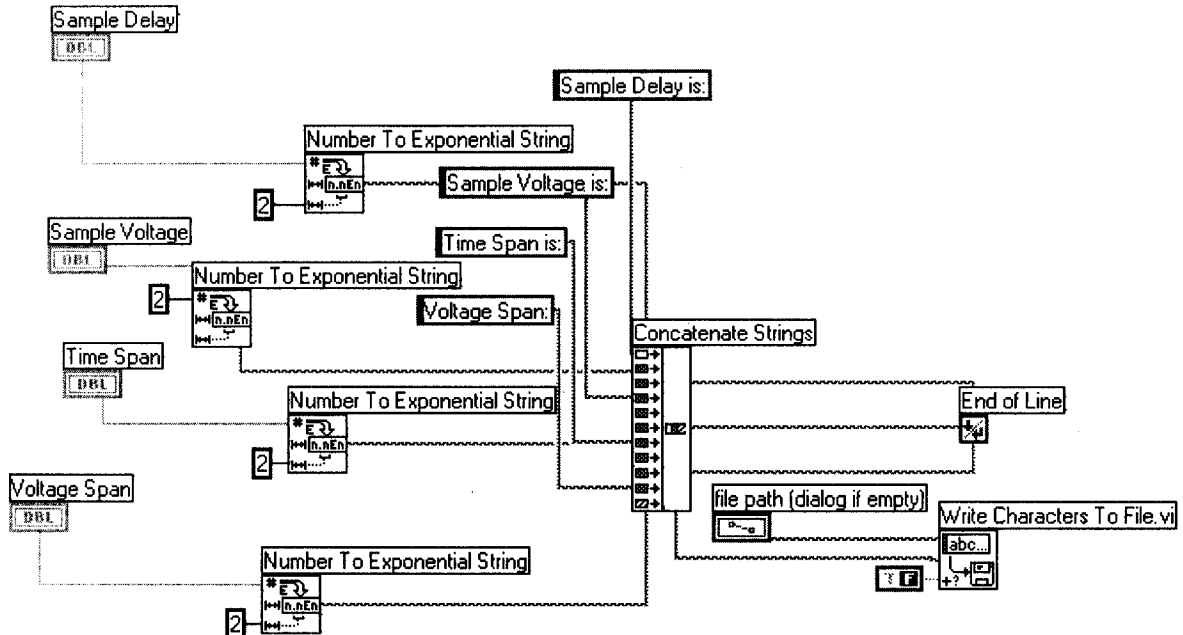
LabVIEW MUI Eye Opening retrieve values

You get to the Invoke node by selecting:

- 1) Functions >> Application Control >> Invoke Node
- 2) Right click on it and choose ActiveX Class
TM_EYE_OPENING_Lib.ItmMeasureEyeOpening
- 3) Right Click on it and select Methods >> GetTermCalculatedValue
- 4) Wire the inputs by right clicking on them, choose Create >> Constant and to choose the right input at nAnalysisTerm pick from the Tools Palette the

 to activate the drop-down menu.

The values that are returned by these methods are stored in the variable Voltage Span, Time Span, Sample Voltage and Sample Delay.
The next subvi – ToFile - writes these values to a file.



LabVIEW MUI Eye Opening create String

The idea behind this subvi is to write the results from the measurement into a file (specified by file path). For this purpose the Write Characters To File vi is used. So we need to convert the results, which are numbers to a String this is done by Number To Exponential String. The vi Concatenate Strings is used to create one string that contains all the measurement results, a label for each number and an end of line character so every result occupies a new line. Here is an overview of the vis that are used:

Number To Exponential String:
Functions >> String >> String/Number Conversion

Concatenate Strings:
Functions >> String
Right click on the left half of the vi and select Add Input until you have enough inputs.

End of Line:
Functions >> String

Write Characters To File:

Functions >> File I/O >> Advanced I/O Functions >> Open File

