

Experiment 4: LabVIEW Graphical Programming – A Primer

References:

1. Lisa K. Wells, *The LabVIEW Student Edition User's Guide*, Prentice Hall, Englewood Cliffs, New Jersey (1995). [<http://www.ni.com/>]
2. Barry E. Paton, *Sensors, Transducers & LabVIEW*, Prentice Hall PTR, Upper Saddle River, New Jersey (1999). [<http://www.phptr.com/>]
3. Nat'l Instruments, *LabVIEW - Proven Productivity (Seminar Series)*, 1997.
4. *LabVIEW* manuals (available in the Physics 405 lab for classroom use).
5. P. Horowitz & W. Hill – *The Art of Electronics* – for GPIB.
6. Physics 405 Lecture Notes (e.g. online) – for GPIB.

LabVIEW Graphical Programming Language

Graphical programming is a natural extension of the way scientists and engineers design equipment, experiments, and also communicate each other. Most of the time when an idea is developed, a general diagram is drawn or sketched and discussed. It is a very productive tool for rapidly designing and obtaining results.

With this in mind, *LabVIEW* (an acronym for **L**aboratory **V**irtual **I**nstrument **E**ngineering **W**orkbench) was developed in late 1980's by National Instruments, Inc for use in the desktop/workstation computer environment. *LabVIEW* soon became very popular and widely accepted in scientific and engineering circles. Graphical programming software packages are also available from many other companies. Hewlett-Packard's *HP-VEE* is another popular package.

In *LabVIEW* one can create a virtual instrument (VI) by selecting various icons from libraries and connecting them together on the computer screen. Each icon (and their interconnections) is translated into code by an internal code generator, such that the program can then be run on the computer. *LabVIEW* uses a graphical programming language called *G*, however you do not need to know this language to use *LabVIEW*.

LabVIEW Program Structure

Under its hood, a *LabVIEW* program is similar to a text-based program with functions and subroutines; however, in appearance it functions like a *virtual instrument* (VI). A real instrument may accept an input, process on it and then output a result. Similarly, a *LabVIEW* VI behaves in the same manner.

A *LabVIEW* VI has 3 main parts: a.) the *Front Panel* window, b.) the *Block Diagram* window and c.) the *Controls*, *Functions* and *Tools Palette* windows, which contain icons associated with extensive libraries of software functions, subroutines, etc. The *Controls Palette* is associated with the *Front Panel* window, the *Functions Palette* is associated with the *Block Diagram* window, and the *Tools Palette* is associated with both *Front Panel* and *Block Diagram* windows.

The appearance of the *Front Panel* window is similar to that of the front panel of an actual instrument. It may contain icons (libraries of which are contained in the *Controls Palette*) associated with graphical elements such as knobs, switches, pushbuttons, x-y plots/charts, etc. Data is input through the keyboard or mouse to interact with those virtual components (for example a switch could be turned on). The result is shown through various output components of the VI (for example on a graph). The *Front Panel* window is actually the main user interface of the VI.

The *Block Diagram* window is the VI's equivalent of source code. It itself is an executable program. Each component of a *LabVIEW Block Diagram* window is represented by an icon (libraries of which are contained in the *Functions Palette*). Here, an icon used in the *Block Diagram* window may be a function, a subroutine (i.e. a lower-level VI), or a control structure, etc. By connecting together the icons in the *Block Diagram* window with wires, one can control the flow/sequencing of the program.

Thus, by choosing specific icons associated with various *Controls* and *Functions* in conjunction with the use of various *Tools*, each segment of the *LabVIEW* VI can be built up in succession, very easily and rapidly, once some familiarity and experience with using *LabVIEW* has been gained by the program developer.

In addition, a *LabVIEW* VI is also inherently modular in nature. A series of sub-tasks can be acted upon to run the desired program. Each VI may be designated as a sub-VI of a larger program and so on to accomplish multitudes of tasks.

Exercise # 1: A Simple LabVIEW VI: A Random Number Generator

We will start with a simple *LabVIEW* program by designing a random number generator and display the results on a graphic screen (a strip chart recorder). Since it is more productive to follow a step-by-step procedure rather than wading through the manuals to learn about this software, a detailed procedure is provided below. Follow these instructions to do this first exercise. In the future you should consult the manuals (paper or on-line *HELP*) in order to understand individual controls, tools and functions.

Note: On your PC, create your own personal folder (e.g. MyFolder) in the Students sub-folder of the **LabVIEW 6.1** folder. Currently, this area on the C-drive of the PC should be:

C:\Program Files\National Instruments\LabView 6.1\Students\MyFolder

If the Students sub-folder is not present in the \LabView 6.1\ folder area on your PC, then create one, and then create your own *personal* folder inside it (e.g. use your initials). Create a shortcut to this Students sub-folder and place it on the desktop of your PC:



Alternatively, you could simply create your own personal *LabVIEW 6.1* folder in a \Students\ sub-folder of the My Documents folder on the desktop of the PC, i.e. C:\Desktop\My Documents\Students\MyFolder. Again, you could create a shortcut to this area and place it on the desktop of your PC.

Save all of your *LabVIEW 6.1* programs in your own personal folder. It is also advisable to back-up (all of) your files before you go home – e.g. either on a floppy disk, or to a special backup area that is specifically dedicated for Physics 405 students on a remote server, this area is: <\\Phyaplportal\phycs405\Common> {to access this area on your PC, go to Network Neighborhood, and then type in <\\Phyaplportal> in the address port, hit <cr> and you should see the PHYCS405 icon. You *must* create your folder as a sub-folder of the <\\Phyaplportal\phycs405\Common> area (because of security and read, write, delete, etc. access permissions. For ease of access to this backup area in the future, we encourage you to e.g. create a shortcut to this area and place it on the desktop of your



PC.

If you instead choose to use a floppy disk to backup your *LabVIEW 6.1* programs, floppy disks are available in the Physics 405 Lab – ask your P405 Lab TA and/or Jack Boparai (6102 ESB) for one. Store the backup floppy in your Physics 405 project drawer for safe-keeping.

Step 1: Starting *LabVIEW 6.1*: Double-click on the *LabVIEW 6.1* icon on the desktop of your PC:



Alternatively/equivalently, you can go to the Start menu of your PC, and select *Start >> National Instruments>> LabVIEW 6.1*. When the *LabVIEW 6.1* program starts up, a window will appear; click on the *New VI* button. Two new panels should appear along with *Tools Palette* and the *Controls Palette*. The two panels are: ***Untitled 1 Front Panel*** and ***Untitled1 Block Diagram Panel***. You should also see *Tools Palette* and the *Controls* (or *Functions*) *Palette*.

If any of these *Palettes* are not visible, go to the pull-down menu in either the *front panel* or *block diagram* windows and select *Window >> Show Tools Palette* and/or *Window >> Show Controls Palette* (or *Show Functions Palette*) to view them. The *Tools Palette* contains the basic tools to build your VI. Familiarize yourself with the icons on each of these *Palettes*. Left mouse click once on a particular icon in order to activate the respective tool. Right mouse click on a specific icon to learn what it is and what it does.


Step 2: At the *Controls Palette* (associated with the *Front Panel* window), use the mouse to first click on the icon associated with the graph menu, and then in the graph sub-menu, select (i.e. click on) the *Waveform Chart* (n.b. not the *Waveform Graph*) icon and place (i.e. drag) it onto the *Front Panel* window, approximately in the middle. If you are not satisfied with the placement/location of the chart in the *Front Panel* window, go to the *Tools Palette* and select the *Position* tool (arrow icon) and then use the mouse to move the chart to where you desire it to be. Then go back to the *Tools Palette* and select the *Operating* tool (hand icon with the upward-extended index finger). You can also label (i.e. name) the waveform chart *Random Number Plot* in its text window by selecting the *Labeling (Editing)* tool in the *Tools Palette* (this icon is marked as an *A* in the *Tools Palette*). If the waveform chart's label is not present/visible, right-mouse click on the waveform chart and use the *Show Label* from the pop-up menu of the waveform chart.

Step 3: Our random number generator will generate numbers from 0.0 to 1.0. However, the default chart Y-axis is from 0.0 to 10.0. To re-label the Y-axis of the chart, use the *Operating Tool* (i.e. select the Hand icon with upward-extended index finger in the *Tools Palette*) and then double-click on the 10.0 in the Y-axis of the chart, and in type in 1.0 where the 10.0 was. Another way to do this is to right-mouse click on the waveform chart to open the pop-up menu and then select *Y Scale >> AutoScale Y*.

Step 4: If you do not see the *Block Diagram* window, go to the *Window* pull-down menu in the *Front Panel* window and select: *Window >> Show Diagram*. When you were placing the waveform chart on the *Front Panel* window, *LabVIEW* put the corresponding icon for it on the *Block Diagram* window. You should see this icon on your *Block Diagram* window. Left-click the mouse on this icon in the *Block Diagram* window, and then press CONTROL-H to show a “help” description of this icon. This is a quick, easy way to get help info on a particular sub-VI.

Step 5: Since we want to interconnect and operate on these items to generate random number, open the *Functions Palette* from *Windows >> Show Functions Palette* menu at the top of the block diagram window. Then select (i.e. drag) the *Random Number function* (double dice icon) from its *Numeric >> Random Number (0-1)*.

Step 6: In order to start and stop the random number generator, we need to set up a condition on our system. This can be done using the *While Loop (Structures)*, which is contained in the *Functions Palette*, associated with the *Block Diagram* window. Select (i.e. drag) the *While Loop* icon – located in *Structures >> While Loop* and place it in the top upper left hand corner of the *Block Diagram* window. Do not release the mouse! Then, still holding the mouse down, drag the *While Loop* icon diagonally downward and to the right, in order to enclose the other icons (as shown below) and then left-click the mouse a second time when the *While Loop* has been expanded to enclose all of the other icons. If you do not see the other icons, or the enclosing of them was not properly done, open up the pop-up menu of the *While Loop* (right-mouse click on the periphery of the *While Loop*) and select *Remove Loop*. Then redo this step. Note that you can also alternatively/equivalently use the *Position Tool* (arrow icon) from the *Tools Palette* to move the *While Loop*'s boundary and/or move the other icons into/inside of the *While Loop* of the *Block Diagram* window.

Step 7: Right-click on the CCW-rotating arrow  of the *While Loop* control in the *Block Diagram* window at the bottom right-hand corner of the *While Loop* – a pop-up menu will appear – and then click on/select *Create Control*. An *On/Off* button will appear in the *Front Panel* window and a green *T-F* icon will simultaneously appear in the *Block Diagram* window, associated with this *On/Off* button. Use the *A* icon from the *Tools Palette* to re-label this switch *On/Off* in the *Front Panel* window. The *On/Off* switch enables a simple way to initiate/stop the generation of random numbers after left-clicking the mouse on this button.

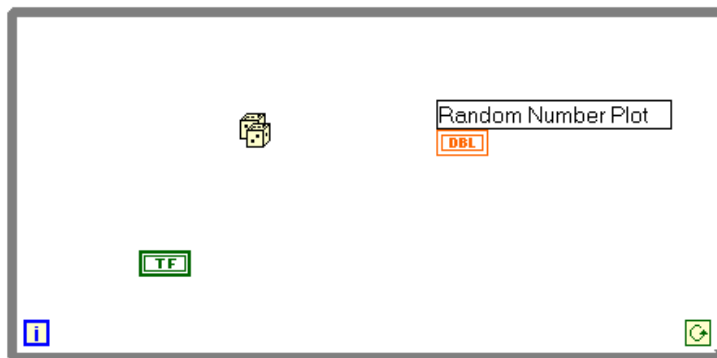


Figure 1. Basic random number generator vi.

Step 8: As in a text-based computer program, the *While Loop* executes all code contained within its boundary until the expression on the conditional input (CCW circular arrow) becomes false. The iteration terminal (marked as an **i** in the bottom left-hand corner of the *While Loop*) indicates the number of times the *While Loop* has executed so far.

In order to make the necessary inter-connections between icons contained within the *While Loop*, click on the *Wiring Tool* (wire spool icon) from the *Tools Palette*. Then go to *Block Diagram* window and move the mouse to the *right-hand* side of the *Random Number* function (double-dice) icon. Wait until the double-dice icon begins to flash, and then left-click on the mouse. Release the mouse button and then move the mouse to the *left-hand* side of the “DBL” icon of the *Random Number Plot* terminal (as shown in the figure below). You will be at this terminal when the icon begins to flash. Then left-click the mouse to *terminate* this wire. You should then see the wire turn solid-orange in color. If you see a dashed black line, the wire is bad. In this case, go to the *Edit* pull-down menu of the *Block Diagram* window, and select *Edit >> Remove Bad Wire*, or alternatively/equivalently, you can use the keyboard combination *Control-B*, or simply right-click the mouse at/on the wire to remove the bad wire. Use the wiring tool to connect the green “*T-F*” icon (associated with the *On/Off* button) to the CCW circular arrow of the *While loop* as shown in the figure below. When finished, the wiring of icons in the *Block Diagram* window should appear as that shown below in Figure 2:

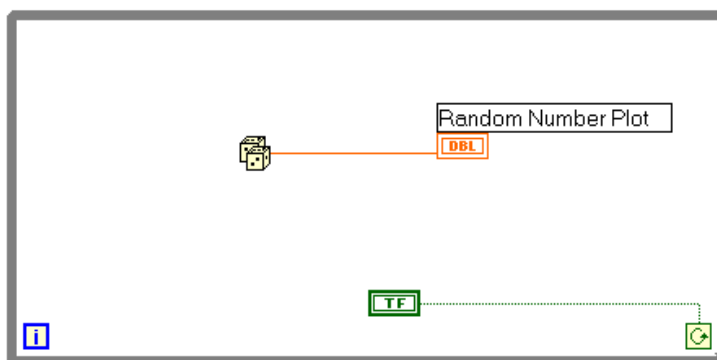


Figure 2. Completed block diagram of the *random number vi*.

Step 9: You have now completed programming of this *LabVIEW VI*. Return to the *Front Panel* window (Remember to use *Windows >> Show Front Panel* or simply click the mouse on the *Front Panel* window.)

Step 10: After selecting the *Operating Tool* (Hand icon with upward-extended index finger), use the mouse to left-click on your *On/Off* switch on the *Front Panel* window to turn it *On*. Setting the *On/Off* switch to *On* simply sets the variable that the *While Loop* tests on to *true* – i.e. the code inside the *While Loop* is ready to be executed when the global *Run* button for this *LabVIEW VI* is engaged. At the *Operate* menu on the top of the VI (usually located below the Pull-down menu), click on the *Run* button icon. To stop the VI, first click your *On/Off* switch to *Off*, and then click on the *Stop* button icon (at the top of the VI). If you wish to clear the chart, right-mouse click on the chart and then select *Data Operation >> Clear Chart*.

Step 11: Explore the effects of *Pause*, *Run Continuously*, and *Abort Execution* buttons associated with the *Operate* menu, located immediately adjacent to the *Run* button near the top of the *Front Panel* and/or *Block Diagram* windows.

Step 12: Save your VI program to your personal folder in the \Students\MyFolder folder by using the *File* pull-down menu of the *Front Panel* window, select: *File >> Save As* and then name your *LabVIEW* program *Random.vi*.

Exercise # 2: Timing, Analysis and File I/O:

In this exercise we will modify the *Random.vi*, adding timing and at the end of the while loop analyze the data and then write the data to a text file, that in turn can read into a spread-sheet program, such as Excel, for subsequent analysis.

Step 1: Open *Random.vi* from the *LabVIEW* menu bar *File >> Open*, if it has not been opened yet. Immediately save this VI as e.g. *Timing.vi*

Step 2: From the *Controls Palette* associated with the *Front Panel* window, select *Numeric >> Knob* (or *Numeric >> Dial*) and add it to the *Front Panel* window. Use the *A* icon in the *Tools Palette* and re-label the knob as *Delay (ms)*. See Figure 3 below.

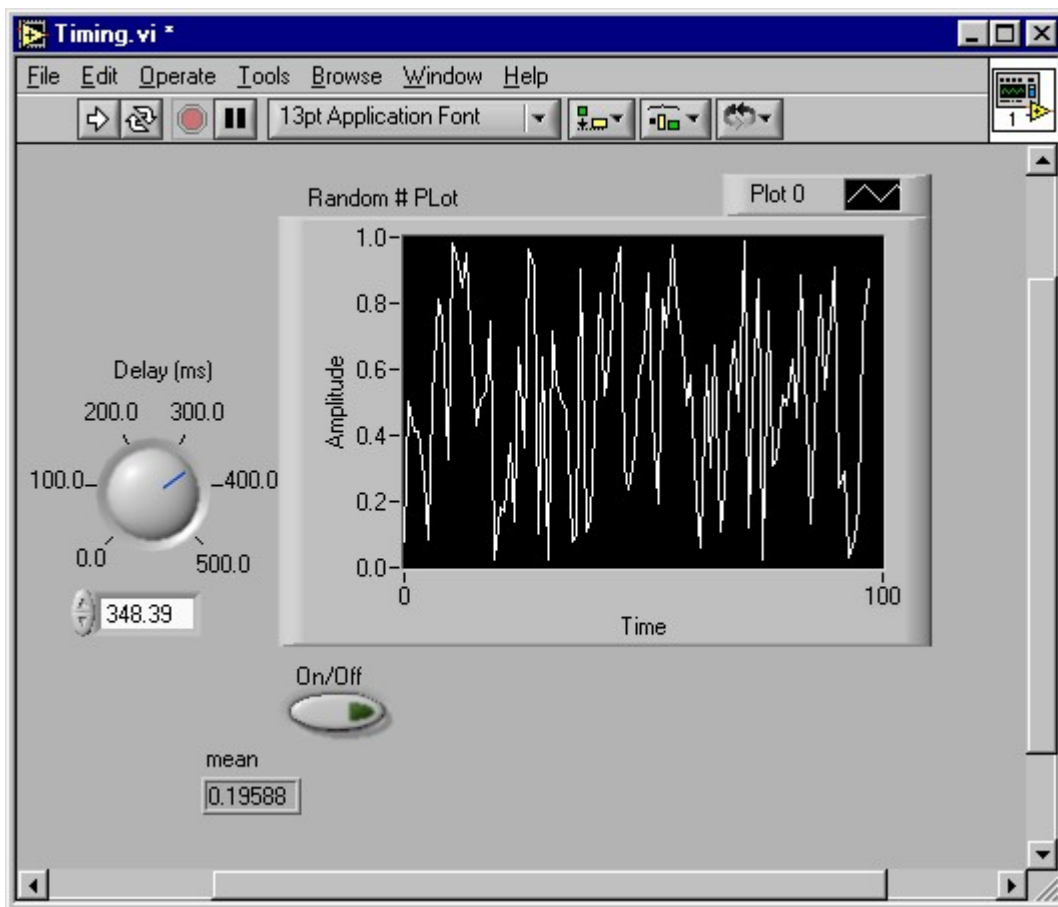


Figure 3. Adding knob, analysis and file I/O to the VI.

Step 3: Use the *Operating Tool* (Hand with upward-extended index finger) to change the scale on the *Delay* knob from 10.0 to 500.0 ms.

Step 4: If you wish to e.g. additionally have a numeric indicator that displays the actual knob value, right mouse click on the *Delay (ms)* control knob and select: *Visible Items >> Digital Display* on its pop-up menu.

Step 5: In the *Block Diagram* window, if the icon for the *Delay (ms)* control knob is not inside the *While Loop*, then select the *Position tool* (arrow icon) in the *Tools Palette* and move the delay (ms) icon inside the *While Loop*.

Step 6: At the *Functions Palette* associated with the *Block Diagram* window, select *Time & Dialog >> Wait Until Next ms Multiple* (timer icon) and place it *inside* the *While Loop* as shown below in Figure 4. The timer icon looks like a metronome. Then use the wiring tool in the *Block Diagram* window to connect the *output* of the *Delay (ms)* control knob to the *input* of the timer icon, on the left-hand (input) side of the timer.

Step 7: Return to the *Front Panel* window, run this VI and observe the effects of varying the delay time on the random number generation.

Step 8: We additionally would like to have this program compute e.g. the mean value of the random numbers generated. This statistical function is available from the *Block Diagram* window's *Functions Palette's Analyze >> Mathematics >> Probability and Statistics >> Mean.vi*. Use the mouse and place it outside the *While Loop* as shown below in Figure 4. Since we would like to view the mean value, *LabVIEW* provides a convenient way to generate such an indicator. Position the mouse over the top right corner of the *Mean* function and open the popup menu. Select *Create Indicator*. This will automatically create a numeric indicator labeled *mean* on the *Front Panel* window.

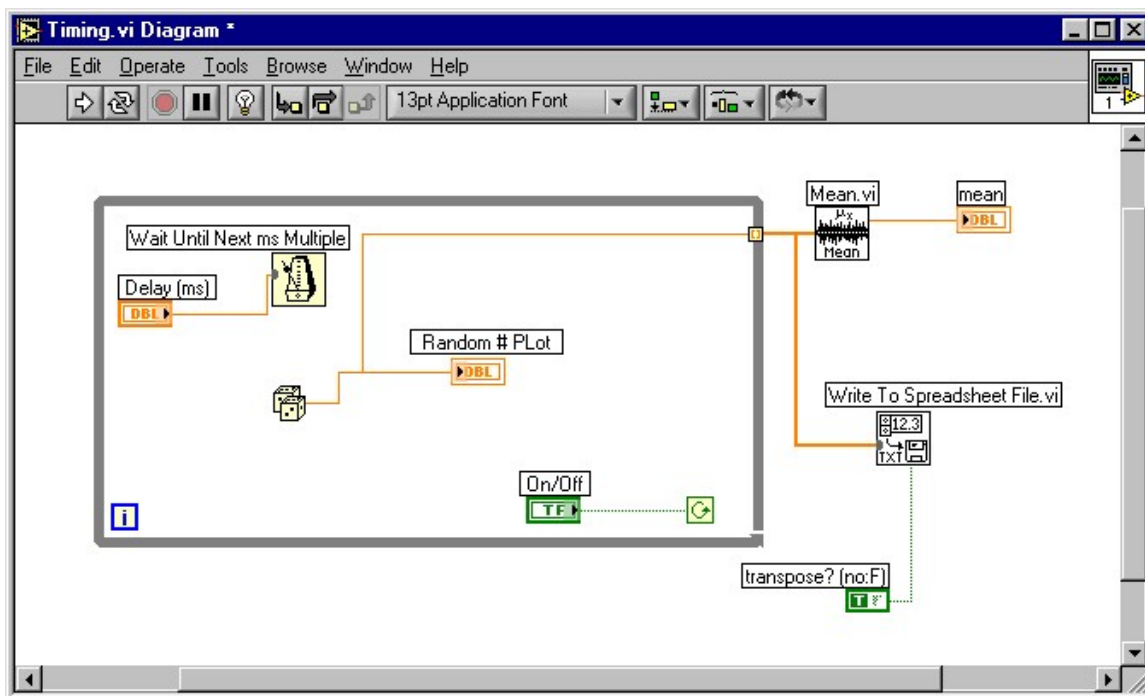


Figure 4. Block diagram of the *Timing.vi* for statistical analysis and file I/O.

Step 9: When a sequence of random number data is generated, we would also like to e.g. save this data into a text file. Once such a text file is generated, it can then read in “off-line”, e.g. using the Microsoft *Excel* spreadsheet program for further processing/data analysis.

To create the sub-VI for File I/O, go to the *Block Diagram* window's *Functions Palette* and select *File I/O >> Write to Spreadsheet File.vi*. Place this VI outside the *While Loop* in the *Block Diagram* window.

Step 10: Wire the icons in *Block Diagram* window as shown above in Figure 4 using the wiring tool. Note that:

- When you wire from inside of the *While Loop* to the mean VI, the wire initially will be broken (as shown by dashes). Right-mouse-click on the black tunnel that appears at the intersection of the *While Loop* and the wire to open a popup menu – Select *Enable Indexing*. The wire will then change to a thick orange line. This procedure allows the data to pass as a set at the termination instead of final data point only.

- (b) When you place the wiring tool at the terminals of the *Write to Spreadsheet vi*, a tip strip or input terminal appears along with its highlighted label. When you wire up the *Write to Spreadsheet vi*, make certain that you wire into the *1D Array* input. You can right-click on the *Write to Spreadsheet vi* icon and select *Visible Items >> Terminals*. When done wiring, right-click again on this icon and deselect this option.
- (c) The default format of the output data file is a row of numbers. This is not very useful for input to a spreadsheet program, e.g. Microsoft *Excel*. An output file with a column of numbers is much easier to use with such a spreadsheet program. Click on the *Write to Spreadsheet vi* icon with the mouse, and then press CONTROL-H to show the help description for this VI. You will see that the bottom-most input terminal of the icon is associated with *Transpose Data* option. Right-click on the *Write to Spreadsheet vi* icon, select *Visible Items >> Terminals* and then by clicking on this bottom-most *Transpose Data* input of the *Write to Spreadsheet vi* icon, and then right-mouse clicking on this input, select *Boolean Control* for this input, and then set this control from *F* (False) to *T* (True). When *Write to Spreadsheet vi* executes, it will then write out the data to a file in *column* (not *row*) format.

Here is a slick editing shortcut: When coding-up a *LabVIEW Block Diagram*, you will find that the most commonly used tools are the *Positioning* (arrow) and *Wiring* (spool) *Tools*. Rather than having to go to the *Tools Palette* each time to switch back and forth between these tools, simply press the <Spacebar> on your keyboard. With each depression of the <Spacebar> key, the mouse cursor will toggle back and forth between the *Positioning* and *Wiring* *Tools*. If you require some other feature in the *Tools Palette*, press your keyboard's <Tab> key. As you press <Tab> repeatedly, the mouse cursor will cycle through all the possible tools in the *Tools Palette*!

Step 11: The above steps complete the *Timing.vi* program. Save your program (e.g. as *timing.vi*), and then from the *Front Panel* window, click on the *Run* button icon to run the program. Once the data is collected, at the end of the *While Loop* a popup window will ask you for a file name and where you would like to save the data. Name your text data file e.g. *mydata.txt* and save your text data file in the same folder where your *LabView* program(s) reside. Then view/read in the data just written out by opening this text file using e.g. Notepad, Wordpad, or the *Excel* spreadsheet program. If you use *Excel*, note that this text data file contains tab-delimited data and appears as one *column* in *Excel*. Make certain that you save your VI before you quit the program – you will need it for the following exercises!!!

Exercise #3 Debugging a LabVIEW Program:

Here we discuss/show several useful techniques of debugging a *LabView* program. These debugging techniques are Execution Highlighting, breakpoints, single step, and probe techniques.

Step 1: In debugging a VI, it is very helpful to see how the program executes. Using the *Execution Highlighting* button, marked as a light bulb and located at the top portion of the *block diagram* window's *Operate* menu (usually located in the top second row), the animated flow of data from one node to another can be visualized. This feature is usually

used in conjunction with single-step mode. Click on this button and run the program. Observe how the data "bubbles" move from one node to another in the *Block Diagram* window. When you are done, turn your *On/Off* switch off and then deselect the *Execution Highlighting* button (i.e. use the mouse and click on this icon again). Save your data in the same *mydata.txt* file by overwriting it.

Step 2: Sometimes one would like to temporarily suspend the execution of a VI in order to inspect the inputs/outputs of a sub-VI while troubleshooting. This function is performed by setting breakpoints in a VI and the activating the so-called *Probe* feature. In order to set a breakpoint for a particular Sub-VI, select the *block diagram* window. Left-mouse click on the *Set/Clear Breakpoint* icon (the stop sign) in the *Tools Palette*. Then position the mouse pointer on the *Random Number* (double-dice) function in the *block diagram* window and click on the mouse. The *Random Number* function will now have a red outline. Then go back to the Tools Palette and click on the *Operate Value* icon (Hand with extended upward-pointing index finger).

Step 3: In the *Front Panel* window, click the *On/Off* button to *On*, then run the VI. The *Random Number* function in the *Block Diagram* window should flash, indicating that it is ready to execute.

Step 4: Then go back to the *Block Diagram* window, and left-click on the *Probe* icon (a circle with P inside) in *Tools Palette*. Then position the mouse (with probe attached to it) at the wire going from the *Random Number* function to the *Chart* function. This wire will then begin to flash. Left-mouse click on this wire to insert the *Probe* between the two functions. The *Probe* will be marked as a number, such as 1 in a rectangular box in the *Block Diagram* window.

Step 5: To execute the program, left-mouse click on the *Step Over* button (green arrow on top of a yellow square) located at the top of the *Block Diagram* window's *Operate Toolbar*. Note that the generated number now displays in the *Probe* window.

Step 6: Left mouse click on the *Step Into* button (green arrow into a yellow square) and *Step Over* buttons a few more times to see how single stepping works. Watch each icon in the *Block Diagram* window flash in succession. Move the *Front Panel* and *Block Diagram* windows apart in order to observe the plot display update on the *Front Panel* window. To run normally, right-mouse click on the red-highlighted *Random Number* (double-dice) icon and select *Clear Breakpoint* in the pop-up window, then deselect the *Pause* button at the *Operate Toolbar*. The generated random numbers will be displayed in the *Probe* window one at a time. Make your delay large to watch the displayed numbers.

Step 7: Turn *Off* your *On/Off switch*, click on the *Stop* button. Then click on the at the top right-hand corner of the *Probe*'s display window to delete the probe. Then exit/quit this program. Do not save any changes to your *LabVIEW* VI when you quit this exercise!!!

Thus, for future *LabVIEW* program development, use combinations of the above techniques to debug/troubleshoot these programs when needed.

Exercise # 4 Adding Modularity to a LabVIEW VI:

In a text-based computer program, we routinely use subroutines (written e.g. in FORTRAN, BASIC), procedures (written in Pascal) or functions (written in C and/or C++). Their purpose is to make the overall program clear and modular. Similarly, we can add modularity to our VI in *LabVIEW*. Many VI's can be formed into a group to create another VI. These VI's can also be used later on in other programs, thereby reducing overall program development time. In this exercise we will create a simple sub-VI for our *timing.vi* program.

Step 1: Open the *timing.vi* program that you created earlier in Exercise # 2.

Step 2: In the *Block Diagram* window, use the mouse to select the *Position Tool* (arrow icon) from the *Tools Palette*. Then select both *Mean.vi* and *Write to Spreadsheet File.vi* by using shift-left mouse click (i.e. holding down the shift key with one hand, use the mouse with the other hand to left-mouse click on each of these icons in succession). Shift-left mouse click is a shortcut method that enables one to select several VI's at the same time!

Step 3: At the top *Pulldown* menu of the *Block Diagram* window, select *Edit >> Create SubVI*. This automatically creates a sub-VI from the section of code you selected. The sub-VI appears as a new icon in the *Block Diagram*. Right-click the mouse on this new Sub-VI icon and choose *Description and Tip* to compose a help and/or description for this new sub-VI, so that days or months later this info will remind you what this VI does.

Step 4: Double click on the new sub-VI to see the location of the new sub-VI in both the *front panel* and the *block diagram* windows. Save this new sub-VI as *mysub.vi*. Also save your original, but now modified *timing.vi* as *timingsub.vi*.

When you create several (or many) related controls and functions, it is logical to group them together and label them as a sub-VI. This is especially true with many instruments and complex programs.

Exercise # 5 GPIB and LabPC+ Data Acquisition Boards:

Instead of directly manipulating the central processing unit (CPU) or microprocessor in the computer, one can use plug-in data acquisition boards and data communication bus to communicate with the CPU. This provides flexibility in reusing the system for many different ways. In our computer we have a GPIB (acronym for **General Purpose Interface Bus**) plug-in board and a Lab-PC+ multipurpose data acquisition plug-in board.

The GPIB plug-in board acts as a controller and provides a quick way to communicate with instruments (such as a DMM, Oscilloscope, Lock-in amplifier, etc.) that already have built-in GPIB instrumentation. In the GPIB environment an instrument acts as a talker (send data), listener (receive data) or controller (manager of the GPIB). In this lab we will generally use our computer to play all three roles. By assigning the unique instrument address (an identifying number string such as 22 for our lab's HP 34401A DMMs) to the instruments, one can communicate with as many as 15 instruments (or more with GPIB extenders) in a daisy-chain of instruments. The typical data transfer rate

of GPIB is about 1 MByte/second. The international standard used to implement its hardware and software is based on IEEE 488 Standards.

In contrast to GPIB, the National Instrument's *LabPC+* DAQ card is a multipurpose data acquisition board. It can measure analog signals, output analog voltages, input and output digital levels, pulses and also provides timing controls. Due to its multi-purpose capability, DAQ boards are more expensive than GPIB plug-in boards. For example, the *LabPC+* DAQ card costs approximately twice as much as a GPIB board. The *LabPC+* DAQ card has 8 analog input (ADC) channels (ranging between ± 5 volts), 2 analog output (DAC) channels (again ranging between ± 5 volts). The *LabPC+* DAQ card also has 24 TTL digital I/O channels. The *LabPC+* DAQ card also has 3 TTL digital counter/timers, which are useful for counting pulses, event timing, pulse generation, frequency measurement, etc.

In this exercise we will explore how to configure (i.e. set up) GPIB and the *LabPC+* DAQ card for data acquisition and communication.

Step 1: Shut down the computer, and then turn off the AC power to the computer and the HP 34401A DMM.

Step 2: Locate the 2 meter long GPIB cable at the back of the computer. It has 24 pin connectors at each end. Note that the connector is dual type – one male and one female. Do **NOT** yet connect the GPIB cable up to the HP 34401A DMM.

Step 3: Now turn on power to the HP 34401A DMM and the computer. Log back into the computer. When powering up the HP 34401A DMM, observe that the DMM will go through a self-test and display its GPIB address number (typically it is 22 by default). Now connect the GPIB cable to the DMM, on its back panel connector.

Very Important: At the end of this lab session, disconnect the GPIB cable from the HP 34401A DMM before powering off the DMM! Otherwise, the GPIB interface in the PC can get “glitched”, which in turn could cause the PC to crash!

Step 4: Double-click on the National Instrument's *Measurement and Automation Explorer (MAX)* icon on the desktop of your PC (or from the PC's Start Menu):



Alternatively/equivalently, you can access the *Measurement and Automation Explorer* directly from *LabVIEW*: Click on the *LabVIEW 6.1* shortcut on the desktop of your PC, and then in the *LabVIEW* main panel, click on the *DAQ Solutions*. Yet another, equivalent/alternative method is to access the *Measurement and Automation Explorer* from the *Front Panel* window of your own VI, select from the pull-down menu *Tools >> Data Acquisition >> DAQ Solution Wizard*.

Step 5: Once *MAX* is up and running, another pop-up window appears, select: “I want *MAX* to search for New Devices every time I launch *MAX*”. Then expand (i.e. double-

click on) “*Devices and Interfaces*” in the Configuration window (left-hand side) of *MAX* – see figure below, then expand (i.e. double-click on) “*GPIB0*” and then expand/double-click on “*Instrument0*”. When there is a GPIB board installed on your PC and the DMM is connected via the GPIB cable to the GPIB interface of the PC, *MAX* will report that it has found the HP34401A DMM and show its address, etc. in the Instrument0 Attributes window (right-hand side) of *MAX* – see figure below. Double check the address of the HP34401A DMM before quitting – left-mouse click on the (for exit). You can also use *MAX* when you have additional GPIB instruments daisy-chained (i.e. added) to the GPIB cable.

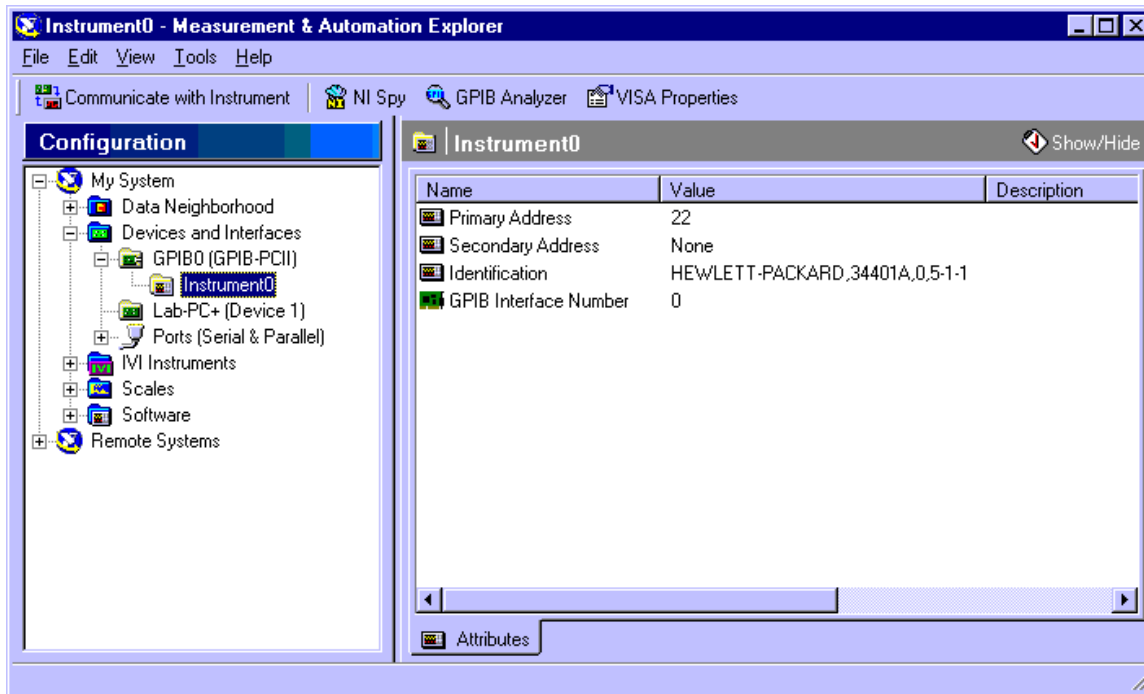


Figure 5. National Instrument’s *Measurement and Automation Explorer (MAX)* – use *MAX* for verifying the correct functioning of DAQ equipment.

Exercise # 6 Experiment with GPIB:

As stated earlier, GPIB is useful for acquiring data from bench-top lab instruments that have built-in GPIB interfaces. In our lab we have the HP34401A DMM, we will use to understand/learn about GPIB operations. In order to learn more about GPIB and its physical and hardware connections refer e.g. to the references for this lab.

As a means to provide better data communication, there exists a standard known as *Virtual Instruments Software Architecture (VISA)* for *LabVIEW*. As an exercise we will use the built-in *LabVIEW* VI using *VISA* Examples. In the next exercise you will build your own GPIB VI to communicate with a DMM.

Step 1: Turn on the DMM and make sure it is connected to the PC via the GPIB cable.

Step 2: The *LabVIEW* HP34401a VI program is located in the following area on the C-Drive of the PC:

C:\Program Files\National Instruments\LabView 6.1\instr.lib\hp34401a\

Find the *hp34401a.lib* program in this folder, right-click on it with the mouse and create a shortcut for it – see below. Then drag this shortcut to your PC’s desktop for future use.



Step 3: Double-click on this newly made shortcut for the *hp34401a*, or equivalently/alternatively, startup *LabVIEW 6.1* directly, open the *HP34401a Getting Started.vi* and observe the controls of the *hp34401a* on its *Front Panel* window. You may wish to double-click on the blue task-bar on the top of the *Front Panel* window to expand this window to full-screen size. Note the current GPIB address of the HP34401A as shown on the *Front Panel*. Change the VI’s GPIB address if it is different from that of the real DMM on your bench. They must agree if GPIB communication is to successfully take place!!!

Step 4: Select the DMM function to 2-wire resistance measurement via a control port located on the *Front Panel* window of the *HP34401a Getting Started.vi*.

Step 5: Run the *hp34401a* VI and measure the open circuit resistance value. Did you hear a “click” inside the HP34401A DMM when the program made this measurement? Observe the *Block Diagram* for the *hp34401a* VI. Use *Execution Highlighting* and trace the signal flow. If you receive any error messages (normally you should not get any such messages), analyze and troubleshoot the block diagram to determine the nature of the problem. If the GPIB address is not set properly, error messages will reveal that the GPIB instrument is not responding.

Step 6: Attach a known resistance to the HP34401A DMM input (use e.g. the bench resistor box) and measure the resistance. To change the precision of the display, right mouse click on the *Front Panel* window on the display that shows the measured value. Select *Format and Precision* to change the precision.

Step 7: Turn on your function generator and set it to output a ~ 1 Volt peak-to-peak, ~ 1.0 KHz sine wave. Change the DMM function to AC Voltage via the control port on the *hp34401a* VI’s *Front Panel* to measure the amplitude of this input signal. Change the DMM function (in same manner) to Frequency and observe the measured value(s). Note the improvement of e.g. 7½ digits of precision on the *hp34401a* VI as compared to that of the 6½ digits of precision on the physical HP34401A DMM’s display. In order for this to occur, consecutive/repeated measurements via the *hp34401a* VI must be fairly identical, i.e. the input signal’s waveform must be quite stable/low noise.

Step 8: In order to revert to manual control of the physical HP34401A DMM, press the blue *Shift* key on the physical HP34401A DMM’s front panel (i.e. the *Local* label).

Do **NOT** save **any** changes to this Example VI when you quit LabVIEW !!!

Exercise # 7 Build your own GPIB VI using LabVIEW:

In order to communicate with the HP 34401A DMM, you can instead develop your own *LabVIEW*-based GPIB VI. *LabVIEW*-based IEEE-488 GPIB functions (such as GPIB read, write, initialize, etc.) are accessible from the *block diagram* window's *Functions Palette* >> *Instrument I/O* >> *GPIB*.

Do **NOT** use **any** of the *LabVIEW*-based **IEEE-488.2** standard **GPIB 488.2** functions at this time!!!

The typical procedure for the development of a *LabVIEW*-based GPIB VI would be to use software to read a value from the physical device by issuing a command (such as *Read the AC Voltage*) from the controller (i.e. the PC). Since this command is coming *from* the controller (the PC) and *going to* the physical instrument, we call it a *write* operation. The next step is to read the value sent back by the instrument to the controller (the PC). In this case the physical instrument is writing, but the controller (the PC) is reading. Therefore it is called a *read* operation.

In order to implement the first step, one must input into the GPIB-Write function the following 3 items:

- (1) GPIB address = 22
- (2) GPIB mode = 0
- (3) A character command string (e.g. such as *Meas:Volt:DC?* to measure DC voltages).

Once the controller (the PC) sends (i.e. *writes*) this information to the physical DMM, the DMM then acknowledges this and sends a data string of the requested information back to the controller.

We must also then explicitly instruct the controller to *read* this returned data string. At least 3 inputs are required:

- (1) the number of 8-bit bytes to read (= 20 bytes, but not more than 20 bytes)
- (2) GPIB mode = 0
- (3) GPIB address = 22

Since errors may occur in write and read operations, the error output from a GPIB-Write should be connected to the error input of a GPIB Read. You may also additionally want to read the error string and status output, in order to obtain more detailed info for each GPIB read and write operation.

A simple GPIB write & read VI is shown in Figures 6 and 7 below. A Chart display is not required, but was added simply in order to be able to easily and visually check the measured data.

Note that to convert the measured data *ASCII string* output from the HP 34401A DMM via GPIB to the PC into *numeric* values (e.g. in order to plot the data on a chart), you will need the *String* >> *String/Number Conversion* >> *Fract/Exp String to Number* function

from the *Functions Palette*. Note that you can click on the *magnifying glass* icon at the top of the *Functions Palette* to facilitate helping you to find/locate the GPIB write and read VI's, Ascii String => Number VI, etc. Note also that you can right-mouse click on each such icon in the *Block Diagram* window, select *view >> label* to see the terminal connections associated with each icon.

This is left as an exercise to practice. Test your VI with various DMM functions, including resistance and frequency. Details of the HP 34401A commands are in *HP 34401A DMM User Manual* starting from page 104, *Remote Interface Reference*. These commands are part of *Standard Commands for Programmable Instruments (SCPI)* commands. A short introduction to SCPI is given in HP DMM user manual page 154.

When done, save your GPIB VI in a file for your next experiments.

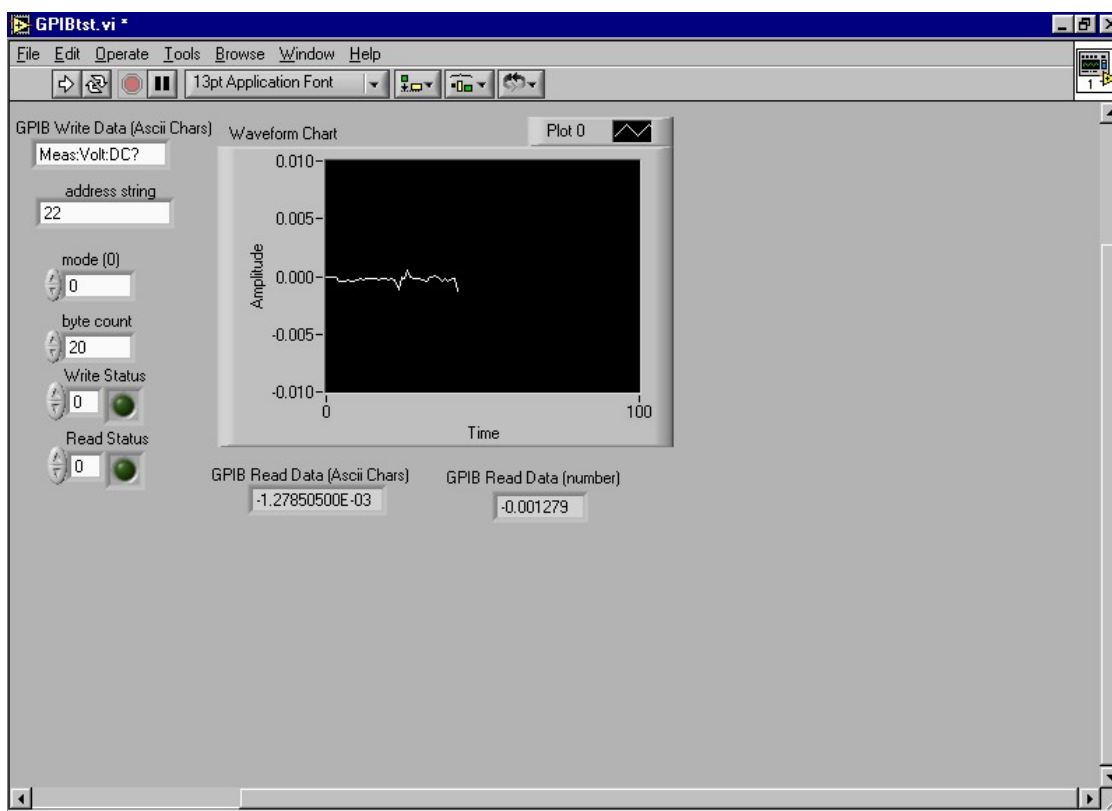


Figure 6. Front Panel window of a simple GPIB VI.

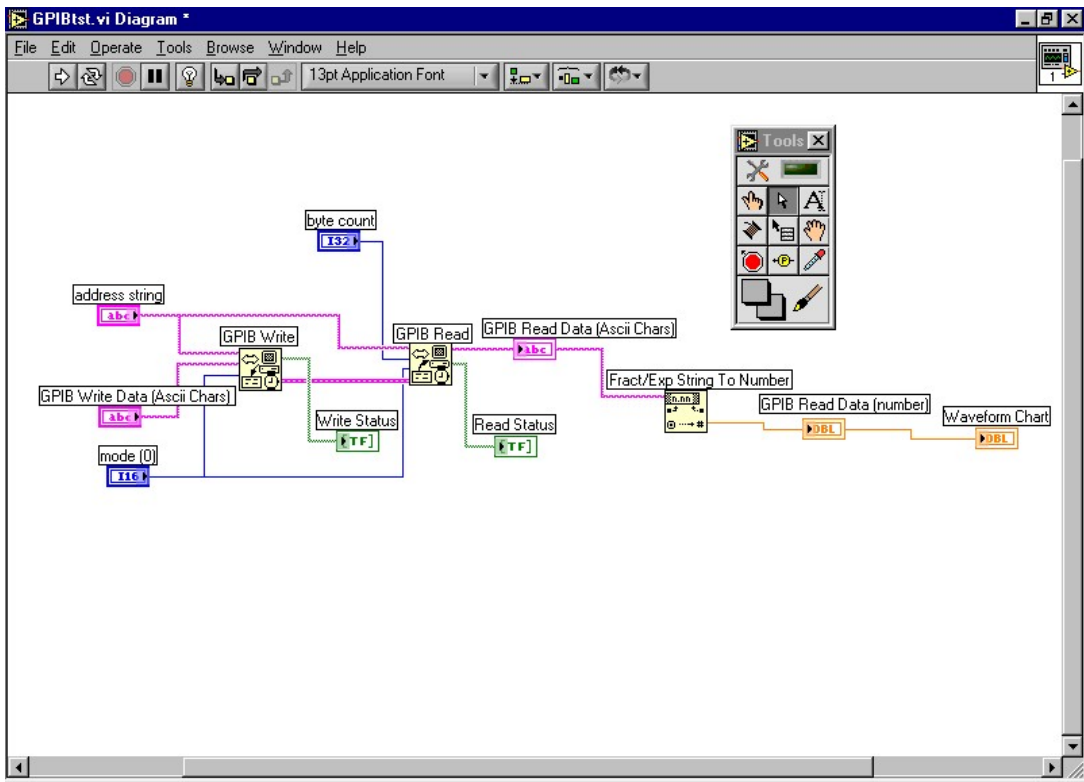


Figure 7a. Block Diagram window of a simple GPIB VI.

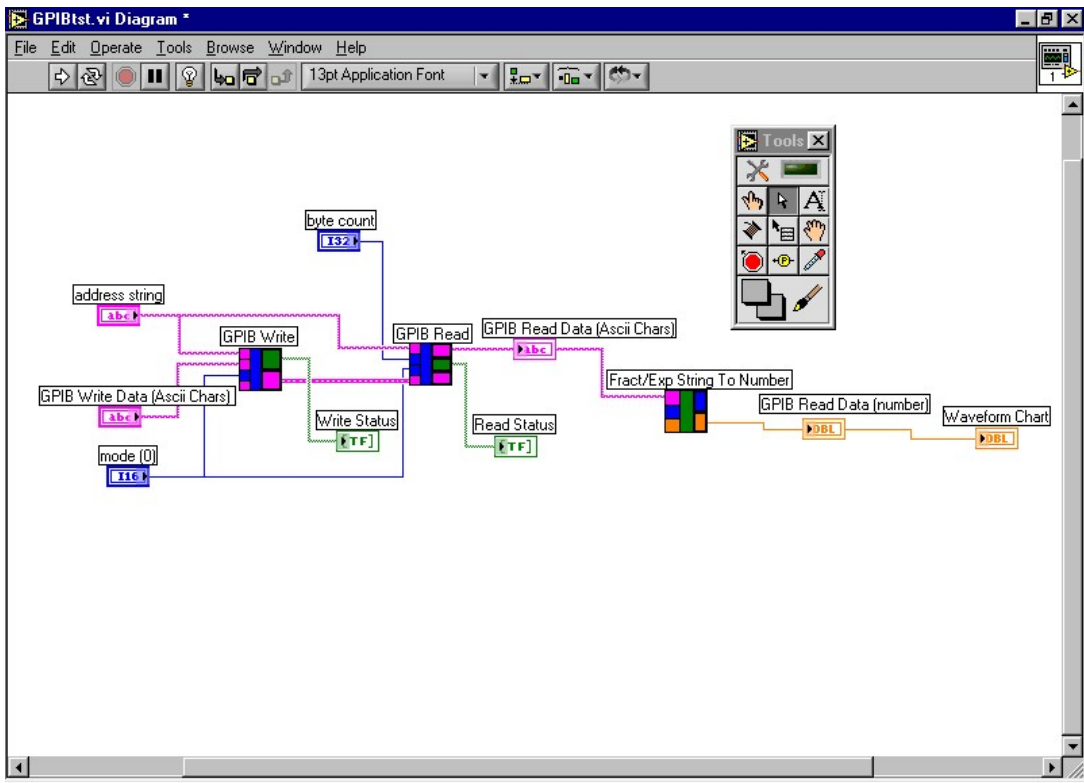
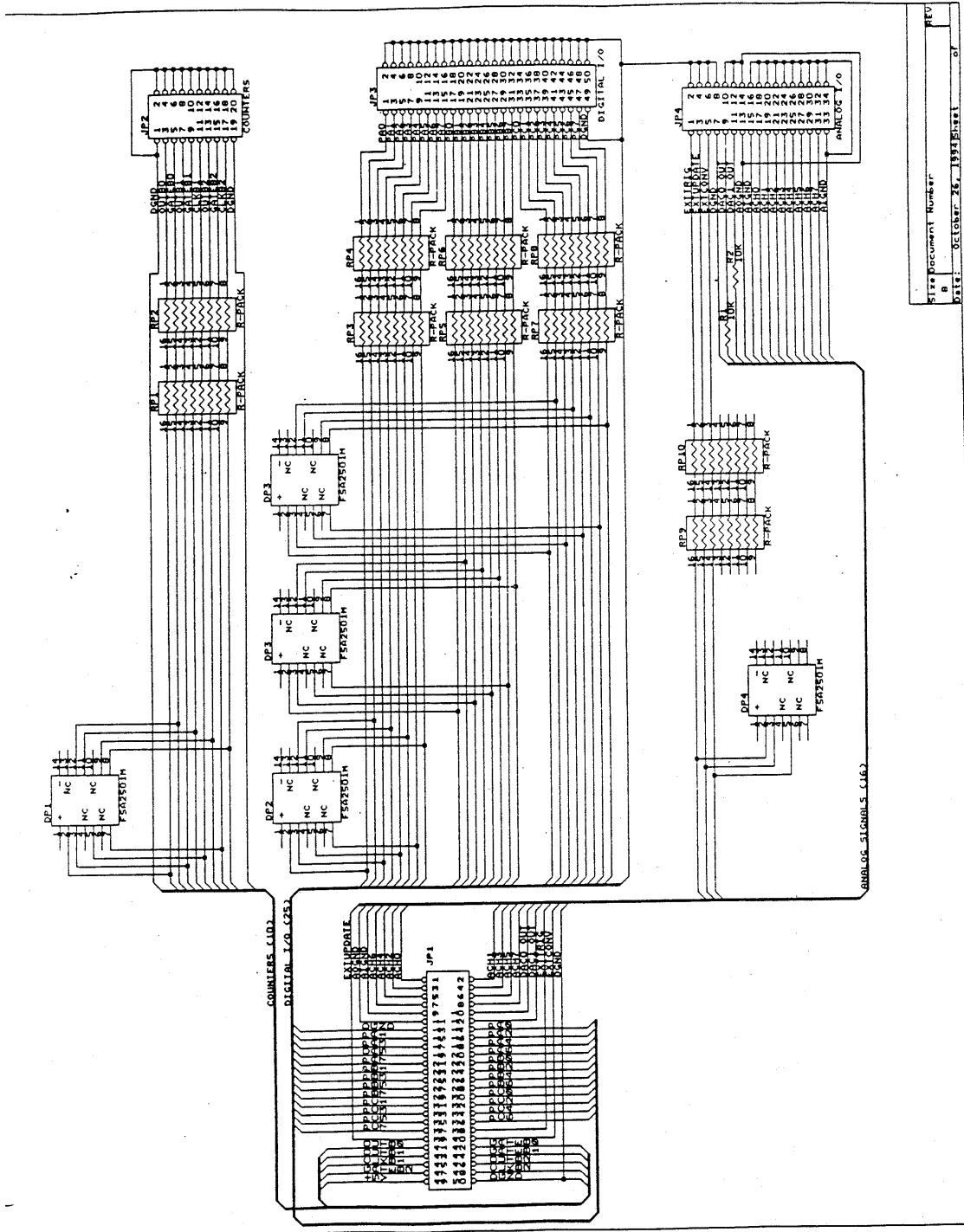


Figure 7b. Block Diagram window of a simple GPIB VI, showing input/output terminals.



The protective circuit installed on the bench for the Lab-PC+ DAQ board.