PHY 406 - Microprocessor Interfacing Techniques LabVIEW Tutorial - Part I Beginning at the Beginning

Introduction

One of the main objectives of this course is to teach you how to gather data using computers and some of the pit-falls and prat-falls which can happen during such an operation. As an adjunct to that it is important that you also do some practical data gathering and try to work through some of the issues on your own. Knowledge that you have wrested from the universe for yourself is often more meaningful than that which you have been handed ready-processed.

The backbone of the practical work in this course is the use of the National Instruments "LabVIEW" package. This package is specifically designed to permit you to quickly implement a computer-controlled data gathering and analysis system which can be extensively customised to suit your needs. It is a very capable package, but is probably unlike anything you have met before. There is therefore going to be a steep learning curve ahead of you before you will be able to be proficient in such work. I wish I could make it otherwise, but I cannot. These notes are designed to make life as easy as possible and to get you started as quickly as possible. In order to make that happen I have made a few restrictions and a few assumptions which are:

- That you are familiar with the concepts of computer systems, such as files, filestore, directories, printing, etc (The notes will help you with the peculiarities of this system)
- That you are reasonably familiar with a "GUI" (Graphical User Interface) and with the use of the keyboard and the mouse.
- That where there are two ways of doing things, I will show you one way - you can look at the full manuals to show you other ways of doing the same thing
- Where there is a system-specific issue I will not discuss other systems
- This is a course in data-gathering, not in pretty LabVIEW programming - what counts is getting the job done, not (at least within reason!) the elegance of the implementation

The Philosophy of LabView

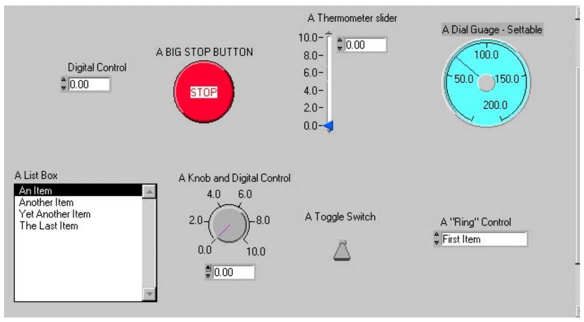
LabVIEW is an entirely graphical language which looks somewhat like an electronic schematic diagram on the one hand and a 1950's vintage style electronic instrument on the other - these are the concepts of the *block diagram* and the *front panel*. LabVIEW is heirarchical in that any *virtual instrument* that you design (any complete functional unit is called a *virtual instrument* and is almost always referred to as a "VI") can be quickly converted into a module which can be a sub-unit of another VI. This is entirely analagous to the concept of a procedure in conventional programming.

LabVIEW is also designed to be extendible. You can add modules through various means. A manufacturer of an interface card or an instrument may provide you with a LabVIEW driver which appears as a VI representing the card and its functionality in the LabVIEW environment. You can also write a LabVIEW module using LabVIEW and present it as a VI to be used in other programs (re-usable code) or you can also write modules which interface with LabVIEW in other languages such as C and C++. These are known as "sub-VIs" and are no different from VIs except that the interface has been defined to the next level. Sub-VIs in C or C++ are very useful if you have a complex numerical procedure to perform on the data which is not covered in a standard LabVIEW routine. Since scientists are rather partial to complex numerical procedures, this can be a very useful property in our context.

Basic Concepts of LabView

As suggested above, there are two "faces" of any LabVIEW VI. They are the *block diagram* and the *front panel*.

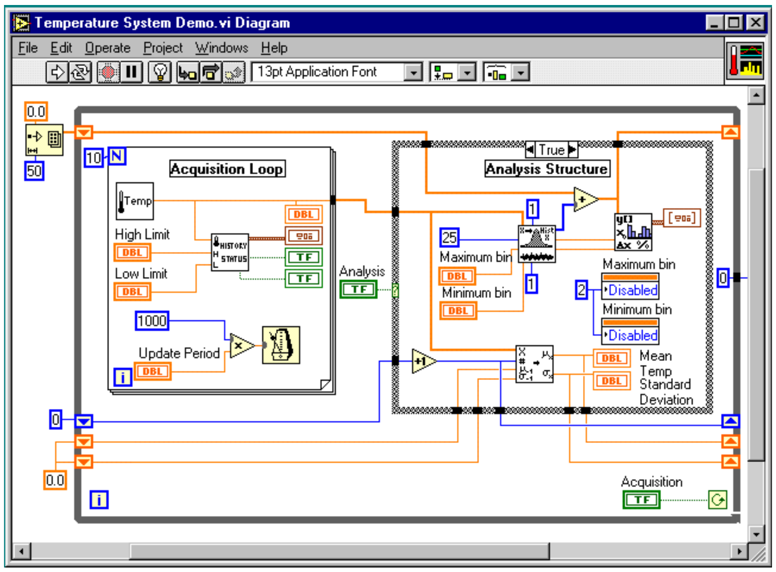
The *front panel* is the face that the user of the system sees. It contains *controls* and *indicators*. LabVIEW has a very rich selection of both (you can even design your own) and this permits a wide range of options to the designer. This is a demonstration of a few of the controls.



A *control* can take many forms. Many of the forms are themselves "pictures" of real controls used on real instruments - rotary knobs for example. Others are strictly digital in concept. All controls have some form of visual feedback to show the user what state they are in. This helps enormously as you do not have to make explicit allowance to show the state of the controls in your design. A second extremely useful property of controls is that you can specify how they are to react if the input given is unsuitable. To give a specific example - if a control should have an input range of 0 to 10 in integer numbers, you can specify what should happen if the value 3.5 is given or -1 or "zero" as a character string. Since a great deal of time can be consumed in "bullet-proofing" a user interface against these sorts of problems, this can be a big timesaver.

Indicators take a large number of forms. Again some are "pictures" of real indicators - lights and meters. Some are more designed for the computer screen. The concept of indicator also includes graphs and charts which is a second major timesaver as you do not have to design any of these elements explicitly.

By intelligent design of the *front panel* of a VI it is fairly simple to produce a simple clean design for the user.

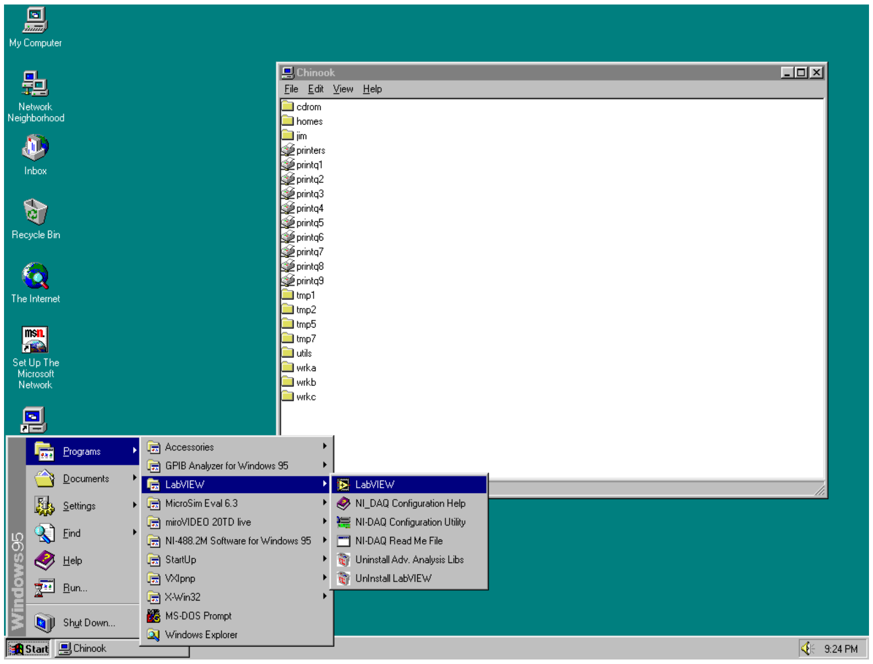


The *block diagram* of the VI is almost the "backside" of the *front panel*. It shows how all the controls and indicators fit together as well as the hidden modules where all the work gets done. It looks somewhat like an electronic schematic diagram and is at least conceptually wired up in the same way. Like a real piece of instrumentation, it is easy for the wiring to look very complex and untidy. One of the major issues in LabVIEW programming is to allocate the timing and ordering of operations. In a conventional programming language this is handled by the order of the statements along with the use of various loop constructs (FOR, WHILE, etc). LabVIEW works in exactly the same way, but the way in which you specify the ordering is more subtle. The concept in LabVIEW is "dataflow" - any item executes when all it's inputs are available. This implies parallelism (or at least pseudo-parallelism). The standard execution is left-to-right because inputs are generally on the left of an item and outpts on the right, but this is a convention, not a requirement. Looping and ordering is handled by structures which look like books with a number of pages or the frames of an old-fashioned cine film.

A Tour of the Interface

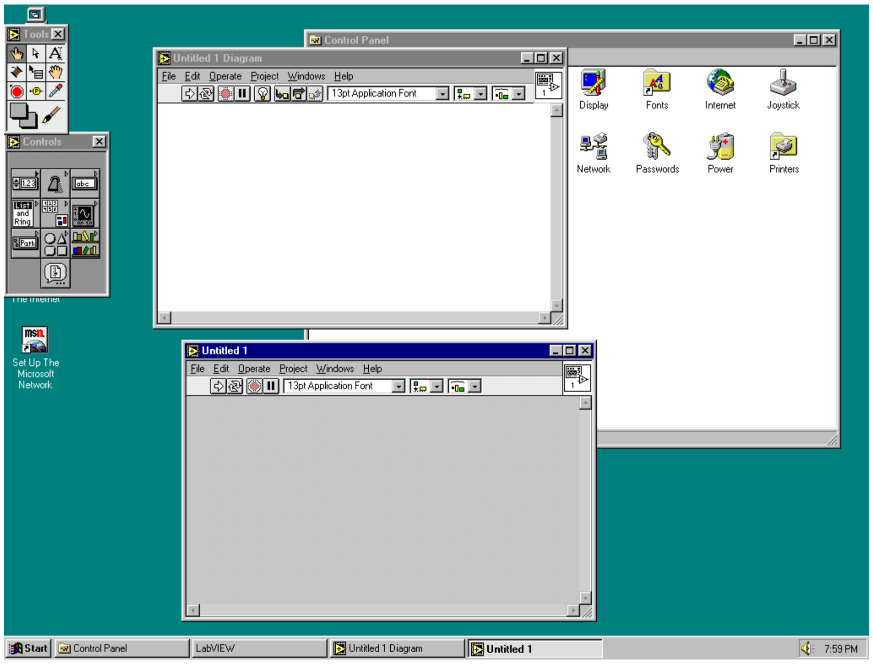
What follows now is a very brief tour of the LabVIEW interface. We are not going to go into all the places possible, but just look at the most common parts. You should read through this section whilst sitting in front of a computer so that you can verify that these notes are correct and you understand them.

First we invoke LabVIEW by the path **Start>>Programs>>Labview>>Labview**. (If there is a LabVIEW icon on your screen you can also start LabVIEW by double-clicking on it).



When we use the notation **Start>>Programs>>Labview>>Labview** we are asking you to follow a series of menus starting from the first item which should be visible in the current window. You can see the entire tree for this operation on the screen below.

Your screen will look something like this (you might have to move a couple of windows around to see everything):



Let’s concentrate first on the window with the grey background - the *front panel.* The window is blank (no panel designed yet) with seven icons along the top of the window and a large one in the top right. The seven icons are used to design a front panel and the large one represents the entire front panel when we are making or modifying a sub-VI.

The seven icons divide into two groups. The first four control the running of the VI:

Icon	Meaning	Explanation
⇨	RUN	Run the VI once. VIs, like conventional programs, do not repeatedly run unless you tell them to. The RUN button changes appearance when the VI is actually running.
↻	RUN REPEATEDLY	Run the VI over and over. Unless you are debugging a VI, this is not a recommended way of repeating any but the simplest of Vis. There are much better ways of doing this using LabVIEW constructs
●	STOP	STOP (unceremoniously) the current VI
⏸	PAUSE/ CONTINUE	Press once for pause, again to continue

The last three icons are used in building front panels

Icon	Meaning	Explanation
13pt Application Font ▾	FONTS	Controls the fonts (size and style of type) used for the front panel
▾	ALIGNMENT	Controls the alignment of groups of controls and indicators - useful for getting things in straight lines and columns
▾	DISTRIBUTION	Controls gaps between things - useful for getting things uniformly spaced.

Above the line of icons (pictures) there are five text menu items:

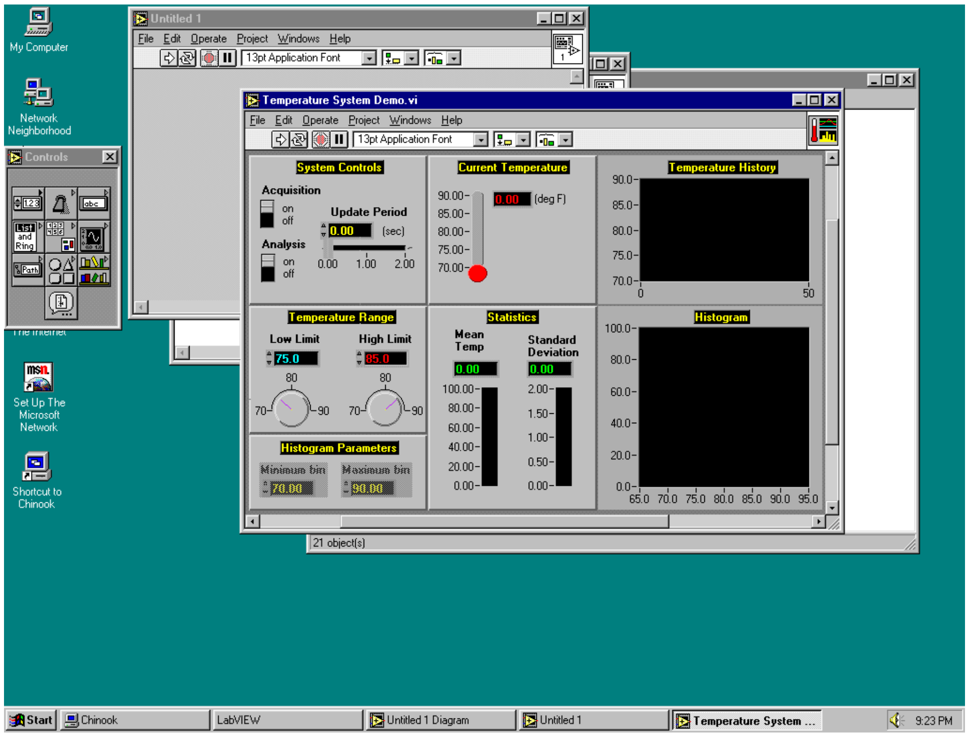
FILE	Is the menu which permits you to do file operations. In this context VIs live in files and therefore this is where you load and save VIs
EDIT	Is the edit menu and contains the commands for editting elements
OPERATE	Operations involving the running of the VI
PROJECT	Operations involving the make-up of the VI - including subVIs
WINDOWS	Select which windows are visible
HELP	One of the most useful areas is the help area. There are several ways of getting help on an item or concept and we will deal with these a lot.

FOLLOW ME

We will now do something very simple - load and run a VI. The VI is a standard example from National Instruments and is a “fake” of a temperature measuring system. The point of this operation is to show you a few of the things that make up LabVIEW and to familiarise you with simple LabVIEW operations.

Select **File>>Open** - double click on “Examples”, then on “Apps” then on “tempsys”. Tempsys is a library of VIs - within the library are a number of items - find the one called “Temperature System Demo.vi” and double-click on it. A box will now appear showing you the loading process and some diagnostics which don’t concern us here.

You should now have a screen which looks like:



On the upper left of the window are a set of controls: Two slider switches which have two states, “on” and “off” - these are boolean controls. There is another control which has a digital and analog part. The digital part is a panel with a number in it and two arrows for raising and lowering the value. Digital inputs like this can usually be changed by either clicking on the arrows are by over-writing the number in the display. The analog part is a slider which can be dragged with the mouse to change the value.

Below that panel in the middle left is a second pair of controls which have a similar digital section, but have rotary controls rather than sliders. Personally I don’t like rotary controls in LabVIEW because I find it hard to rotate the knob using the mouse, I much prefer the sliders.

In the middle of the screen are two panels with indicators on them. Both indicators have a digital and an analog section - the digital section looks pretty much like a control without the arrows to change the value. The analog section of the indicator is a “thermometer” display.

The right-hand side shows some of the most powerful LabVIEW indicators - charts and graphs. These are graphical records of the output and can be displayed in a number of forms which we will discuss later.

There are a few things on the front panel. On the bottom left are two controls which are “greyed out” - this means that they cannot be accessed (changed) at the moment. There are also a couple of other indicators which don’t show up yet.

Now we will run the VI. First, I must explain what it is supposed to do. The VI simulates reading a temperature at a rate determined by the “update period”. It displays the current temperature and also computes the mean and standard deviation of the last 10 readings. It displays a “strip chart” of the history of the temperature readings with the current high and low limits shown and a histogram of the readings. Finally it activates and under- and over-temperature warning.

Before we run the VI I want to show you one important thing- how to print things. There are two procedures depending upon what exactly you want to print:

Print the Contents of the VI

1) Select **File>>Print**
2) You will be presented with a selection menu which asks what you wan to be printed

Print an Image of the Entire Screen

1) If required, maximise the VI by pressing the maximise button (second from top right)
2) Press the “PrintScreen” button
3) Reset the size of the VI by pressing the restore button (second from top right)
4) Open the paint program **Start>>Programs>>Accessories>>Paint**
5) Import the clipboard **Edit>>Paste** - at this point you can edit the picture to take out bits that you don’t want - but you’ll need to learn a bit about the paint program to do that.
6) Print the picture **File>>Print**.

I am assuming that you now have a paper copy of the VI in your hand - sorry it’s not in colour, but funding is still tight!

Press the RUN icon at the top left Notice that the icon changes to the running form and the front panel starts to update. This VI does not end because it contains an infinite loop as we shall see later.

Using the operating tools we can slide, twist, poke and over-write the controls. To slide and twist knobs and switches put the operating tool on the control and use the left mouse button. To poke a digital control, put the tool on the appropriate arrow and click (the amount the control changes can be varied as we shall see later). To over-write a value, double-click on the old value and then type the new one in. Please try changing a few controls and observe the effect. Notice that the analog controls (sliders and knobs) naturally show you the limits of the input, whereas the digital controls don’t

When you have finished playing, stop the VI by sliding the aquisition switch to “off”. The VI may not stop immediately because it has to finish what it is doing. This is a much better way of stopping a VI than pressing the “STOP” on the toolbar because it permits a clean end to the VI operation, the STOP button can leave unfinished business around.

Now we are going to look at the “backside” of the *fromt panel* - the *block diagram.* Select **Windows>>Show Diagram**. Notice that the block diagram doesn’t replace the front panel - both are visible at the same time, useful in development operations.

The first thing that should strike you about the block diagram is that it is a bit more complex than the front panel! However most equipment looks nicer on the outside than the inside!

The next thing is that there are three shaded boxes - a big outer one and two inside it. These are the looping boxes. The outside one is a “while” loop which continues until a boolean condition becomes false. The left-hand inner one is a “for” loop which executes once for every iteration of the while loop (these loops are therefore “nested” as the diagram suggests) and the right-hand one is an “if-else” condition (not a loop). The “true” part is showing at the moment. Click on the arrows by the word “true” at the top middle of this box to swap to the “false” side which is a lot less crowed than the “true” face. Click on the arrow again to swap back.

The “wires” connecting the parts and some of the boxes are in different colours. LabVIEW uses different coloured wire to tag different types of variable. Green (dotted) is boolean, blue is integer and brown/orange is double precision. Notice that the brown/orange come in two thicknesses - the thin lines represent scalars, the thick lines arrays.

Since LabVIEW operates left-to-right it is clear that within each iteration of the outer while loop, the left-hand for loop runs completely through once and then the right-hand if-else clause is executed.

By the way if you are having trouble swapping between the windows, arrange them so that they are offset slightly along a diagonal line with enough offset to show a little bit of the other window area even when it is not selected. Then you can click on this small area to swap between windows. If the overall screen is large enough, you can arrange them so that they don’t overlap and then you can see both at once, but this isn’t always practical.

You will see a number of boxes with single coloured lines around then with a number inside. These are constants. The boxes with two lines, outer thick - inner thin, are controls - the “backside” of the things on the front panel. Double-click on the one marked “update period”. The display swaps to the front panel and you can see that the control marked “update period” is highlighted there. These are the front and back side of this control. Move back to the block diagram.

The boxes with double thin lines are the indicators. Try a similar selection process on the one marked “DBL” in the top right of the for loop. It corresponds to the thermometer and digital value in the current temperature area.

The squareish boxes mostly in black-and-white are very significant. They are also VIs. Try double-clicking on the one in the middle of the for loop (left-hand inner box). It will pop up another front panel. The “controls” on this front panel are in fact controlled by the input wires to the VI and the indicators feed their output through the attached wires elsewhere. Notice that there are three controls and three indicators. Close this window and see that there are three inputs (left-hand) and three outputs (right-hand) to this VI. This is the “procedure” or “subroutine” of LabVIEW. Each of these sub-VIs can be independently made and then strung together to form a more complex whole.

Now close both the front panel and block diagram - you shouldn’t have changed anything, but if you did please ask the system to ignore the changes. (Just say no!).

Now we are going to build a very simple VI. In order to make sure that we all start from the same point, please close LabVIEW now by closing the blank windows. There will be a final little box which says “There are no VIs open” and you should choose “Quit” from this.

Summary

So far we have:

- ‣ Introduced you to the basic philosophy of LabVIEW, a graphical processing language. It uses the concepts of a *front panel* and a *block diagram* to show you the user and designer views of the VI (*virtual instrument*)
- ‣ Given you a tour of a moderately complicated VI which contained *controls*, *indicators* and sub-VIs
- ‣ Shown you how to start LabVIEW and load a VI