PHY 406 - Microprocessor Interfacing Techniques
LabVIEW Tutorial - Part III
Going Forward and Round and Round

While Loops and Switches

We are now going to progress with our LabVIEW by learning how to loop a VI and stop it in a civilised manner. The loop we will use is a "WHILE" loop which runs round and round as long as a control variable is "true".

In order not to tax things too much, we will re-use the VI we made last time and put it in a loop so that we can keep changing the inputs and see the outputs changing. We will also adopt the practice of having a nice switch to stop the VI when we are ready.

If you are continuing from the previous example, then you already have the right screens in front of you, if not restart LabVIEW and bring that practice VI up again.

> Some thoughts on modifying things so that you add to them and getting them saved properly. I have found that if you want to take a VI and then upgrade it to be saved under a new name, the best thing to do is to save it under the new name now. The reason for this is that I have a habit of saving things from time to time as I reach "plateaus" of functionality or get more worried about how much I will lose if the system crashes or the power dies. If I have the VI saved it under the new name now, I only have to "save" from there on - I don't need to remember to "save as" the first time - which I always forget and then overwrite the original VI. So if you agree with my philosophy, now is the time to save this VI under a new name!

We now need to put all this in a loop on the block diagram. Using the select tool, find the "while" loop under **functions>>structures**. Start by moving to the upper left of the digram and the holding the left button down drag the loop out until it encloses everything. (This is the only time you can do this - you can't arbitrarily "enclose" compenents at any other time - they should be created either inside or outside the loop structure).

Notice the two squares in the lower corners. The left-hand *I* is a variable which counts (from 0) the number of times the loop has executed. The right-hand is the boolean control variable. If the input is "true" then the loop executes, if it is "false" then it stops at the end of the current iteration - ie cleanly.

Now let's learn another LabVIEW shortcut. Right-click on the loop and select the "create indicator" option - a digital indicator pops up on the front panel. This is a really quick way to create new indicators when you need them. Yous can move the indicator around and label it by selecting the indicator and then selecting **show>>label** from the pop-up menu (right-hand mouse

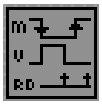
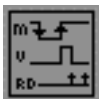
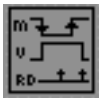
button).

The same thing can be done with the loop control, use the pop-up menu to create a control. Now using the operating tool you can set the control to "on" and run the VI. The loop indicator should show that it is running round and round very fast. You can now dynamically alter the input variables and the output changes. When you push "off" - the VI stops (run arrow changes and stop button dims). Now try to restart the VI by pushing "run". Nothing happens because the push-button control still says "off". To run again you have to change the pushbutton to "on" and start the VI - inconvenient. What we need is a smarter pushbutton.

Now you might think that boolean switches are fairly simple, but that is not true. There are a number of essential questions to ask:

- Do you want the switch to latch a new value or just react when you "push" it?
- Do you want this to happen when you push the mouse button or when you release it?
- Do you want to be sure that LabVIEW will read the switch in it's new state?

This leads to six different types of switch:

Icon	**Latch?**	**Action on Press or Release?**	**How many times will LabVIEW read it?**
	YES	PRESS	Many times
	YES	RELEASE	Many times
	NO	PRESS/RELEASE	Maybe never (released too quick) Maybe many times (released "slowly")
	NO	PRESS	Once - revert when LabVIEW reads
	NO	RELEASE	Once - revert when LabVIEW reads
	NO	PRESS	At least once, maybe more - revert when LabVIEW reads after button released.

The one most suitable for our purpose is the non-latching, revert after LabVIEW reads -

the fourth in the list. Change the pushbutton to that type from the pop-up menu item **mechanical action**. You should also change the default state to "on" (manually set the pushbutton "on" and then set that as the default state using the pop-up menu).

Now when you run the VI it should only require you to press the run button to start it.

From now on, I will assume that you are going to save your VI at appropriate times. It is a good idea to save things to a new name every time you reach a satisfactory state of operation. This is more than paranoia - it is a sensible policy to make sure you don't lose an afternoon's work because of "finger trouble".

It's a bit static isn't it?

The world of video games makes us like movement and this VI is a bit static, only a counter (which might by now have reached a very high value!) Counting upwards. Time to add a bit of action. First start changing one of the input numbers about 10 times a second, then.... Well maybe there's an easier way.

LabVIEW is very rich in functions. What we are going to do is to change one of the input variables to the output form a random number generator. First delete the control by selecting it on the front panel with the select tool, then pressing **Delete** (you can't delete a control form the block diagram that way. Go back to the block diagram and remove the broken (bad) wire by **Ctrl-B**. Now find the random number generator on **function>>numeric** (it looks like a couple of fuzzy dice ) and put it on the block diagram in the while loop and wire it up with the wiring tool to the multiply operator. You might wonder how the generator knows when to "produce" a new value. In effect Labview asks every component whether it has a value available. The digital control says "yes" and produces the value of the control, the random number generator says "yes" and produces a new random number.

Now when you run the VI, the thermometer should "shiver" very satisfactorily assuming that you have the right scaling.

It would be nice to have a record of the output over time - a graph or a chart. LabVIEW distinguishes between a graph in which all the data are available together (as in an array) and a chart when only one value is available at a time (as in a strip-chart recorder). We will add a chart to the VI. At this stage you might want to expand the VI by enlarging the front panel window (drag on right-hand edge with left mouse button) until there is a space about as large as the rest of the window spare on the right. Now select **controls>>graph>>waveform chart** and place the chart on the right of the front panel. Add the label before pressing **return**.

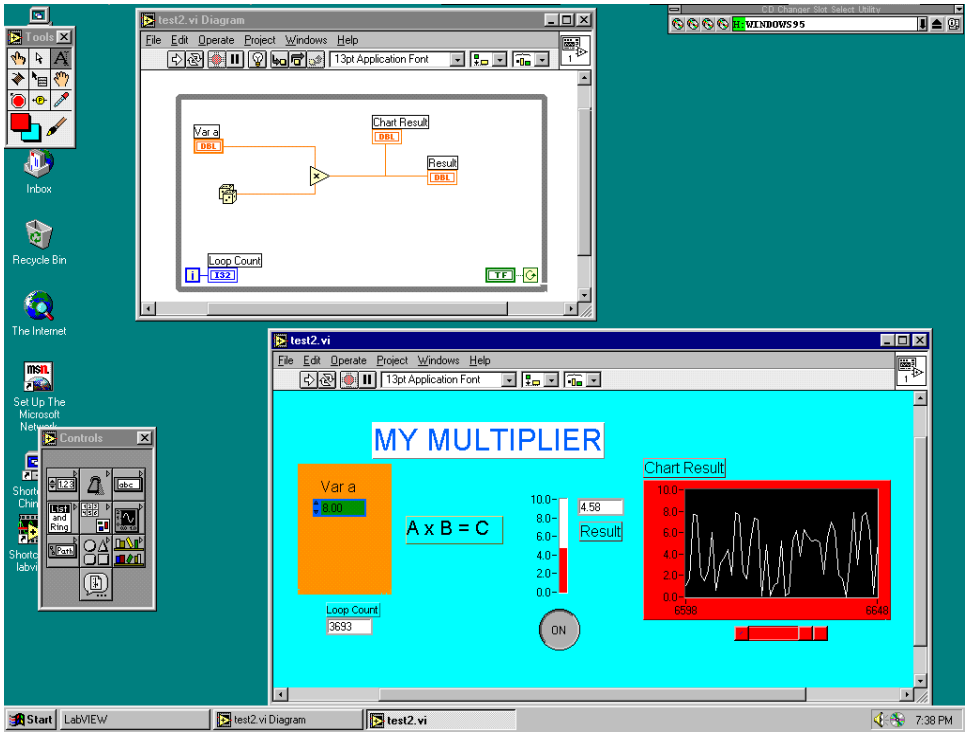
The chart can be wired up in parallel with the thermometer on the block diagram by starting with the wiring tool at the wire between the multiplier and the output indicator and wiring

to the chart from there - one output can feed any number of inputs of the same type. Make sure that the indicator is inside the confines of the while loop.

Now when you run the VI, you see a chart of the last 50 values and the chart presents a running window of the last 50 values. From the pop-up window you have control over a large number of parameters. A new set is the set that controls the rolling window. You can control the number of values visible by changing the horizontal scale and control how these are spread out by changing the size of the box by selecting it and dragging the size out horizontally. You can also review the history of the values using a storage capability and the horizontal scale with a scrollbar. By default 50 values are shown and 1024 values are stored. By activating the scrollbar from the pop-up menu **show>>scrollbar** you can use a simple scrollbar to run through the stored values (You probably need to colour the scrollbar white to see the parts properly, for some reason the default values don't show up too well). You can also change the length of the stored dataset from **Chart History Length**

Those of you who have shrp eyes will notice that the VI is running distinctly slower with this chart in place. There is no such thing as a free lunch and more displays equals more processing required equals more time per loop equals fewer loops per second. Something to be borne in mind for the future.

My design now looks like this:



Too Fast!

Now that you can see how fast the VI is going, you might decide that it is too fast. A new value every 250mS would be just fine. There is a way to time a loop and that is to insert a timing element. From **functions>>Time & Dialog>>Wait Until Next ms Multiple** get the little picture of a metronome, place it in the loop and use the pop-up to generate a constant for the input and set the value to 250 (the value is in mS). Move the constant over and wire it to the input (left-hand) side of the timer. Your VI should now loop four times a second.

Summary

- While loops have an iteration count and a boolean control variable
- Pushbuttons can be of six different types depending upon the action required.
- The random number generator creates a new random number between 0 and 1 every time it is interrogated.
- Charts are used to display the sequence of outputs from a process and are highly programmable.
- LabVIEW may be "slowed down" by timing units

Exercise

Instead of the random number generator, use the **Digital Thermometer.vi.** From the tutorial functions - use the Boolean "mode" input so that you can have the display in either Farenheight or Centigrade - add a third option for Kelvin. (You will find it useful to call up the help (**Ctrl-h**) on the Thermometer to see how to wire it up).