ME 3200 Mechatronics I Laboratory

Lab 1: An Introduction to CVI

This exercise introduces LabWindows/CVI, a powerful computer program for data input and output. The software is used in conjunction with a Data AcQuisition (DAQ) card mounted inside each computer. Connections to the DAQ cards are accomplished via the terminal blocks at each lab station. A project file can be created for a specific task. The project file consists of three other files: a *.c file, a *.h file, and a *.uir file, with the "*" being the project name. The C file is written in the familiar ANSI C programming language and tells the computer how to interact with the data acquisition board. The H file is known as a header file. It allows the user to call on previous written functions organized into libraries. The UIR file is the graphical User InteRface built by the engineer to suit his/her experimental needs. In this exercise, you will design your own graphical user interface to take a voltage reading with the DAQ, compare it to a setpoint (a number you choose), and output the difference to the DAQ. A simple schematic of this is shown in Figure 1.

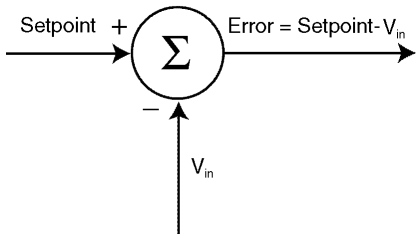

Figure 1. Summing junction for CVI experiment.

Getting Started:

1. Locate the directory C:\CVI\PROGRAMS\MECH_I and make a folder for your lab team. This folder will be used to save the lab data for the rest of the semester, and will be deleted at the end of the semester.
2. Open CVI by double clicking on the icon. Create a new project file by selecting *File/New/Project*, accept all the defaults, and then save it to your folder by selecting *File/Save As.* Find the folder you created for your team as the destination folder, and give the file an appropriate name.
3. Locate the template application file by selecting *File/Open/Source (*.c)*, and then locate C:\CVI\PROGRAMS\LABS\TEMPLATE.C. Save this file to your directory, giving it the same name you gave your project file.

The template file you just saved is a basic application file that can be modified to suit the needs of different experiments. To create your project, lines of code will be inserted or modified.

4. Create the graphical user interface by selecting *File/New/UserInterface (*.uir)*, and then save it to your directory with the same name used for the other files.

5. Locate the header file for your project by selecting *File/Open/Include (*.h)* from the project window. The file should already be there and it should have the same name as the project and application files. Selecting this will bring up a window with the header file in it. Any changes to the header file will be automatic.
6. Add the separate files to the project by selecting *File/Add file to project* on each of the component files. There should now be three files listed in your project window (i.e., *.c, *.h, *.uir, where "*" is the name you chose for your project.)
7. Bring the C file to the front. Look a few lines down to the list of header files that are included. Find the one that says, "include name.h" and change "name" to the name of your project.
8. Look down a few more lines to find a line that says:

   panel_handle=LoadPanel(0,"name.uir",PANEL)

   and change name to the name of your project.

Now that you have all of the necessary files in the right place and properly interconnected, it is time to put together your graphical user interface.

Creating the Graphical User Interface:

1. Bring the UIR window to the front and select *Create/Panel*. Your panel will then appear. It will say "untitled panel" at the top. You can resize it by dragging the corners, just like with other Windows applications.
2. Double click anywhere on the panel to play with settings like color, title, etc. Just do not change the constant name on this panel. If you look on the right, you will see a field that says "constant name: PANEL". Do not go near it. Your panel is a constant in the C file as verified here.
3. Create a knob, dial, slider, or gauge to control your setpoint. The setpoint is a voltage that you will choose. Think of it as a desired voltage. To create the knob, dial, slider, or gauge, select *Create/Numeric*, and then select the type of numeric that you want. Double click on your new numeric. An information window should appear. Name the numeric appropriately in the "constant name" field. The name you give it will be used in the C file, so choose wisely. Remember also that CVI is case sensitive. Give the numeric a label, perhaps the same name as its constant name. This name will appear on the display above the component.
4. Create numeric items for the input voltage and the error. You will be reading the input voltage from the DAQ using the input voltage numeric. This value will be subtracted from your setpoint to give the error, which will be read on the error numeric. Follow the instructions from the previous step to create these numeric items. Give your numeric items constant names and labels as before.
5. Your panel should now have three numeric items on it. Notice how you can move them wherever you choose by clicking, dragging, and dropping. Notice also how double clicking on an item brings its information window to the front.

6. Create a button to take a sample. Every time the user clicks on this button, CVI will run through the program. To create your start button, select *Create/ToggleButton*. There will be a variety of buttons from which you can choose. Give your button the proper constant name, label, etc. In addition, in the "Callback Function" field, type "START_FUNC". This tells the C program to run the START_FUNC function, which will be written later.
7. Create a quit button by following the instructions in the step 6. When this button is pressed, your program will shut down and your panel will close. In the "Callback Function" field, type "QUIT_FUNC". This tells the C program to run the QUIT_FUNC function.
8. Save your UIR file. When you save the UIR file, your header file is automatically updated and brought to the front. Remember that CVI automatically modifies header files, so there is no need for you to ever modify them yourself. With this in mind, when your header file comes up, do not change anything.

Modifying the Application Code:

1. Check near the bottom of your header file. Look for two lines that look similar to:

   ```
   int QUIT_FUNC(int panel, int control, int event, void *callbackData, int eventData1, int eventData2);
   int START_FUNC(int panel, int control, int event, void *callbackData, int eventData1, int eventData2);
   ```

   These lines reference the start function and quit function in the C file.

2. Bring the C file to the front. Look for the quit function. It starts in the same way as the line in the header file, and should end with *return(0)* and a comment similar to */*End of QUIT_FUNC*/*. Highlight the quit function and copy it to the clipboard (ctrl+c). Paste the contents of the clipboard underneath the quit function (ctrl+v). Replace QUIT with START and omit the line that reads *QuitUserInterface(0);*. Your new start function should look like this:

   ```
   int START_FUNC(int panel, int control, int event, void *callbackData, int eventData1, int eventData2)
           {
                   if(event==EVENT_COMMIT)
                   {

                   }
                   return(0);
           }  /*End of START_FUNC*/
   ```

   The space between the braces in the middle is where you will modify the code to interface the panel on the graphical user interface with the DAQ.

3. Leave your cursor in the space between the two middle braces. Use your mouse to select *Library/UserInterface/ControlsGraphicsStripcharts/General/GetControlValue*. A new window will appear. Locate the "panel handle" field and enter the formal

name of your panel, namely "PANEL", remembering that CVI is case sensitive. Locate the "Control ID" field and enter the path to the setpoint control (i.e., PANEL_SETPOINT). Locate the "value" field and enter the variable name (i.e., &setpoint). (The ampersand (&) is a pointer in C. Sometimes C needs to be pointed to its variables introductions. A quick test to see if a pointer is needed is to right click in the "value" field, and if a message that says "passed by reference" appears, C needs a pointer.) The field at the bottom of the window shows how these decisions modify the function call. Select *Code/Insert Function Call*. Close the window.

4. Check your C code. The previous step wrote and inserted the line you see there. This line of code instructs CVI to check the entered value for the setpoint and assign that value to the appropriate variable.
5. Select *Library/DataAcquisition/AnalogInput/SinglePoint/MeasureVoltage*. At the bottom of the window, locate a box with the function call *AI_VRead* (analog input voltage reading). Set the board slider to "1", pick an input channel (usually channel 0, since it is in the top left corner of the DAQ board), and set the gain to "1" (meaning that you multiply the value by one). Locate the "voltage" field and enter the variable name (i.e., &v_in). The field at the bottom of the window shows how these decisions modify the function call.Select *Code/Insert Function Call* to update your application file. Close the window. Check your C code to make sure that it was updated.
6. On the next line of the C code, type in *error=setpoint-v_in*; or its equivalent with the variable names you chose. Press Enter to get to the next line.
7. Select *Library/DataAcquisition/AnalogOutput/SinglePoint/GenerateVoltage*. A window should appear. Move the board slider to "1", choose the output channel (typically channel 0), and enter *error* or the variable you are using in the "voltage" field. The field at the bottom of the window shows how these decisions modify the function call. Select *Code/Insert Function* to update the application file. Close the window. Check your C code to make sure that it was updated.
8. To display the v_in and the error on the numeric items you created, select *Library/UserInterface/ControlsGraphicsStripcharts/General/SetControlValue*. Follow the same steps to fill in the fields as you did in step 3. Select *Code/Insert Function* to update the application file. Close the window. Check your C code to make sure that it was updated. Locate the line that says, "*int panel_handle;*". On the next line declare your new variables as doubles (i.e., *double setpoint, v_in, error;*). Save your code.

Testing the Panel:

The code that was just completed reads the voltage level at the input channel you chose, subtracts it from the setpoint you can actively select, and outputs the difference from the output channel you chose. To test this, you need to concoct a variable voltage source. This will be accomplished with the voltage divider, as shown in Figure 2.

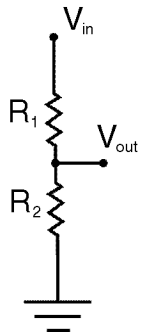

Figure 2. A voltage divider.

The formula that relates V_{in} to V_{out} in a voltage divider is as follows:

$$V_{out} = V_{in} \frac{R_2}{R_1 + R_2} \quad (1)$$

where V_{in} is the input voltage in volts, V_{out} is the output voltage in volts, R_1 is the output resistance in ohms, and R_2 is the input resistance in ohms. The way that you will build a voltage divider is using the potentiometer on your bench. A potentiometer is a resistor that has voltage applied across it. A wiper (an electrical contact attached to a knob) is used to sweep along the resistor and create a voltage divider anywhere along its length.

1. Locate the potentiometer on your bench. It is accessed through three connectors underneath a knob. Take a cable with banana plugs on both ends and connect one side of the potentiometer to the 5V power supply. Take another banana plug cable and connect the other side of the potentiometer to ground. Connect the power supply ground to the appropriate AI_GND (the left one corresponds to the first row of AI channels, and the right one corresponds to the second row). Connect the middle of the potentiometer (the wiper) to the AI channel you chose in your code. Switch on the power supply.
2. Bring the PRJ window to the front. Select *Run/Run Project*. A window will appear that looks just like your UIR that you created. This is where the actually interfacing occurs. Click your start button.
3. The numeric items should change instantly, reflecting the v_in and the error. Verify that these are reading correctly using your bench multimeter. Remember to use the proper ground on the DAQ when making this reading. When reading v_in use the proper AI_GND, and when reading the error use the AO_GND.
4. Change the setpoint on the display. Turn the knob on the potentiometer. Click the start button again and notice the changes. Verify the values with the multimeter. Show your TA that your panel works.
5. When you are satisfied with the experiment, click on the quit button. This will shut down the display. Print a copy of your panel from the UIR window, and save a copy of the C file to disk for your report. Exit CVI, turn off the power supply and multimeter. Clean up all of the wires.
6. ANSWER ALL OF THE QUESTIONS ON THE FOLLOWING PAGE AND CHECK OFF WITH YOUR TA.

Questions:

1. What is the function call for the DAQ to take a voltage reading? What are the arguments used, and what do these arguments mean?

2. What is the function call for the DAQ to output voltage? What are the arguments used, and what do these arguments mean?

3. Explain how pushing a button on the UIR relates to the program execution. Why did you enter in the "Function Call" field on the start and stop buttons, but not on the numeric items?

4. Write a short subroutine to acquire data from AI_1, multiply the data by 1.5, and then write the data to AO_1. Fill in the braces in START_FUNC.

5. Suggest two applications for the LabWindows software.