Virtual Lab 5: Capture that Waveform!

By: Steve Badelt and Daniel D. Stancil
Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA

Purpose:

- Incorporate User Functions to make code more efficient in space and time
- Learn to capture waveforms on the Oscilloscope
- Use Dialog features to remind the user of important possibilities

Equipment:

- Agilent 34401A Multimeter
- Agilent 54601A Oscilloscope with Agilent 54657A Measurement/Storage Module
- Agilent 8116A Pulse/Function Generator or Agilent 33120A Function/Arb Generator
- Agilent 6236B Triple-output Power Supply
- Connectix Video Camera
- Intel 100MHz Pentium computer, 32MB Ram with GPIB Card
- HP VEE for Windows software, Version 3.2
- Timbuktu (remote access) software
- Text: Robert Helsel, “Graphical Programming: A Tutorial for HP VEE”, Prentice Hall, 1995
- Download Agilent VEE executable files

Introduction:

In the next three labs, we will be designing code to examine the frequency spectra (magnitude and phase) of different waveforms. All of the final program coding will be up to you. When the labs are completed, you will have a flexible spectrum analyzer at your disposal.

User Functions:

`User Functions` allow you to perform a series of tasks, which are the same or very similar, several times within the same piece of code. Put simply, a User Function is a special way of calling a **User Object**. Indeed, common tasks can easily be repeated by cutting and pasting a User Object. However, making modifications would then entail finding and changing each individual User Object. The User Function leaves a single copy which will effect all calls to it. It therefore saves space on the hard drive and in memory. In addition, libraries of User Functions can be created, making their use more convenient than cutting User Objects out of old programs you have written.

To create a User Function, first create a User Object that will perform desired functions on the inputs and outputs you specify for it. Then choose **`Make User Function`** from its object menu. You will be prompted to enter a name for the function, and a Call Function object will replace the old User Object. Anytime a Call Function object is given the name of your function, it will add inputs and outputs to match the names of those in your original User Object.

In order to modify a user function, first create a Call Function object which uses it. Then choose **`Edit User Function`** from the object menu. Once modifications are made, you may need to **`Configure Pins`** if you have added any new inputs our outputs. User Functions are saved to disk with the program in which they were written. If you wish to call a User Function from another program, use **`Device`**→**`Function`**→**`Import Library`**. This object must be placed on the screen and accessed before making any calls to the library’s functions. Any file which contains a User Function may be imported.

Dialog Messages:

Dialogs are helpful for informing the user of a condition or prompting the user for input, including strings, numbers, and file paths (as you have already discovered). When designing a program for users other than yourself, they are particularly useful for making sure bad things don't happen. The objects which call up dialogs are in the **Data→Dialog Box** menu.

Junctions:

The **Junction** object outputs whatever data input is the most recent from among its multiple inputs. It is particularly useful if multiple, mutually exclusive datastreams converge in one place. Another function that controls dataflow is the **gate** object. A Gate outputs whatever is input to it, but only when its sequence in pin has been activated. Both functions are available in the Flow menu.

Data Structures:

In the next few labs you will start doing more programming independently, without step-by-step instructions. It may be helpful to know what the data types within Agilent VEE are. Many are specific to Agilent VEE, and don't commonly exist within a language like C. Look at pages 4-2 and 4-3 in the **Graphical Programming** text to see what data types are available for your use. In this lab we will be using the waveform data type.

Procedure:

Part 1: Practice

The functions which retrieve a waveform from a scope channel are specific to each channel. Yet, sometimes it is helpful to simply pick a channel number to read. This program will write a User Function to do this. A Dialog input is used to acquaint you with its usage.

1. Create a User Object on the workspace and maximize it. Add a data input labeled "Channel", and a data output labeled "Wave." The labels are very important for User Functions. The Call Function object will use these labels to tell you which input is which.
2. Create an If/Then/Else object and add three more Else/If transactions. Type "A==#" (# 1-4) in each.
3. Add four Scope component drivers to the right of the If/Then/Else objects. To each add a **VIEW_CH#** input and an **AUTOSCALE** input. Wire an integer constant with value 1 to each input.
4. To each component driver add a single data output, **WF_CH#,** one for each channel of the scope. The **WF_CH#** retrieves the waveform as seen on the scope. The output is a Waveform data type.
5. Wire the input of the If/Then/Else object to the Channel input of the User object. Each of the Else outputs should be hooked up to the sequence in pin of its respective oscilloscope component driver.
6. Create a **Junction** with four inputs (**Add→Data Input**).
7. Connect the output of each component driver to the Junction, and hook the Junction output up to the Wave output pin of the User Object.
8. Choose **Make User Function** from the object menu. Name it whatever you would like. Notice how the User Object is converted to a Call Function object with the name of your function in it. Minimize the object.
9. Select an **Integer Input** from the **Data→Dialog Box** menu. Set the **Prompt/Label** to "Enter Channel Number", the **Default Value** to 1, the **Value**

Constraint to between 1 and 4 ("(1<=value) and (value<=4)"), and the **Error Message** to "You must enter a number between 1 and 4."

10. Add a **Display→Waveform** to the workspace. Wire up all the appropriate inputs and outputs of the existing objects. Confirm that it works as you would expect. Save your program as lab5pt1.vee.

11. To call a function, select **Device→Function→Call**, then choose **Select Function** from the Object Menu. A menu will appear listing the available functions.

Part 2: Assignment

Design an object or function that will accept the following inputs:

- Waveform: The numbers 1-3 representing a waveform function on the function generator. These represent Sine, Triangle, and Square, in order.
- Duty: Percent duty cycle. Changes the width of a square wave, and alters the symmetry of the other waveforms.
- Amplitude, Offset, and Frequency: Should be self explanatory!

Given these inputs, create a function/object to set the function generator to the appropriate settings, and capture the resulting waveform on channel 1 from the oscilloscope. Make sure the entire waveform is shown on the screen and use bandwidth limiting. It will be sufficient to use the scope's Autoscale. Note that the waveform inputs of 0,4 correspond to DC and pulse, respectively. These will not be used in our program.

Once you have finished, hook up appropriate constants to each of your object's inputs. Create an additional object which executes before your object and initializes the instruments. It should turn on channel 1 of the scope, set the probe to 10X, set the waveform to Sine, and enable the output of the function generator. Save your program as lab5pt2.vee.

You should be able to:

- Demonstrate your waveform capture program, lab5pt1.vee.
- Demonstrate your function generator control/waveform capture program, lab5pt2.vee.

Demonstration checklist for lab5pt2.vee:

1. Initialization object to turn on scope channel 1, set probe to 10x, set waveform to Sine, and enable function generator.
2. Generation and capture of Sine, Triangle, and Square waveform types.
3. Dialog to prompt the user if waveform values other than 1,2,3, are specified.