Prelab 4 - Input Offset Voltage (V_{OS})

Introduction

In this lab exercise, you will write and test the code to perform the offset voltage test. This quarter, we will be using the false-summing amp to perform this measurement. Since V_{OS} is a precise measurement, we will be using the QMS for our voltmeter.

Prelab Assignment

a) Print out a copy of the DIB schematic for each site. Show all paths necessary to perform this test. In this manner, you will determine which relays must be closed and which resources you will use. Label the resistors in the same manner that we did in class (e.g. R1, RF, RA, and RB).
b) Calculate the expected output voltage of the DUT to verify that it will not rail.
c) Calculate the expected worst case gained-up offset voltage (e.g. the QMS measurement) so that you know what range to set your QMS at in lab.
d) Using the ETS index, look up the QMS unit. What are the possible ranges allowed for this unit? What are the accuracy for each range? Which range will you use?
e) Will you need to add any more instrument groups or #defines? Explain your answer.
f) Write the test plan for the Vos test. Add this to your existing test plan.

Lab 4 - V_{OS} Test

In this lab, you will add the input referred input offset (V_{OS}) test to your project. For the most part, I will let you write this function on your own. However, there are a few new coding issues that must be discussed.

A. QMS Unit

In this lab, we will be using the QMS unit. Note that the QMS is ONLY a high precision voltmeter. It does not measure current and it does not source voltage or current. The QMS can be set to 9 different ranges, 4 different filter cut-off frequencies, high precision mode, fast measurement mode, variable number of averaged samples, and time between samples.

B. Calculations and Datalogging

Unlike the previous functions, we must perform math on the resulting measurements to get the datasheet result. In order to do this, we must retrieve the data from the datastructure, perform the math, and then store the result back into a new array that is a regular array.

C. Debug routines

When doing math on a measurement, it is often nice to know the actual value measured during debug. In order to get the information, you must use C printf commands to send the information to the screen. However, you don't want this code to waste time during production, so you should set it up so that you must enable it when you want it.

1. Set up a global variable in UserInit that allows you to enable or disable this debug mode. This is done just like the "Site 1 Enable" switch.
2. Place the printf commands in an if statement that will only happen if the debug mode is enabled.
3. Write the printf commands that allow you send measurements from the datastructure to the screen.

Example:

```
      //check for the button to be enabled
      for (int i=0; i<2; i++)
      { if(debugGOL)  //skip this code if debug is not set
            { etsprintf("GOL Test site %d \n VoNullA: %f VoNullB: %f
\n",i+1, VoNullAdata[i].value,VoNullBdata[i].value);
            }
      // \n means new line; %d means replace with an integer; %f means
      // replace with a float; the string sent to the screen is in
      // quotes with the % fields filled in according to the list at the
      // end.
      }
```

D. Write the Function

Reminders of the key coding elements are shown below. Please refer to labs 1 and 2 for details.

1. Set up any additional instrument groups. Some functions will require no additions.
2. Create a new function.
3. Set up the datasheet file.
4. Write the V_{OS} test.
5. Don't forget to modify TestCompletion and FailSite if necessary.
6. Compile, build, and run your program. Fix any errors.
7. Backup your program onto your memory stick and let the professor know that you are ready to go to the tester.
8. First, run your test empty socket and debug any final errors. Then, test both sites simultaneously. Debug any errors. Once both tests pass, play with the accuracy/time tradeoff parameters. Swap your chips. How consistent are your measurements from site to site? Record your results.
9. If you made any necessary changes to your code, make sure you back up the new version and restore it onto your computer for next week.

E. Write up Your Results

Record your test results and your test time. Turn in your final code, test results, and a discussion of the time/accuracy trade-off and site-to-site reproducibility.