Prelab 2 - Continuity

Introduction

In this lab exercise, you will write and test the code to perform the continuity test. The purpose of the continuity test is to determine if there is a complete connection from the ATE instruments to the device. The primary function of the continuity test is to determine if the device has been loaded properly into the DIB. Additionally, it checks for fabrication errors in the device pads. Since most devices failing continuity do so because of improper loading (devices are loaded automatically using a handler and sometimes misalignment causes loading errors), often the devices that fail this test are reloaded and tested again. Therefore, the test is designed to ONLY check for a continuous connection from instrument to device pin. The easiest method of performing this test is to check for the pad protection circuitry that is built into the pad. Fig. 1 shows an example of typical pad protection circuitry. The diodes are reverse biased, and thereby have no impact on the input signal unless the pin goes to a voltage higher than VCC or lower than VEE. Other examples will be presented in class.

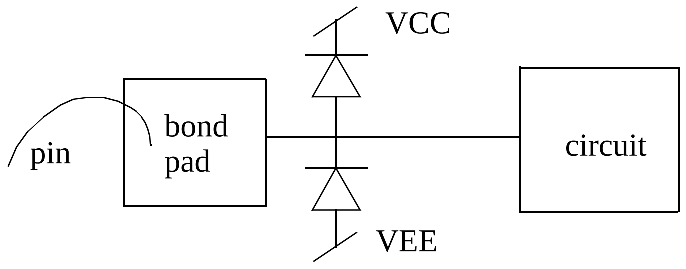

Figure 1: Pad protection circuitry

The simplest and most controlled method of performing this test is presented below and shown in Fig. 2:

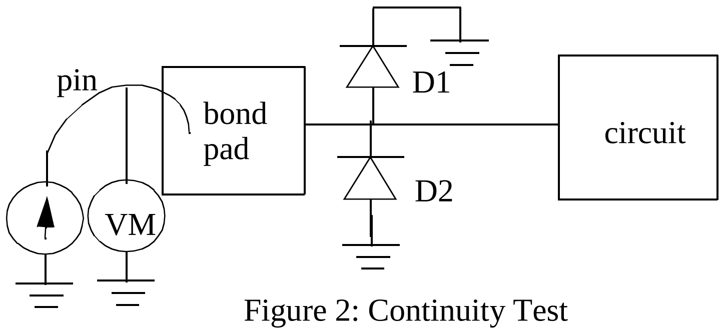

Figure 2: Continuity Test

1. Turn off the rails, VCC and VEE
2. Apply a small current (1mA) to all input and output pins of the chip. Positive current should forward bias D1 and negative current should forward bias D2. Some pads only have D1 or D2.

3. Measure the resulting voltage (Volt Meter, VM). The resulting voltage should equal the diode drop across the forward biased diode (0.2V to 1V). If you are measuring D1, VM should be positive. If you are measuring D2, VM should be negative.
4. Log the results and compare them to the datasheet limits.

Prelab Assignment

a) Be sure and read the introduction so that you understand the test that we will be performing. I will not have time to get to this in class before we do this in lab.
b) Print out a copy of the DIB schematic for each site. This is posted on Angel in the Lab section. You only need pages 2 and 3.
c) 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.
d) Will you need to add any more “groupset” functions, nameset functions, #defines, or does anything need to be added to the msGroupSite? Explain your answer. (Note: the VOS trim pins should be left open for all tests on the LM741. These pins go to resources so that some tests may be performed on other devices.)
e) If you want to start thinking about how you will program this test, that is great. However, please do not actually add any code to your existing project. New functions MUST be added in a particular method or things won’t work. I will show you this method in lab.

Lab 2 - Continuity Test

In this lab, you will add the continuity test to your project.

A. Modify UserInit

While most of the groups were formed last week, we need to add one more group. That group is the relay continuity group. Note: relays are controlled by a transistor whose input (called a CBIT) must be set high or low to activate or deactivate the relay. Assume that the state presented in the schematic is the default state. Closing the CBIT will change its state. Opening the CBIT will return it to its shown state.

1. Identify all relays that must be closed to perform the continuity test.
2. You may use CBIT numbers (the boxed number next to the relay on the site 1 schematic) or relay names. If you use relay names, all relay names must have a #define statement mapping them to the CBIT number. Note that all relays are the same for site 1 and site 2. A single CBIT command sets them together. The only time this would not be the case is for individual site calibration, which we are not doing at this point.

B. Create a new function

1. You MUST ALWAYS add and delete functions using the function wizard. The function wizard not only creates the function, but it also adds the function to the datasheets file which is used for datalogging.
2. Add a function called Continuity.

3. Select the number of “rows” for the datasheet file. The number of rows is equal to the number of values you want to compare to a set of limits for a given site. The multi-site elements will be handled later when we save the values to the datalog. For the continuity test, you want to test Vplus, Vminus, and Vo for each site. Therefore, you need 3 rows.
4. The rest of the fields should remain at the default values. For learning purposes, the function should be added to the “ProductDatasheetFlow”. You can set up multiple Test Flows (e.g. order of tests to be run). We will always use the datasheet to determine our test flow. Having multiple test flows is useful when you have some tests that aren’t run all the time because their intention is to make sure that there is no drift in the process, but they aren’t production tests. You can choose the Test Flow and some test flows will have a different subset of tests than another.

C. Fill out the Datasheet

The datasheet file contains all of the information that you want to save in the datalog. This is the information that you want sent to the screen and saved to the files. It also contains the test limits. The measured value will be compared to the test limits and if it falls within range, the test will pass. If it is out of range, the test fails. The reason the test limits is stored in a different file is so that you can use the same program to test a family of parts with different datasheet files for the different limits (e.g look at the LM741 datasheet - 3 parts are listed with slightly different values for each part. This is a family of parts with basically the same pinout and set of tests, but slightly different values for a given test).

1. Start filling out the datasheet file. This is the first test, so the Test Number should be 1 for all 3 rows.
2. We will separate the different measurements with the Subtest Number. Set these to the values 1-3.
3. Provide a Datalog Description with enough detail that you will know what each measurement is for. In this case, ORDER MATTERS! You must set these in same order as your instruments were grouped or datalogging will be harder later. Set up the description in the same order as this will be the order in which you measure the results.
4. Set the low and high limits. Since you are measuring the diode drop, the resulting value should be between 0.2V and 1V. Note: you will be to be careful of the sign. If you are assuming a positive voltage, you are assuming the you are checking the diode to VCC and thereby are sourcing current out of the instrument. The conditions for test must match the sign of the results.
5. Set the units to be the units that will be measured. If you are datalogging a single measurement, it should be in the units measured (e.g. V for voltage and mA for current). If you have performed math on the measurements, you need to account for the resulting units.
6. Set the Low Fail Bin and the High Fail Bin. When i.c.’s are tested, they are typically sorted as to what passed and what failed. Physically, any chip that fails is scrapped, so further separation is unnecessary and impractical since there are only a limited number of physical bins to choose from. However, a product engineer might like to know which tests failed and how often so that he can propose improvements and choose which test to work on improving. This information can easily be tracked in software where there are a large number of

resources. Therefore, there are “hardware bins”, where the devices are physically sent when they pass or fail, and “software bins”, where the information about each chip is stored. 1 is reserved for passing devices. Map the failing device to a hardware bins to the software bins.

D. APU, SPU, and CBIT Utilities

All functions will generally have the following steps:

(i) Set up the hardware. This involves closing necessary relays and setting the mode and ranging of the source/measurement units, and connecting them to the device.
(ii) wait for the circuit to settle
(iii) measure the resulting currents and/or voltages
(iv) if necessary, perform math on the measurements to obtain the proper spec.
(v) store the results into the datalog
(vi) turn the appropriate equipment off

In order to set up the hardware, we need to know what the standard instruments are: APU, SPU, and CBIT.

1. APU - analog pin unit.
   Sources and measures DC and time-varying current and voltage.

2. SPU: smart pin unit
   Sources and measures DC and time-varying current and voltage. Provides more power and higher resolution.

3. CBIT: control bit
   Opens and closes the relays on the DIB.

E. Write Continuity Function Code

The continuity function should have each of the following steps:

(i) Set up the hardware.
(ii) Wait for the circuit to settle
(iii) Measure the resulting currents and/or voltages
(iv) Store the results into the datalog
(v) Turn the appropriate equipment off

1. Step (i) includes closing the necessary relays and initializing the equipment. You will need to wait for the relays to close and charge to redistribute appropriately.
2. Next, set up your continuity APU’s and your supply SPU’s as directed in the prelab schematics. Example: Set the APU continuity group (consisting of Vmin, Vplus, and Vout for both sites) to source 1mA of current into the continuity pins. You should expect a diode drop voltage measurement, which determines the range of the instrument. Note: positive current leaves the instrument.
3. Set up the SPU’s to control the rails.

Note: you will generally source +/-15V to the chip for your other tests. Since this is not a high precision test, it's best to set the mode of the instrument to something that you will be using later.

4. Step (ii): wait for the circuit to settle with the input values set. At this time, estimate a wait time of 1000usec. However, we will play with this wait time and see if it makes a difference on the ATE.
5. Step (iii): Measure the instrument readings and store the results. We can change the number of averaged samples and time between samples on the ATE and see the impact on the measurements.
6. The measurements will be stored into a datastructure that is set up in the same order as the instrument group. Each element in the datastructure contains 4 fields (resource or instrument number, site, value, and whether the measurement Passed or Failed), so the data for each site must be stored along the y-axis, as shown in Table 1. Since we have 3 instruments with site 1 and 3 instruments with site 2, the final datastructure should contain 6 elements as shown.

Table 1: Datastructure for continuity

Resource	site	value	PassFail	
APU0 (Vmin)	1	.7	1	site 1
APU1 (Vplus)	1	.72	1	
APU2 (Vout)	1	.8	1	
APU8 (Vmin)	2	10	0	site 2
APU9 (Vplus)	2	10	0	
APU10 (Vout)	2	.75	1	

Once the information is stored in the result structure, you can perform math functions on it if necessary. This is not necessary for the continuity test.

4. Step (iv): log the results. Once all data is stored, you must place the results into the datalog. The measured results must be compared to the datasheet file and then determined whether or not the measurements pass or fail the limits.
5. Once the results are properly stored, you must turn the equipment off.

F. Modifying the Set-Up Code

Each time you have written a function, you must also go back to the functions FailSite and TestCompletion to make sure that any new instruments that you have used are turned off if a site fails or when the program is done. That way, you will not blow the next device that is placed into the socket.

G. Test and Debug

1. *Compile and correct errors.*
   The results and any error messages should be displayed at the bottom of the screen. If you have an error, you can double click on the error message and it will automatically take you to the line of code that threw the error.
2. *Build and correct errors*.
   The results and any error messages will display at the bottom of the screen.
3. *Run and correct errors*.
   Run with or without debugging depending on whether or not you want to debug your code or not.
   If you have no run-time errors, you should see a display showing the results of the tests for both sites.
   All continuity tests should fail without a chip.
   If you have run-time errors, break points can be set by clicking on the gray area beside the code. They can be removed by clicking again. Set a break point, run with debugging and find which message causes the error.
4. *Backup your code to a memory stick.*
5. *Take your memory stick and restore your code onto the ATE computer*
   Make sure the correct board is on the ATE, that power to the board is on, and that the board has a vacuum.
6. Select your test then open it.
7. Run your test code without debugging on site 1 with a chip in site 1. If all results seem normal, continue. If not, debug your code on the ATE.
8. Activate site 2 and see if the test passes site 1 and fails site 2.
9. Test both sites simultaneously.
10. Play with some of the wait times, number of samples and time between samples to see how the measurement is affected.
11. 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.

H. Write-up your results

Record your test results and your test time. How do they compare with another students results in the class? Turn in your continuity function code, test results, and a discussion of similarities and differences between your results and someone else in the class or the impact that changing wait times or number of samples had on the measurements.