**Capacitance measurement**

The approach we used was similar to that used on the ECE 4760 Labs listing from Fall 2014, which is what we will base this section of the project description on. The idea is to measure the time it takes for a RC circuit to charge a capacitor to a given level. If R3=R4 in the schematic below then the level will be v(t1/2)=Vcc/2. Specifically, we used the internal analog comparator as shown in the following diagram to trigger a timer1 event. In our case, we connected a trimpot in place of R4 to fine tune the capacitance meter for accuracy. Since R2 will be known, we can get C because the voltage on the capacitor, v(t)=Vcc(1-exp(-t/τ)) with τ=(R2)\*C. We used an analog DEMUX to select different values for R2 (1k 20k and 100k) to autorange the capacitance readings.

The resistor going to PortB2 should be around 100 ohms to limit current. R2 must be chosen such that the capacitor charging time is not too short or too long, so that must be considered when implementing autoranging. If the charge time is too short you will lose measurement accuracy. It if is too long, the timer will overflow.



**Figure 3:** Simplified Circuit for Capacitance Measurement (Credit Bruce Land)

Our program executes in the following order:

1. Set PortB3 to an input.
2. Drive PortB2 to zero by making it an output and wait long enough to discharge the capacitor through 100 ohms. Clearly, to discharge to zero volts with 1% accuracy, R2>100\*(100ohms).
3. Convert PortB2 to an input and start a timer. The capacitor will start to charge toward Vcc.
4. Detect when the voltage at PortB2 is greater than the voltage at PortB3. That is, you will have to record when the comparator changes state using the timer1 input capture function, which is set up to be triggered by the comparator. Using input capture gives better timing accuracy and more dynamic range.
5. Switch to a smaller resistance on the analog DEMUX if your timer overflows, or if you read an unusually small charge time (such that your resolution is not enough for good accuracy) switch to a larger resistance.
6. Repeat

If you are using an AVR Microcontroller, something the 4760 lab page mentions is that if you decide to print floating point numbers to the LCD you need to follow the directions in [Using sprintf function for float numbers in AVR-GCC](http://winavr.scienceprog.com/avr-gcc-tutorial/using-sprintf--function-for-float-numbers-in-avr-gcc.html) the AVRstudio directions are at the bottom of the page, but **NOTE** that one library is left out. The correct summary is:

Open the Project>Project Configuration dialog box, then go to the custom options , then select linker options, and then add the

-Wl,-u,vfprintf -lprintf\_flt -lm and -Wl,-u,vfscanf -lscanf\_flt -lm switches.

**Low Voltage (<= 5V), High Voltage (5V< v < 20V), Resistance, and Frequency:**

Again, for voltage, resistance and frequency measurements we referenced work done in a 4760 lab from Fall 2012, which can be found on the course page. Something to first note are that the 1k resistor and diodes protect the analog input (A0) from over/under voltage. See Figure 3 below.



**Figure 3:** Circuit for Resistance, Voltage, and Frequency Measurements

Low Voltage measurements were taken by setting up the internal ADC to take measurements from A.0 on the MCU. We set Vref to Vcc, and found that there were good accuracy measurements above 100mV. However, one could autorange voltage better by changing Vref internally for smaller, more accurate measurements. High voltage measurements were taken using a voltage divider that divides any input voltage by 10.

We have 5 decade resistors (Green Box in Figure3) from 100 Ohms to 1 Mega Ohm (shown in Figure3) connected to an analog DEMUX. We turn them on (one at a time) to autorange and measure resistance. The tristate i/o pins make it very easy to connect a resistor to ground (output-low), Vcc (output-high), or disconnect (input-no pullup). We are measuring the voltage that results from division with a known resistance value coming from the DEMUX, so if we denote one of the five resistors as R, and the resistor-under-test connected to the probe as Rt (Blue Box in Figure3), then Rt = R/(Vcc/Vin - 1). The sensitivity of the resistance measurement decreases as Vin increases, suggesting that we should keep Rt<<R. However, low Vin means significant quantization error in the ADC. We keep Vin between about 0.1\*Vref and 0.9\*Vref during resistance measurements for 5% accuracy. We autorange resistance by turning different resistors on to divide voltage closer to logic level for MCU operations.

B2 is the input to the internal comparator, which we also use for capacitance measurements, but we use it here for frequency measurements. B3 is the reference pin for the comparator, which is divided to Vcc/2. Similar to making capacitance measurements, we set up the timer1 interrupt to trigger at the positive clock edge, and set a timer that counts on a 0.5us timescale as the prescalar for timer1 is set to divide the crystal frequency by 8 in our code. We autorange frequency by changing the timer 1 prescalar.

Additionally, we also detect the pulse width of an input signal by interrupting on each positive and negative edge on the comparator connected to timer1 and keeping track of the timing for both positive and negative edges. We are therefore able to calculate the duty cycle of an incoming wave using $Duty Cycle=\frac{Pulse Width}{T}$, where T is the period of the incoming signal.

The probe input goes to the ADC and internal comparator. This example from the ECE 4760 course page ([ADCtestGCC644.c](http://people.ece.cornell.edu/land/courses/ece4760/labs/s2012/lab4code/ADCtestGCC644.c)[uart.c](http://people.ece.cornell.edu/land/courses/ece4760/labs/s2012/lab4code/uart.c), [uart.h](http://people.ece.cornell.edu/land/courses/ece4760/labs/s2012/lab4code/uart.h), [project zip](http://people.ece.cornell.edu/land/courses/ece4760/labs/s2012/lab4code/ADCtest.zip)) shows how to set up the A/D converter to read a trimpot.

This [example](http://people.ece.cornell.edu/land/courses/ece4760/labs/s2012/ADC_ISR_GCC644.c) (also from the course page) shows how to get higher ADC accuracy by putting the MCU to sleep during the conversion to cut down digital noise. Note that some special timing code has to be added to make the UART work correctly when you put the MCU to sleep.

Something Important the 4760 lab we referenced mentioned is that you may want to check the V-reference impedance and the capacitance (if any) placed from Aref to ground on your board. On the Mega644 board it is 30 kohm with a 20 nF capacitor placed from Aref to ground. This implies the time-constant to CHANGE Vref is 3e4 x 20e-9 = 60e-5 or 600 microseconds! This means that to get 8-bit accuracy after you change Vref, you must wait 1/256 = exp(-t/600) or t~3 mSec before doing another conversion! The calculation is similar for any board after inspecting the datasheet.

**Transistor Testing**

The test sequence for a transistor works like this in our code:

1. Set D2 High and read D1 and D3. If D1 is High, BE junction conducts, otherwise not. If D3 is High, BC junction conducts, otherwise not.
2. Set D1 High and read D2. If D2 is High, EB junction conducts, otherwise not.
3. Set D3 High and read D2. If D2 is High, CB junction conducts, otherwise not.



**Figure 4:** Transistor Testing Circuit

Now, if only the BE and BC junctions conduct, the transistor is of NPN type and is working fine. And, if only the EB and CB junctions conduct, the transistor is still normal but the transistor type is PNP. All other cases indicate the transistor is no good (such as EB and BE both conduct, or BC and CB are both not conducting, etc.).

**Diode Testing**

This is also taken care of at the low voltage probes. This essentially checks to see if the diode approximately drops 0.7V (as per standard designs) through ADC0 on our MCU. If 5V is passing to the ADC, we know the diode has broken down and basically become a short, otherwise, if there is no voltage on the ADC, we know that either nothing is connected or the diode is connected in reverse bias, which we display on our LCD.

**Temperature Testing**

For Temperature Sensing we used the LM35 Precision Centigrade Temperature Sensor. A quick look at its [datasheet](http://www.ti.com/lit/ds/symlink/lm35.pdf) reveals that the LM35 has a range of 2 ֩C to 150 ֩C when no load on the output. The voltage change with respect to temperature is basically linear, so we simply multiplied the incoming voltage detected by ADC7 with a scalar.

**Connectivity Testing**

Connectivity testing was done by connecting an LED from pin D.0 to the cathode probe. When in this mode we set D.0 high, so when something is connected across the probes, current flows from D.0 through the LED and to ground, illuminating the LED.

Everything was carefully soldered onto a two level, through hole PCB configuration, which we then mounted on an arm strap which holds the MCU. Connections are then soldered from there to the LCD display, touch button, and testing circuits on the gloves.

**LCD Display**

The LCD display used in the setup is a 16x4 dot matrix display – EA DOG-M 163W-A series LCD. More information about the display can be found here in the datasheet. We had to write our own library functions for interfacing with the code. The header and C files for the LCD are attached at the end in the appendices section.

The backlight for this LCD comes as a separate module- we got the EA LED55x31-B backlight module from DigiKey and interfaced it with the LCD module package by soldering the back light LED pins. The LCD was interfaced using +5V 4 Bit interface connected on the PORT pins C0-C3 and control pins on the C5-C7.

Detailed connections of the LCD can be found here in the overall schematic.

**Programmer**

We used an AVRISP MK II programmer provided by Prof. Bruce R Land. The ISP programming frequency was set to 4 MHz for higher writing speeds. Fuse bits and JTAG connections were checked as per ECE 4760 Lab 1 instructions. We did experience some glitches while programming the MCU when it would automatically reset itself while programming. Resetting the system helped us to overcome this problem.

**The Overall Multimeter Schematic can be found here!**