

The City College of New York

Grove School of Engineering - Electrical Engineering Dept.

EE 42500 – Computer Engineering Lab

Professor Feng

Spring 2020

**Lab 2: Data Acquisition: Analog-to-Digital and Digital-to-Analog Conversions**

**Prepared By**: Ben Variano

**Lab Partner:** Huiwen Li

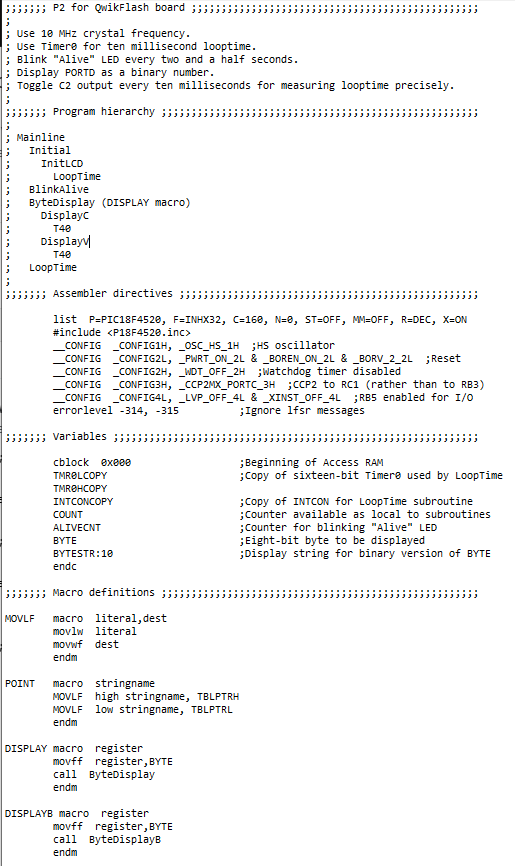
**Objective**

The purpose of this lab was to understand the ability of the built-in Analog-to-Digital converter module and the external Digital-to-Analog converter module. In Part 1 we were asked to perform an Analog-to-Digital conversion of an analog input controlled by a potentiometer to a 10-bit binary number and display it on the LCD screen. In Part 2 we did the opposite, and converted an 8 bit binary number into an analog output voltage. As a result of this lab we learned the power of the PIC controller to convert between the analog and digital worlds, and learned how to control the function of this converter through the use of registers and assembly code.

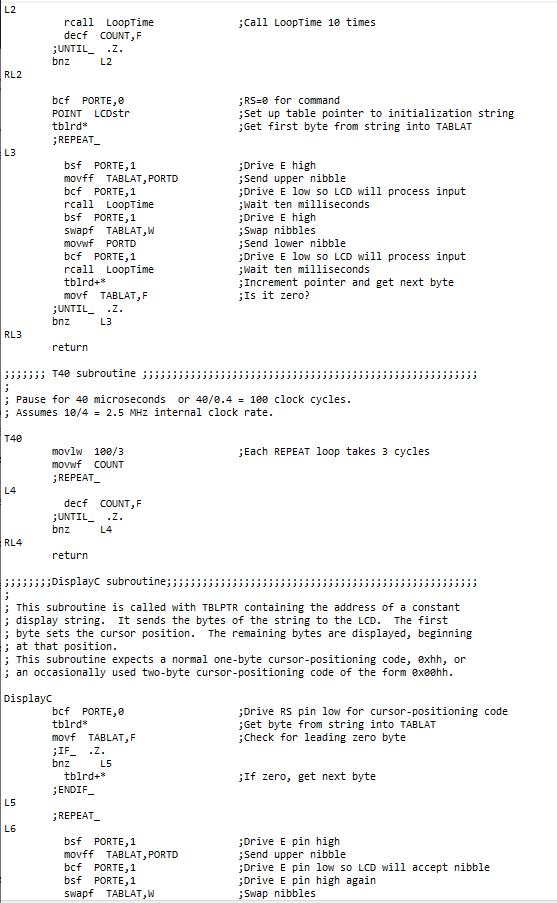
**Part 1**

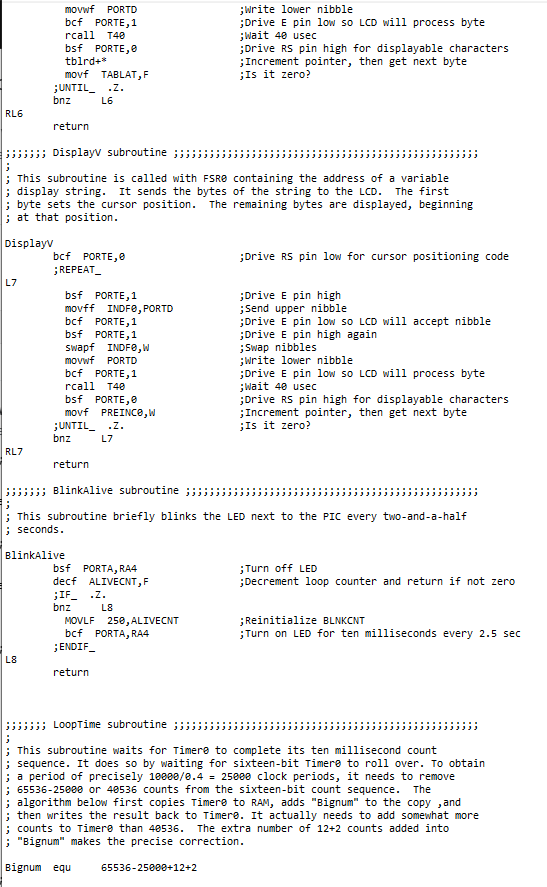
In Part 1 the task was to convert an analog input voltage to a 10-bit binary number that a computer can understand using the PIC converter. We wired the input voltage through a potentiometer so that we can change it, and observe the change of the binary number that we then displayed on the LCD screen. We recorded both the measured input and the binary output and compared them to examine the accuracy of our conversion.

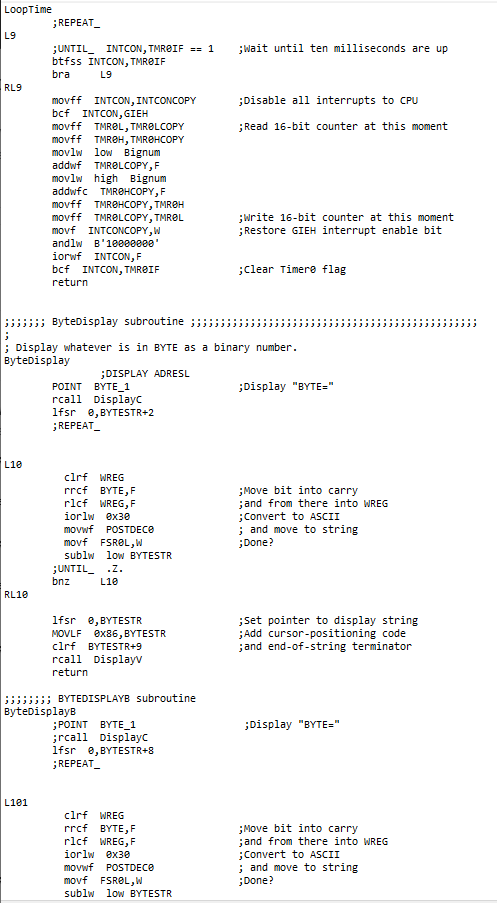
The main aspects in part 1 that we had to pay attention to in the code was our ADCON0, ADCON1, ADCON2 registers that control the ADC module. From ADCON0 we are able to enable the A/D converter, start the conversion, check the conversion status, and select the input channel. In ADCON1 we were able to set the configuration of the ADC by selecting the voltage reference and input type(A/D). ADCON2 is used to select the output as left or right justified, set the clock, acquisition time. The ADRESH and ADRESL are used to hold the converted signal as binary numbers. Using the ADFM bit in the ADCON2register, we could control whether the 10-bit binary number was left or right justified, which is important when displaying it on the LCD screen. The following code shows how we set the ADCON registers, as well as the check for conversion, and output to an LCD display.

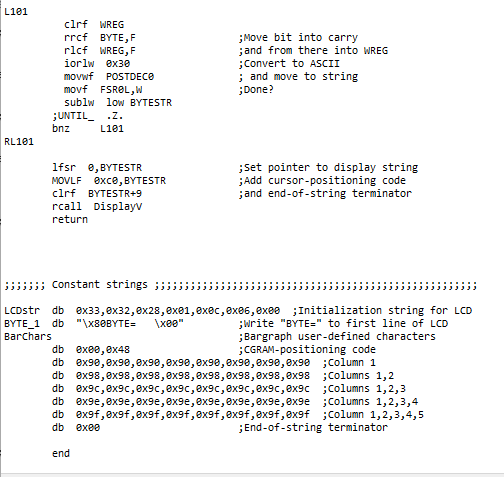


****

****

****

****

****

**Part 2**

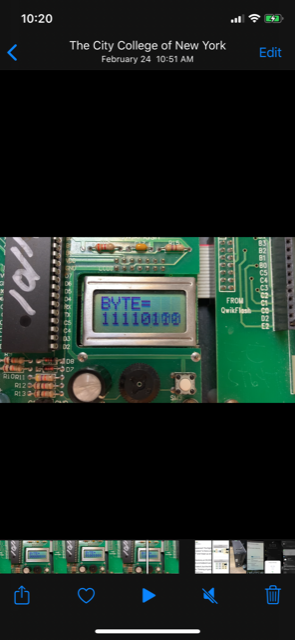
In Part 2 the task was to take an analog signal that we converted to an 8-bit binary number, via a similar process seen in Part 1, and convert it back to an analog signal to be outputted to an oscilloscope beside the original input signal, so that we can see the accuracy of the PIC conversion. We set a function generator to High Z mode, with a 5 Vpp sine wave and a 2.5 V offset as our input signal. The PIC would then convert this signal to an 8-bit binary number that is stored in the ADRESH register. Instead of displaying this binary number, we instead converted it back to an analog signal using a D/A conversion via the PIC board. We displayed this output alongside the input on the Oscilloscope.

The main aspects of Part 2 we focused on was the procedure for converting the signal. We had to decide with output to use, we used output A: 0x21. We then had to initialize the ADCON0, ADCON1, ADCON2, SSPIF, SSBUFF, TRISC registers in order to initialize everything (SCK, SDO, RC0 outputs) and configure the converter for Digital-to-Analog conversion. We then manipulated these registers and bits so that we could write the 8-bit ADRESH register to the SSBUFF for conversion back to analog. The following images show the input and output offset from one another on the oscilloscope. As we increase frequency we see a discrepancy since the input is changing faster than the clock pulse on the PIC board so it can’t convert quick enough to keep up. Otherwise we see how accurate the conversion is at appropriate frequencies. Our final code for this project is on the Lab computer on campus, and cannot be retrieved due to the Coronavirus closing campus.

**Images**

**Part 1**

*********Figure 1: Potentiometer and Displayed Binary* *Figure 2: Potentiometer at Full (Infinite)*   
 *Output Resistance*

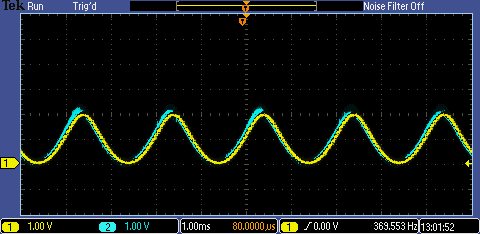
 

*Figure 3: Potentiometer at some Resistiance* *Figure 4: Poteniometer at some less* *Less Than Infinity* *Resistance greater than 0, Less Than Figure 3*

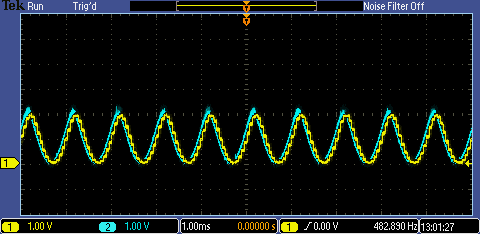


*Figure 5: Potentiometer at 0 Resistance*

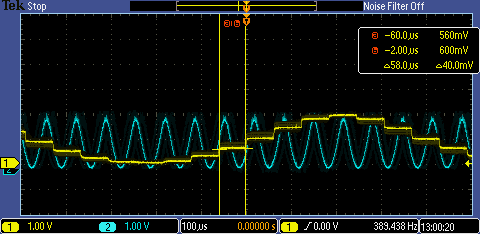
**Part 2**



*Figure 6: Channel 1 Output vs. Channel 2 Input; Accurate Conversion Leads to Identical Sine Waves*



*Figure 7: Channel 1 Output vs Channel 2 Input; Increased Frequency, Sine Waves Remain Identical*



*Figure 8: Channel 1 Output vs Channel 2 Input; as the Frequency is Increased to Faster than the PIC Board Clock Cycles Can Read, We See Discrepancy in the Input vs Output Signals*

**Conclusion**

As you can see the PIC controller has a very accurate Analog-to-Digital and Digital-to-Analog conversion. We were able to properly convert any input signal to an accurate 10-bit binary representation of that signal such that a computer can understand the signal and read. We were also able to accurately transform a given 8 bit binary signal back to an analog signal to be displayed on the Oscilloscope. Because of this, we could take in an analog signal, convert it to binary, and then convert it back to analog to be displayed on the oscilloscope next to the original input signal, showing it as almost perfectly identical to the input signal. As we increased the frequency higher, we began to see discrepancies in the input verse output signal since the clock pulse couldn’t read the signal in fast enough to properly convert it.