Microavionics Lab 1 (5067)

Sage Herrin
University of Colorado Boulder

This lab touched on the basic operation of the EasyPicPRO V7 board and how to upload programs such as a test suite to test the basic operation of the board. Later a serial capture program is used to read out the temperature and voltage read by the board.

A. Written Lab Questions

1.

The entire provided test suite did pass. LEDs flashed and the tune was played shortly after the board was powered up, a tone was emmited when the RD3 push button was enabled, and the frequency was effectively changed when the P2 potentiometer was rotated. The LCD display showed the temperature read from the LM35 sensor and displayed the current voltage as well (roughly 3.23 V). Figures one through three below show a variation from two to four KHz read by the AD2 logic analyzer.

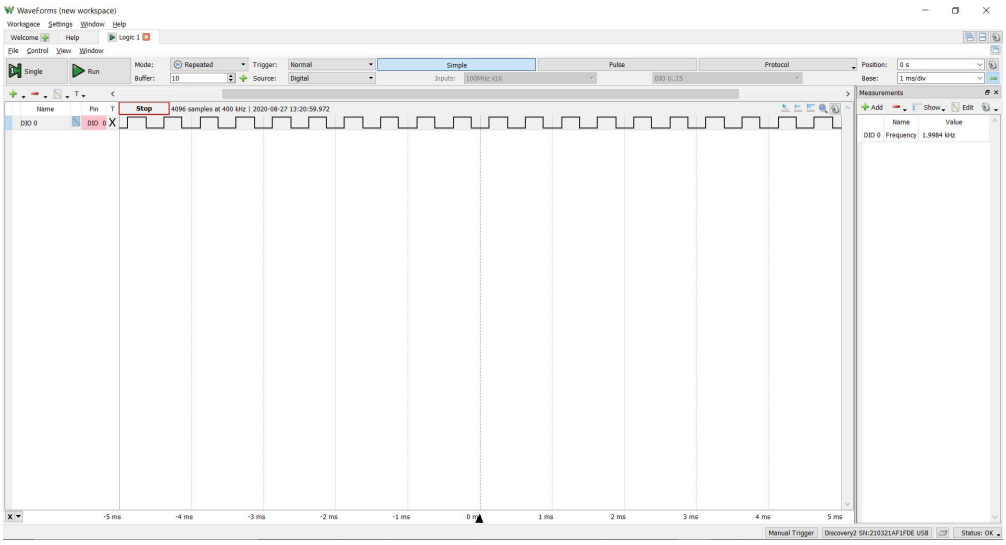

Fig. 1 2 KHz logic analyzer readout

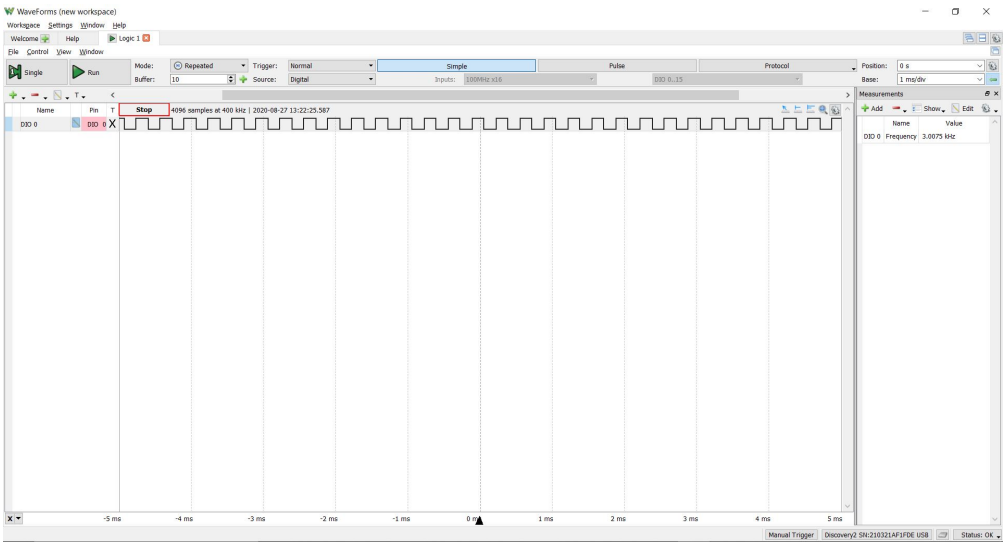

Fig. 2 3 KHz logic analyzer readout

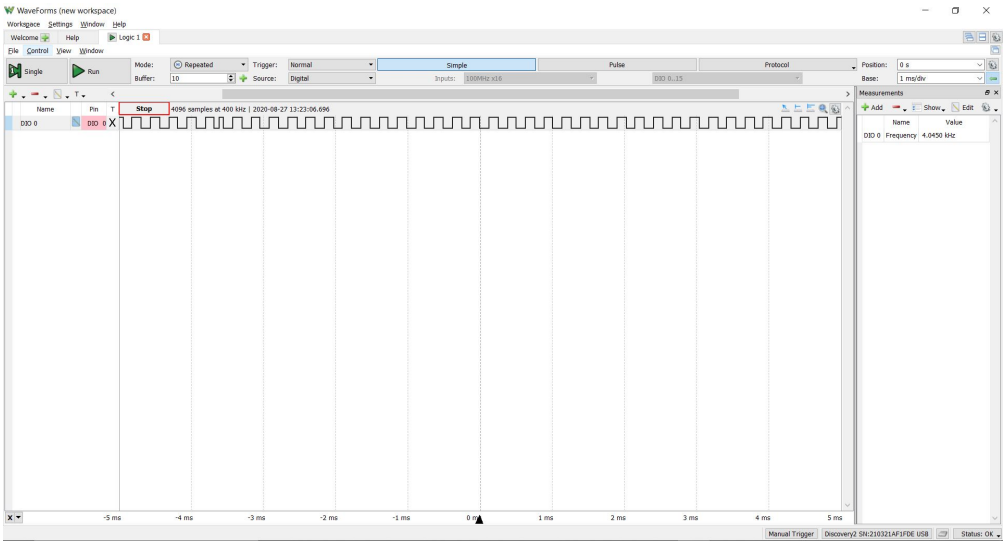

Fig. 3 4 KHz logic analyzer readout

Figures four through seven below show the boards response to the following USART commands from the RealTerm terminal.

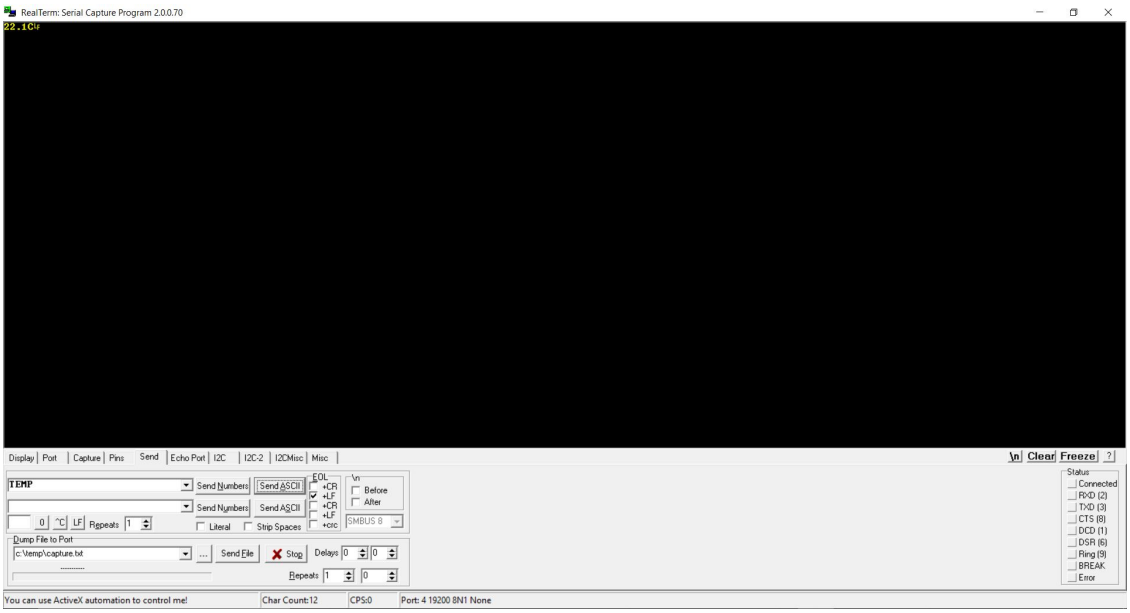

Fig. 4 Temperature value displayed in RealTerm terminal

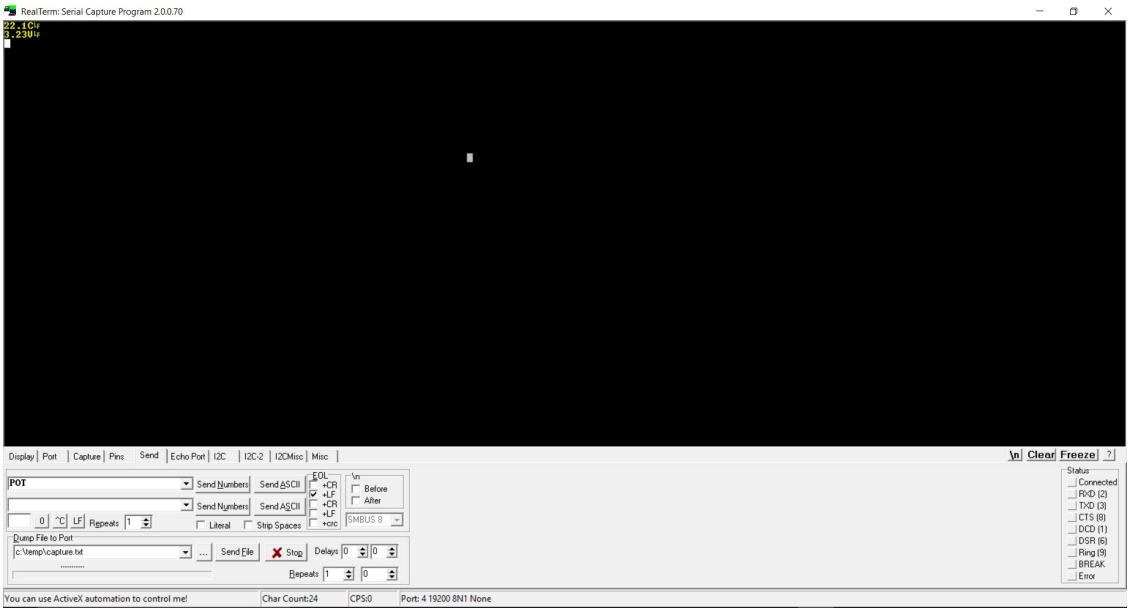

Fig. 5 Volatage value displayed in RealTerm terminal

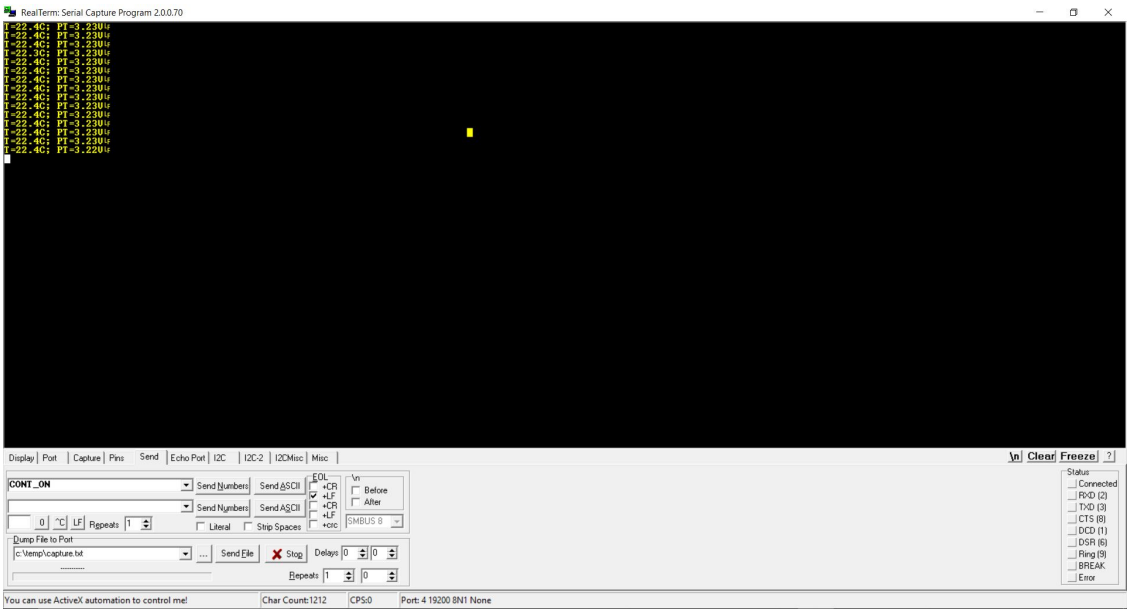

Fig. 6 Continuous temperature and voltage values in RealTerm terminal

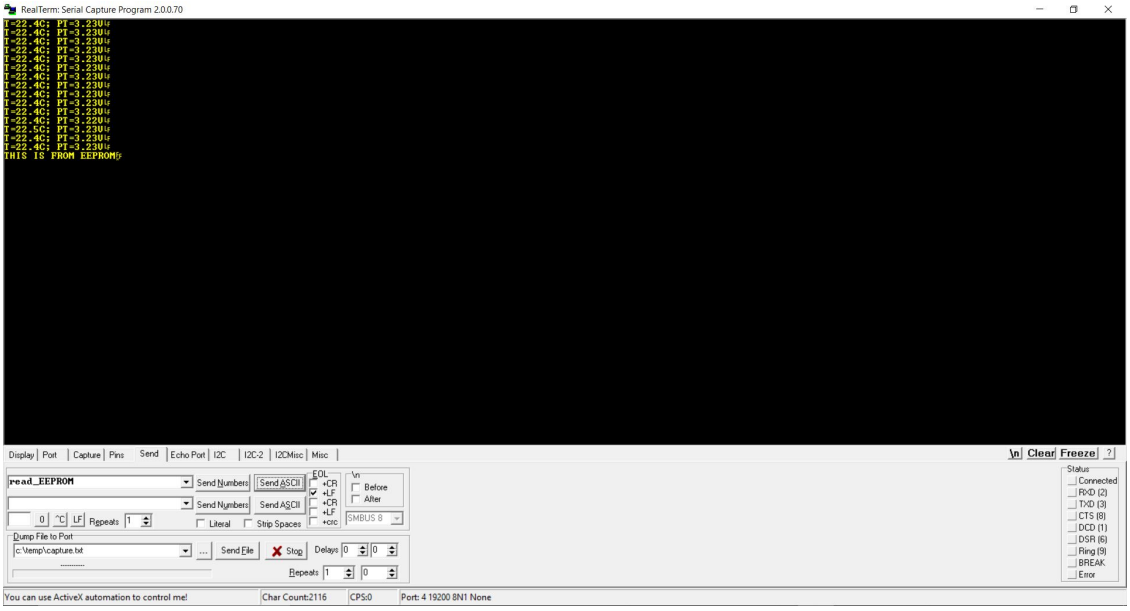

Fig. 7 EEPROM readout displayed in RealTerm terminal

Figure eight below shows the 0.5 Hz triangle wave output from the DAC to the scope in the AD2

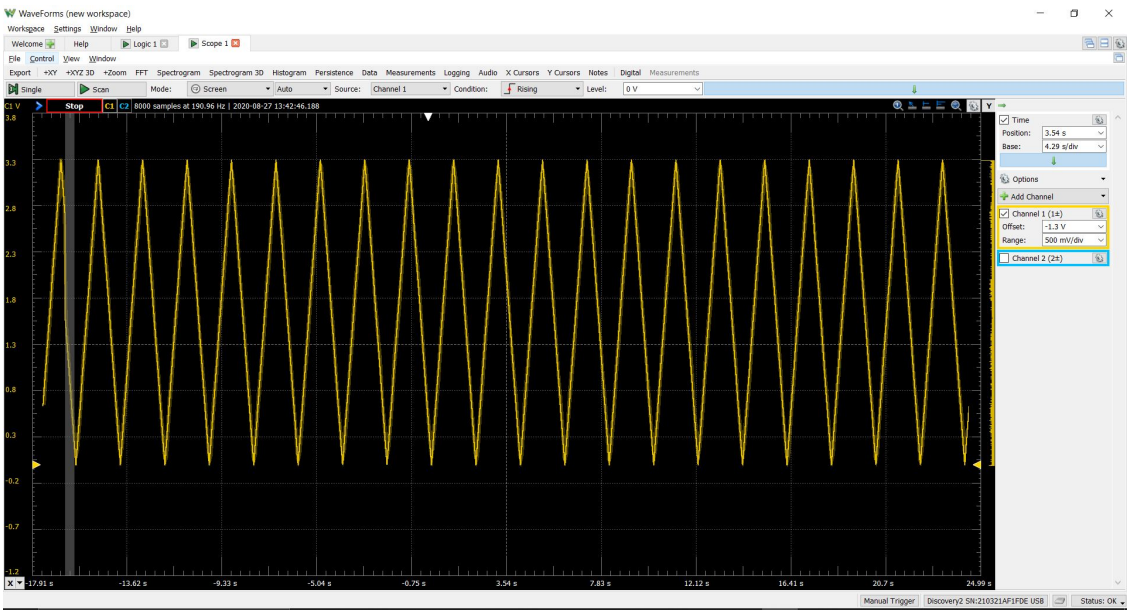

Fig. 8 Triangle wave output from DAC to AD2

2.

The three pins on the J4 header connected to the board are TS2, the RF6, and the RA3 pins. These three pins allow the PIC board to obtain analog temperature reading from the LM35 temperature sensor. The current configuration of the yellow indicates that the PIC is using the RA3 pin to obtain temperature readings from the LM35.

3.

The Piezo buzzer on the board makes noise when the board is programmed because the buzzer and the signal/channel that the program is being uploaded on are common or shared between the buzzer and the RX pin.

4.

When the switch for RG7 is in the up position, the LED remains on. When the switch is in the middle, the LED is off, and when it is in the down position the LED is also off. While the switch is in the up position and the LED is on, pushing the RG7 button makes the LED slightly brighter while the button is held down. When the switch is in either the middle or the down position, pushing the button turns the LED on while the button is pressed, but turns off once the button is released. This is because putting the switch in the up position connects the resistor in pull-up state to the selected pin, while middle position disable both pull-up and pull-down states, and the down position connects the resistor in the pull-down state, so in both middle and down position, the board is seeing a logical low, so the light is off, until the push button sends it to high. The resistors on SW13 are used to make sure the proper current is delivered to each of the ports on the board, and the resistors after the LEDs are used to limit the current delivered to the LED so that the LED is not damaged.

5.

High level programming is a type of programming that uses a language that is hardware/OS independent, so that it can be used with many different boards and computers. An advantage to this is how universal this type of programming is as well as how readable some of the languages used can be, such as Python. However the need for high-level code to be translated into lower level languages such as assembly and machine language necessitates more memory and longer execution times.

Low-level languages are advantageous in that they have much quicker execution times compared to high level languages and require less memory when used, but they can be hard to read and slow to debug and fix errors when they

inevitably arise, and are hardware specific, so assembly language for an Intel processor for example is not portable to a Motorola processor.

B. Gaonkar Questions

1.8

The function of the the address bus on a board is so that the processor can identify memory registers and I/O devices. The data bus is used to transfer binary data or instructions to and from the processor, memory, inputs, and outputs, such as when the processor reads instructions from memory. The Control lines, also called Read and Write, are used to timing signals to enable memory and I/O devices, for example asserting the Read timing signal in order to enable the memory chip so that it can read instructions from memory.

1.9

The address Bus is unidirectional because the microcontroller is addressing to specific locations in memory, and nothing is ever writing to the processor. The data bus is bidirectional because data is flowing both from the processor to memory and outputs as well as from inputs and memory back to the processor, e.g. when the processor is reading data from an input, reading from memory, writing to memory, or outputting a signal.

1.10

The number of bits that can be stored in 1 K-byte (KB) of memory are 1 byte = 8 bits, 1 KB = 1000 bytes = 8,000 bits

1.11

The number of registers in 8 KB of memory assuming 8 bits per register $\rightarrow$ 1 KB = 1,024 bytes $\rightarrow$ 8 KB = 8,192 bytes $\rightarrow$ 8,192 registers. If there are 8,192 registers, and the first starts at hex address 0000, then the last register has an address equivalent to 8,191 in hex, by dividing by 16 and taking the remainder $\rightarrow$ 1FFF is the address of the last register.

1.15

converting 07FF from hex to decimal $\rightarrow$ starting from the right, $F = 15 * 16^0 = 15$, $F = 15 * 16^1 = 240$, $7 * 16^2 = 1,792$, $0 * 16^3 = 0 \rightarrow 07FF = 15 + 240 + 1792 = 2047 \rightarrow$ there are 2,048 memory addresses, so this is a 2KB memory chip.

2.11

The purpose of the program counter is to hold program memory addresses of instructions that are to be read next my the MPU. It is also used a locator to fetch instructions for the MPU. Once an instruction is fetched from memory, the PC increments to point/fetch the next instruction for the MPU. The PC is a 21-bit register in the PIC18 MCU.

convert 47,82,127,129,243,-23,-67, and -255 to binary and hex and 0x33,0x19,0x64,0x4C, and 0xAB to decimal for both signed and unsigned numbers

Starting with binary conversions:

$\frac{47}{2} = \frac{23}{2}^{1} = \frac{11}{2}^{1} = \frac{5}{2}^{1} = \frac{2}{2}^{1} = \frac{1}{2}^{0} = 0^{1} \Rightarrow 47 =$ **00101111**

$\frac{82}{2} = \frac{41}{2}^{0} = \frac{20}{2}^{1} = \frac{10}{2}^{0} = \frac{5}{2}^{0} = \frac{2}{2}^{1} = \frac{1}{2}^{0} = 0^{1} \Rightarrow 82 =$ **01010010**

$\frac{127}{2} = \frac{63}{2}^{1} = \frac{31}{2}^{1} = \frac{15}{2}^{1} = \frac{7}{2}^{1} = \frac{3}{2}^{1} = \frac{1}{2}^{1} = 0^{1} =$ **01111111**

$\frac{129}{2} = \frac{64}{2}^{1} = \frac{32}{2}^{0} = \frac{16}{2}^{0} = \frac{8}{2}^{0} = \frac{4}{2}^{0} = \frac{2}{2}^{0} = \frac{1}{2}^{0} = 0^{1} =$ **10000001**

$\frac{243}{2} = \frac{121}{2}^{1} = \frac{60}{2}^{1} = \frac{30}{2}^{0} = \frac{15}{2}^{0} = \frac{7}{2}^{1} = \frac{3}{2}^{1} = \frac{1}{2}^{1} = 0^{1} =$ **11110011**

$-23 \Rightarrow$ start with 23 & use 2's complement

$\Rightarrow \frac{23}{2} = \frac{11}{2}^{1} = \frac{5}{2}^{1} = \frac{2}{2}^{1} = \frac{1}{2}^{0} = 0^{1} \Rightarrow 00010111 \Rightarrow 11101000$ (2's comp.)

add 1 $\Rightarrow$ **1110 1001 = -23**

$-67 \Rightarrow \frac{67}{2} = \frac{33}{2}^{1} = \frac{16}{2}^{1} = \frac{8}{2}^{0} = \frac{4}{2}^{0} = \frac{2}{2}^{0} = \frac{1}{2}^{0} = 0^{1}$

$\Rightarrow 01000011 \Rightarrow 10111100$ (2's comp.), add 1 $\Rightarrow$ **1011 1101 = -67**

$-255 \Rightarrow \frac{255}{2} = \frac{127}{2} = \frac{63}{2} = \frac{31}{2} = \frac{15}{2} = \frac{7}{2} = \frac{3}{2} = \frac{1}{2} : 0$

$\Rightarrow 1111\ 1111 \Rightarrow$ 0000 0000, add 1 $\Rightarrow$ **1111 1111 0000 0001**

2's comp,

Now doing conversions from decimal to hex:

$\frac{47}{16} = \frac{2}{16} = 0$ (15) $\Rightarrow$, $15 = F \Rightarrow 2\ 15 \Rightarrow$ **$2F_H = 47$**

$\frac{82}{16} = \frac{5}{16} = 0 =$ **$52_H = 82$**

$\frac{127}{16} = \frac{7}{16} = 0 \Rightarrow$ **$7F_H = 127$**

$\frac{129}{16} = \frac{8}{16} = 0 =$ **$81_H = 129$**

$\frac{243}{16} = \frac{15}{16} = 0 =$ **$F3_H = 243$**

-23, starting with previously found binary rep. & converting to hex

$\Rightarrow 1110\ 1001$, $1110\ 1001 \Rightarrow$ **$E9_H = -23$**

14 → E 9

-67 = 1011 1101 (found previously) $\Rightarrow$ $1011\ 1101 \Rightarrow$ **$BD_H = -67$**

11 → B 13 → D

$-255 = 1111\ 1111\ 0000\ 0001$ (found previously)

$\Rightarrow$ 1111 1111 0000 0001 $= FF01_H = -255$
15→F 15→F 0 1

converting hex to decimal for both signed & unsigned #s:

0X33

$\Rightarrow 3\times16^1 + 3\times16^0 = 48 + 3 = 51$ for unsigned #

in binary, $\frac{51}{2} = \frac{25}{2}^1 = \frac{12}{2}^1 = \frac{6}{2}^0 = \frac{3}{2}^0 = \frac{1}{2}^1 = 0^1 \Rightarrow 0011\ 0011$, since the MSB is 0, the signed number is also 51

$0X19 \Rightarrow 1\cdot16^1 + 9\cdot16^0 = 16+9 = 25$

in binary, $\frac{25}{2} = \frac{12}{2}^1 = \frac{6}{2}^0 = \frac{3}{2}^0 = \frac{1}{2}^1 = 0^1 = 00011001$. MSB is 0,

so signed number is also 25

$0X64 \Rightarrow 6\cdot16^1 + 4\cdot16^0 = 96 + 4 = 100$ for unsigned number

in binary, $\frac{100}{2} = \frac{50}{2}^0 = \frac{25}{2}^0 = \frac{12}{2}^1 = \frac{6}{2}^0 = \frac{3}{2}^0 = \frac{1}{2}^1 = 0^1 = 01100100$,

MSB is 0, so signed number is also 100

$0X4C \Rightarrow 4\cdot16^1 + 12\cdot16^0 = 64 + 12 = 76$ for unsigned number

in binary, $\frac{76}{2} = \frac{38}{2}^0 = \frac{19}{2}^0 = \frac{9}{2}^1 = \frac{4}{2}^1 = \frac{2}{2}^0 = \frac{1}{2}^0 = 0^1 = 01001100$,

MSB is 0 so signed number is also 76

$0XAB => 10 \cdot 16^1 + 11 \cdot 16^0 = 160 + 11 =$ **171 for unsigned number**

in binary, $\frac{171}{2} = \frac{85}{2} = \frac{42}{2} = \frac{21}{2} = \frac{10}{2} = \frac{5}{2} = \frac{2}{2} = \frac{1}{2} = 0$

(remainders, read bottom to top: 1 0 1 0 1 0 1 1)

$=> 10101011$, since MSB is 1, subtract 1 & invert,

$=> 10101010 => 01010101 => 2^6 + 2^4 + 2^2 + 2^0 = 64 + 16 + 4 + 1 = 85$

$=>$ **-85 for signed number**

14

If the J1 header were switched to the 5V position, this particular board could still use the 5V logic instead of the usual 3.3V because this orientation of the J1 header enables a voltage regulator on the board to convert the 5V input to 3.3V to be used for the logic on the board.

15

Using the AD2 voltmeter function, the measured voltage of the RA3 pin on the board measured 285 mV. Using the scaling factor in the LM35 data sheet of 10 mV/degree Celsius, this translates to 28.5 degrees Celsius. Using the conversion from Celsius to Fahrenheit of $C * 1.8 + 32$ yields a temperature of $28.5 * 1.8 + 32 = 83.3$ degrees Fahrenheit. This temperature makes sense for a late afternoon day in August in Colorado due to the higher average temperature for this time of year.

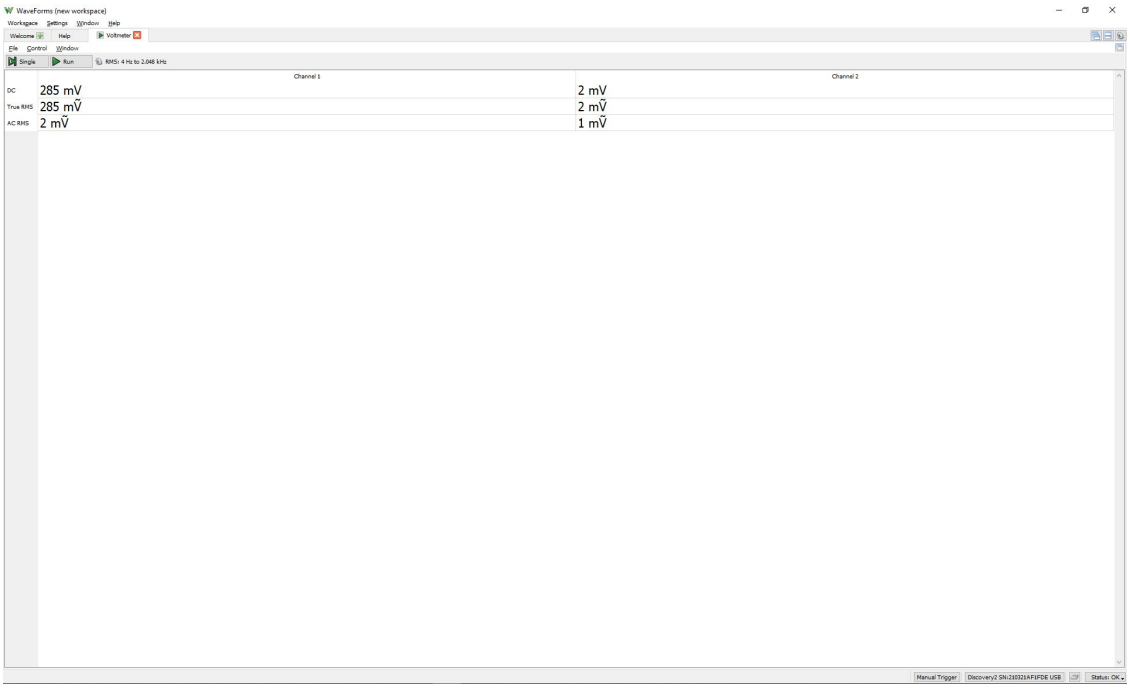

Fig. 9 Voltage reading of RA3 pin for LM35 voltage

16

While measuring the voltage from pin RA3, there was approximately ± 2 mV of noise during the measurements. This converts to ±0.2 degrees Celsius variation in the measurements. One solution that could be used to mitigate this noise would be to use a pull-up or pull-down resistor on the board, or to ground the board itself to an external ground so the board no longer has a floating ground.

17

According to page 485 of the data sheet for the PIC, the output current limit for PORTB is 25 mA and the output limit for PORTG is 2 mA. This implies that if we wanted to use the PIC to drive a typical LED that requires 20 mA, it would require the LED to be driven from specific ports, for example only off of PORTB and not PORTG.