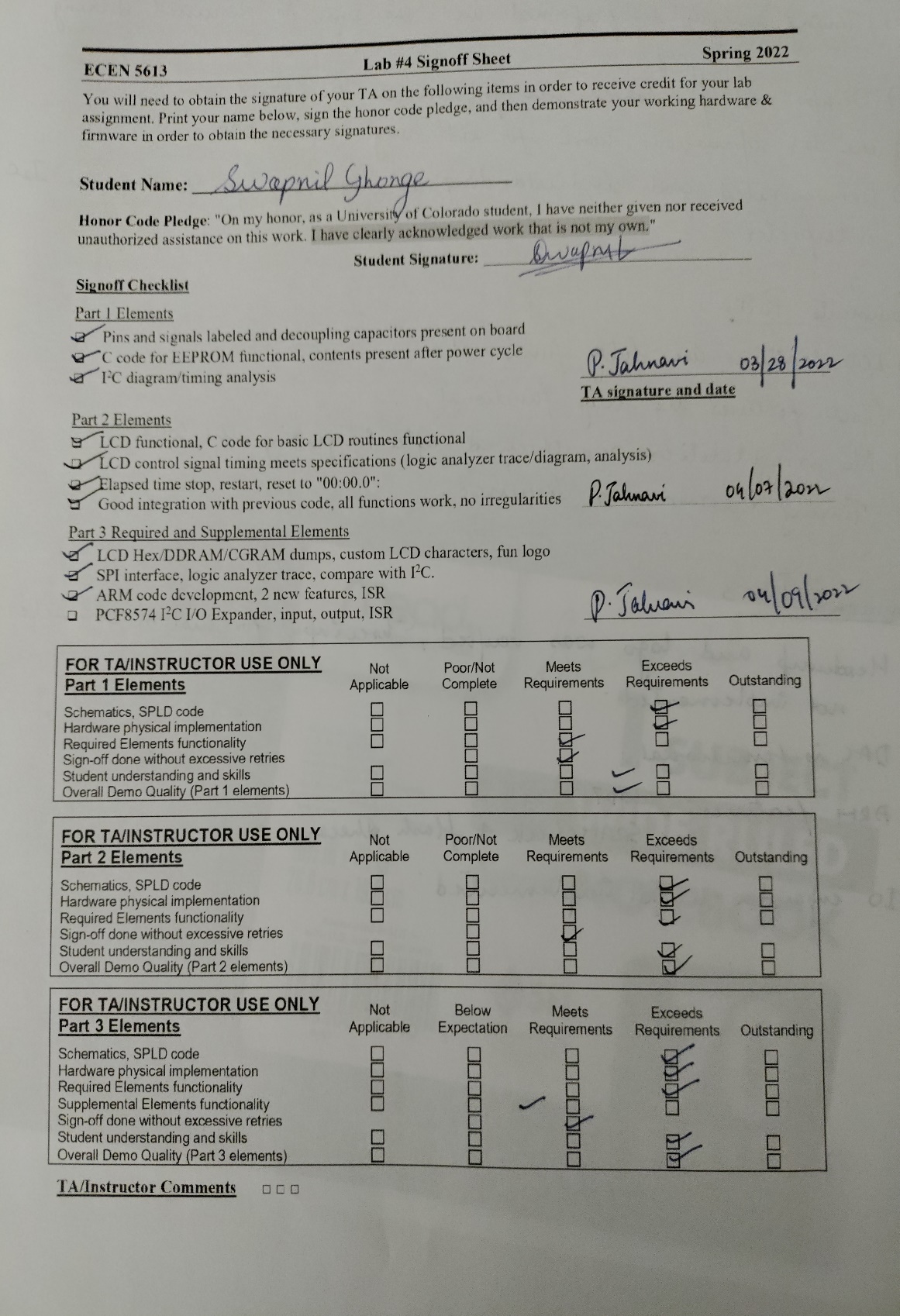
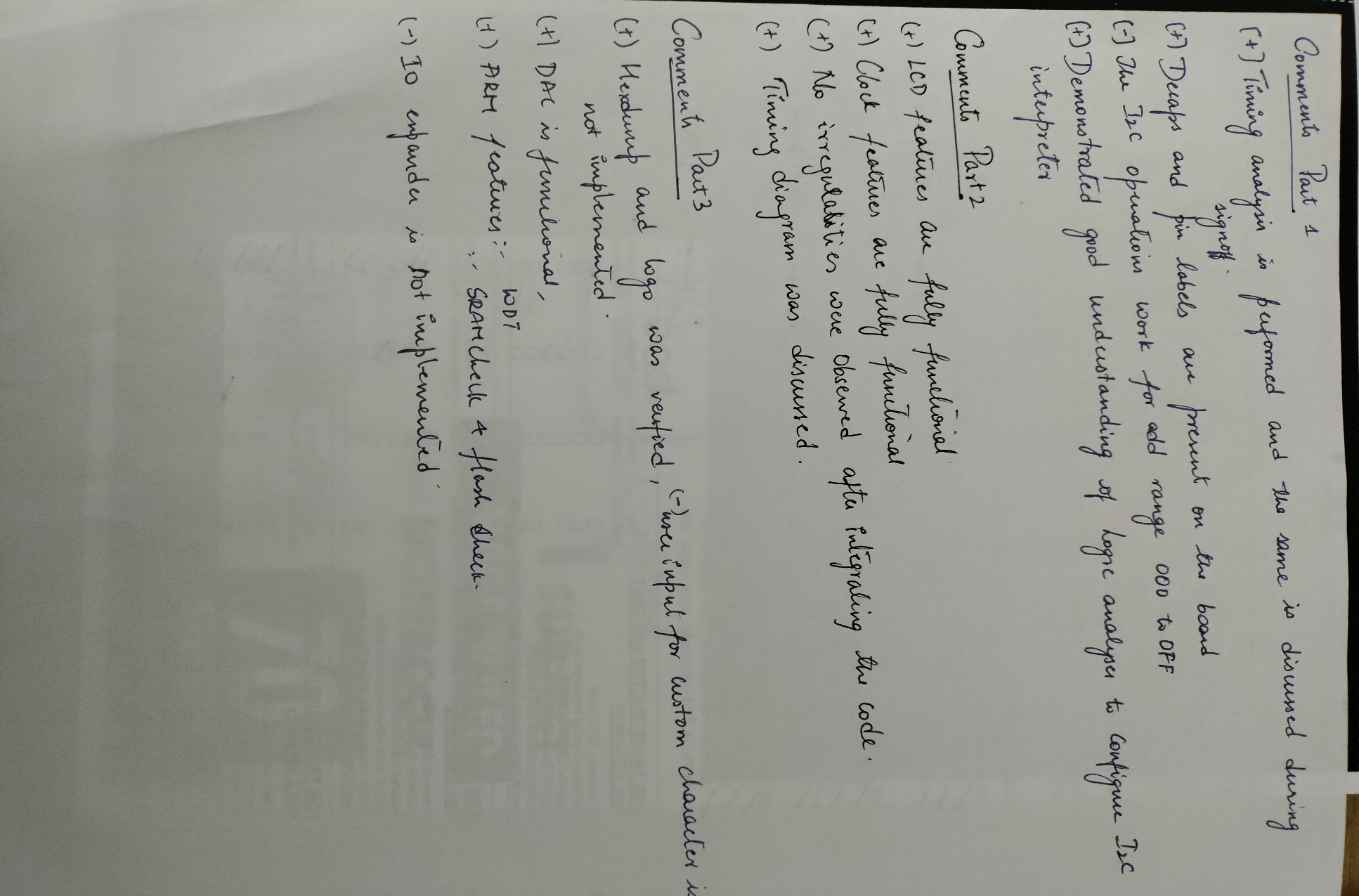
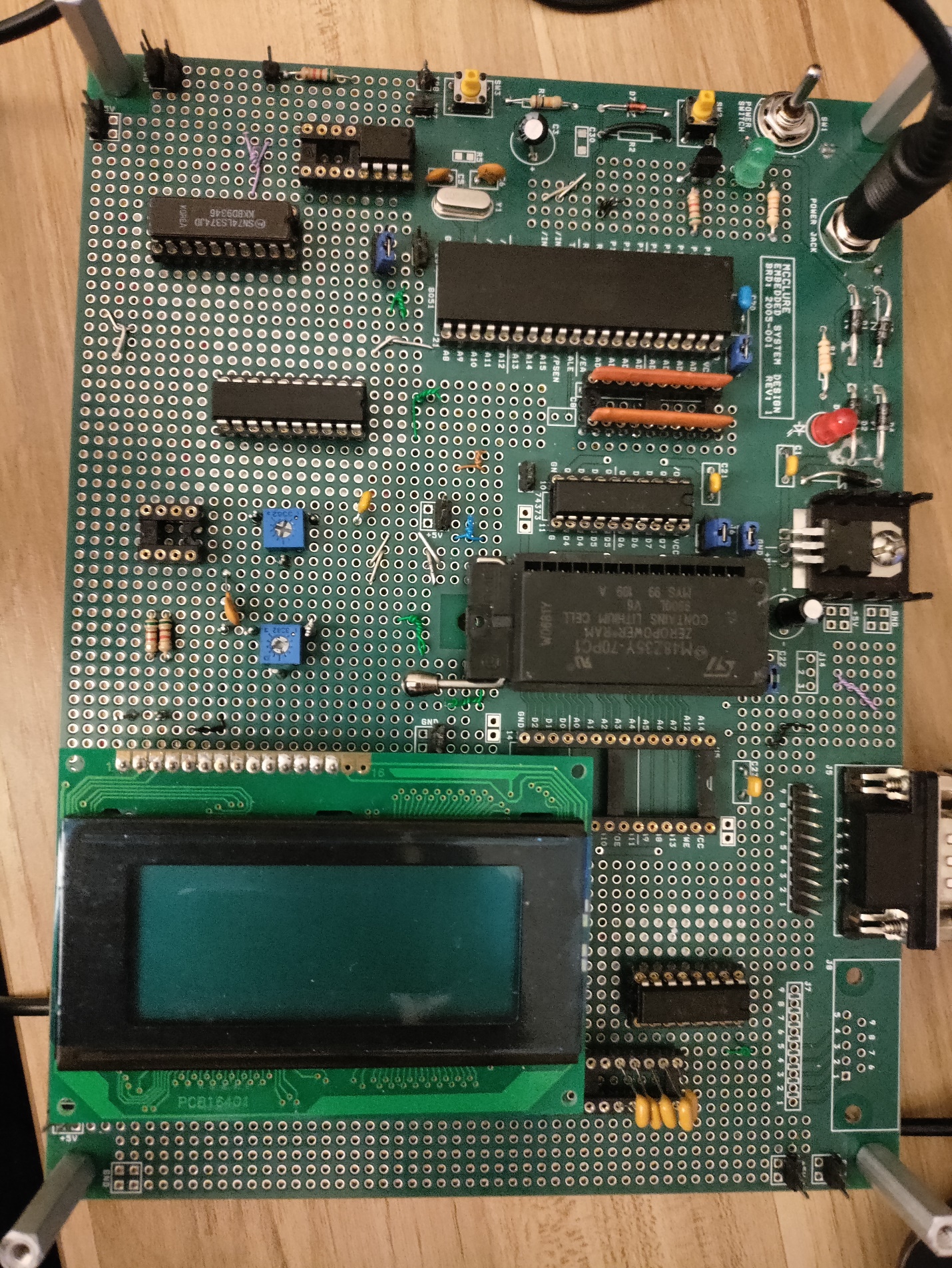
**ESD LAB 4 REPORT**

**Sign-off Sheet:**

****

****

**Board Top**

****

**Board Bottom**

****

**Part-1**

Brief: The of Lab 4 Part 1 was to write I2C driver with the ability to bit-bang write and read a byte at any EEPROM I2C address using function calls from C.

In this part of lab I implemented an I2C device driver EEPROM IC NM24C04. Where I connected the SCL and SDA to the pins of AT89C51RC2 microcontroller IC.

To implement bit bang technique I deviced 2 function, so that I could read, write the data void weeprom (char page, char address, char datum); char reeprom (char page, char address);

Then I programmed about the acknowledge bit after we send the address and the data.

Key Learning:

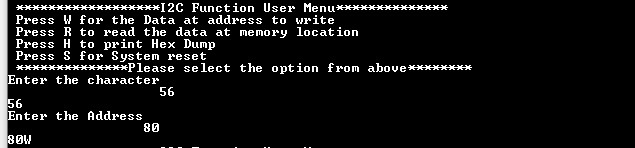
Was able to Write I2C driver and perform the bit bang technique.

At every clock cycle which is connected to GPIO, 1 bit of data was sent.

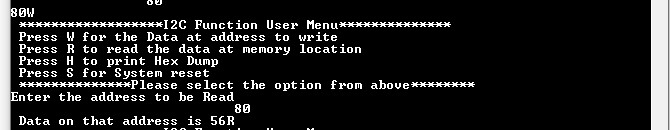
Challenges faced:

To debug the master and slave was most difficult task, for which I used a logic analyzer to debug my code. I also faced a lot of difficulty in receiving the acknowledge bit which was hard to debug, for which I made some changes in my code which led to accurate value of acknowledge from the slave.

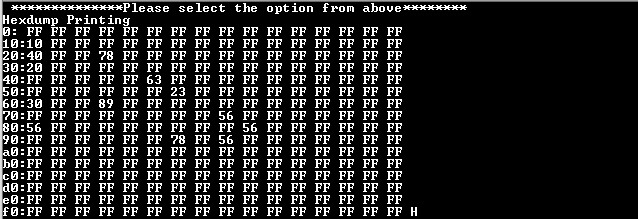
Below are the screenshots of the implementation of I2C using bit banging technique.



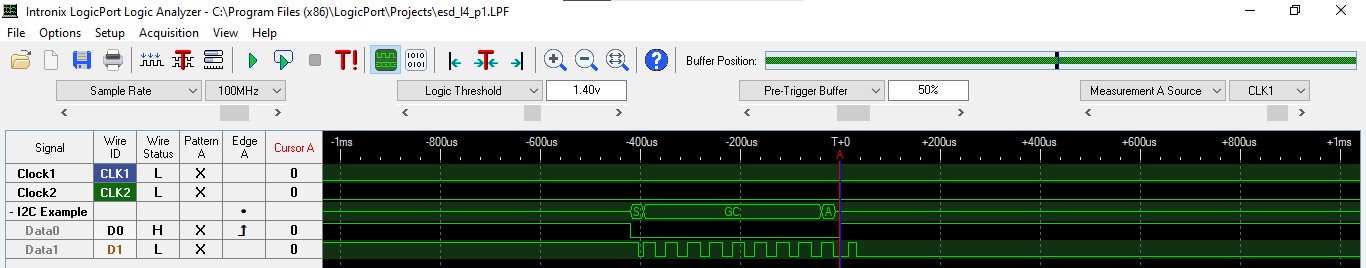
User menu on Serial Communication to write a character at specific location.



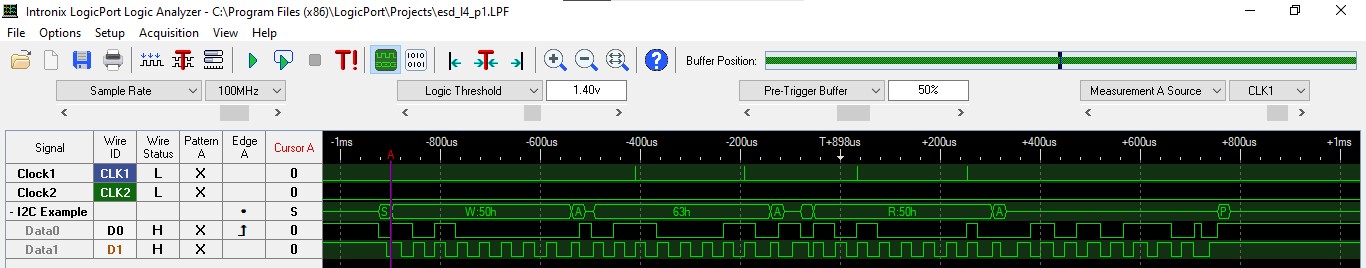
Reading the same written value on address 0x80h



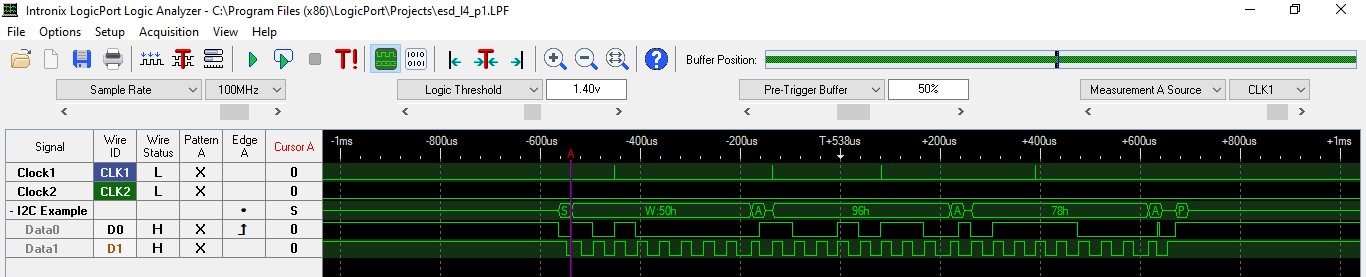
Printing the dump of values of memory location using Hex dump function.



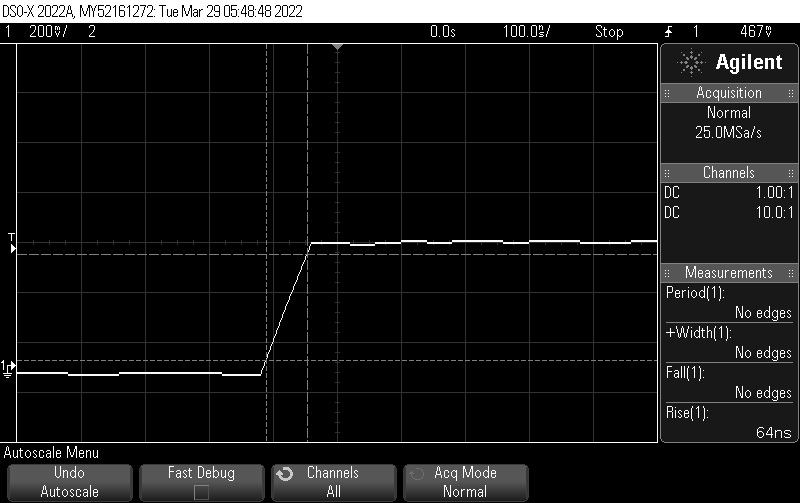
Resetting the EEPROM and data seen on Logic Analyzer



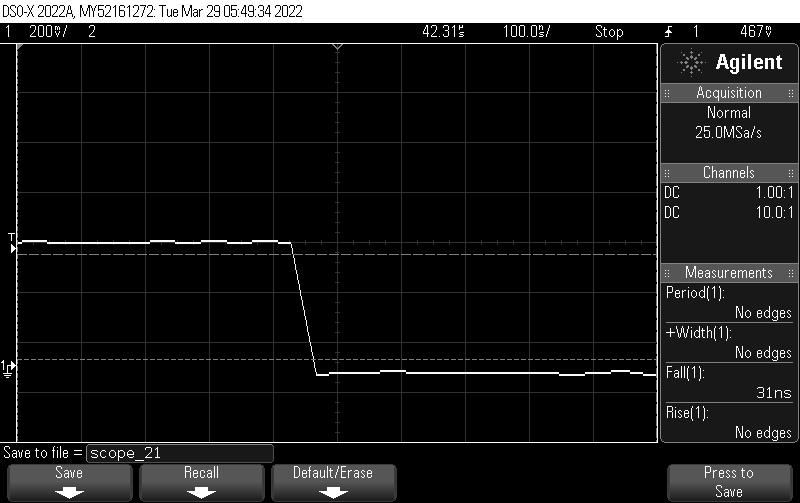
Writing value 60h at 50h memory location.



Reading the same 60h value at 50h memory location.



Rising time of the EEPROM IC using Oscillloscope: 64ns



Falling time of the EEPROM IC using Oscillloscope: 31ns

**PART-2:**

Brief: The objective of this part was to Interface LCD HD44780U, to integrate the LCD with AT89C51RC2 microcontroller and display string, character, time elapsed, restart time, stop time.

To integrate the LCD with AT89C51RC2 I wire wrapped the DB0- DB7 pins of LCD to AD0-AD7 pin of the microcontroller. The Register select pin was connected to P1.4 and Register Write P1.5 GPIO pins. To initialize the LCD the address pointer at 8000h. The lcd init function uses 0x30h value with a delay of 150ms while continuously checking the LCD is busy or not. Now here the RS and RW pins are set as low. To turn the cursor-on the LCD is checked of Buffer flag is zero.

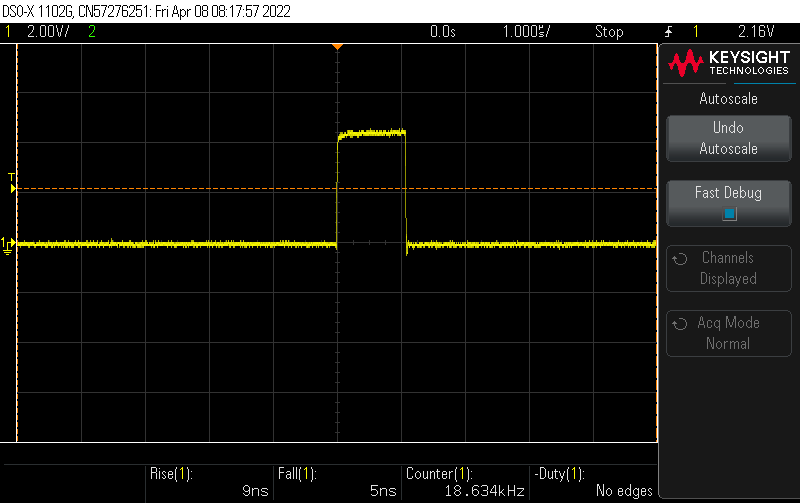
Key Learning:

I was able to successfully to initialize the LCD with writing characters on LCD, writing strings on LCD at any coordinates. I was able to initialize the timer in LCD and by the help of putchar() function I was able to print string on LCD. I was able to print to timer with millisecond and second and minute.

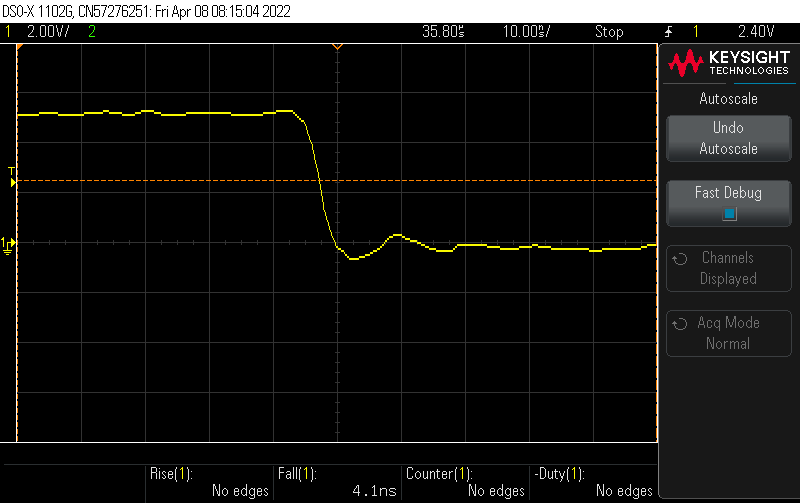
Challenges:

The biggest challenges I faced was contrast of the LCD was not optimum. I started debugging the code, but I found out that the program was correct and according to datasheet. I checked connection according to datasheet which were proper, Then I shifted my attention to working of potentiometer. The potentiometer when powered on the resistance was not changing, to I took help for TA Maanas and I changed the potentiometer. After tweaking the potentiometer a bit I was able to some brightness on LCD.

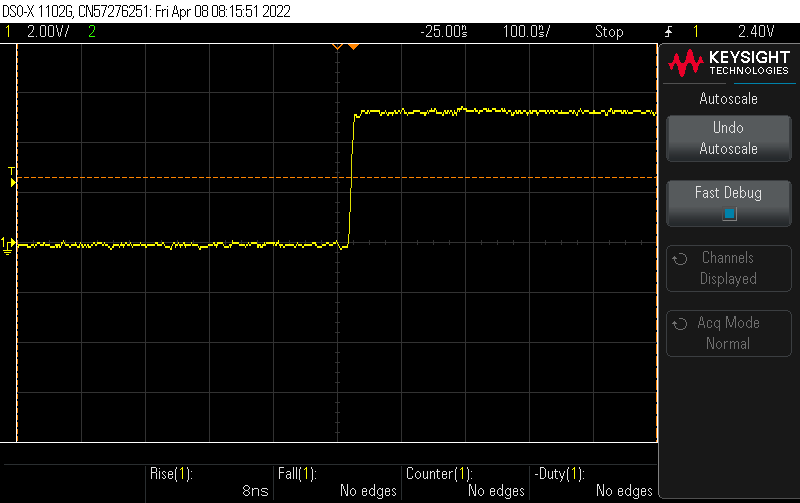
Below are the Screenshots of the reading I took



Rise time and Fall time of the Enable.



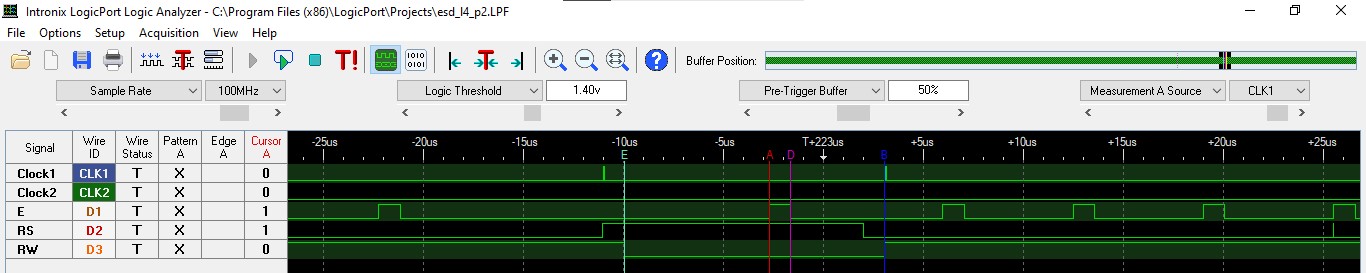
Register Select RS fall time: 4.1ns



Register Select RS rise time: 4.1ns



Register Write RW fall time: 6ns



Logic Analyzer Enable/RS/RW following the timing analysis according to datasheet

**PART-3**

Brief: LCD DDRAM and displays it in hex on the PC screen in a clean and logical format. Also, LCD characters and print fun logo. LCD HexDump the values of DDRAM, CGRAM. Interface SPI with MCP4802(DAC) and compare with I2C. Develop ARM code of two features, the new I implemented was watch-dog timer, SRAM check and Flash Check and blink the LED.

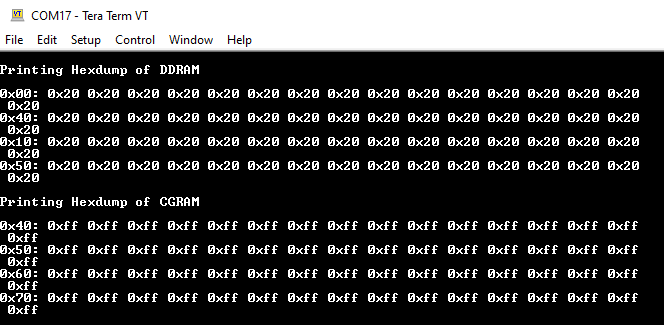
Key Learning:

I successfully created the Fun logo of a character of ‘CU’ by writing the values on CGRAM at address starting from 0x40h. I learnt how to write the values in CG RAM and DDRAM and integrating with putchar function. I also learn to integrate timer with LCD and successfully integrated the timer interrupt with LCD. I also learnt about how to configure the DAC through SPCON register and sending message and converting to voltage, I sent the valye 0x7450h and converted voltage I got was 0.81 volts according to the calculation Vo = Vin/2^N. For the ARM code development I chose to learn more about watchdog timer and its operation. Secondly in ARM code, I also learnt how to check the code spaces in SRAM and Flash drive and blink an LED when one of the memory is accessed while checking.

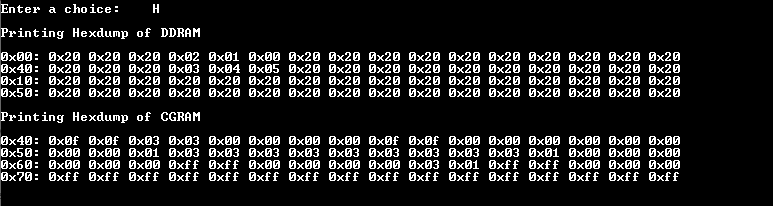
Challenges:

The challenges I faced configuring the correct value to initialize the DAC by SPCON register. I took help from classmates and learnt how to configure the DAC correctly. The next challenge I faced was I was not receiving data on Logic Analyzer, to which I checked my code how the value progressed. Another challenge I faced was printing the value of Hexdump of CGRAM, I was not able to configure it correctly, I took help from my PES lecture and tried to follow the steps to print Hexdump and I successfully implemented the hexdump.

Below are screenshots of the part 3 of Lab 4



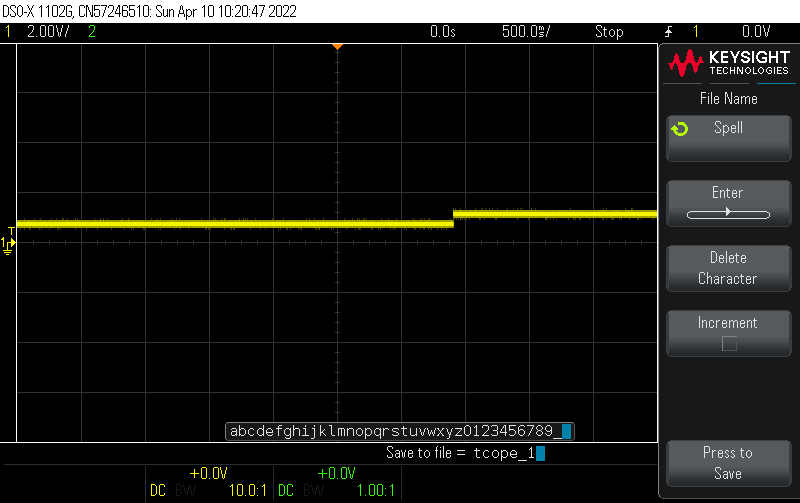
Printing of HexDump when NO value is written.



Printing the HexDump when after printing the CU Logo.



Printing HexDump values when I pass String in the DDRAM



Output of DAC as 0.8 volts as discussed above