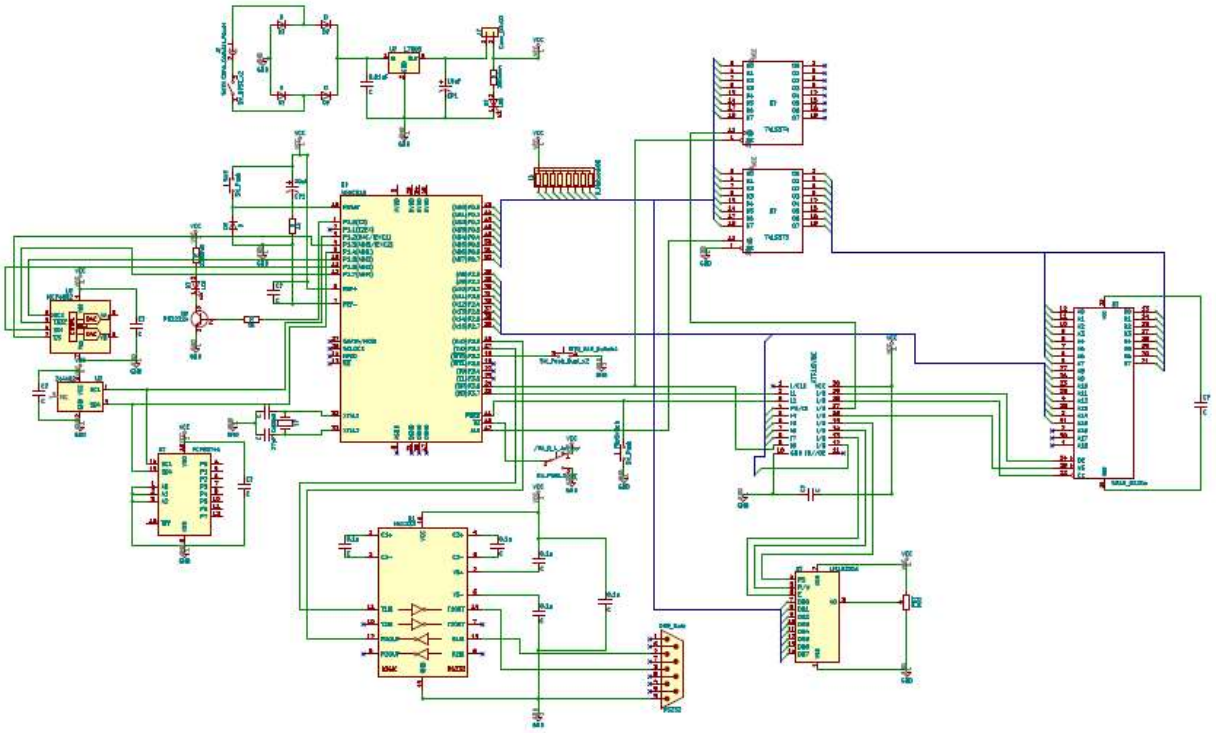
Monish Nene

ESD LAB4 Report

Schematic



Questions

a) What operating system (including revision) did you use for your code development?

Answer: Windows 10

b) What compiler (including revision) did you use?

Answer: SDCC 2.6.0

c) What exactly (include name/revision if appropriate) did you use to build your code (what IDE, make/makefile, or command line)?

Answer: IDE -> CodeBlocks for AT89C51 and Code Composer for MSP432

d) Did you install and use any other software tools to complete your lab assignment?

Answer: Tera Term for ARM

e) Did you experience any problems with any of the software tools? If so, describe the problem

Answer: Yes,

1) The interrupt handler of AT89C51 can't access global variables. The Compiler doesn't throw an error for it.
2) The WinCUPL doesn't throw a compiling error if the name defined in the input/output differs with the logic but the spld code doesn't work for that particular pin. (ENABLE v/s Enable was my case)

Things Learnt in Lab 3

Wrote my own I2C driver

Wrote my own SPI driver

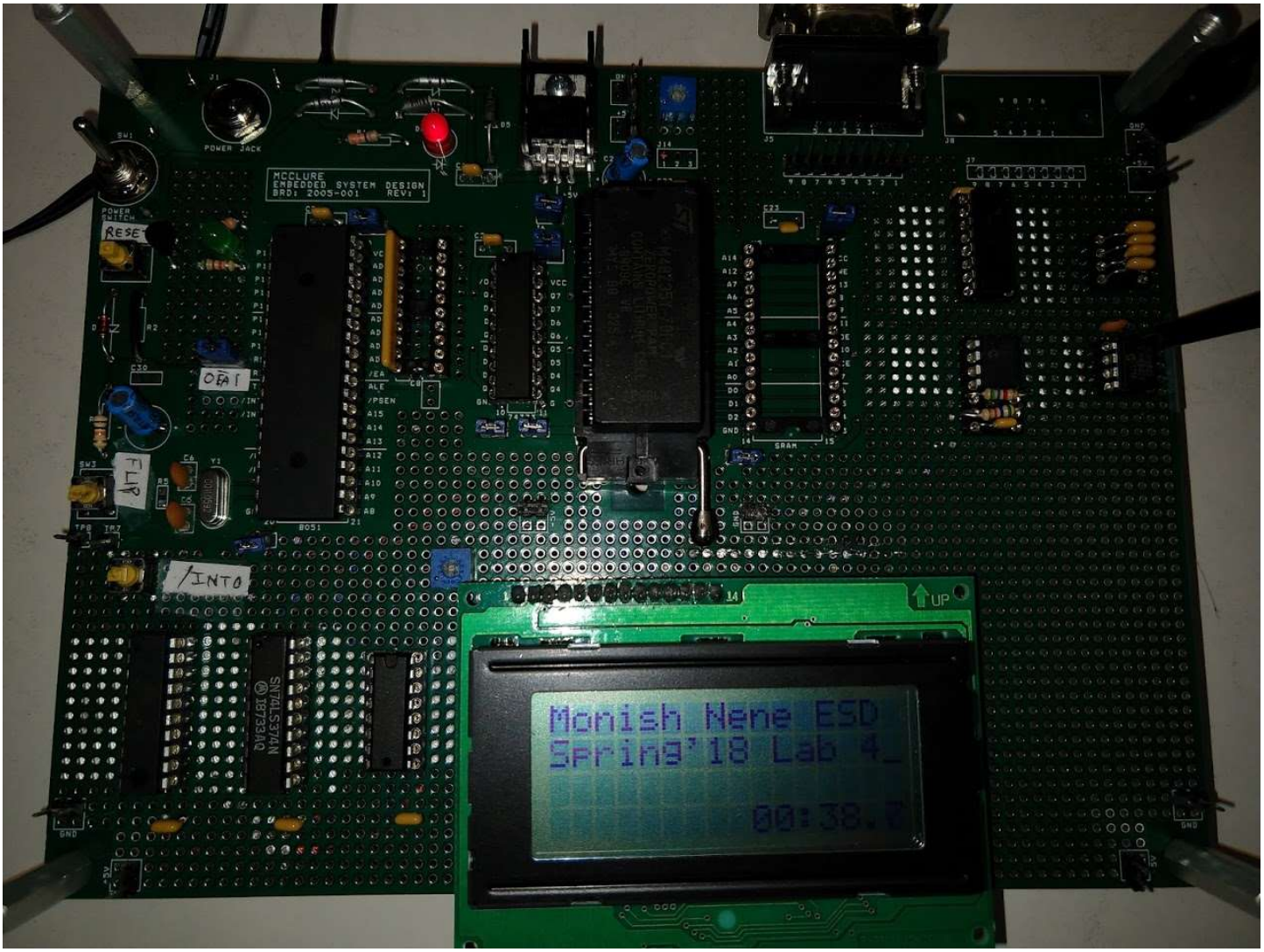
LCD interface

Interrupt handling for AT89C51

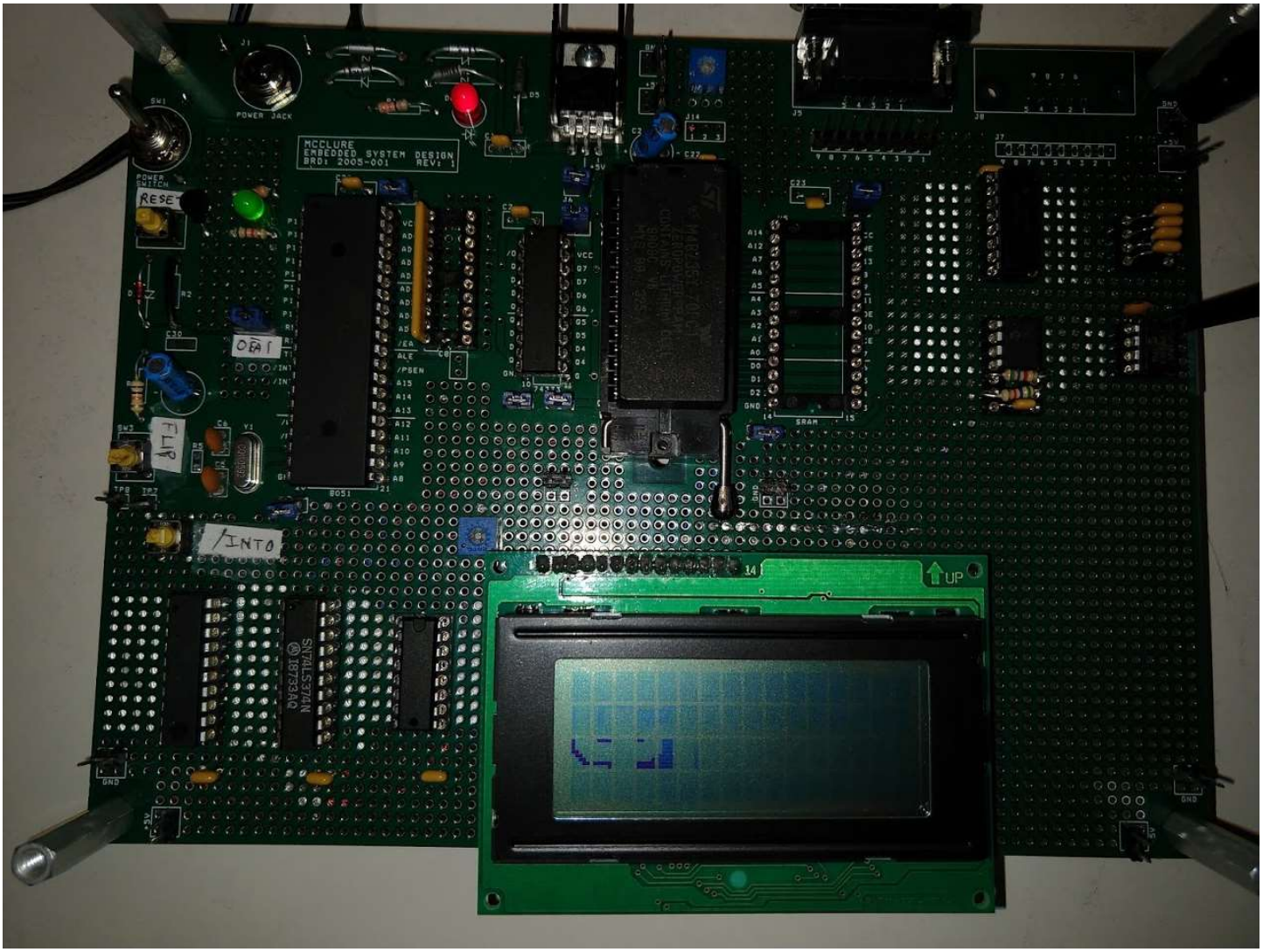
Toughest Thing to do

Making LCD work properly with RTC clock updates

Board



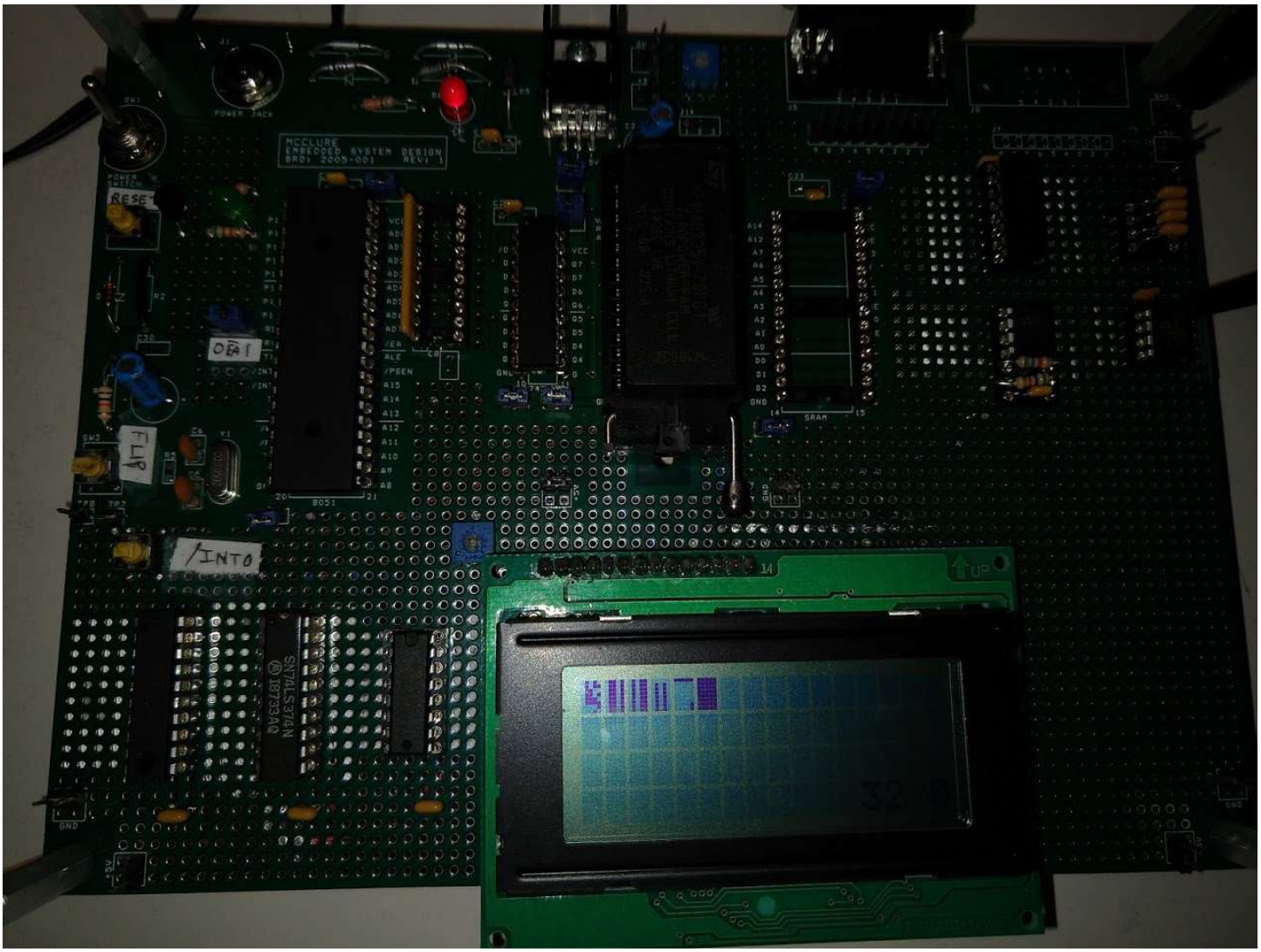
CU Boulder Animated Logo



Screenshots

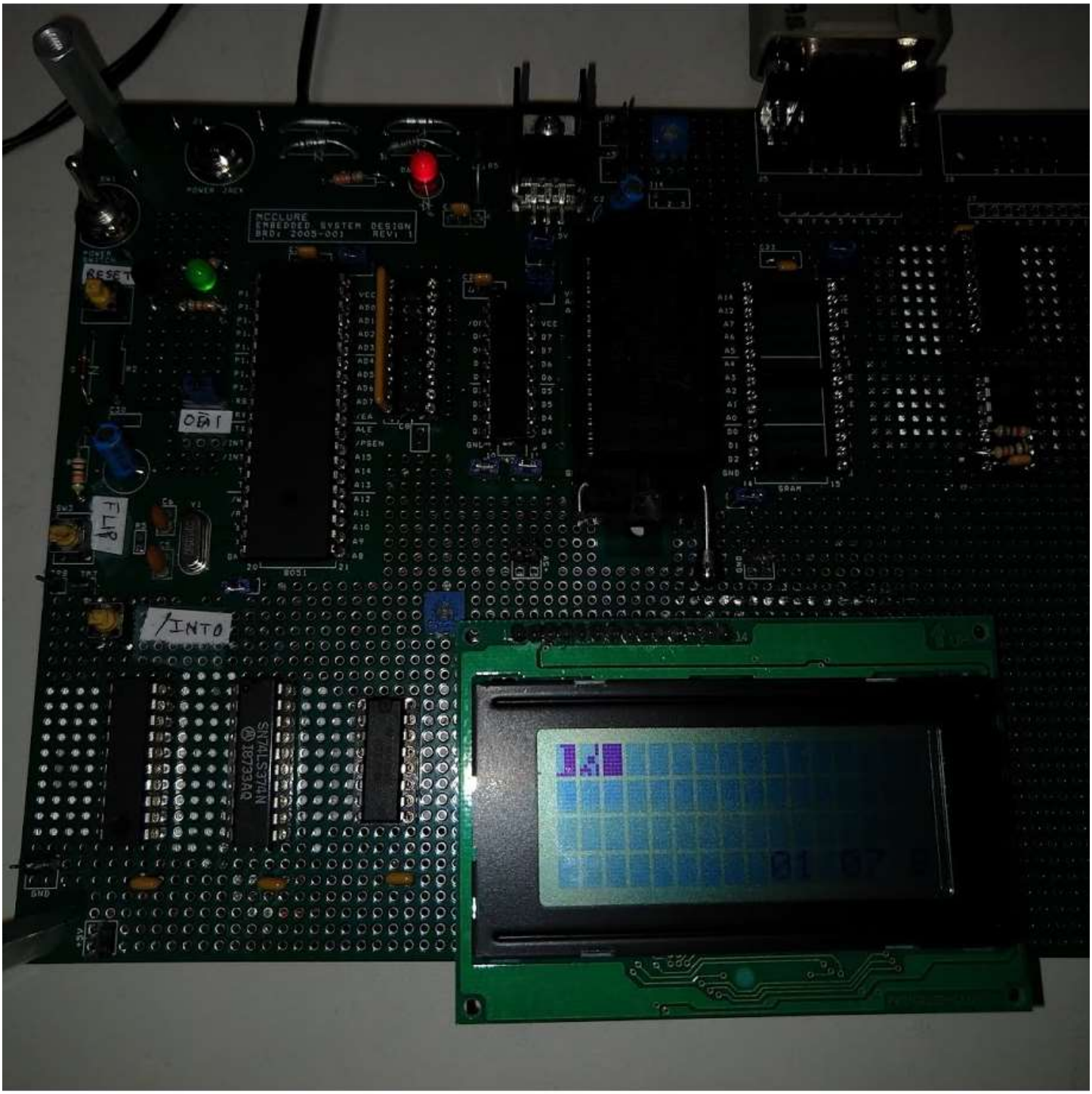
Custom Character

```
Monish Nene ESD Spring 2018 Lab 4 Required
Press  'Backspace'-> Clear LCD ,'X'-> Jump Co-ordinates, 'D'->LCD DDRAM DUMP, 'G'-> LCD CGRAM DUMP,
'B'->CU Boulder Logo, 'C'->Custom Character Builder, 'W'->Write Data I2C, 'R'->Read Data I2C, 'I'-> EEPROM DUMP
'T'->Start or Stop Time, 'Y'-> Reset Time, 'P'-> Write to io expander Port, 'Q'-> Read from io expander port, '?'-> Display Menu
Custom Character mode
How many custom characters do you want to create?(upto 8)4
Enter base of your data entry16
Character 1
Row 0A
Row 1B
Row 2C
Row 3D
Row 4E
Row 5F
Row 61
Row 72
Character 2
Row 0D
Row 1D
Row 2D
Row 3D
Row 4D
Row 5D
Row 6D
Row 7D
Character 3
Row 0B
Row 1B
Row 2B
Row 3B
Row 4B
Row 5B
Row 6B
Row 7B
Character 4
Row 0
Row 1AA
Row 2A
Row 3A
Row 4A
Row 5A
Row 6A
Row 7A
```


MCCLURE
EMBEDDED SYSTEM DESIGN
BRD: 2003-001 REV: 1
POWER JACK
POWER SWITCH
RESET
OE1
CLK
/INT0
8051
SRAM
SN74ALS138AD
UP

CGRAM Hexdump and DDRAM Hexdump

```
Custom Character mode
How many custom characters do you want to create?(upto 8)2
Enter base of your data entry2
Character 1
Row 01111
Row 1111
Row 2111
Row 3111
Row 4111
Row 5111
Row 6111
Row 711111
Character 2
Row 0001
Row 1001
Row 2001
Row 3110
Row 4110
Row 51001
Row 611101
Row 710101
CGRAM Hexdump
00: 0F 27 47 67 87 A7 C7 FF 01 21 41 66 86 A9 DD F5
10: 0B 2B 4B 6B 8B AB CB EB 00 AA 4A 6A 8A AA CA EA
20: 0F 3E 5E 7E 9E BE CE EE 1F 20 40 60 80 A0 C1 E1
30: 18 28 48 68 88 A8 D8 E0 FF FF FF FF FF FF FF FF
DDRAM Hexdump
80: 00 01 20 20 20 20 20 20 20 20 20 20 20 20 20 20
90: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
A0: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
B0: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
C0: 20 30 32 20 33 30 20 31 20 20 20 20 20 20 20 20
```


MCCLURE
EMBEDDED SYSTEM DESIGN
BRD: 2005-001 REV: 1
RESET
/INT0
8051
GRAM

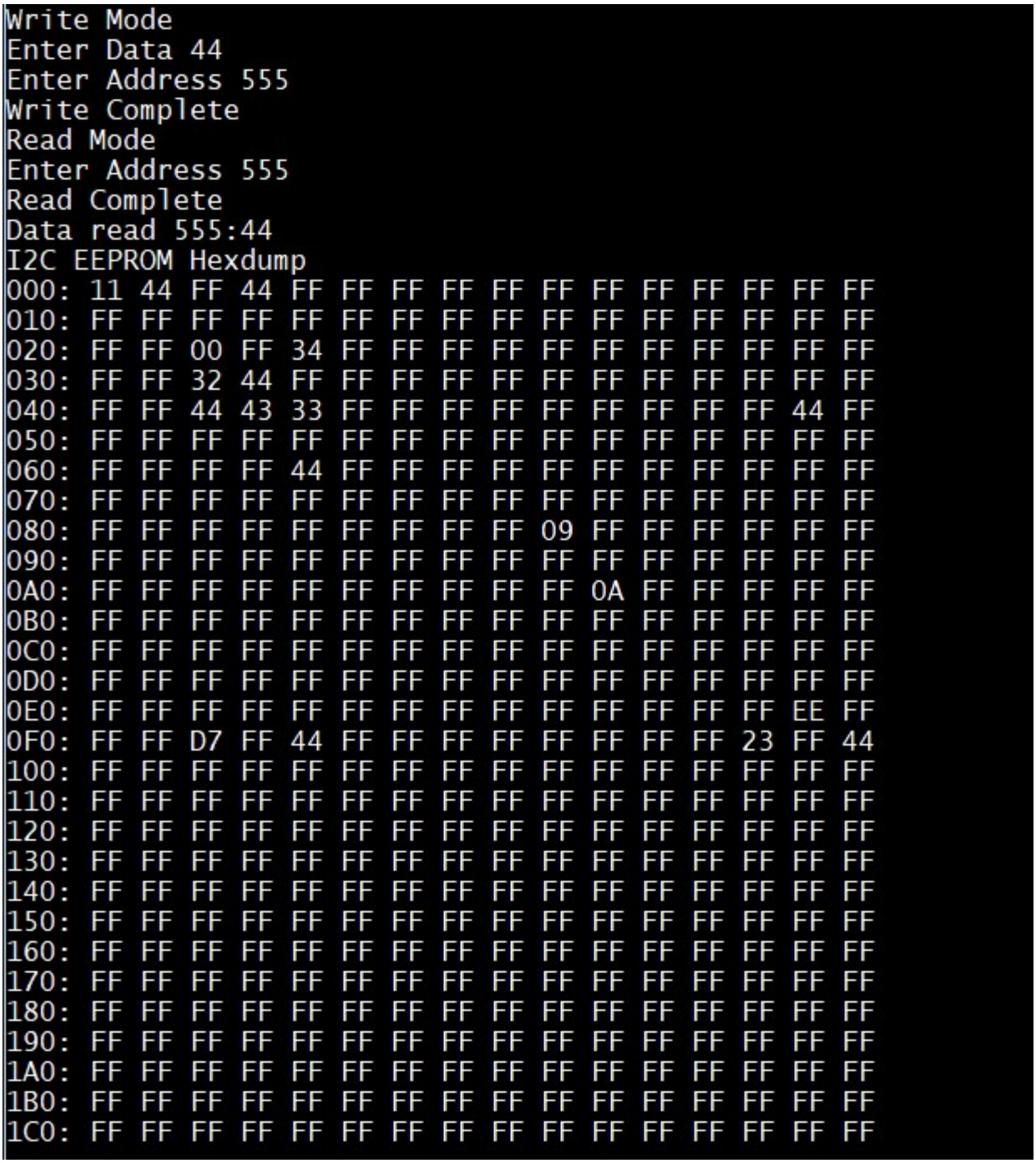
IO EXPANDER Interface (1's Complement P0-P6 if P7 low detected)

```
Monish Nene ESD Spring 2018 Lab 4 Required
Press  'Backspace'-> Clear LCD ,'X'-> Jump Co-ordinates, 'D'->LCD DDRAM DUMP, 'G'-> LCD CGRAM DUMP,
'B'->CU Boulder Logo, 'C'->Custom Character Builder, 'W'->Write Data I2C, 'R'->Read Data I2C, 'I'-> EEPROM DUMP
'T'->Start or Stop Time, 'Y'-> Reset Time, 'P'-> Write to io expander Port, 'Q'-> Read from io expander port, '?'-> Display Menu
Read Complete
Data read 80
Write Mode
Enter Data 12
Write Complete
Read Complete
Data read ED
```

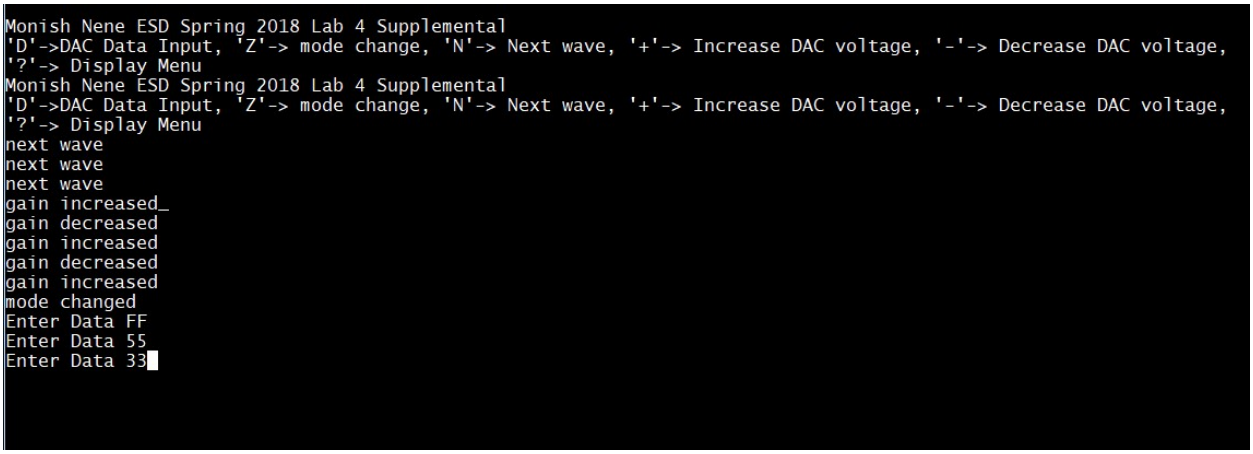
I2C HexDump

```
Write Mode
Enter Data AA
Enter Address 00A
Write Complete
Read Mode
Enter Address A
Read Complete
Data read 00A:AA
I2C EEPROM Hexdump
000: 11 44 FF 44 FF FF FF FF FF FF AA FF FF FF FF FF
010: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
020: FF FF 00 FF 34 FF FF FF FF FF FF FF FF FF FF FF
030: FF FF 32 44 FF FF FF FF FF FF FF FF FF FF FF FF
040: FF FF 44 43 33 FF FF FF FF FF FF FF FF FF 44 FF
050: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
060: FF FF FF FF 44 FF FF FF FF FF FF FF FF FF FF FF
070: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
080: FF FF FF FF FF FF FF FF FF 09 FF FF FF FF FF FF
090: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0A0: FF FF FF FF FF FF FF FF FF FF 0A FF FF FF FF FF
0B0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0C0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0D0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0E0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF EE FF
0F0: FF FF D7 FF 44 FF FF FF FF FF FF FF FF 23 FF 44
100: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
110: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
120: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
130: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
140: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
150: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
160: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

I2C HexDump

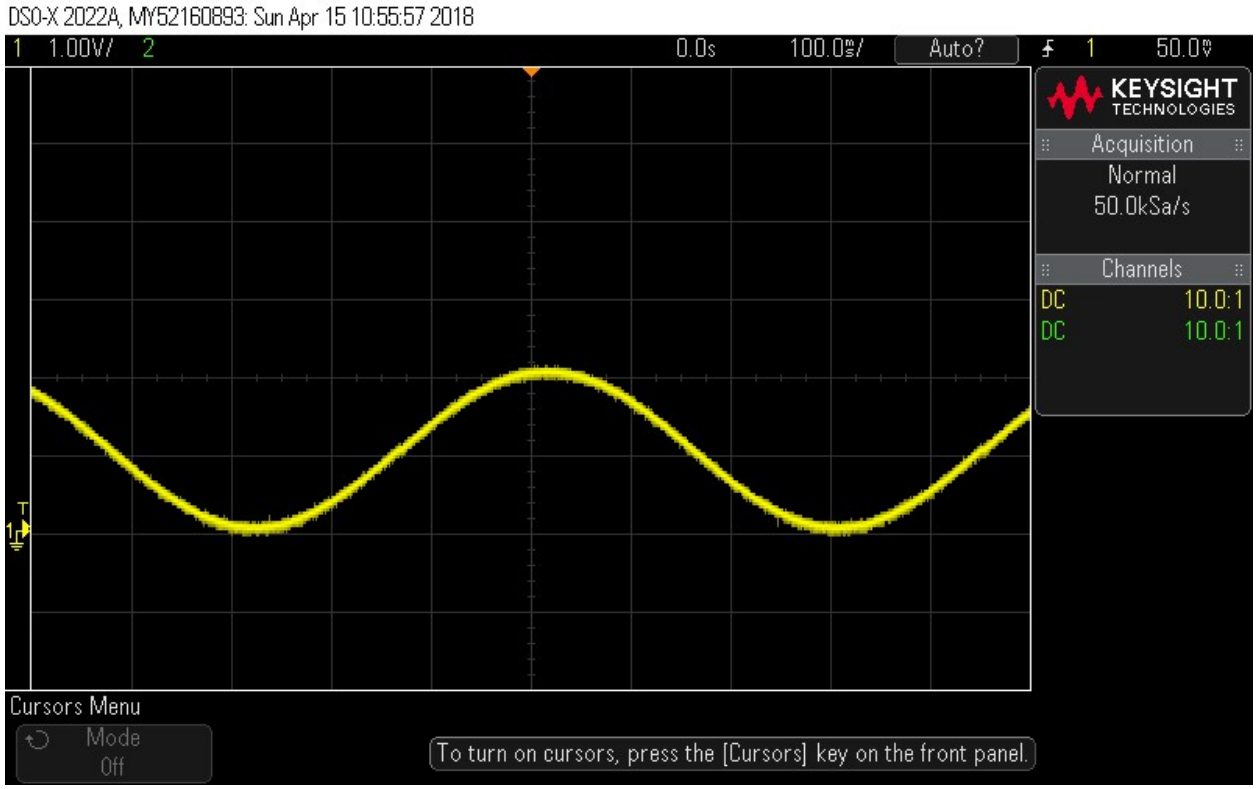

Write Mode
Enter Data 44
Enter Address 555
Write Complete
Read Mode
Enter Address 555
Read Complete
Data read 555:44
I2C EEPROM Hexdump
000: 11 44 FF 44 FF FF FF FF FF FF FF FF FF FF FF FF
010: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
020: FF FF 00 FF 34 FF FF FF FF FF FF FF FF FF FF FF
030: FF FF 32 44 FF FF FF FF FF FF FF FF FF FF FF FF
040: FF FF 44 43 33 FF FF FF FF FF FF FF FF FF 44 FF
050: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
060: FF FF FF FF 44 FF FF FF FF FF FF FF FF FF FF FF
070: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
080: FF FF FF FF FF FF FF FF FF 09 FF FF FF FF FF FF
090: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0A0: FF FF FF FF FF FF FF FF FF FF 0A FF FF FF FF FF
0B0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0C0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0D0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0E0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF EE FF
0F0: FF FF D7 FF 44 FF FF FF FF FF FF FF FF 23 FF 44
100: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
110: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
120: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
130: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
140: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
150: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
160: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
170: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
180: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
190: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1A0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1B0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1C0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

DAC control GUI

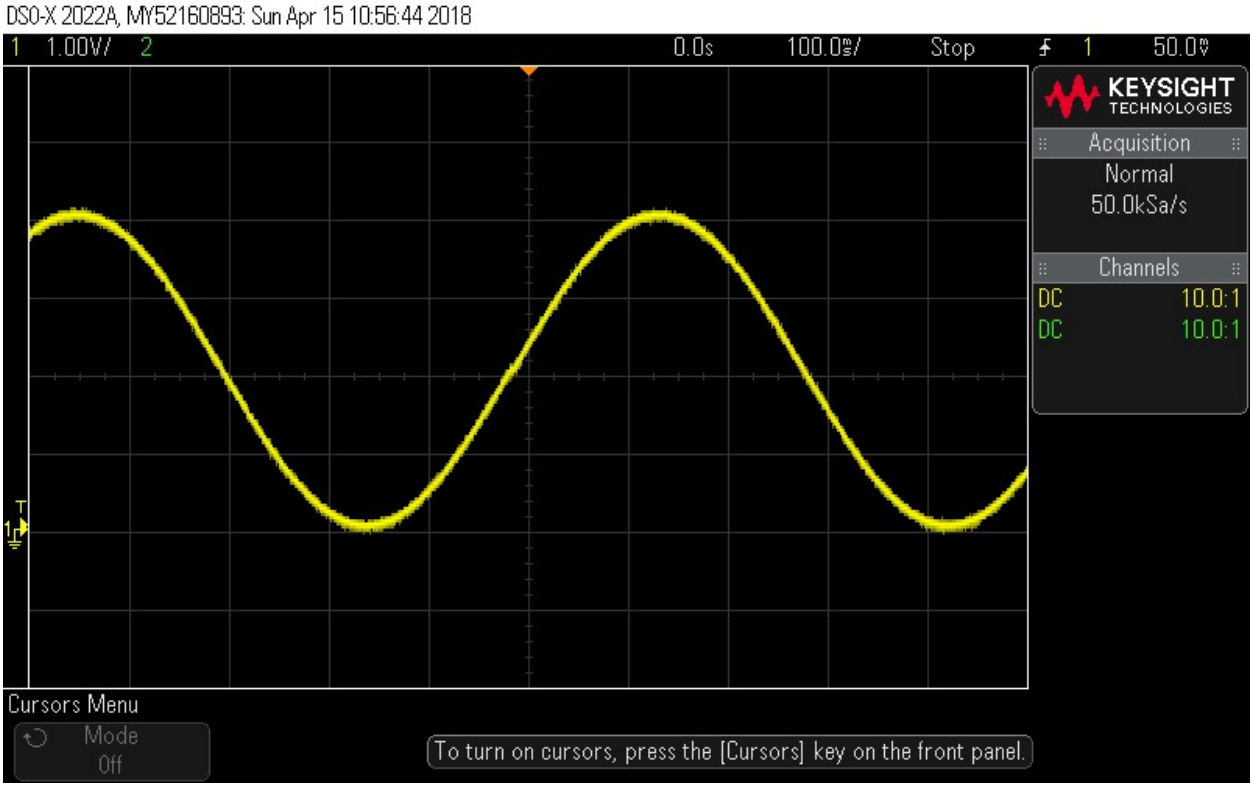


Sinewave DAC

Low gain

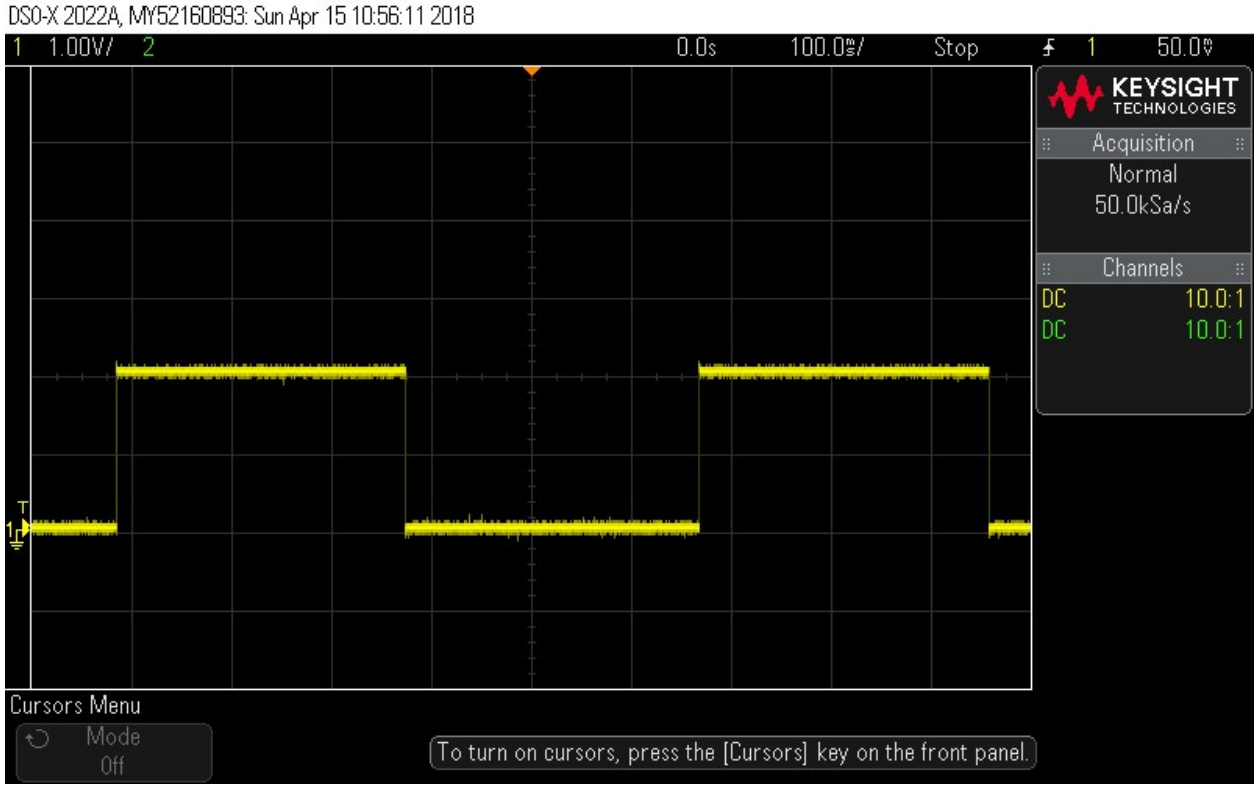


High gain

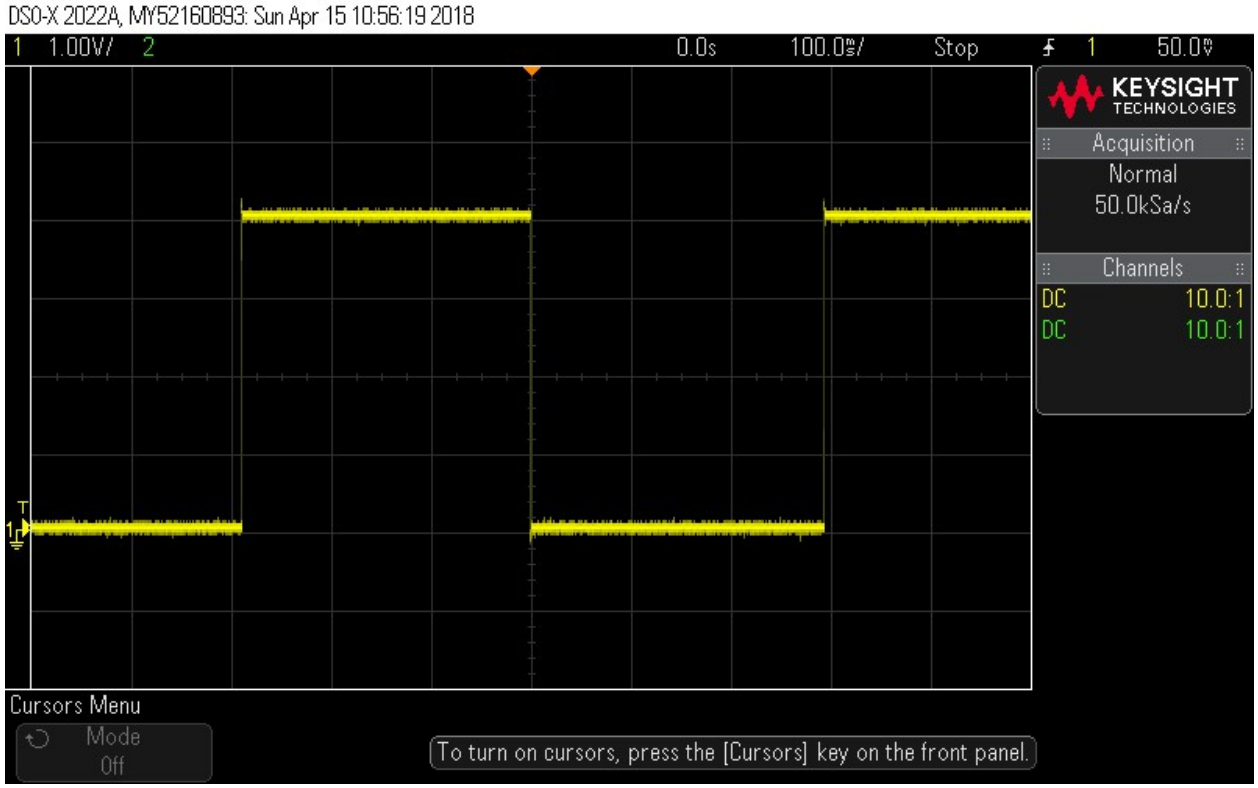


Squarewave DAC

Low gain

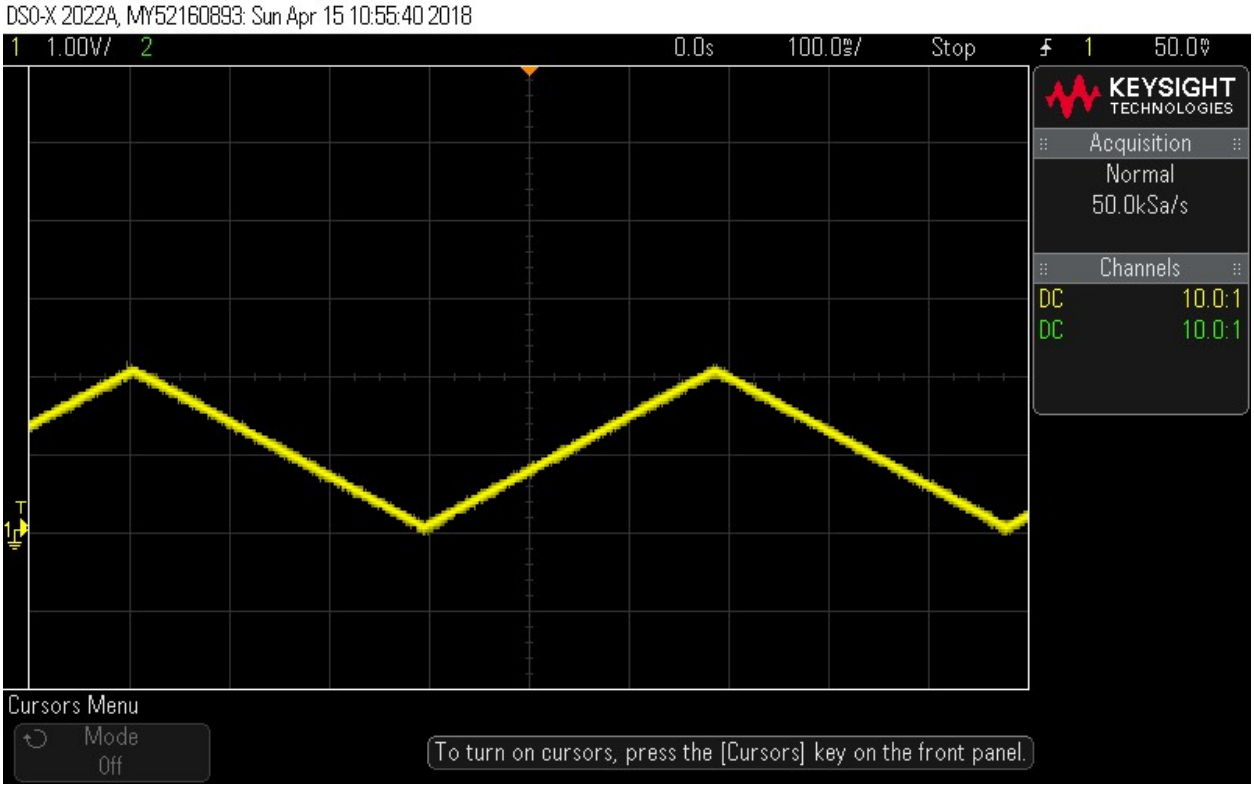


High gain

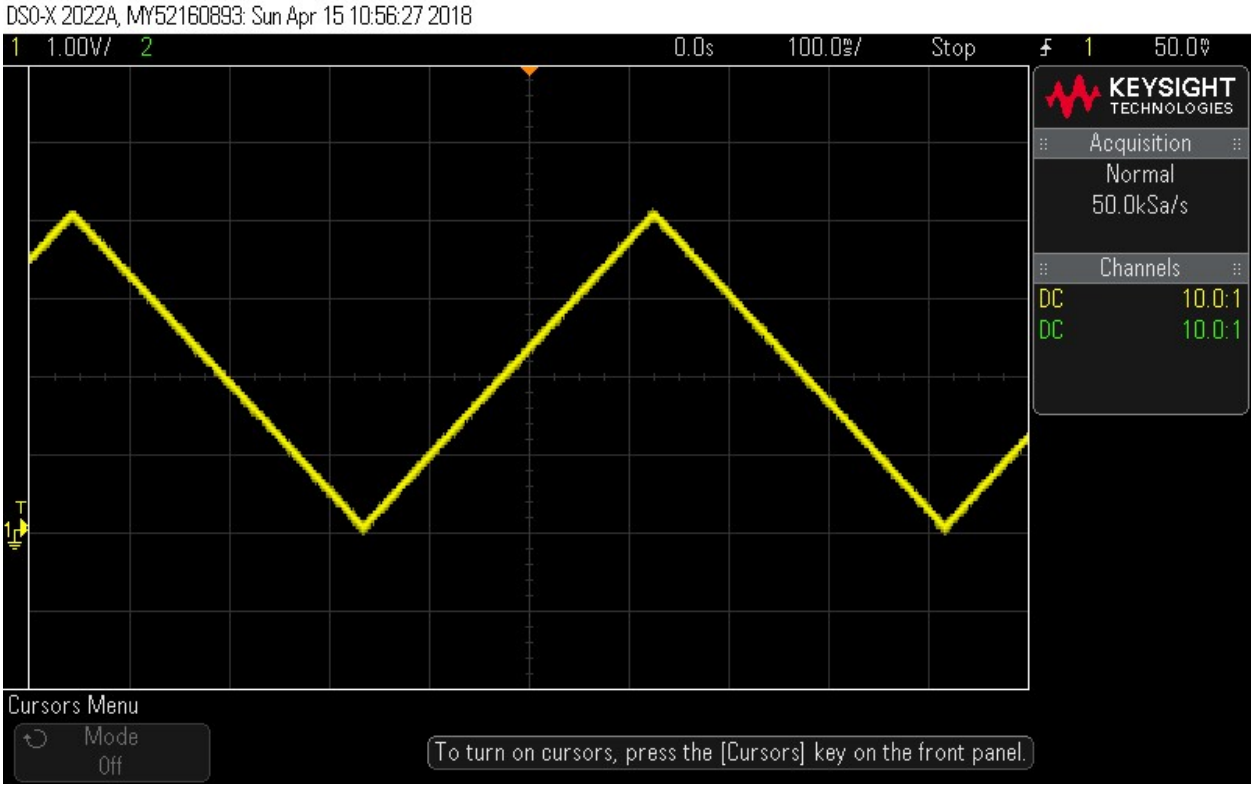


Triangular wave DAC

Low gain

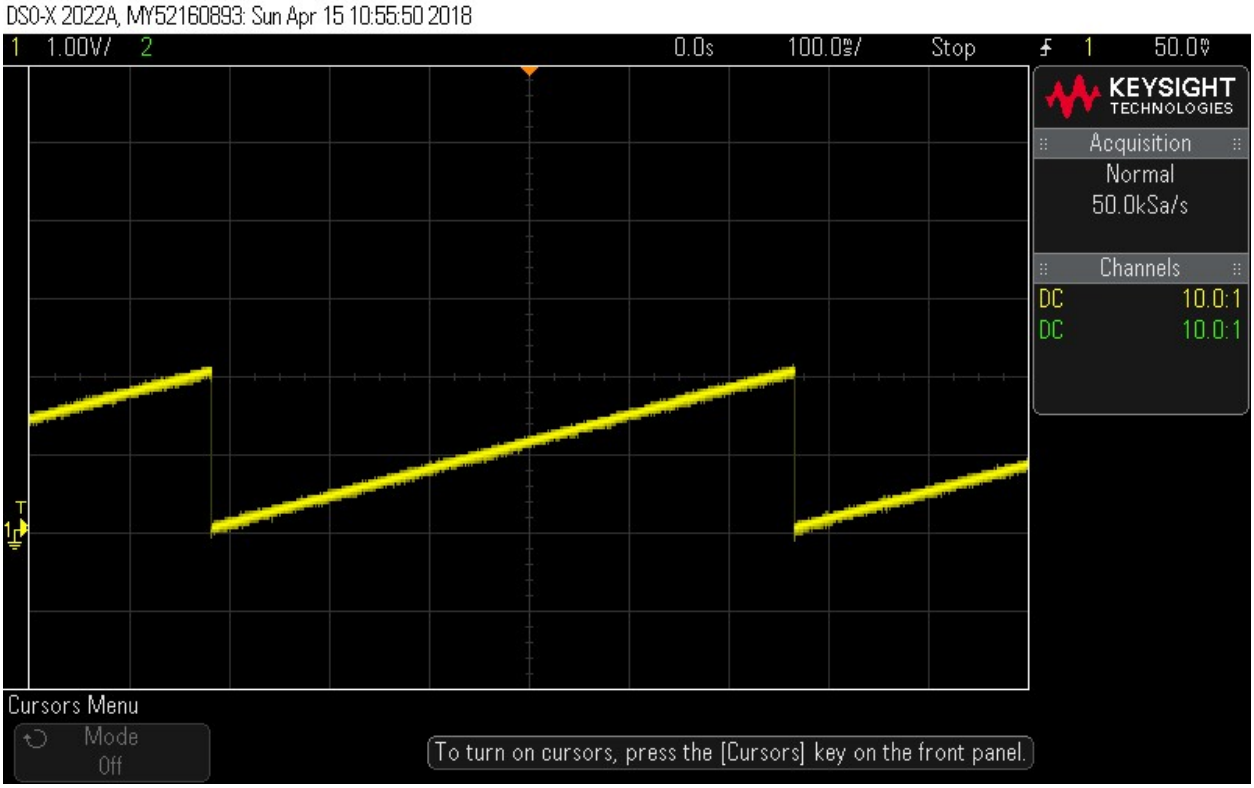


High gain

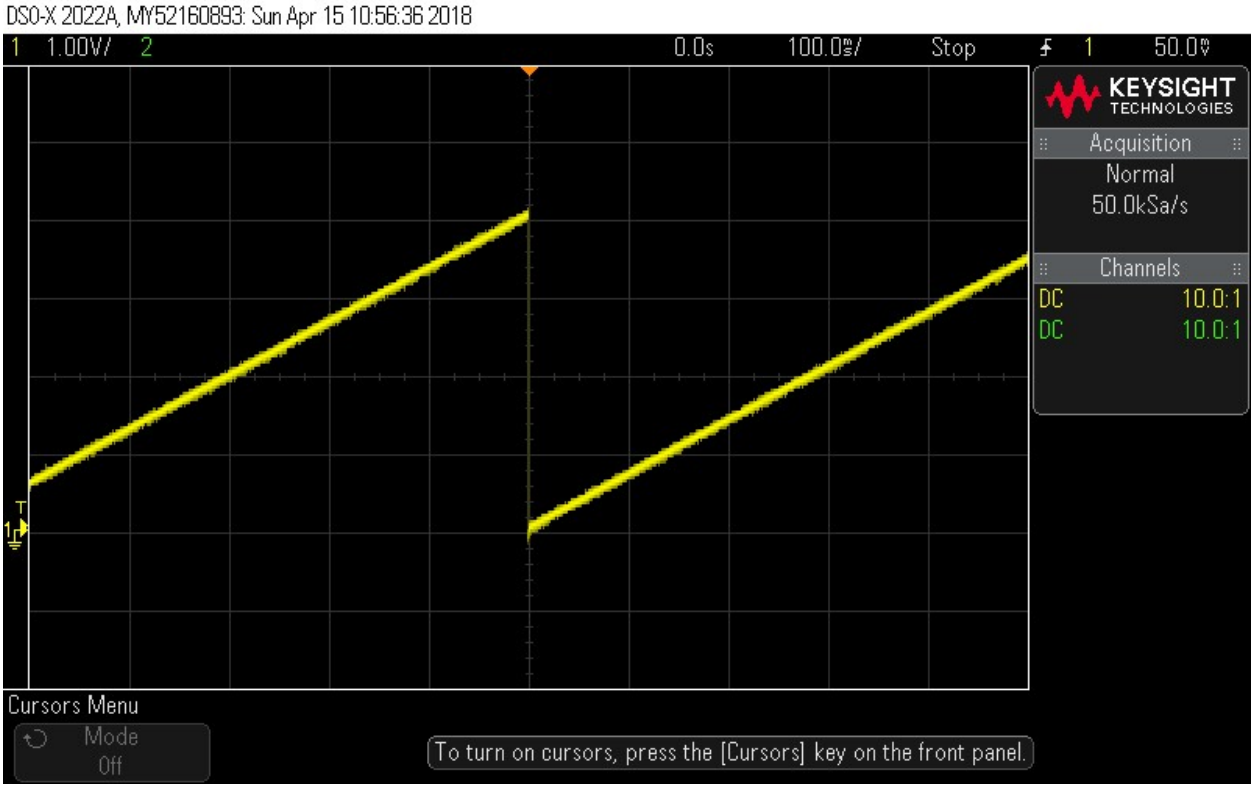


Sawtoothwave DAC

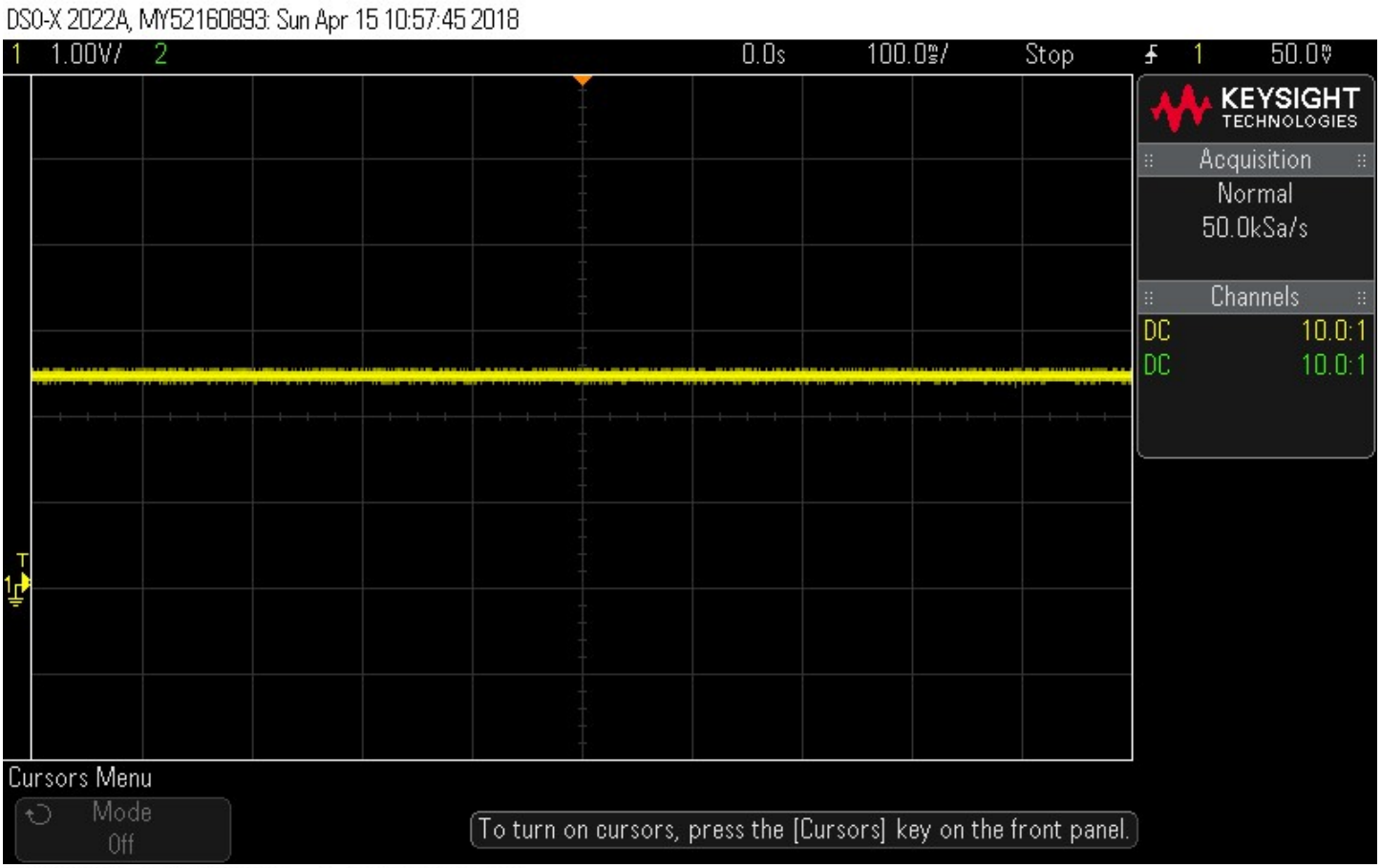
Low gain

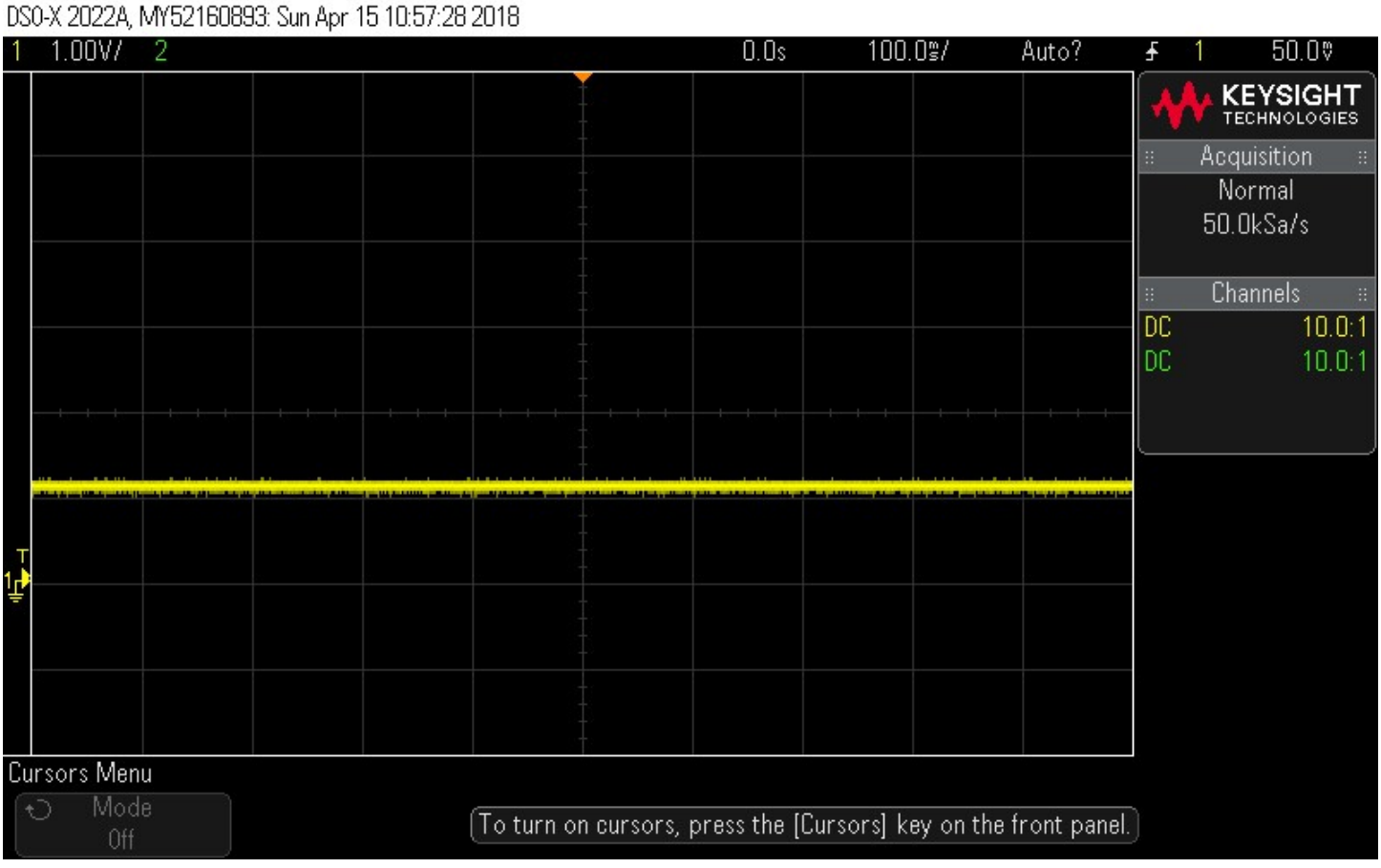


High gain



DAC Value Set Mode





ECEN 5613

Lab #4 Signoff Sheet

Spring 2018

You will need to obtain the signature of your TA on the following items in order to receive credit for your lab assignment. Signatures are due by **Friday, April 6, 2018 (Part 1 Required Elements) and Friday, April 13, 2018 (Part 2 Required/Supplemental/Challenge Elements).**

Print your name below, sign the honor code pledge, circle your course number, and then demonstrate your working hardware & firmware in order to obtain the necessary signatures.

Student Name: Monish H. Nene.

Honor Code Pledge: "On my honor, as a University of Colorado student, I have neither given nor received unauthorized assistance on this work. I have clearly acknowledged work that is not my own."

Student Signature: Monish

Signoff Checklist

Part 1 Required Elements

- ☑ Pins and signals labeled and decoupling capacitors present on board
- ☑ LCD functional, C code for basic LCD routines functional
- ☑ LCD control signal timing meets specifications (diagram)
- ☑ Serial EEPROM functional, contents present after power cycle
- ☑ C code for EEPROM functional, I²C timing correct
- ☑ EEPROM `eereset()` functional and correct
- ☑ Support for custom LCD characters, fun logo
- ☑ LCD Display, Clear, and Hex/DDRAM/CGRAM dumps
- ☑ Support for custom LCD characters, fun logo

[TA signature] 04/08/2018

TA signature and date

Part 2 Required and Supplemental Elements

- ☑ Elapsed time display (accurate 1 second resolution)
- ☑ Elapsed time stop, restart, reset to "00:00.0":
- ☑ Good integration with previous LCD code, all functions work with no irregularities
- ☑ SPI interface
- ☑ PCF8574 I²C I/O Expander
- ☑ TI MSP432 ARM code development

[TA signature] 04/15/2018

FOR TA/INSTRUCTOR USE ONLY **Part 1 Elements**	Not Applicable	Poor/Not Complete	Meets Requirements	Exceeds Requirements	Outstanding
Schematics, SPLD code	☐	☐	☑	☐	☐
Hardware physical implementation	☐	☐	☑	☐	☐
Required Elements functionality	☐	☐	☐	☑	☐
Sign-off done without excessive retries		☐	☑		
Student understanding and skills	☐	☐	☐	☑	☐
Overall Demo Quality (Part 1 elements)	☐	☐	☐	☑	☐

FOR TA/INSTRUCTOR USE ONLY **Part 2 Elements**	Not Applicable	Below Expectation	Meets Requirements	Exceeds Requirements	Outstanding
Required/ ~~Supplemental Elements~~ functionality	☐	☐	☐	☐	☐
removed ~~Sign-off done without excessive retries~~		☐	☐	☑	☐
Student understanding and skills	☐	☐	☐	☑	☐
Overall Demo Quality (Part 2 elements)	☐	☐	☐	☑	☐

TA/Instructor Comments ☐ ☐ ☐

☐ Optional Challenges: DMA, Level Translation

\+ LCD wrap over all lines.
\+ CU logo (21 custom characters) animation.
\+ custom characters (any base in hex)
* clock struggles when doing Cr to DD RAM dumps
\+ sine, triangle, square and sawtooth (SPI DAC)