Laboratory Notebook

Name Simon Opsahl

Email sopsahl@mit.edu

Subject 6.115

Notebook # 1

Laboratory Notebook for 6.115: Microcontroller Project Lab. This contains work for labs and for the final project.

Contact:

Simon Opsahl
(909) 645-0645
sopsahl@mit.edu

550 Memorial Dr
Tang 03A
Cambridge, MA 02139

Table of Contents

Title Lab 1 Report (p2)	Date
Name	Partner

Exercises (cont.):

7. Build your Kovid Konsole.

8. Test your Konsole.

Exercise 1:

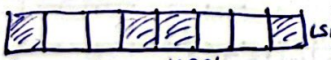
I started by adding minmon31 as found on the course website to my ROM. The EEPROM is the CAT28C64B variety. When the switch is set to mon, we are able to communicate via TeraTerm.

I then uploaded the test program, and successfully turned on only the rightmost LED. The code is in the appendix.
The loop exists in order to keep the squirrel busy so it doesn't start executing tasks that we don't tell it too. The PC will stay in the deterministic region.

To verify what light is in P1 MSB, I can use ~~#80h~~ #80h instead of #01h to P1 and only the 8th ~~bit~~ LED from the right should light up. This worked, and the code is in the appendix.

We can use setb P1.0 instead of mov P1, #01h to set only the first bit. I expect this will accomplish the same as the test program. As expected, only the LSB LED turned on.

To make a static pattern, I chose the pattern below.



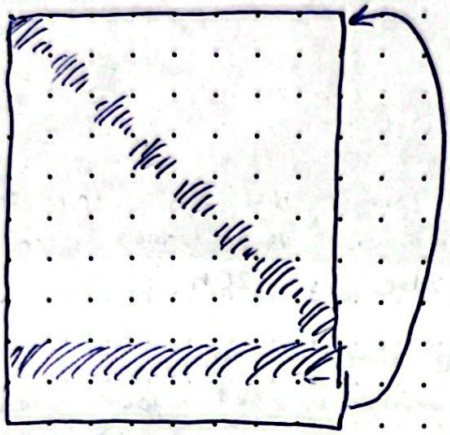
This corresponds to #99h.

Title Lab 1 Report (P3) | Date

Name | Partner

Exercise 1 (cont):

We can use NOP to introduce a delay, but if we only use mov and nop, we'll have to hard code everything into the file. djnz can help loop over the nop functions so they don't have to be typed out many times. I will do a program that looks like the following.



I plan to use clr and ~~set b for~~ sets for the ~~singles~~ single lights, then mov for the full and empty displa In order to make the flash last a reasonable amount time, I will use the ~~accumulator~~ registers and djnz ~~and do nop~~ fo 11E6 cycles.

I encountered difficulties making the timer delay long enough to visualize, but ultimately embedding two djnz lo with a counter of 255 worked.

Title	Lab 1 Report (p 4)	Date 2/21
Name	Partner	

Exercise 2:

In order to initialize the typewriter program, we need to find the register constants for 9600-baud communication.

tmod: selects the mode of the timers
- we want to use timer 1, so we will ignore bits 0-3
- we plan to use mode 2, which corresponds to ~~and~~ an 8 bit reload with TL1 starting from the value in TH1. Each overflow leads to an interrupt.
  + This corresponds to 10b for M1M0.
- Finally, Gate# is low enabled, so it must be 0.

The final value is 20h

tcon: controls the timer
- Everything will be set low except the bits pertaining to timer 1 (6-7). TR1 triggers the timer when set, and TF1 is a flag initialized LOW.

The final value is 40h

th1: stores the starting counter for TL1.
- If we want a baud of 9600, we must use -3d as the value for th1, by the following formula:

$$\frac{11.0592e6 \cdot 2^{SMOD}}{12 \cdot 32 \cdot (256 - TH1)} = \text{baud rate}$$

$$(256 - TH1) = \frac{11.0592\,e6 \cdot 2^0}{12 \cdot 32 \cdot 9600} = 3$$

Thus, th1 holds FDh
- Note: if we want a baud rate of 2400, we can just multiply -3 by 4. TH1 would thus hold -12.

scon: controls serial port behavior
- we don't need TB8 or RB8 as we only need 8 bit communication. TI and RI are involved in actual function, so can be initialized low.
- we want receive enabled, so we set REN. we want mode 1, so SM0, SM1, SM2 are 010

The final value is 50h

Title Lab 1 Report (p5)	Date
Name	Partner

Exercise 2 (cont.):

The typewriter program appeared to work, and I was able to display "Hello World!" in TeraTerm.

```
Welcome to 6.115!
MINMON>
*D
>.....
*Hello World!
```

Result of "Hello World!"

We could have used less program memory by not having the main body at 100h, but instead at the start of the memory. We also could just paste in the init, getchr, and snd functions instead of making function calls. The current structure, however, thrives ~~exactly~~ off its modularity, as it is very easy to understand.

Exercise 3:

In order to also show the character on the LED bank, we can just move the value in the accumulator to P1 when we call sndchr.

In order to create the auto-wrap, we can use djnz instead of loop with a counter of 65, and every 65th cycle do a carriage return and line feed.

I will determine the ASCII codes of each character by doing a sweep of every character on the keyboard and looking at what is shown on the LEDs. The completed table is pasted on the following page.

Title	Lab 1 report (p6)	Date
Name		Partner

Exercise 3 (cont.):

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2	SP	!	»	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Title Lab 1 report (p7)	Date 2/22/24
Name	Partner

Exercise 4:

In order to make a simple calculator, several considerations must be had:

1. 3 numbers ⟶ value

   We must convert the 3 integers in the 1s, 10s, and 100s place into a corresponding value in hex. To first get the hex value for the number from the ~~dec~~ ASCII representation, we can add 30h. Then, we can use mul Ax, where A holds 1, 10, or 100, and x holds its factor. Finally, we can add the three results. We have our 3 digit number translated from ASCII!

2. +, −, and other

   We must determine whether an input is '+', '-', or something else when the two numbers are stored. At this point, we can just compare the ASCII values, and I will arbitrarily designate the default output to be 0h. For the others, the subroutine will run, output the result on the LEPs, and restore to an initial state.

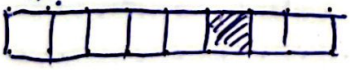
3. Restoring to an initial state

   We must have all registers restore to an initial state, jump to the very start of the loop, and be ready to receive the two new numbers.

2/24/24

Exercise 4 (cont.):

When we overflow the program, the numbers wrap around. 250 to 10 becomes 4, because 5 to become 255 (the maximum value), 1 to become 0 again, and 4 more. The result is:



Title: Lab 1 report (p 8) 2/24/

Name	Partner

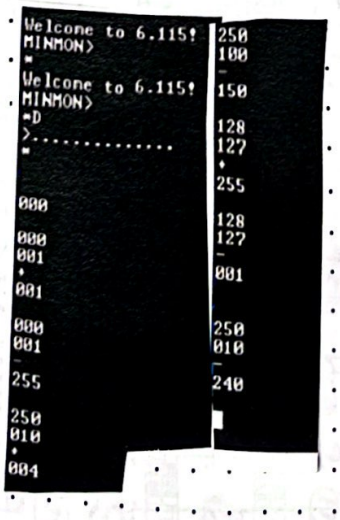
Exercise 4 (cont.):

I tried a bunch of different operations, and this is what resulted:

Term 1	Term 2	Op	Output
000	001	+	001
000	001	-	255
250	010	+	004
250	010	-	240
128	127	+	255
128	127	-	001

Exercise 5:

In order to display the result, we need to convert the hex value into power of 10 ASCII representation. This can be done by first finding number of 100s, then number of 10s in the remainder, then number of ones for each digit. Finally, we subtract #0d0h to get the ASCII. I tried each of the combos above, and this is what I got:



Title Lab 1 : Up: (1)	Date
Name	Partner

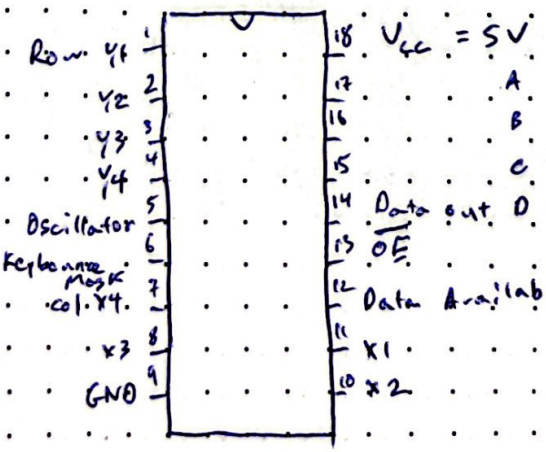
Exercise 6:

I chose to go with the more ~~physical~~ thin keypad, primarily because I ~~tried~~ don't care about the feel of pressing an actual button. The pinout is described below:

Pin				
X_8	1	2	3	A
X_7	4	5	6	B
X_6	7	8	9	C
X_5	*	0	#	D
Pin	~~6~~ 4	~~4~~ 3	~~3~~ 2	2 1

Notes from the 74C922 Datasheet:

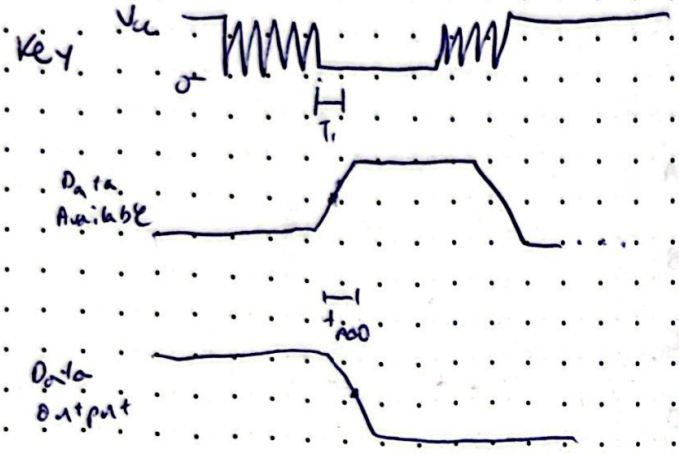
- Supports $\leq 50\ k\Omega$ on resistance switches.
- Only one external capacitor is needed for debounce.
- Data available only goes high when a keyboard entry has been made.
- It takes one debounce period for DA to reset when a key is released.
- 2 key roll-over (one, then another is pressed) is supported.



Title	Lab 1 write report (p10)	Date
Name	Partner	

Exercise 6 (cont.):

Waveforms:

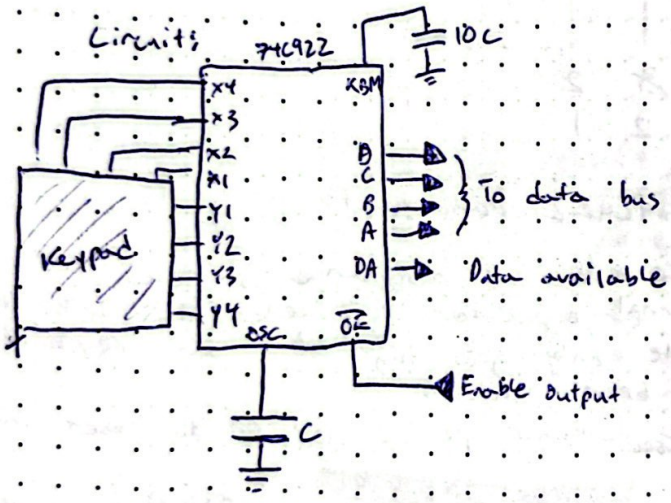


$T_1 = RC_{KBM}$

$R = 10k\Omega$

$C_{KBM} = ?$

Circuit:



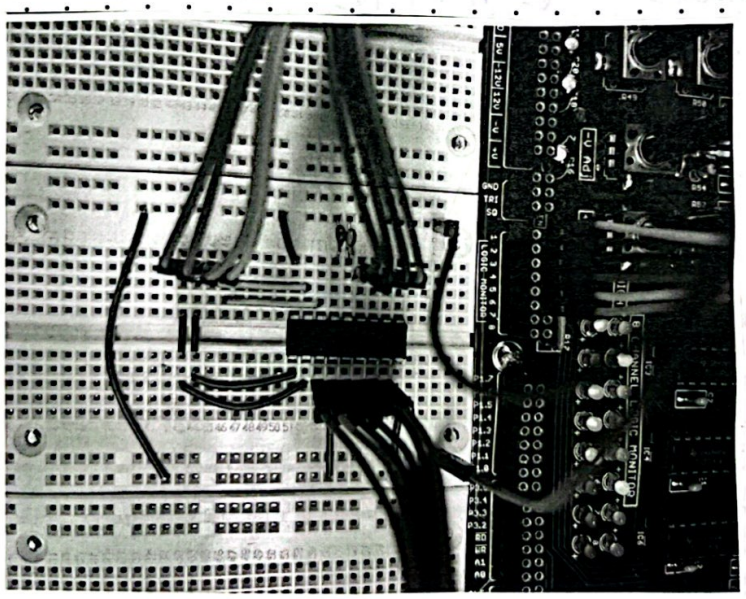
To find C, we want a capacitor that is ~ in the typical range. I looked on the graphs and decided on 0.1 μF using the 100 nF capacitor rated for 50 V. This is for C_{OSC}. C_{KBM} is 10× C_{OSC}, so I'll use the 1 μF capacitor rated for 50 V.

Title	Lab 1 report (p 11)	Date	2/26/24
Name		Partner	

Exercise 6 (cont.):

I wired up my circuit with the chosen capacitors, connected GND to GND and VCC to 5V. The other pins I connected to pins 1-6 of the logic monitor. When I connect $\overline{OE}$ to ~~0~~ +5V, DA will switch with any button press, while no other pin switches.

When I connect $\overline{OE}$ to GND, however, every ~~5~~ output is held to GND, and when a button is pressed, the DA light triggers and the value is outputted and stored. The values cycle from A=0h, 1=3h, ..., D=Ch, *=Fh. Below is a picture of my circuit.



Title	Lab 1 report (p12)	Date
Name	Partner	

Exercise 6 (cont.):

I chose to connect the pins based on the following arrangement.

A — P1.3 P1.0
B — P1.2 P1.1
C — P1.1 P1.2
D — P1.0 P1.3
~~A~~
$\overline{OE}$ — P3.3
DA — P3.2

I wrote code that used the value ~~from~~ the key masked with 0100 1111 so that the output would be an ASCII @, A-O. The result is below

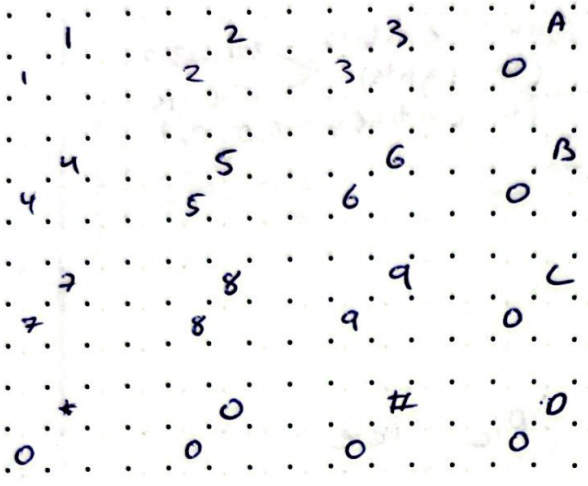
1	2	3	A
C	B	A	C
4	5	6	B
G	F	E	D
7	8	9	C
K	J	I	H
*	0	#	D
O	N	M	L

When making the table lookup, I decided to have all non-digit characters be 0.

Title Lab 1 report (p 13)	Date
Name	Partner

Exercise 6 (cont.):

When this sent to terminal, this is the result:



Now all that is left is to make the calculator. This can done using a very similar structure to the calculator from before, except that now the character is fetched by the keypad press.

Note: I plan to use A for '+' and B for '-'

I tested the same ~~out~~ inputs as the previous calculator, and the results all hold.

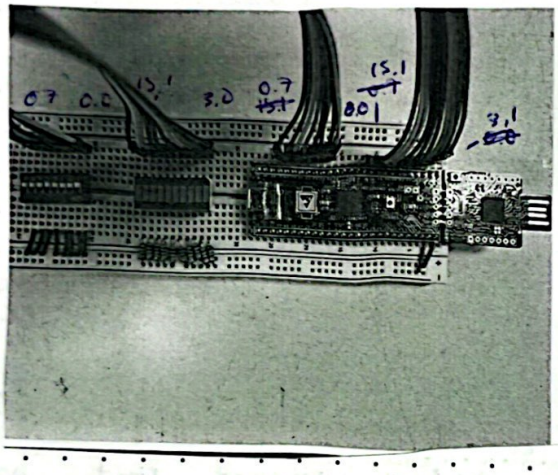
-Note: I used P3.4 to track whether or not a button press is a new one or ongoing so that the input doesn't get flooded.

Title	Lab 1 report (p 14)	Date
Name	Partner	

Exercise 7:

I wired up the breadboard as instructed. A photo is attached below. Elements include:

- 1kΩ pull down resistors — 3.0 - 3.7
- LED bank (10 lights) — 15.0 - 15.1
- Switch bank (8 switches - 0.0 - 0.7

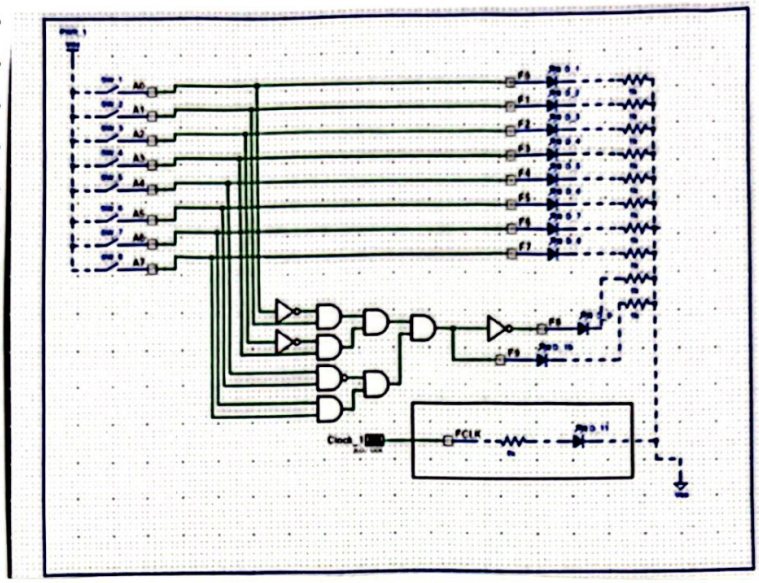


~~Exercise~~ Exercise 8:

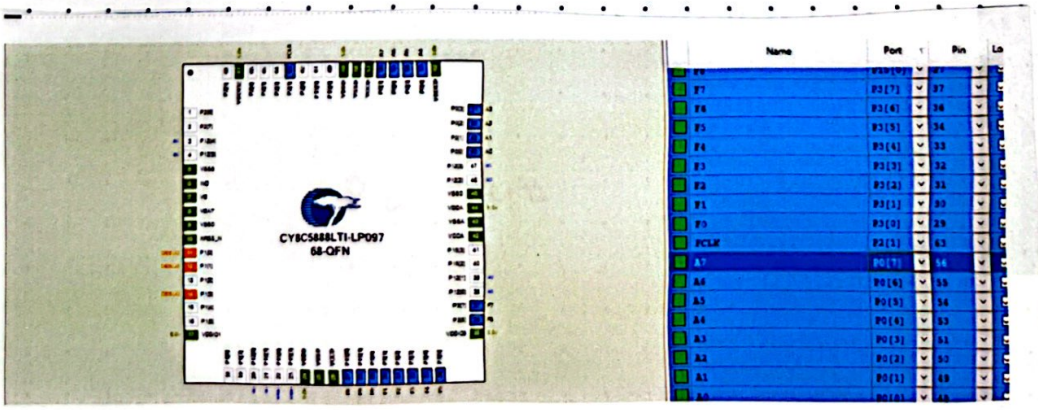
In my PSoC creator design, I made sure to specify I was building for the PSoC 5LP with chip CY8C5888LTI-LP097. I made the design very similar to the staff example, with a few notable exceptions.

Title	Lab 1 report (pls)	Date
Name	Partner	

Exercise 8 (cont.):



- The top level diagram



- The pin assignments

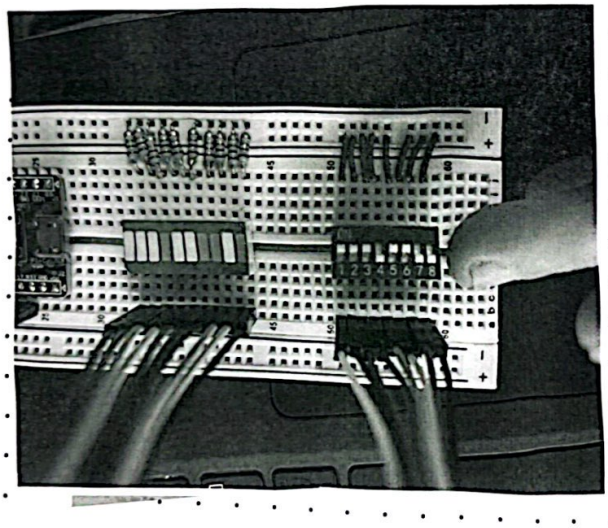
Title	Lab 1 report (p 16)	Date
Name	Partner	

Exercise 8 (cont.):

I chose to have the 9th LED be a wrong, and the 10th a ~~wr~~ right for a specific combination: 0101.0011b, or 'S' in ASCII. That's the first letter of my name, and a difficult code to crack. I have the pictures below:

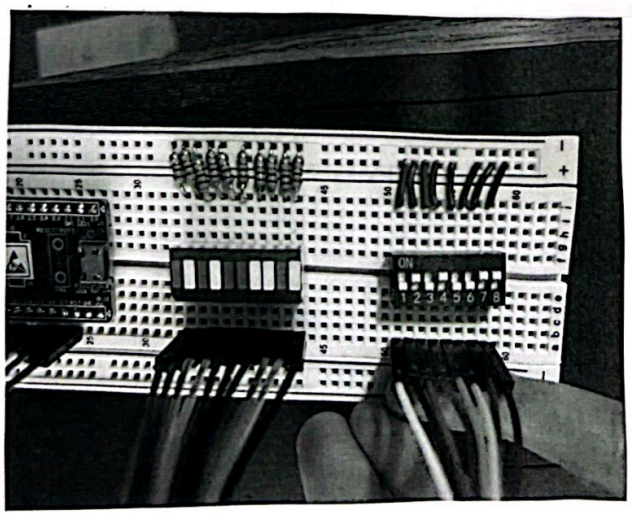
Code: 11101100

WRONG



Code: 01010011

RIGHT



Title Lab 1 report (p 17)	Date
Name	Partner

10/10 KL 2-27

End

Lab 1

Report

Title	Lab 2 report (p1)	Date	2/27/24
Name	Simon Opsahl	Partner	

6.115 Lab 2

Goals:

- Understand and modify the monitor code running on your R3IJP board.
- Understand the structure of an Intel Hex file.
- Learn to use interrupts and timers.
- Connect a peripheral chip to your R3IJP.
- Take control of a physical system (LED lamp) using your R3IJP.
- Learn about the PSoC "Programmable System on Chip" and the C programming language.

Exercises:

1. Expand MINMON
2. Reverse assemble a hex file
3. A simple calculator
4. Get comfortable with the fluorescent lamp
5. Make some square waves
6. Understand the 8254 counter/timer chip

Title 6.115 lab 2 report (p 2)	Date
Name	Partner

Exercises (cont.):

⑦ Control the lamp.

⑧ Understand the 8051 hardware.

⑨ Learn about the PSoC Big Board

Exercise 1: Adding Read/Write to MINIMON

We will add read to 'R' and write to 'W'

Read:

- We want to read the hexadecimal contents of a byte in external memory.
- We then want to display the output,

  Ex: * R9000
  FF
- We can use printhex, which prints the contents of A as 2 digit ASCII hex, and getbyt, which reads in a 2 digit ascii from the serial port,
- We can also use the DPTR, and DPL, its low byte, and DPH, its high byte.
- We can move and pop the values into the pointer and indirect address.

Title Lab 2 report (p3)	Date
Name	Partner

Exercise 1 (cont.):

Write:

- Write can be implemented in a very similar way, but instead uses movx @dptr, a. We need to check formatting, but besides that, it should be straightforward.
- We can only write to memory locations accessible in the RAM, which is 8000-ffff, as the EEPROM is read-only.
- The result of several read/writes are below.

Title	6.115 lab report (p4)	Date	2/24
Name		Partner	

Exercise 2: Reverse Assemble Hex

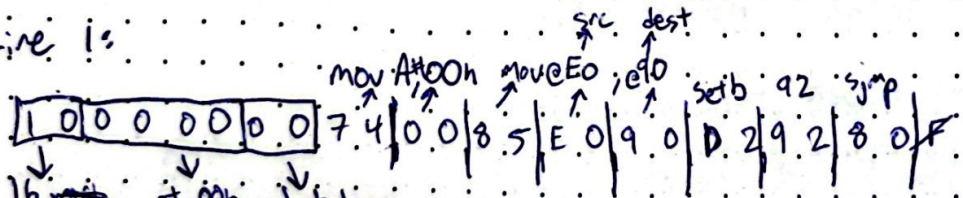
In order to reverse assembly, we must deconstruct the inputs. The hex is in the following format:

: XX XXXX XX XX ... XX XX

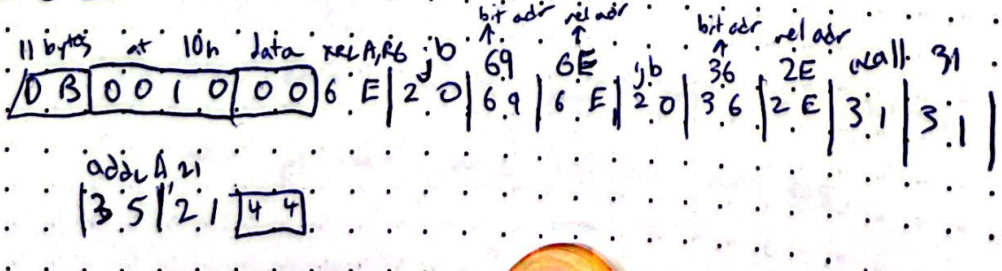
record length | address | data type | data bytes | checksum

Record length will say the number of words in the data, and address will say the location. Everything else is unimportant to the decoding. The data types are consistent, the checksum is just checking for corruption.

Line 1:



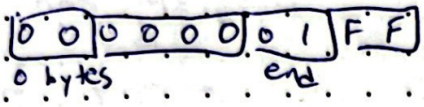
Line 2:



Title 6.115 lab report (p5)	Date
Name	Partner

Exercise 2 (cont.):

Line 3:



The code, thus, is as follows:

```
start: org  00h
       mov  A, #00h
       mov  90h, E0h   → 90h=P1, E0h=Acc.
       setb 92h        → 92h=P1.2
       sjmp start      → sjmp F7
                         = -9 bytes
                         +2
                         = -7 bytes => mov A, #00h

       jnc  61h        → if C==0, branch
       orl C, 74h      → 74h=   if C==1, proceed
       mov R1, #20h     → 61+2 = 63+9= 6Ch
       xRL A, R7

       xRL A, R6
       jb  69h, 6E     → 11+3+6E = 82h
       jb  36h, 2E     → 14+3+2E = 45h
       acall, 31       → 001_0011_0001 = 131h
       add A, #21h
```

Although we have a number of instructions, the only important ones are at in the first block. In this, we load P1 (the LEDs) with 00h, then set P1.2, then loop over it again, the rest of the code is not reached, and when converted to ASCII, says "Party on in 6.115!" The code is in the Appendix, and the hex is included with it.

Title 6.115 lab report (p6)	Date 2/24
Name	Partner

Exercise 3: A simple calculator

In order to have a program run from mon and update locations in memory with the output of arithmetic operations, we can use the dptr, and have subroutines for each of +, -, *, and / add their results and increment.

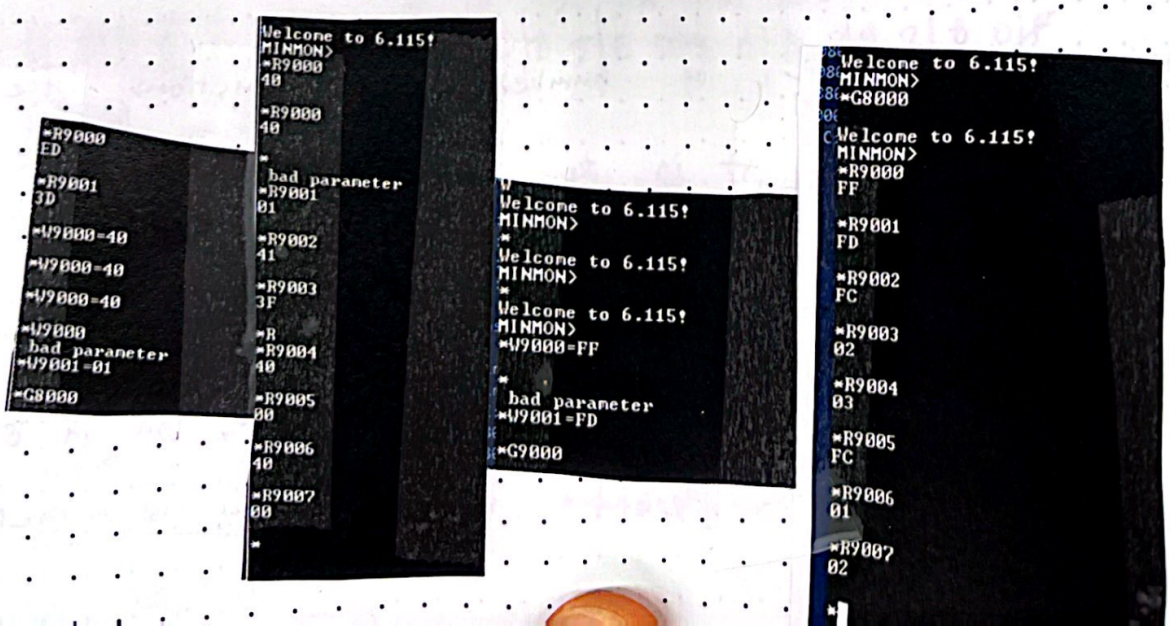
I will try two sets of numbers:

40 and 01

~~25~~ ff and ~~251~~ fd

	sum	diff	product	quotient	remainder
40 and 01	41	3F	0040	40	00
ff ~~25~~ and ~~*~~ fd	fc	02	fc03	01	02

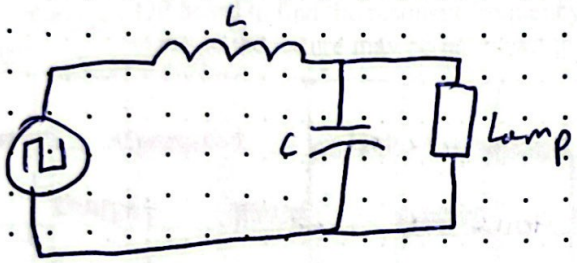
* Note, I started the program at 8000h so lcall works for the subroutines.



Title 6.115 lab report (p7)		Date 3/11/24
Name	Partner	

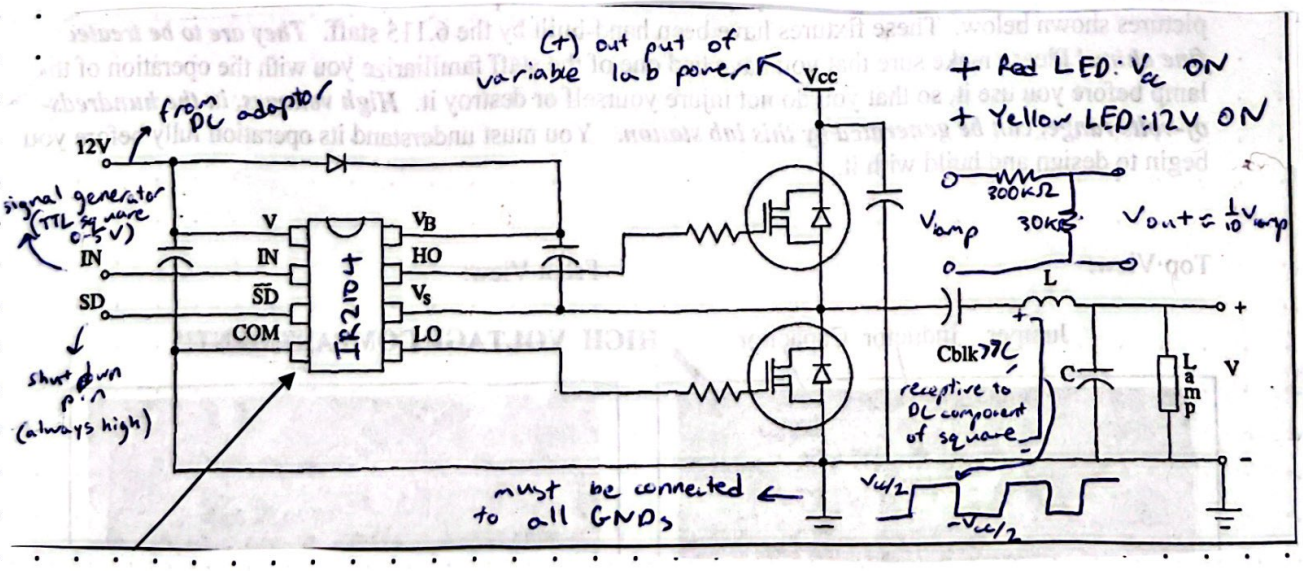
Exercise 4: Understanding the Lamp

When working with a fluoresecent lamp, our strike voltage ~ 200 V, and after, we have a negative V-I relation in the range ~100-500 mA. In order to strike and maintain low current, we need the following circuit:



We plan to tune the circuit assuming L and C are poorly controlled.

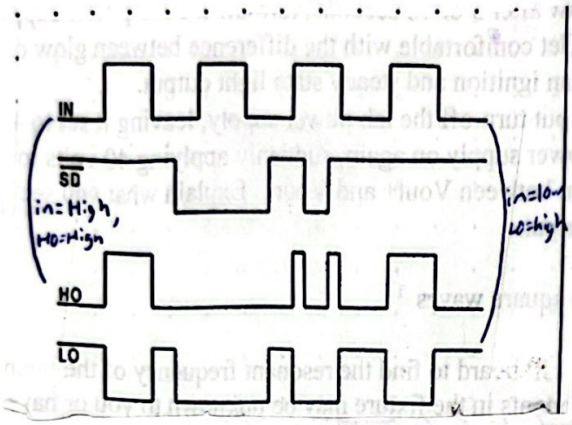
Our lamp ballast is wired up the following way:



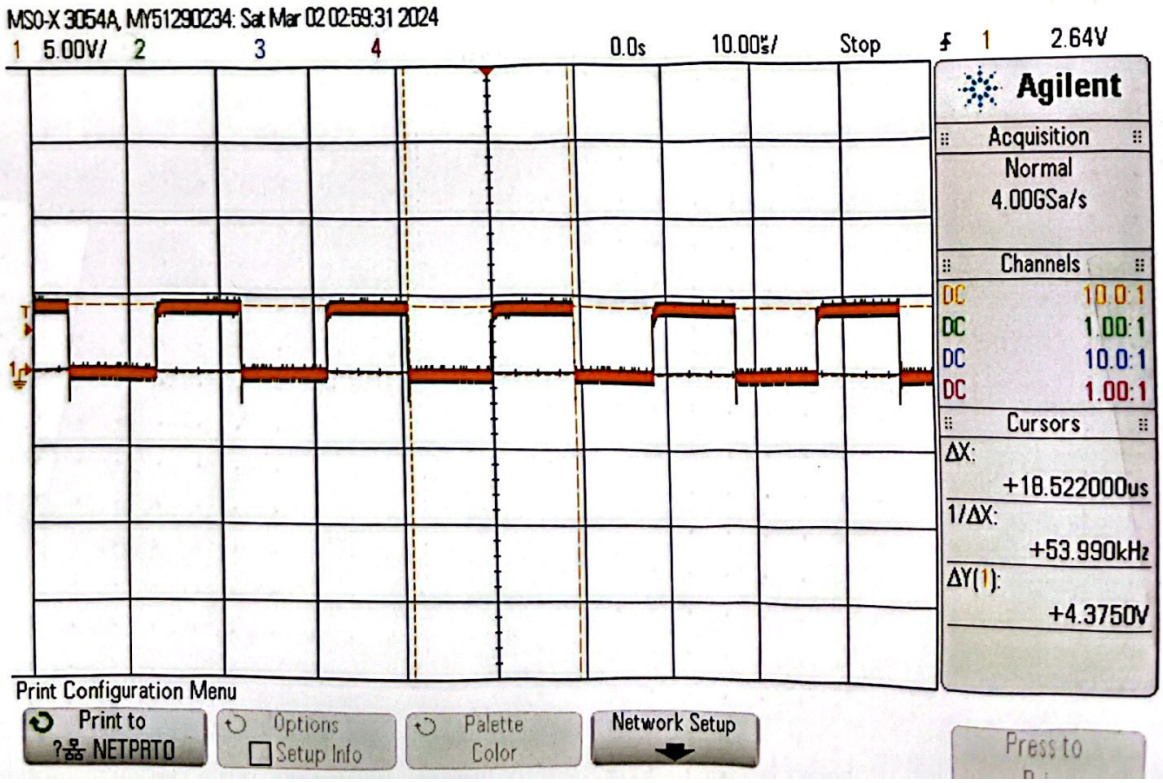
Title 6.115 lab report (p8)	Date
Name	Partner

Exercise 4 (cont.):

The timing is like below:



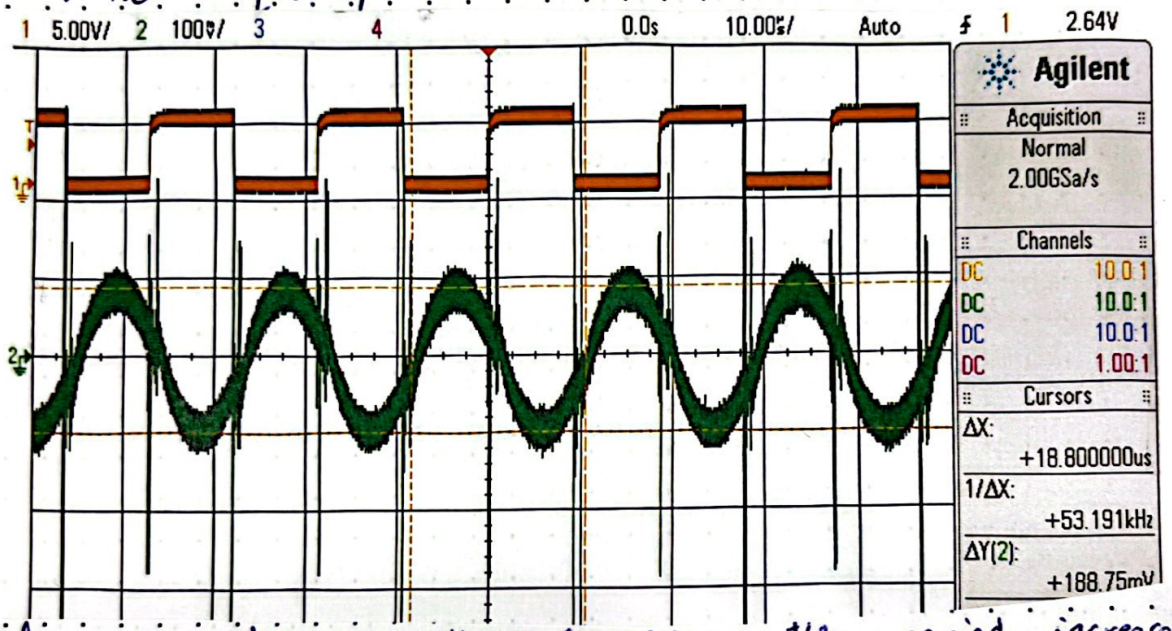
To set up, I connected the BK precision 5 MHz function generator 4011A. I used the TTL square wave source. When connected to the probe, the plot at 54 kHz looks like the following:



Title 6.115 lab report (p 9)	Date
Name	Partner

Exercise 4 (cont.):

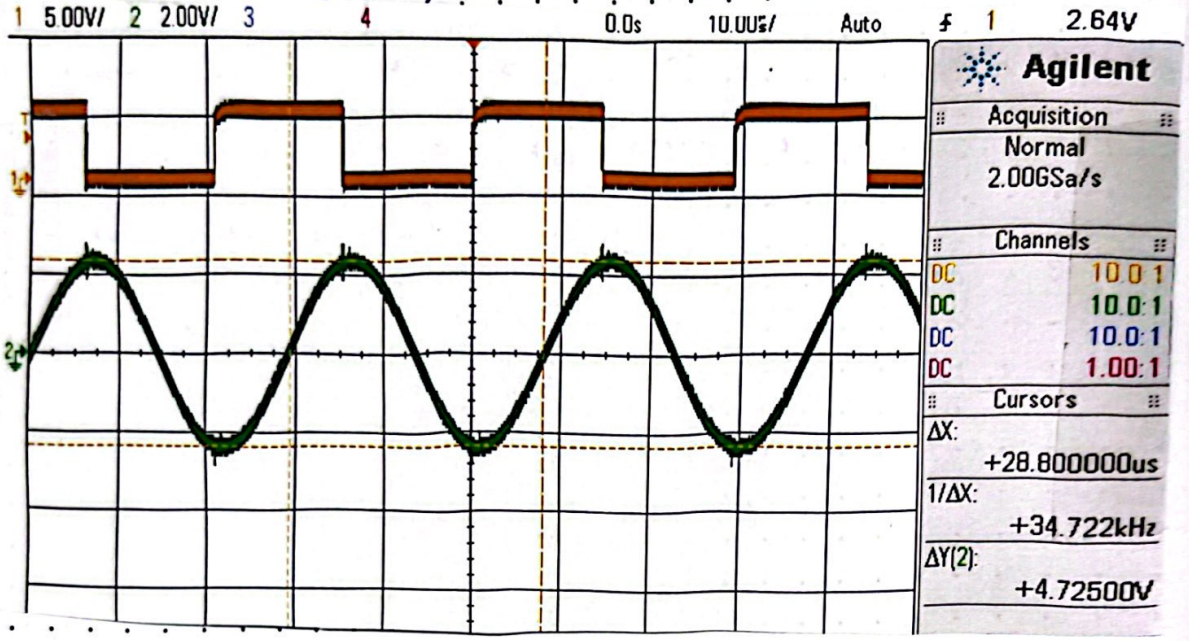
I set the signal generator at 60 kHz, then put the lab supply at 3.0 V. When we see the output, we have a sinusoid with an inverse waveform. The peak of the sinusoid corresponds to the TTL low. At 3 V, 54 kHz, the amplitude is ~94 mV. The frequency is ~54 kHz.



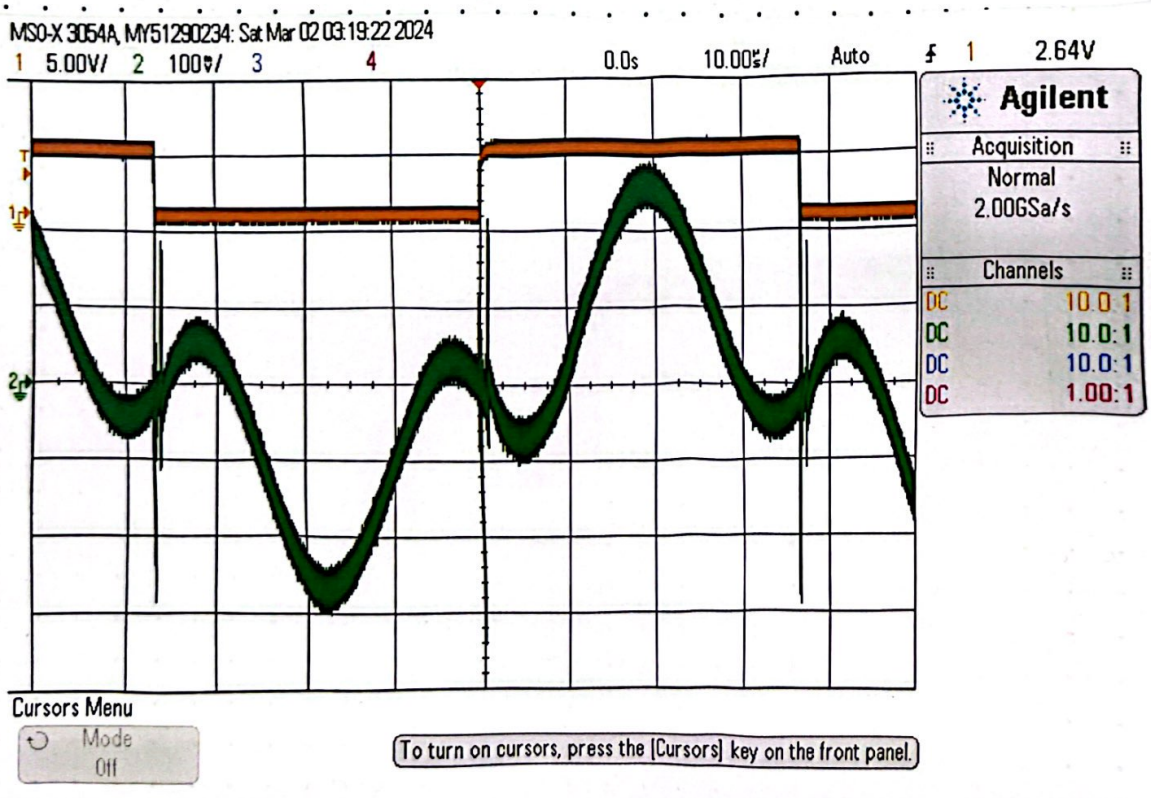
As we decrease the frequency, the period increases and so does the amplitude. The phase also shifts so that the peak of Vout is at the end of the square HIGH. The maximum amplitude is ~4.73 (p-p) at 34 KHz. The dot is on the previous page. This maximum is at $f = \frac{1}{2\pi RC}$. ~~If me~~

Title G.115 lab report (p 10)	Date
Name	Partner

Exercise 4 (cont.):



If we bring the frequency down to 15kH, the sinusoid starts to separate into a half-period wave

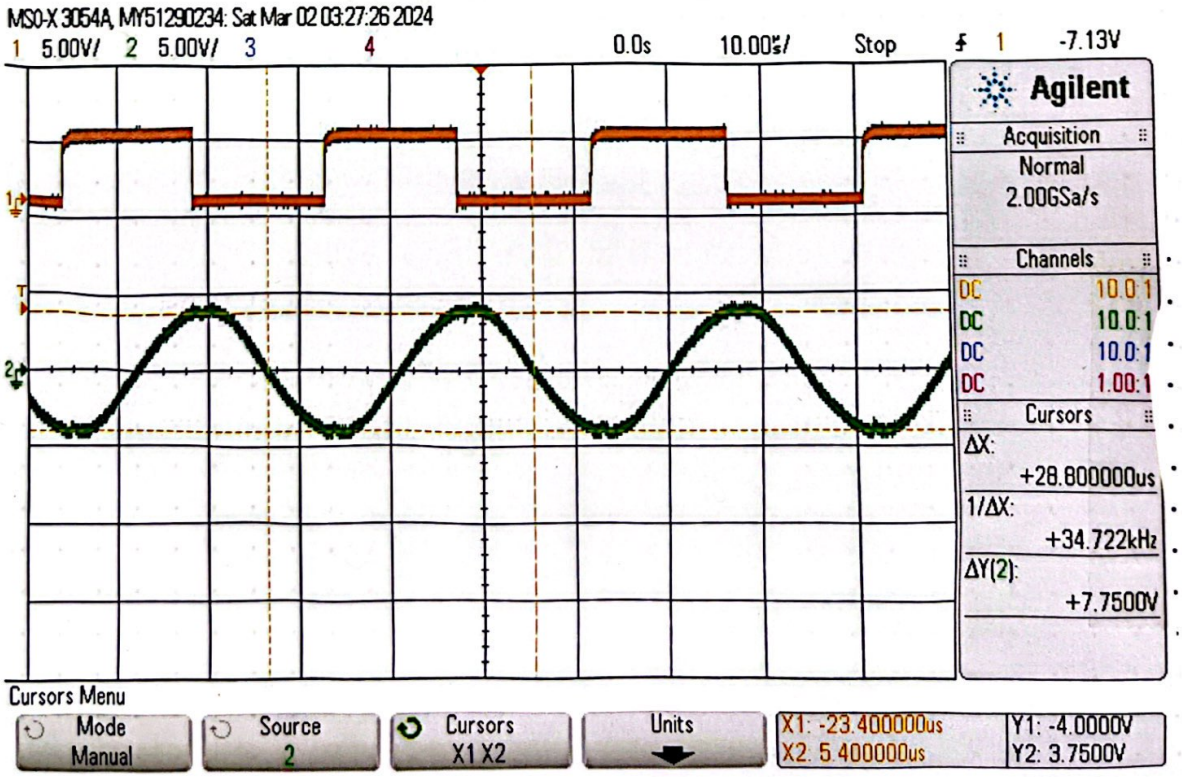


Title 6.115 lab report (p11)	Date
Name	Partner

Exercise 4 (cont.):

So I will use 34.5 kHz as the resonant frequency. The lamp light struck once I had crossed the 20 V threshold, it flickered, then started glowing. The power supply maxxed out at ~30 V, so I kept it there.

When I turned on the bulb, we see a small glow; when observing the ~~plot~~ plot of $V_{out}-V_{in}$, we see an initial spike in the sinusoid, then a restoration to around ~~80~~ 8 V on $V_{out}-V_{in}$, which corresponds to about 80 V across the bulb, a typical operating equilibrium voltage.



Title	Lab 2 report (p12)	Date	3/
Name		Partner	

Exercise 5:

~~In order~~ I will make the offset accessible from 1000 ~~0000~~h in the RAM (set by ~~W9000h~~ W9000:?). ~~It can be~~

The code will follow as:

```
main:
    ~~movx a, @dptr @100h~~
    ~~movx~~                    ~~mov a, R0 ; R0 holds the~~
    loop:                       mov R0, R0 ; R0 holds the from 100h
                                R0 R1.
                                1 cycle
    djnz ~~R0~~ R0, loop        2 cycles
    cpl P1.0                    2 cycles
    sjmp main                   2 cycles
```

The total cycles are 2(53 + 2·(R0)). Each takes 12 clock cycles, and the clock is 11.0592 MHz. Thus, our range is

$$\frac{11.0592\ MHz}{12 \cdot 256 \cdot 2} \le f \le \frac{11.0592\ MHz}{12 \cdot 2(256 \cdot 2 + 5)2}$$

~~13,057 KHz~~ ~~62.83 KHz~~ ? f ? 891 Hz

76.8 KHz

At around 34 KHz, the resolution is

$$\frac{11.0592\ MHz}{12 \cdot 2(x \cdot 2 + 5)x} = 34\ KHz$$

$$x = 4$$

Title	Lab 2 report (p 13)	Date
Name	Partner	

Exercise 5 (cont.):

At $x = 5$, $x = 3$

$x = 5$:

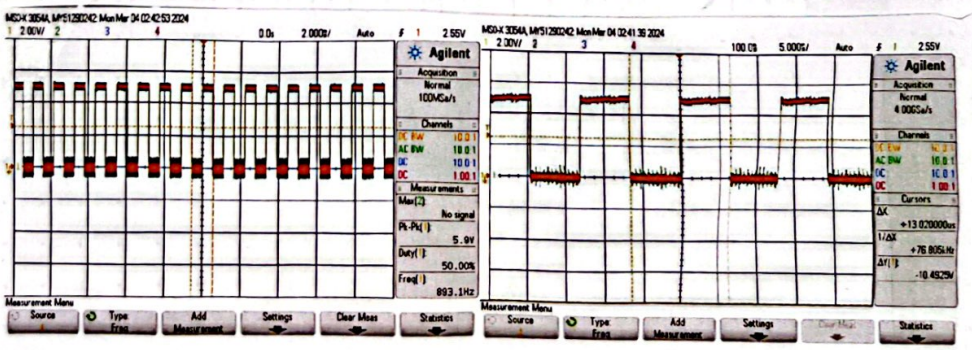
$$f = \frac{11.0592\ MHz}{12 \cdot 2(\frac{255}{5} \cdot 2 + 5)} = 30.72\ KHz.$$

$x = 3$:

$f = 41.9\ KHz$.

The resolution is around 5 kHz in the region we want it to be, which is not ideal resolution for fine tuning.

The plots of W9000 = 01 and 00 are below. The actual range appears to be 76.8 kHz to 893 Hz.



Title	Lab 2 report (p 14)	Date
Name	Partner	

Exercise 5 (cont.):

We want higher resolution in the 34 kHz range, which means we want it in the center of our range. We can ~~user~~ use the timer in mode 1 for a 16 bit counter and check if TF0 is set. We also need to disable interrupts if possible if we clr ea at the beginning. (It defaults to the

We can do this as

```
cycles      main:
2           mov TL0, R0     ) Have low and high values
2           mov TH0, R1     ) from the data
1           set TR0
            loop:
2           jnb TF0, loop
1           clr TR0
2           cpl P1.0
2           sjmp main
```

cycles = 10 + 2(10000 − x)

max: x = FFFFh

$$f = \frac{11.0592\ MHz}{12 \cdot 2(10 + 2(1))} = 38.4\ kHz$$

min: x = 0h

$$f = \frac{11.0592\ MHz}{12 \cdot 2(10 + 2(10000))} = 56\ Hz$$

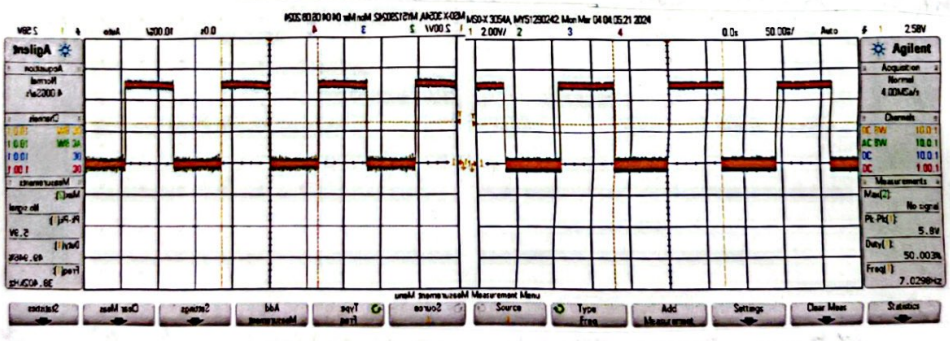
Title Lab 2 report (p 17)	Date
Name	Partner

Exercise 5 (cont.):

$$f = 34\ kHz = \frac{11.0592\ MHz}{12 \cdot 2(10 + 2(10000_h - x))}$$

$$x = FFFE_h$$

The resolution barely improves to around 4 kHz in our desired region. We need something better. The plots are below, showing a max of ~~41.4~~ 38.4 kHz at FFFFh and a minimum of 7 Hz at 0000_h



Title Lab 2 report (p16)	Date
Name	Partner

Exercise 5 (cont.):

For the auto reload, we can use the interrupt handling to toggle P1.0, but in all other cases, just need to be in a do-nothing loop. We need EA to be set and IE.1 to be set for Timer 0 overflow. We also want falling edge control, so we'll use 1 for IT0.

In total:

IE = 10000010

Tcon = 00010001

Tmod = 00000010

TH0 = x

We handle the interrupt at 00BH. The resulting range is 1.8 kHz for x=00h, and 65.83 kHz for x=ffh. 34 kHz is in the center of this range, which is ideal.

The range around 34 kHz (F2, F3) goes from 32.9 kHz to 35.45 kHz, a resolution of 1.25 kHz, which is much, much better.

Title Lab 2 report (p 1?)	Date
Name	Partner

Exercise 5 (cont.):

The timing comes from:

interrupt —jmp→ cpl P1.0 (2 cycles)
ret (2 cycles)

As a result, the total time is

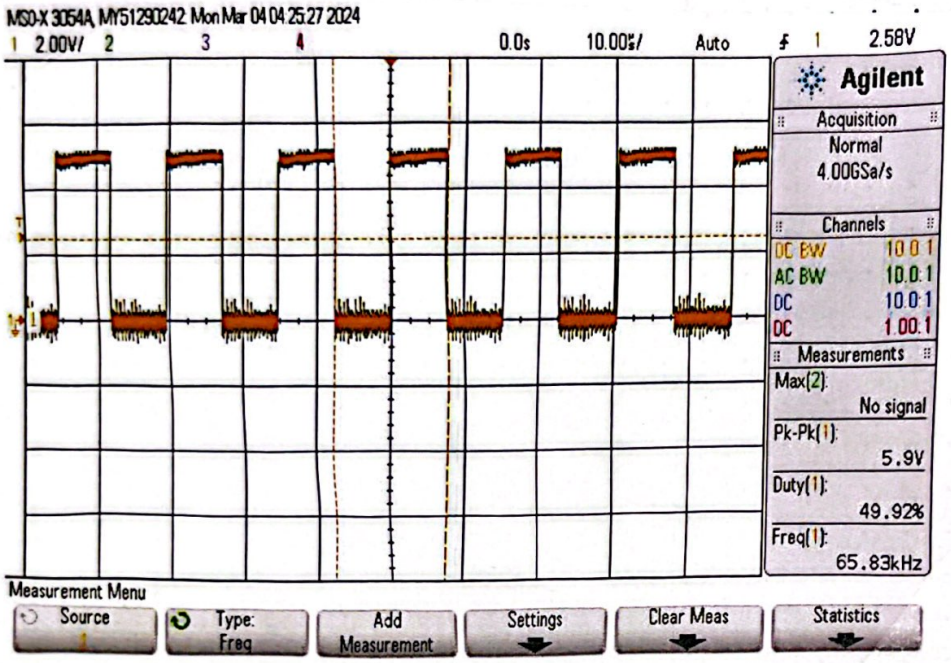
$\frac{11.0592\ MHz}{12 \cdot 2(256-x)} \Rightarrow 1.79\ KHz$ to ~~75~~ 65.83 KHz.

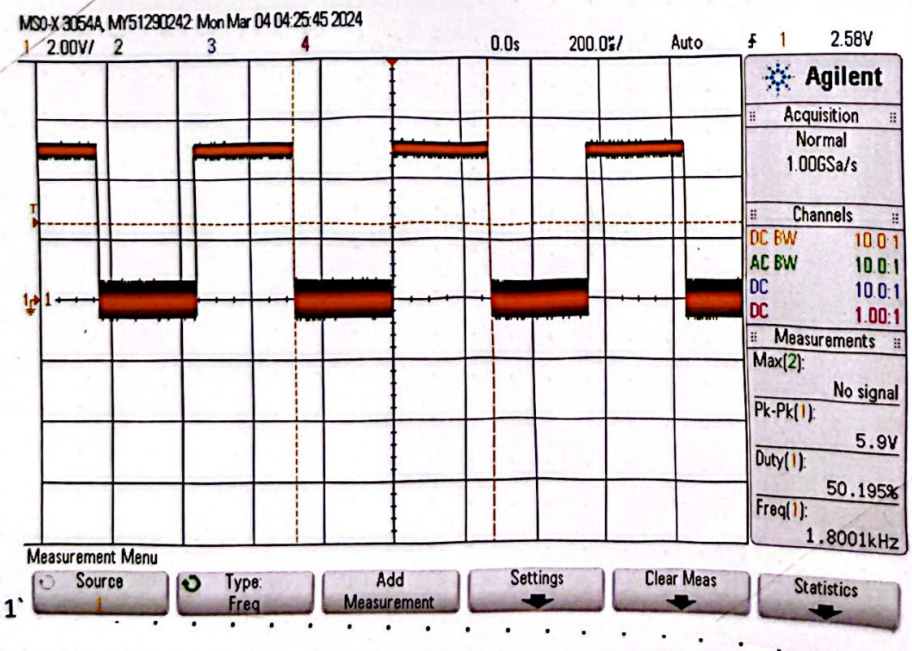
The reason the third option is so much better is because the delay from overflow to restart is more due to the autoreload. Thus, the only kinks in the time are the interrupt handling, which are much less than the other routines.

The plots are on the following page for the third controllable frequency.

Title	Lab 2 report (p18)	Date
Name	Partner	

Exercise 5 (cont.):

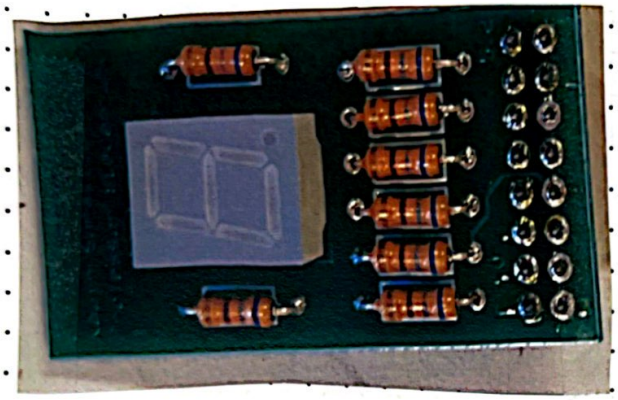


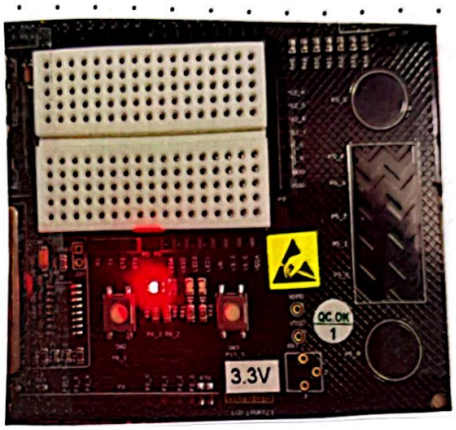


Title Lab 2 report	Date
Name	Partner

Exercise 9: PSoC Big Board

I went to a solder clinic on 2/29, and a picture of my result is below.





When the project is built and flashed, it displays "6.115 IS COOL" on the led display.

In working with Exercise 1, I created a digital output pin w/ no external or HW connection. It is set to P6.3, or LED4 on the Dev Board. When built and programmed, we get a red flashing LED4.

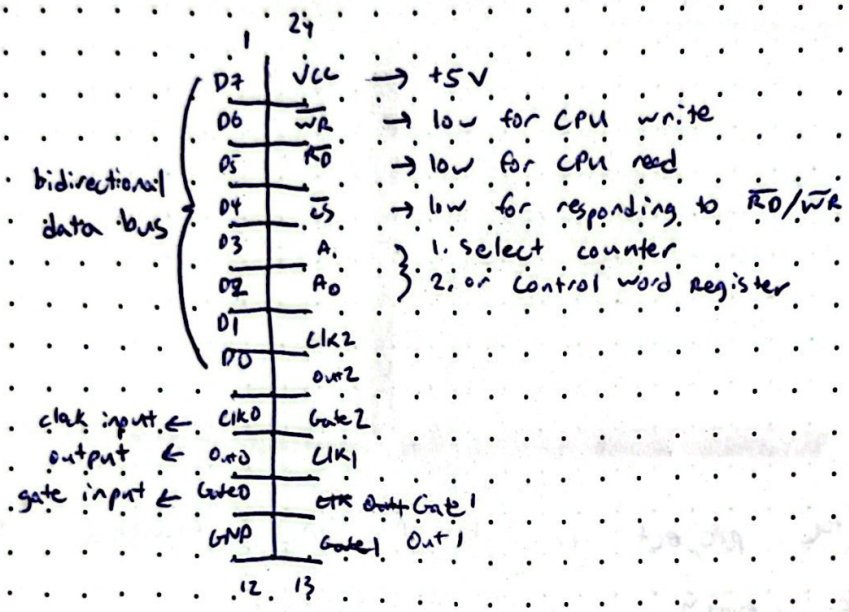
To make a flashing SOS signal, we need to add code to the main.c file using CyDelay(), which specifies a number of milliseconds to wait. The led on photo is above.

Title	Lab 2 report (p 20)	Date	3/
Name		Partner	

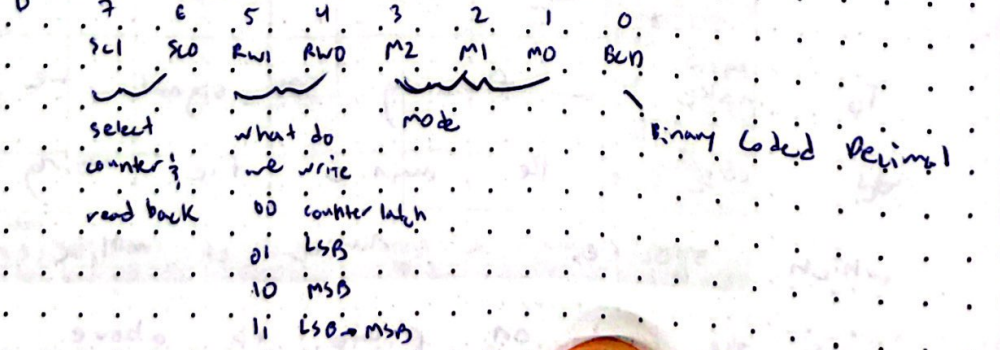
Exercise 6: The 8254 Timer

Notes:

- Contains bank of 3 timers, 16 bit, up to 10 MHz.



- Control word Register (A=11d) stores operation of counters
- Any read measures the contents of a latch downstream of the counting element.
- Any write puts the value into the upstream count registers.
- To use, you write a control word then add a count.
- CW format:



Title Lab 2 report (p 21)	Date
Name	Partner

Exercise 6 (cont.):

- The control word must be written before initial count.
- A counter latch latches the count when the command is received, but doesn't disrupt the CE.
- Read back can be used to latch on a count or status of multiple counters:

  CW: 1 | 1 | $\overline{CT}$ | $\overline{status}$ | CNT2 | CNT1 | CNT0 | 0

- Status reads can show output:

  Output | Null | RW1 | RW0 | M2 | M1 | M0 | BCD

- Modes:

  0: Interrupt on count running out
  1: One shot
  2: Divide by N counter
  3: Square wave

- Mode 3:

  + Out is initially high when half of count expires out goes low. Initial count of N is N period wave
  + Gate=1 enables counting, used to synchronize the counter.
  + Gate is sampled on the rising edge.

Setting up the 8254:

1. Connect D0-D7, $\overline{RD}$, $\overline{WR}$, A0, A1, V_{cc}, and GND to their appropriate spots.
2. GND Gate & CLK for counters 1 & 2.
3. Connect 00 $\overline{CS}$ to $\overline{XIOSel}$ on the XIOSELECT line.

Title	Lab 2 report (p 22)	Date
Name	Partner	

Exercise 6 (cont.):

4. Connect Gate0 to +5V, connect 10 MHz clock to CLK0.

5. Use mode 3 for ~~clock 0~~ counter 0

0 0 1 1 0 1 1 0 = 36h
(counter 0; LSB + MSB; mode 3)

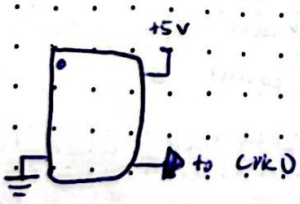
6. For FE00-FEFFh, the xiosElect is active, we must look at A0, A1.

~~A1, A0~~

- All addresses with A1A0=11 write to the control register (64 addresses).

FE00h (04,...) = Counter 0
FE01h (05,...) = " 1
FE02h (06,...) = " 2
FE03h (07h,...) = Control word

For the 10 MHz clock, we have the pin out

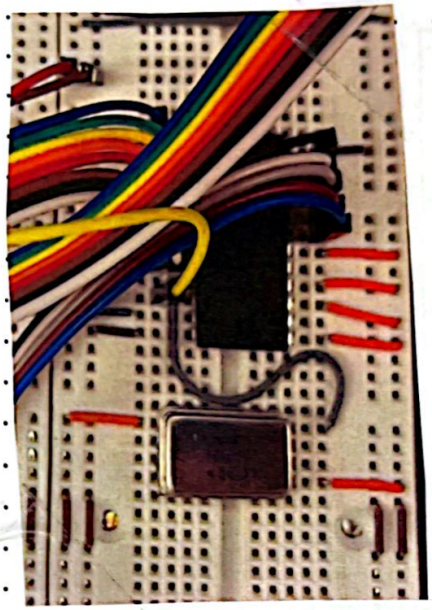


Note: If we had a 1 MHz crystal instead, it would have a lower possible output freq, but the max and resolution would be worse. It would have lower Δf, however.

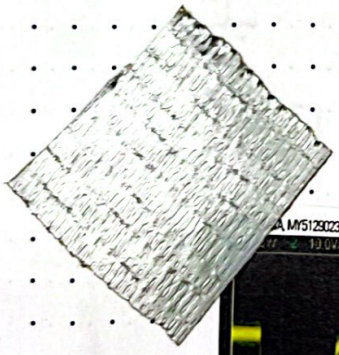
I used WFE03 = 36h and 0126 h to provide the values for 34 KHz. The wiring and plots are shown to the right.

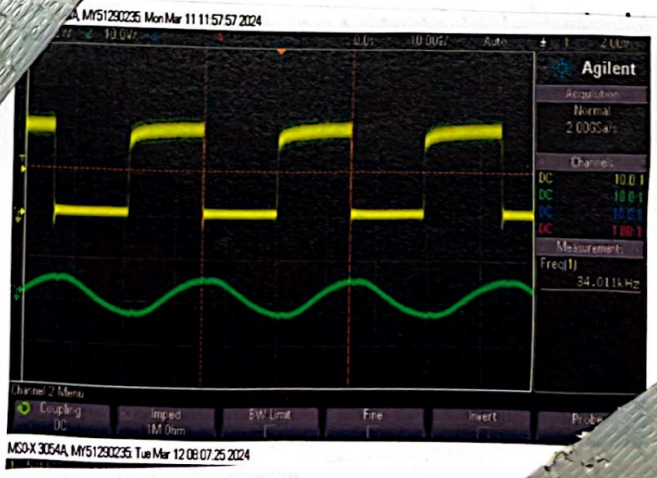
Title	Lab 2 report (p 23)	Date
Name	Partner	

Exercise A.6 (cont.):

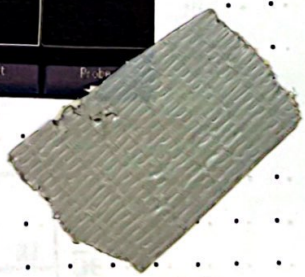


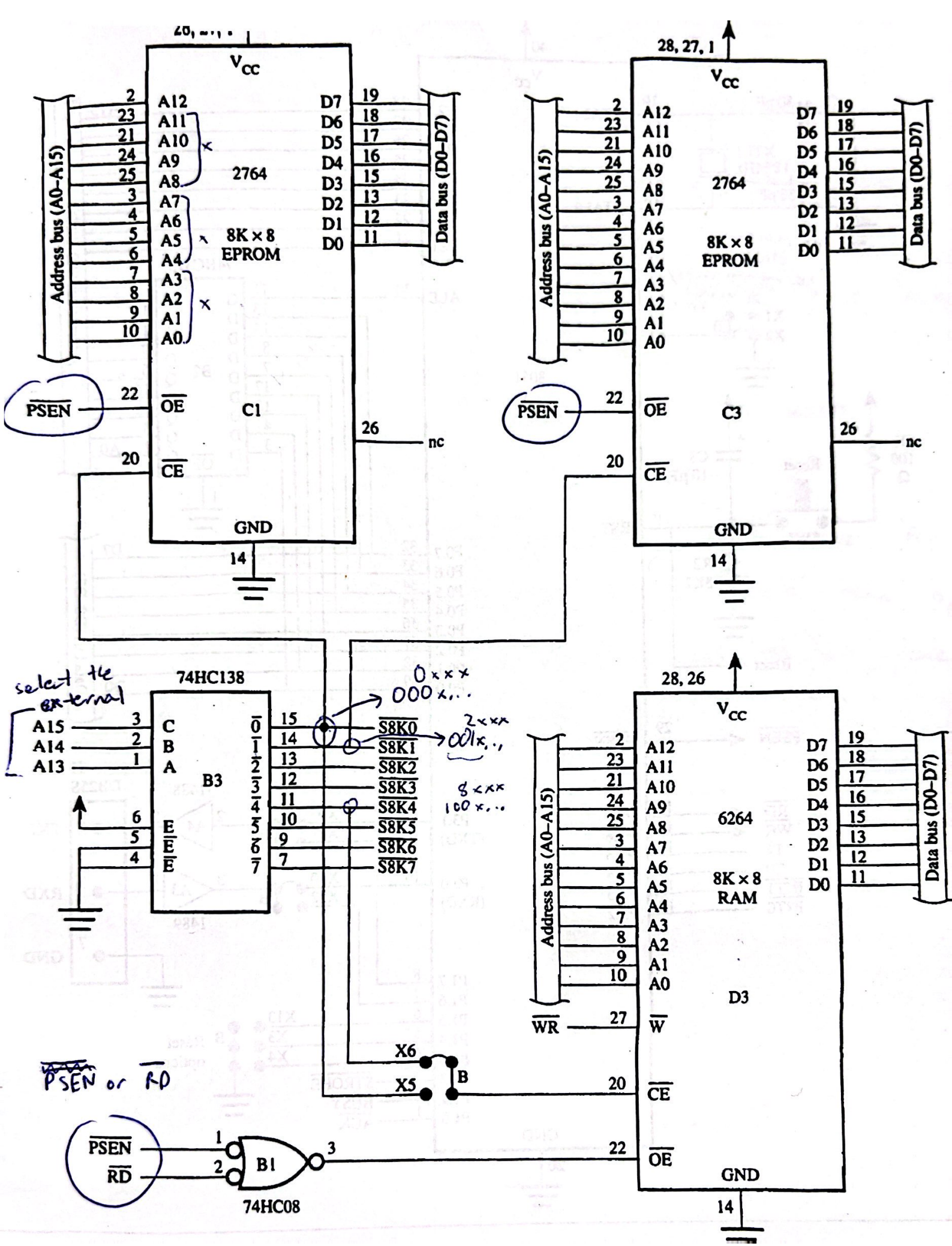
The circuit of the 8254 and 10 MHz crystal





square wave generated w/ 0126

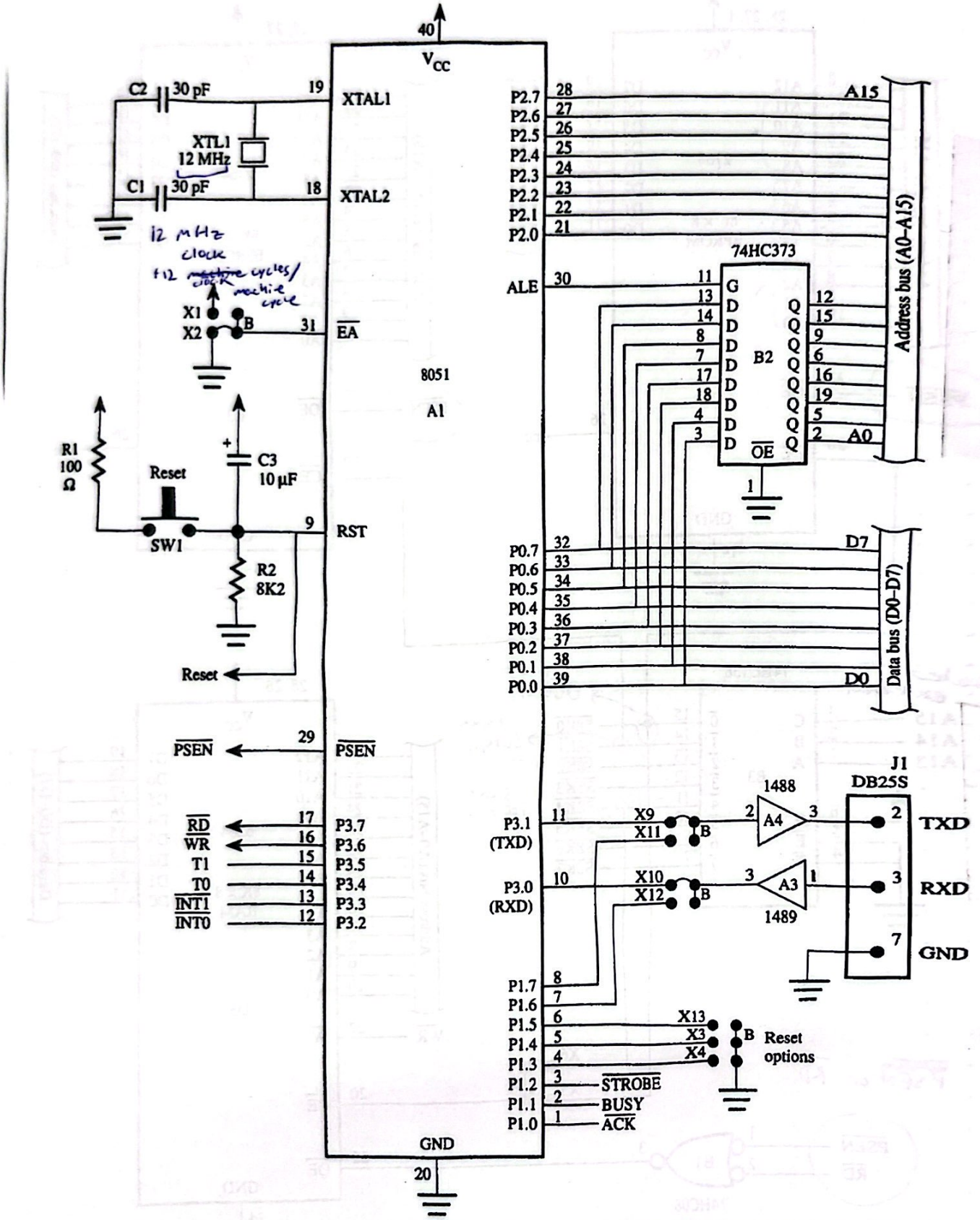




Scanned with CamScanner

Title	Lab 2 report (p.24)	Date
Name	Partner	

Exercise 8: Understand the 8051



Title Lab 2 Report	Date
Name	Partner

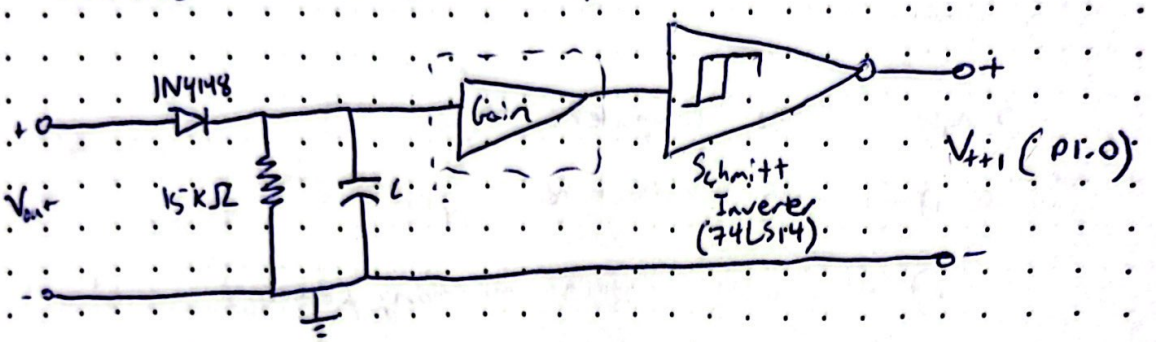
~~Exercise~~ Exercise 8 (cont.)

Single Board Computer:

1) 1 μs machine cycle.
2) C1 could contain mnemonic address stored at 0xxxh and is ROM.
3) After power-on reset, the routine could fetch code from internal ROM or C1.
4) G800 would jump to D3, the RAM.
5) G200 would jump to C3.
6) Because movc sends a low pulse for $\overline{PSEN}$, any of C1, C3, or D3 could read by movc.
7) Because movx doesn't send the low pulse, only D3 (RAM) can be read.

Title	lab 2 report (p 26)	Date
Name	Partner	

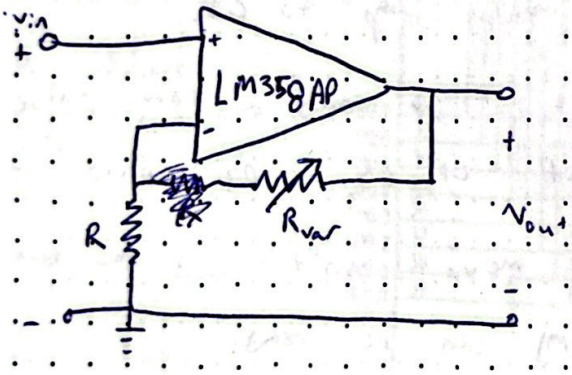
Exercise 7: Control the lamp



$RC = 1\text{-}15\ mS \Rightarrow 10\ mS$

$$C = \frac{10 \times 10^{-3}\ s}{15 \times 10^{3}\ \Omega} = 0.667 \times 10^{-6}\ F = 0.67\ \mu F$$

Gain Block:

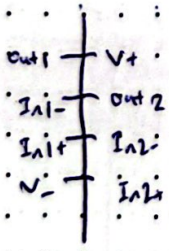


$R = 200\ \Omega$

$R_{var} = 10K$ POT

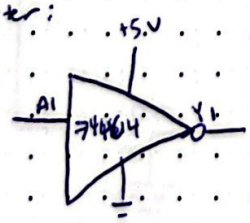
$G \begin{cases} Min = 1 \\ Max = \frac{200 + 10K}{220} = 50 \end{cases}$

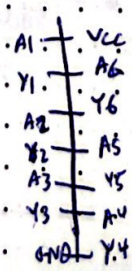
LM358AP:



V+/V- = +5V / GND to saturate
Out/In 2 = GND

Schmitt Inverter:

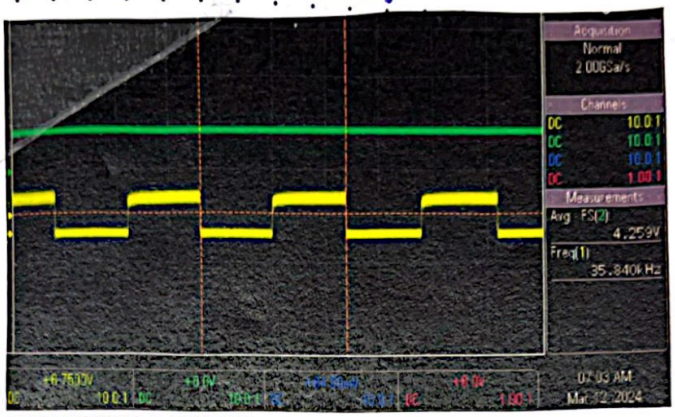


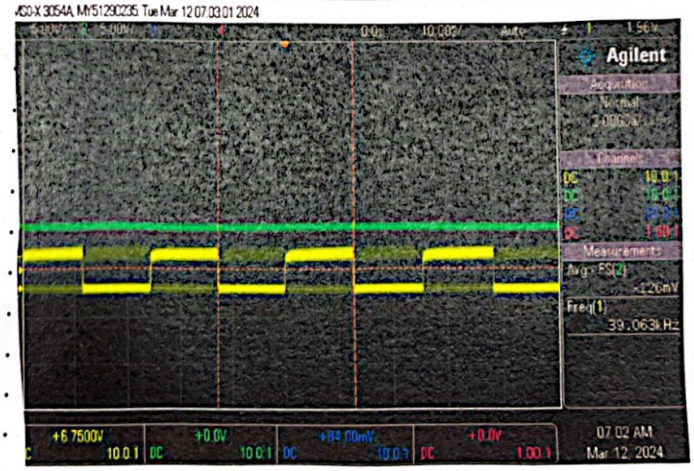


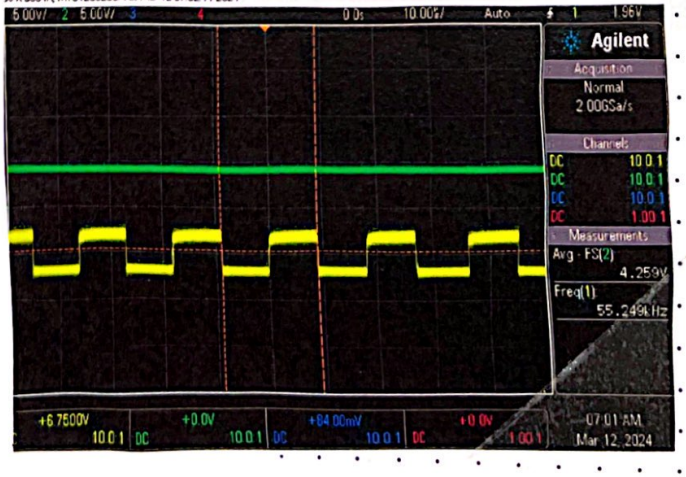
Title	Lab 2 report (p 27)	Date 3/11/24
Name	Partner	

Exercise 7 (cont.):

I was able to establish a threshold, where the Schmitt output is high when out of range, low when close, and high again when triggered. The plots ~~and circuit~~ are shown below.



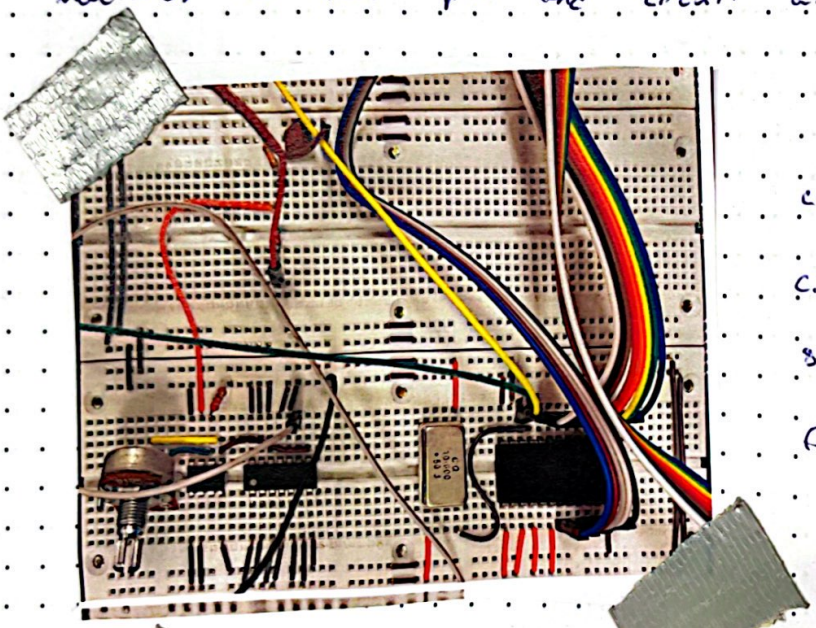




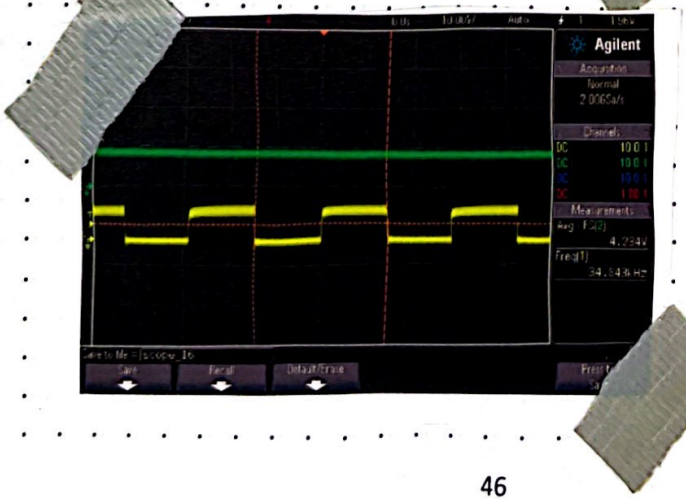
Title	Lab 2 report (p 28)	Date
Name		Partner

Exercise 7 (cont.):

When striking the lamp, I decided to step down in frequency in large steps ($\frac{1}{x10}$) at first, then decrease to 1 when close (Schmitt low), and 0 when struck (schmitt high again). This can be done by using a timer delay. I used a 16-bit timer with E000h load. This allows the real time system to not fall behind the machine. The final plot and circuit are shown below.



complet Lab circuit w/ 8254 timer and feedback circuit



Driven frequency once striked

Title	Date
Name	Partner

checkoff
16/10

End of Lab 2 Report

Sümeyra Gürsoy
3-12-24

Title	Lab 3 Report (p1)	Date	3/16
Name	Simon Opsahl	Partner	

~~Exer~~ Lab 3 Report

Goals:

- Further understand and modify the monitor code
- Finish getting familiar with 8051 - assembly.
- Connect multiple peripherals to the R31JP.
- Make a collection of digital waveform synthesizers
- Explore the PSoC!

Exercises:

1. Modify MiniMon
2. Analog output from digital squirrel
3. Make a sine wave generator
4. Add some more useful peripherals
5. Lissajous curves
6. Secrets of the Squirrel

Name | Partner

Exercise 1: Modify MINMON

In this exercise, we use "V" to enter a vector of bytes to create waveforms. I want to automatically print the dptr each time, then enter will escape the entry. Entering 256 bytes also escapes the entry. The result is below:

Exercise 2: Analog Output

We must overcome the sharing of P0-7 to add another IC to our R31JA. We will be adding the AD558 (JN 40609 39743), an 8 bit DAC.

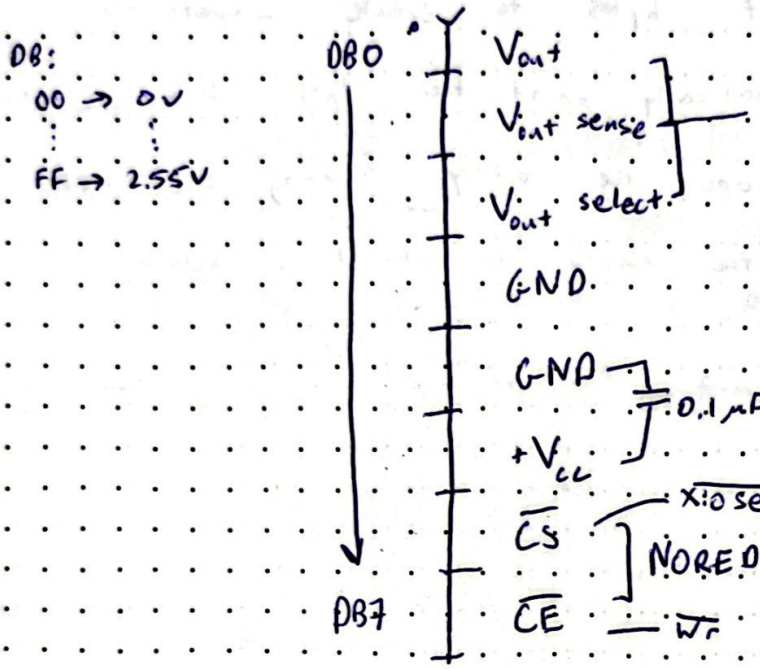
Notes:

- Operates on a single +5 - +15 V supply.
- Has ±1 LSB resolution.
- The J is for use in 0-70°C.
- The output range is 0-2.56 V

Title	Lab 3 report (p 3)	Date
Name	Partner	

Exercise 2 (cont.):

- The pinout of the AD558 is as follows



- We must connect the V_{out}'s for the 0-2.56 V

and add a bypassing capacitor of 0.1 μF

To allow for multiple ICs, we need a demultiplexer to control the $\overline{CS}$ line.

DM74LS138N:

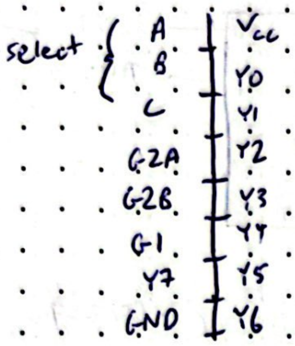
- 3 select inputs → 8 lines
- 3 enable pins:

G1 → H enables an output
G2(A, B) → L enables a low-active output

Lab 3 rep. 11 | Date
Name | Partner

Exercise 2 (cont.):

- The pinout is as below:



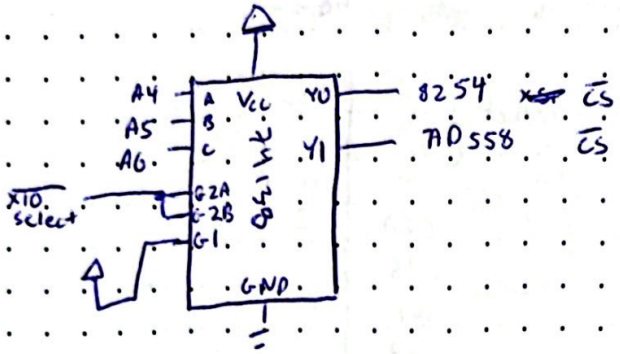
- We need G2A and G2B to be low and G1 to be high to enable the chip. This can be done by tying G1 to +5V and G2A & B to $\overline{XIOSELECT}$.

- For the select pins, we want each to be addressable by while leaving open the first ~~8~~ addresses. Thus, A4, A5, and A6 will be A, B, and C select pins.

	Addresses	Device
Y0	Fe00 - 0f	8254
Y1	Fe10 - 1f	AD558
Y2	Fe20 - 2f	-
Y3	Fe30 - 3f	-
Y4	Fe40 - Fe 4f	-
Y5	Fe50 - Fe 5f	-
Y6	Fe60 - 6f	-
Y7	Fe70 - Fe 7f	

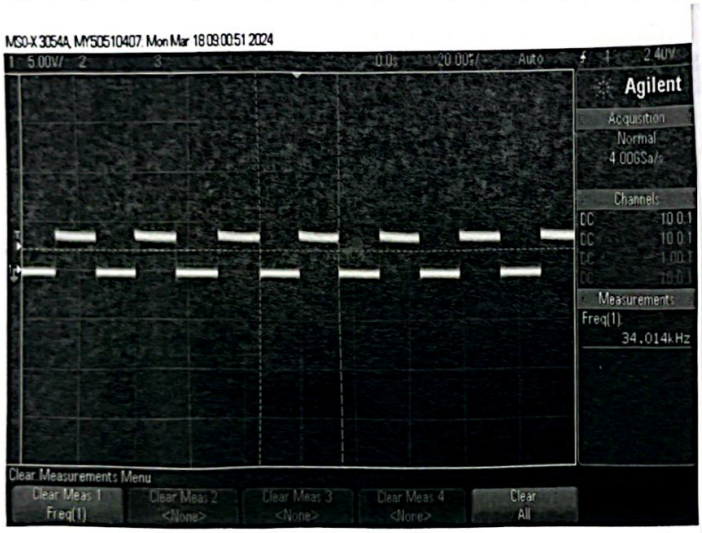
Title	Lab 3 report (p5)	Date
Name	Partner	

Exercise 2 (cont.):

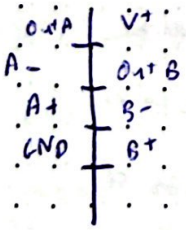


when I tried 34 kHz (WFE03=76, WFE04=26, WFE00=01)

the plot looked like the below.



In order to protect the DAC, we can use the LM358. Op Amp

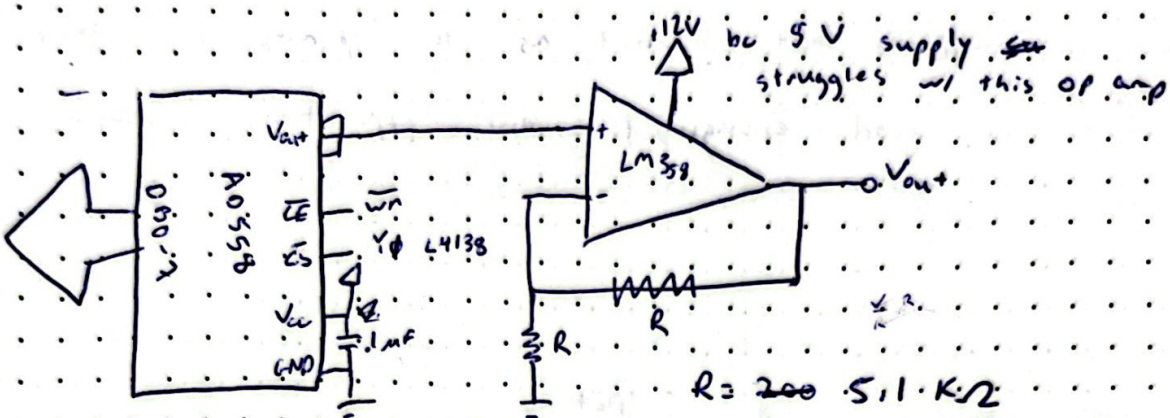


Lab 7

Name	Partner

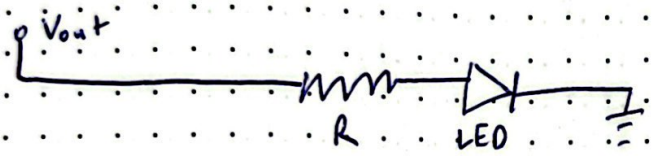
Exercise 2 (cont.)

The final pt circuit looks like the following



$R = 200.5 \pm 1 \cdot K\Omega$

we want current maxxed at 10 mA at 5 V



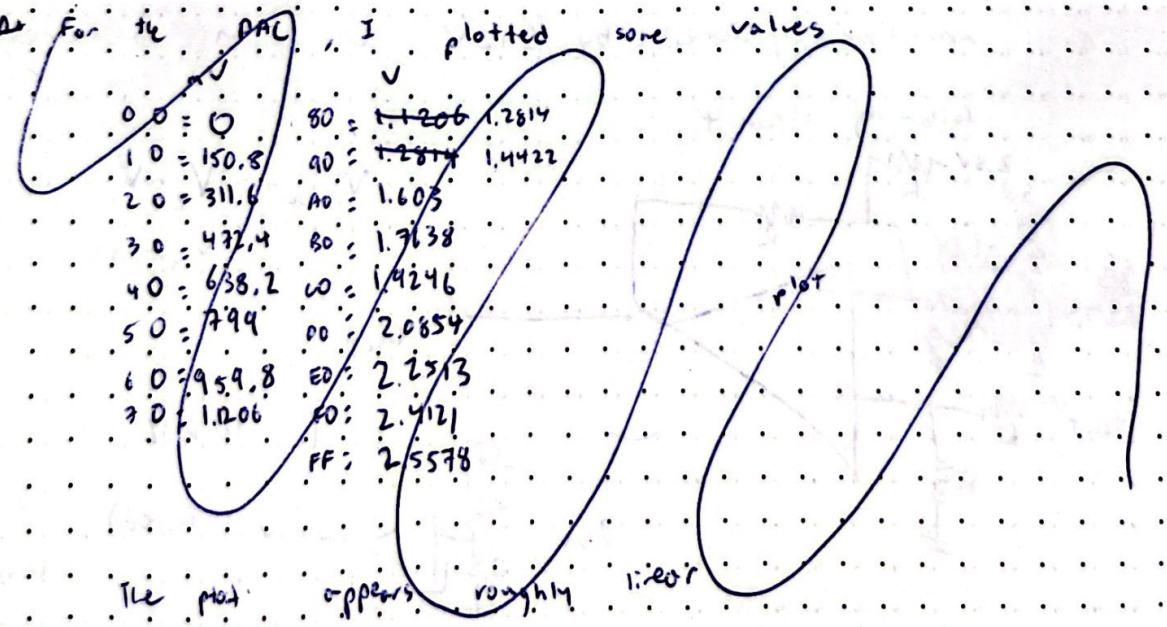
$R = 300\ \Omega$

At 5V, green LED has 2 V = V_f.

$$R = \frac{V_s - V_f}{I} = \frac{5-2}{.01} = 300.\ \Omega$$

As For the DAC, I plotted some values

	V		V
00 = 0	~~0~~	80 = ~~1.1206~~	1.2814
10 = 150.8		90 = ~~1.2814~~	1.4422
20 = 311.6		A0 = 1.603	
30 = 472.4		B0 = 1.7638	
40 = 638.2		C0 = 1.9246	
50 = 799		D0 = 2.0854	
60 = 959.8		E0 = 2.2513	
70 = 1.1206		F0 = 2.4121	
		FF = 2.5578	

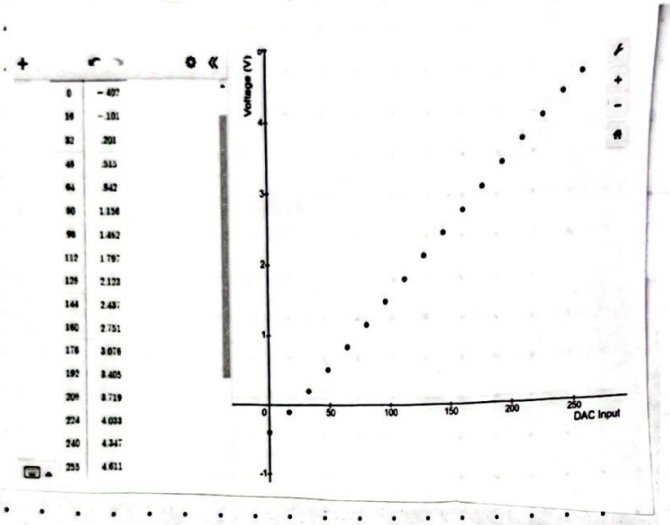


The plot appears roughly linear

Title	Lab 3 report (p#)	Date 3/18
Name	Partner	

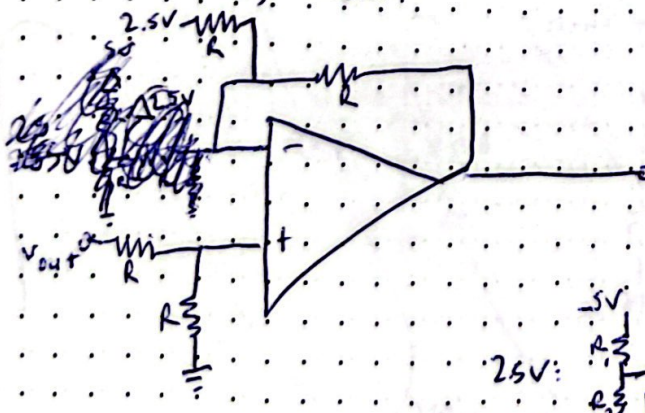
Exercise 2 (cont.):

I plotted the results on desmos. Note: the range is not perfect, as the op amp is cheap. It is good enough for our applications, however.



Exercise 3: Sine wave generator

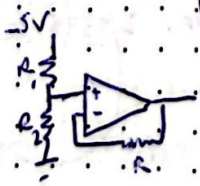
We can use another LM358 to make a difference amplifier, powered by ±12V. We want to create the following circuit:



$V_{offset} = V_2 - V_1$

$= V_{out} - 2.5V$

$R = 10\,k\Omega$



$R_1 + R_2 = 10k$

Used a po

Lab 7		Date
Name	Partner	

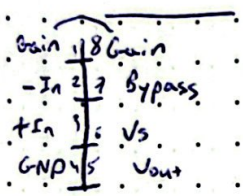
Exercise 3 (cont):

The result is a variable amplitude between -2.5 and 2.5 V. The actual output is -2.6 to 2.4, within a reasonable range for our application. By using a 10K Ω POT for the offset, we can tune this to be in range

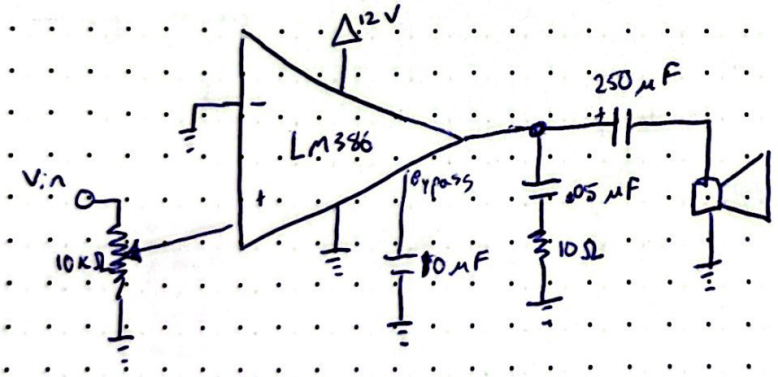
---

3.19.

To make the audio amplifier, we can use the following circuit:



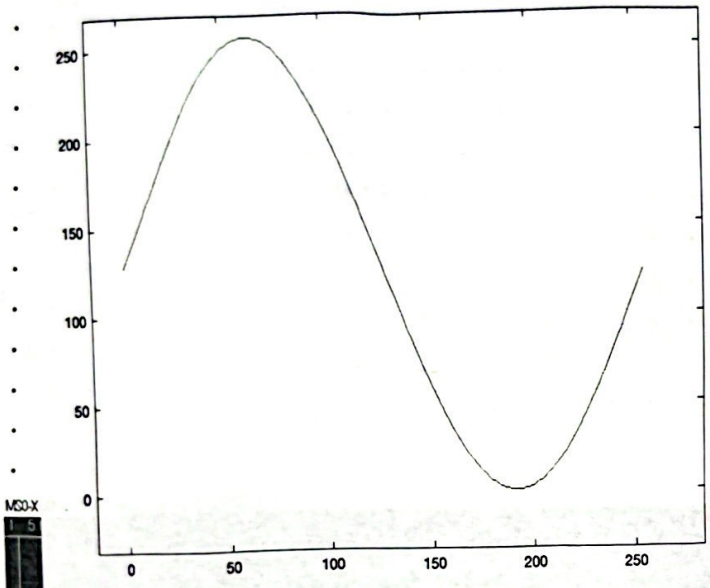
Initially set to 20
ranges 20-200

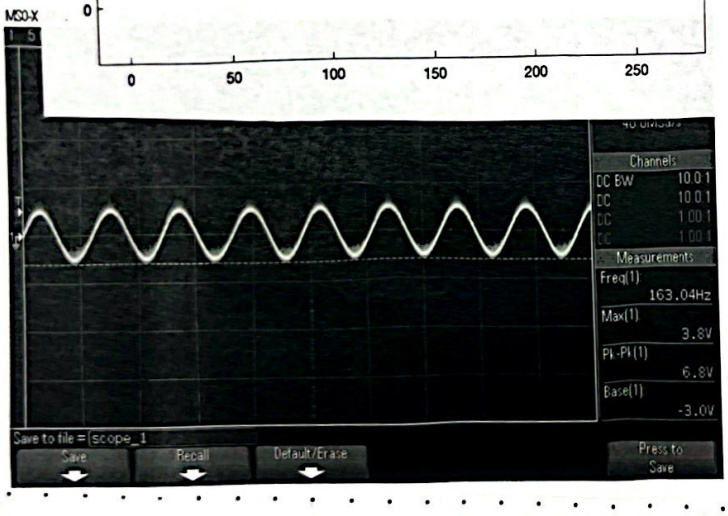


Title Lab 3 report (p9)	Date
Name	Partner

Exercise 3 (cont.):

When I run the generated sine wave (generated in matlab), the below plot is what I get on the offset op-amp.

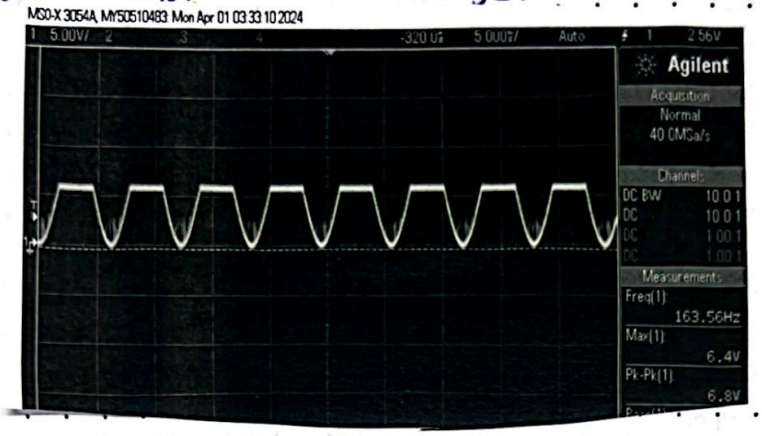




Title	Date
lab 3 report (p 10)	3/31/24
Name	Partner

Exercise 3 (cont):

When the output is connected to the speaker, the output saturates around 6.5 V, which is good enough for our range.



The output gets more grainy as we increase the volume and the output saturates.

Timing around 163.64 Hz

	# cycles
lcall	2
mov	1
inc	1
push	2
push	2
mov (to dptr)	2
movc	2
pop	2
pop	2
ret	2
movx	2
sjmp	2
	= 22 × 12

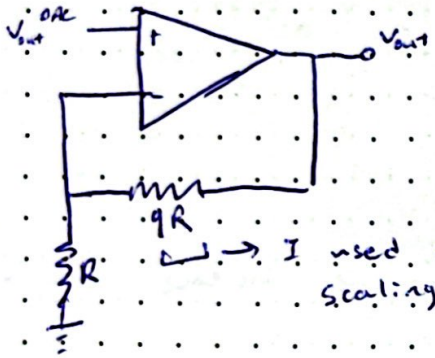
} done 256 times for the 256 entries

$$f = \frac{11059200}{22 \times 12 \times 256} = 163.64 \text{ Hz}$$

Title	Lab 3 report (p11)	Date
Name	Partner	

Exercise 3 (cont.):

In order to complete the 10-fold scaling, we can add div a,b, where b holds 10 before sending to the DAC. The scaling op-amp must look like the following:

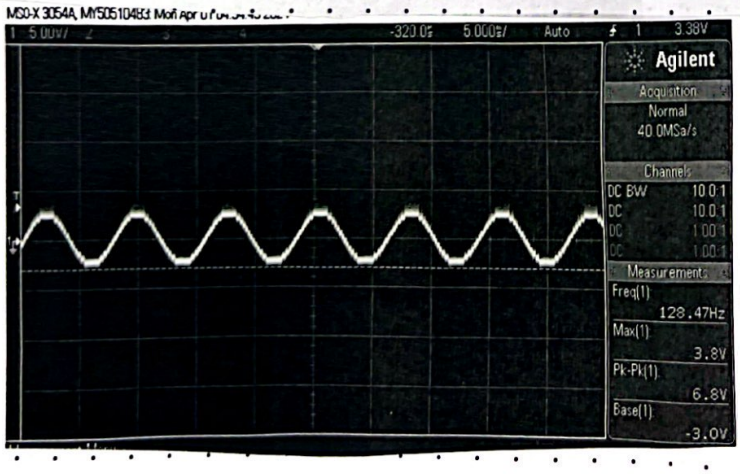


$V_{out} = V_{in}\left(1 + \frac{R_2}{R_1}\right) = 10\,V_{in}$

$R = 3.3\,K\Omega$

→ I used a pot here to tune so the scaling works

When we connected the output to the scope, we see something with more step sizes, or more discrete.



Title Lab 3 report (p 12)	Date
Name	Partner

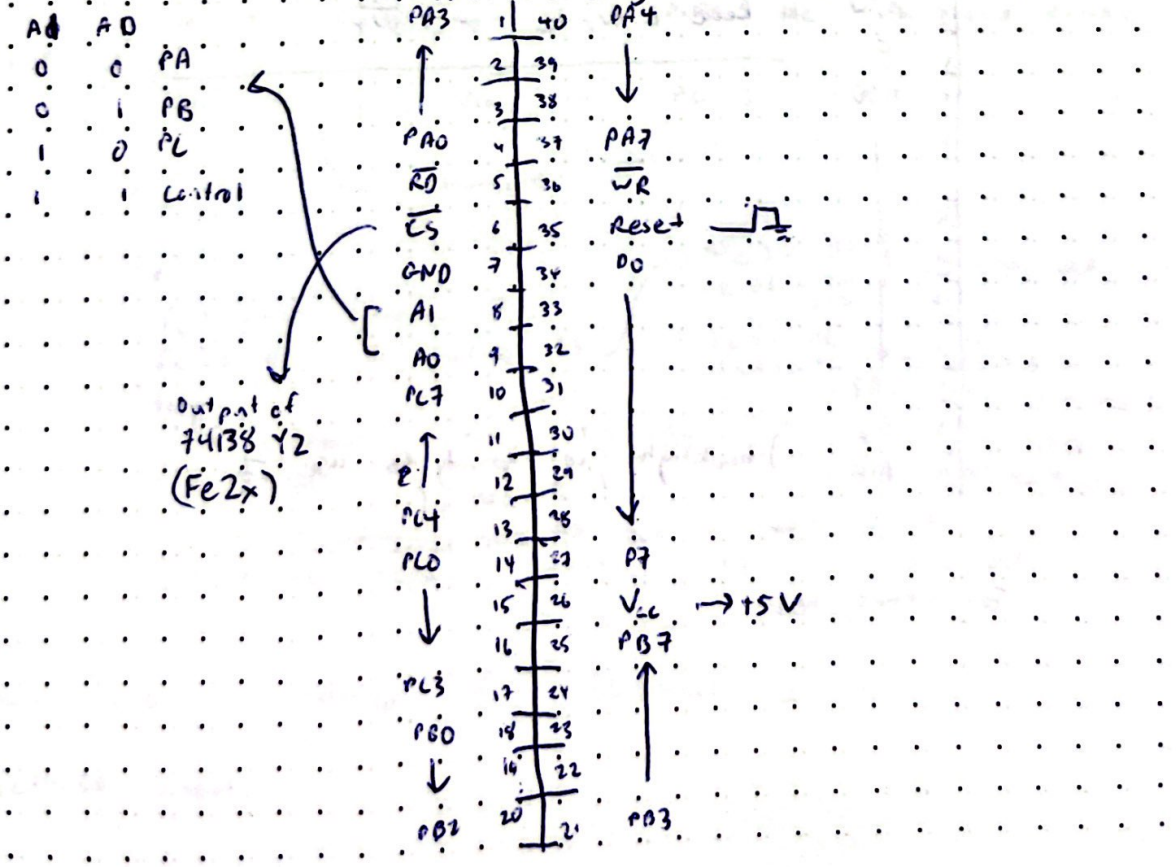
Exercise 3 (cont):

The new sound output is even more grainy, but behaves similarly to the result before. The wave shape appears more discrete because we cannot track fractions. The new freq, 128 Hz, corresponds to the addition of 6 more operation cycles per loop. We added div and mov b, #div, which add 4 and 2, respectively.

Exercise 4: Adding more peripherals

The 8255:

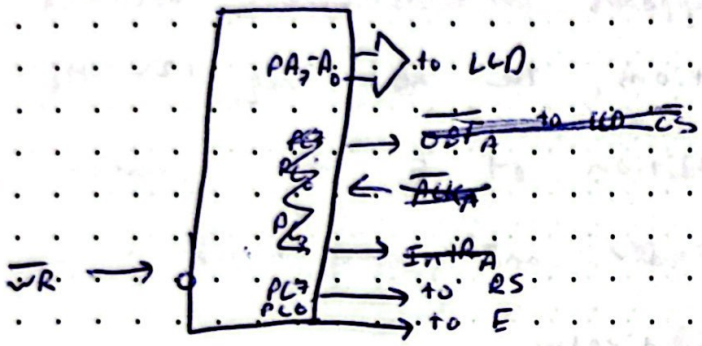
This is essentially adding 3 new P1 outputs.



Title	Lab 3 report (p13)	Date
Name	Partner	

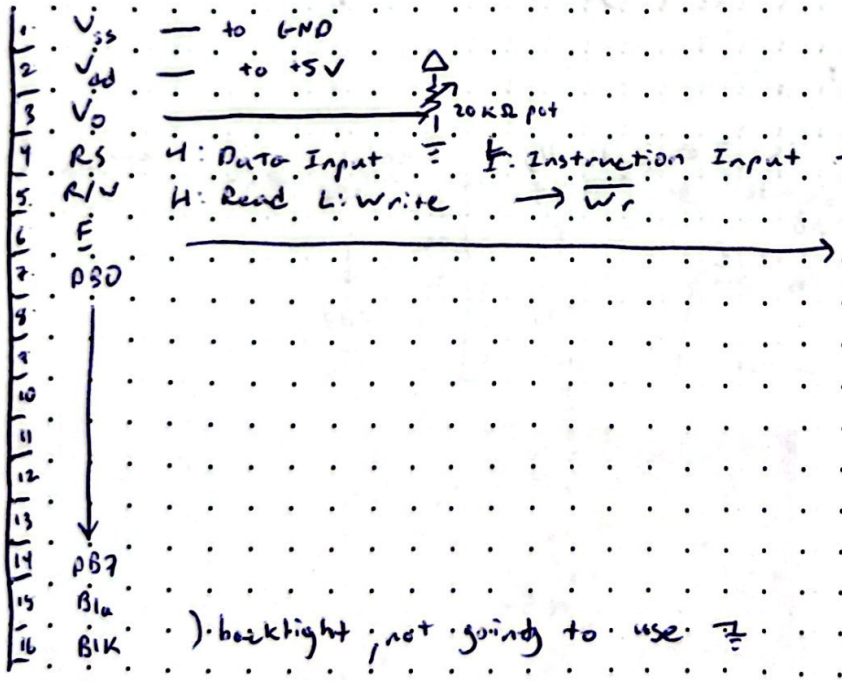
Exercise 4 (cont.):

In mode 0 (output), we can use port A and the following configuration:



FE23=
mode 0: 81

For the LCD (1602A-1), we can use the following pinout:



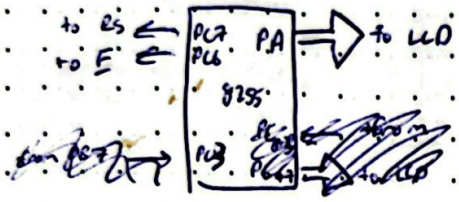
~~To start for~~

Title Lab 11 | Date

Name | Partner

Exercise 4 (cont.):

The intuition is that we will have PA send info to the LCD, PC6/7 control whether we have data/instructions and whether we are enabled. PC3/2 will be inputs from DB7 for the RD busy flag. I will use RT delay instead



To start, we use CW SI at Fe23, initializing mode 0 with A, B, and C4-7 output and C0-3 input. We then set PC7,6 to 10. We then write to PA with the character. We then write the enable, or 11 to PC76. We then and enable. We then RD the busy flag in PC0-3, until it's 0. We then write the next character.

```
init:
    Fe23 = 80    mode 0
    Fe22 = 00    initialize RS
loop:
    Fe20 = char  writes the character
    Fe22 = C0
    Fe22 = 80    ) pulses E
    Fe22 = 00    sets RS low to read out bf
    wait until Fe22b3   waits until busy flag set
    jump loop
```

Fe20 = 38, Fe22 = 40, 00) 8x7 char set.
Fe20 = 0C, Fe22 = 40, 00) display on, no cursor.
Fe20 = 01 clr display
Fe20 = 80 set RAM to 0

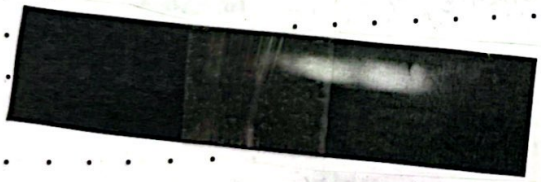
There are a number of additional initial steps needed, detailed above.

Title	Lab 3 report (p15)	Date
Name	Partner	

Exercise 4 (cont):

The result is below. The output, "6.115 Rules!" is as follows

	output
6	36
.	2E
1	31
1	31
5	35
	20
R	52
u	75
l	6C
e	65
s	73
!	21
.	00 term

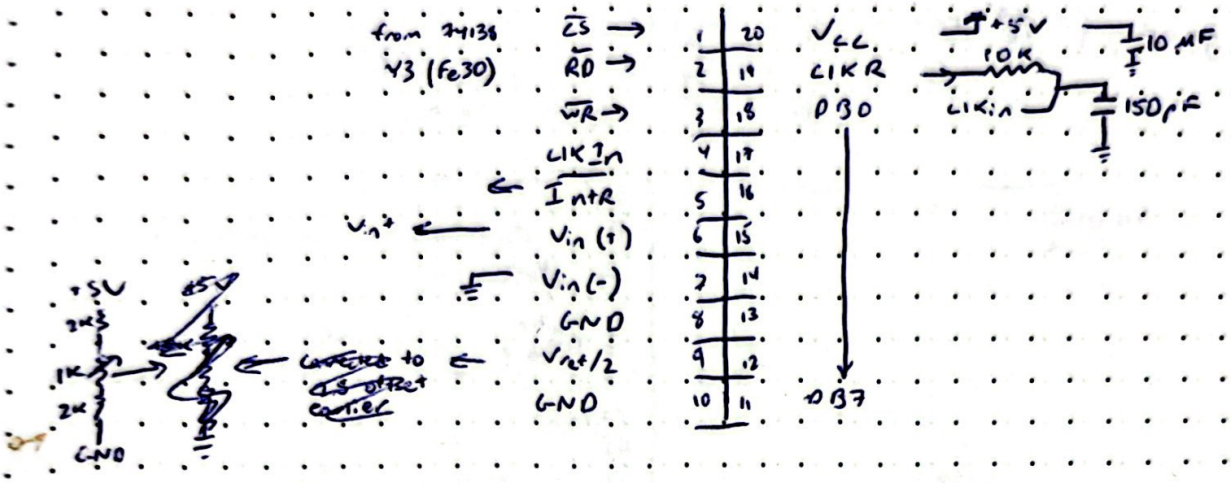


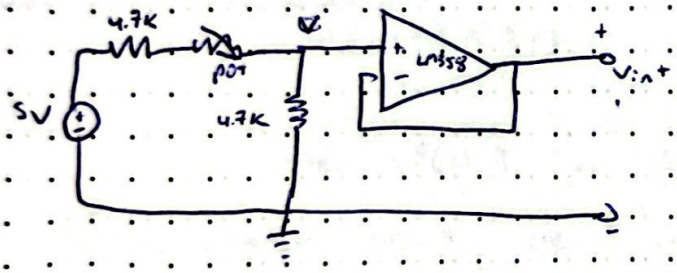
It's faint but it's there

Title Lab 3 report (p16)	Date
Name	Partner

Exercise 4 (cont.):

The ADC0804:





Testing circuit:

R_{pot} = 0–100 kΩ

should range 0–25 ✓

To display on the LCD, I used the same code as the lcd display before, but it loops by measuring voltage. The voltage is clocked, then uploaded, then clocked again... the output is like follows:

x 0 x . V

30 2E 30 20 56

The output ranges 0-255 for 0-5 V. Dividing by 51 will give the ones place, and dividing the remainder by 5 will give the tenths

Exercise 4 (cont.):

We can see the resolution is pretty good, ie:

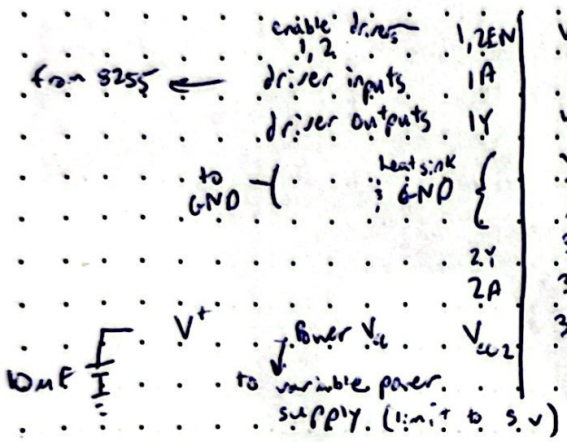
Lab 5 | Date 4/11/24

Name | Partner

Exercise 4 (cont.):

The L293D:

- A driver (quadruple high-current half-H)
- Bidirectional drive currents ≤ 1 A
- Voltage range 4.5 – 36 V



We can use PC0-3 on the 8255 to drive xA. Nothing changes about the cw (80), we just need to write to FE22 to get our desired output (+V/0).

Exercise 5: Lissajous Curves

The schematic for the Lazerdillo is attached on the following page.

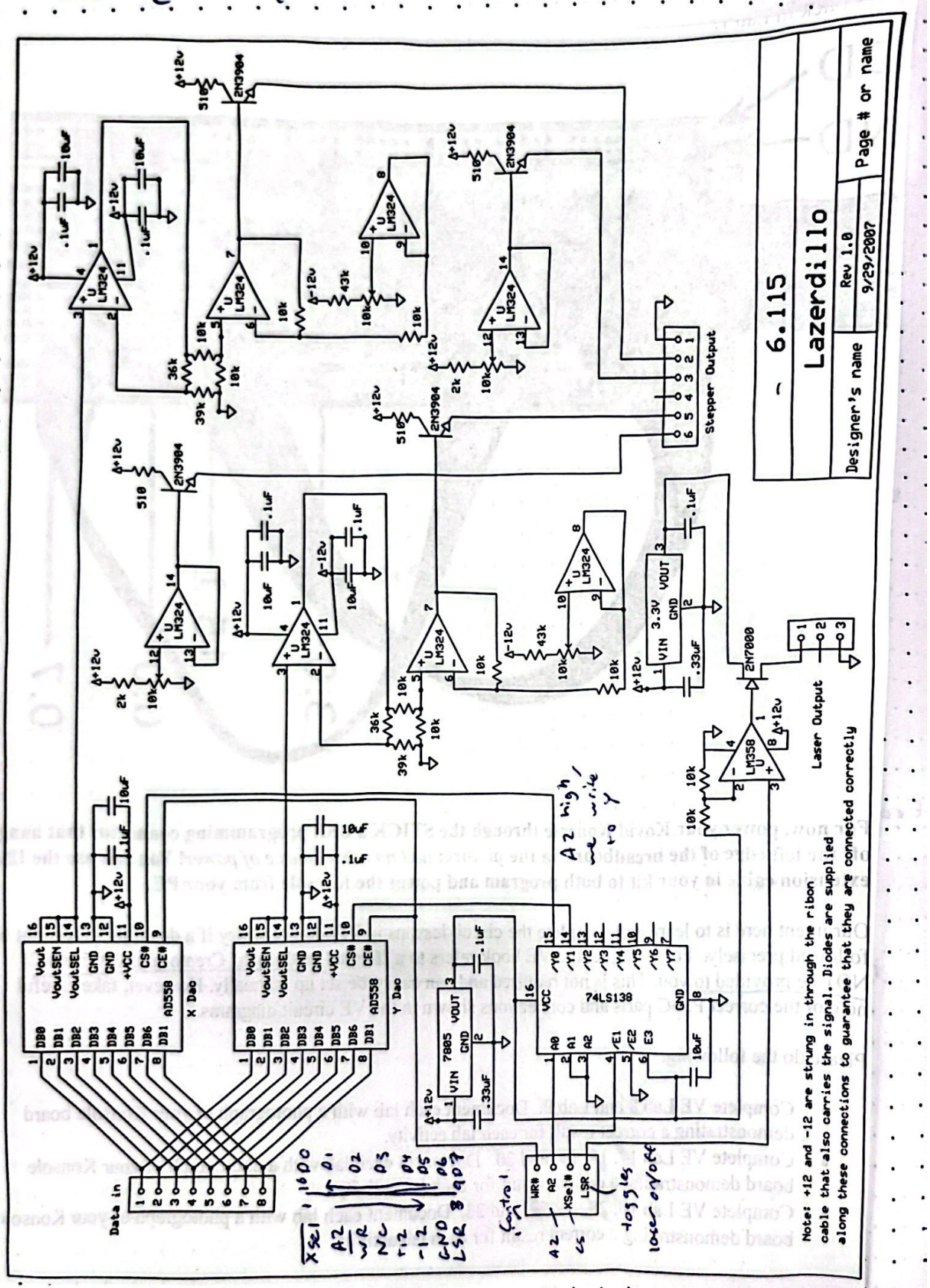
Lissajous curves:

$x = A\sin(2\pi at)$ $y = B\sin(2\pi(bt + \phi))$

◯ $\Rightarrow \frac{a}{b} = 1$, $\phi = \frac{1}{4}$ ⟋ $\Rightarrow \frac{a}{b} = 1$, $\phi = 0$

Title	Lab 3 report (p19)	Date
Name	Partner	

Exercise 5 (cont):



Title Lab 3 report (p20) | Date

Name | Partner

Exercise 5 (cont.):

- The mirrors cannot go above 80 Hz.
- The figure must be completed in 1/20 th of a second or we see movement.

Interfacing:

- using the 256 byte sine wave table and
  movx a, @a+dptr
- Table advance by frequency (how much we inc each time):
  - Timer 0, TH0 = #04Ch
    + we need to leave enough time for computation
  - Leads to 5120 interrupts/second
  - for 80 Hz, inc by 4 = $\frac{256}{5120} \cdot 80$
  - for 54 Hz, inc by 2.7 ? what do we do ☺
    + use a 16 bit count
    + add bottom (fractional count), then add top with carry
  - we can add rotations by offseting freq. by 1 or ... ie.

$x = \sin(2\pi \cdot 54t)$ $y = \sin(2\pi \cdot (81t - t))$

rotates one a second

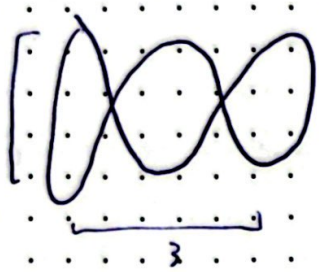
Title Lab 3 report (p21) | Date

Name | Partner

Exercise 5 (cont.):

A2 selects between X & Y

Tie Tie fighter:



$\frac{a}{b} = \frac{1}{3}$

3

we can use 80 and ~~64~~ as our values

$\frac{256}{5120} \cdot 80 = 224$ $\quad$ $\frac{256}{5120} \cdot \frac{80}{3} = \frac{4}{3}$

b a

~~02 0110 0110~~
~~0266h~~ 0400h

00 1100 1100 1.01010101
CCCh 0155h

~~When we scope x, and y, we get~~

~~IV~~ found that this doesn't result in perfect matching, so I'll use

$\frac{1}{20} \cdot x$ and $\frac{1}{20} \cdot \frac{y}{3}$ to find good values

offset: 60 3 and 1
y:60 x:20

Note: we need a phase shift of $\frac{\pi}{4}$ to x to make the shape

Lab 3 Prep

Name | Partner | Date

Exercise 5 (cont.):

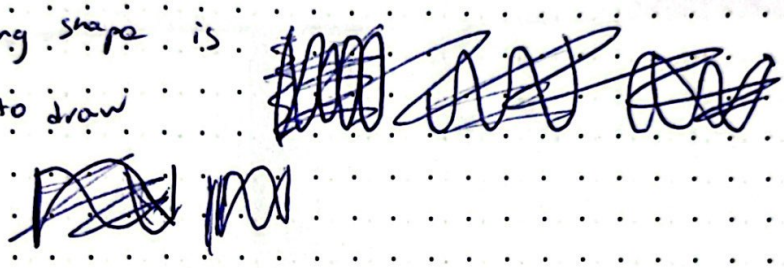
Circle: $\frac{a}{b} = 1$ $\phi = \frac{1}{4}$ (start y at 40h)

$a = b = 80$, offset = 4

$a = 1$, $b = 4$, phi = 0

b = 80 Hz → 4 starts at 0000 0000
a = 20 Hz → 1

The resulting shape is hard to draw



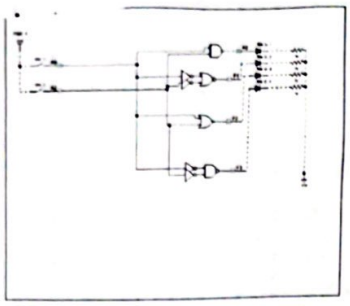
To rotate, we can use a offset on ~~b~~ to make it rotate faster. phase offset of 1 => 1 Hz. phase offset of 1.5 leads to 1.5 Hz. I will use ~~0x50~~ 1.25 instead of ~~0x20~~ to get the tie fighter rotation. ~~59.50~~ 58.75 / 20 = ~~3~~ 3 - ~~1.25~~ / 20 = 3 - 0~~8~~ = 3 - ~~0.125~~ = 02EF

.25
.0625
20
1.625
1.25

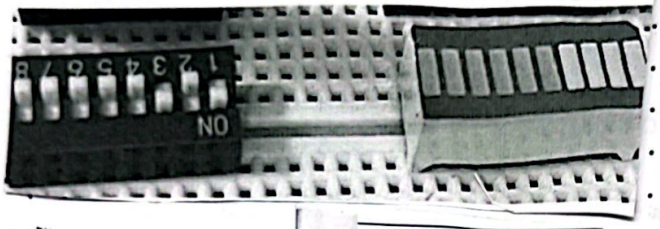
Note: I also added a rotation up/down button to the keypad relay!!

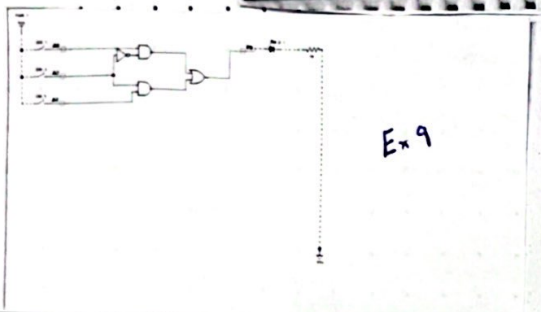
Title	Lab 3 report (p231	Date 4/27/2
Name	Partner	

Exercise 6: PSoC11

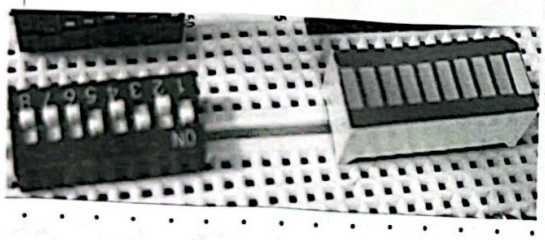


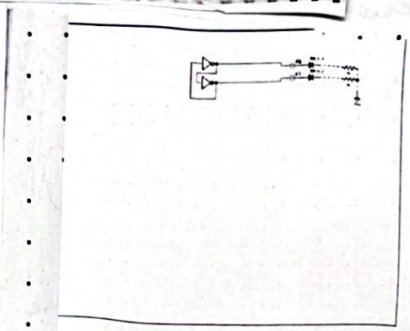
Ex 8



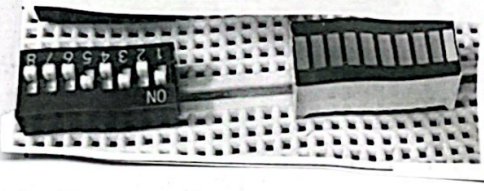


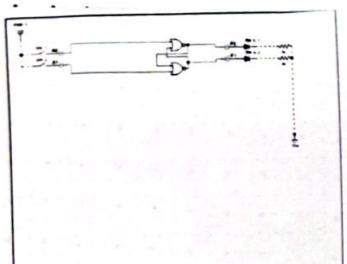
Ex 9



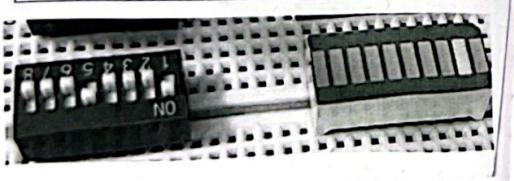


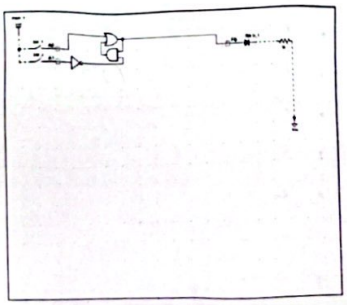
Ex 17



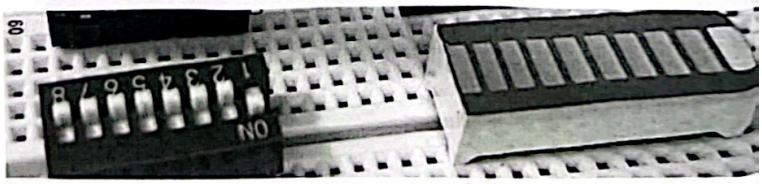


Ex 18



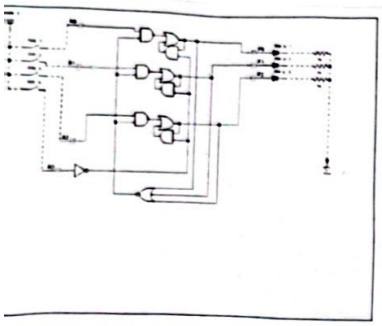


Ex 19

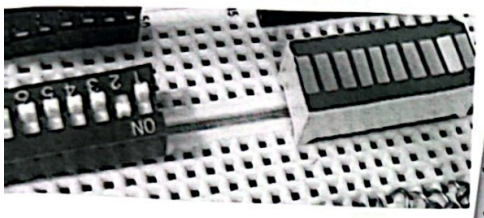


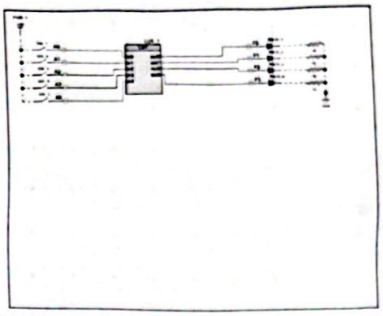
Partner

Exercise 6 (cont.):

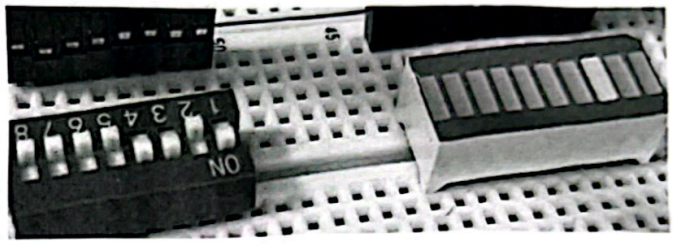


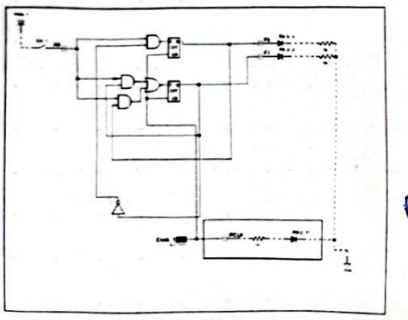
Ex 20



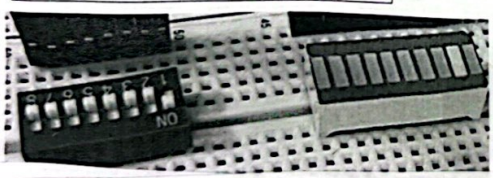


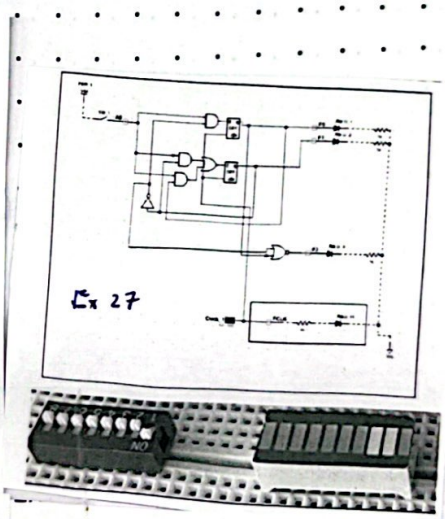
Ex 16



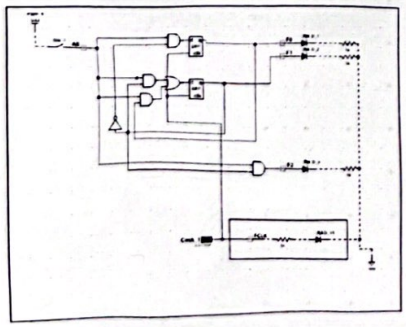


Ex 25

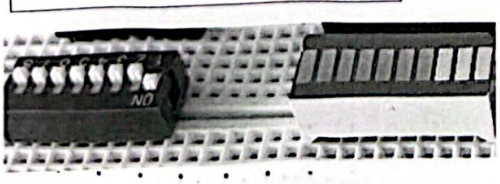


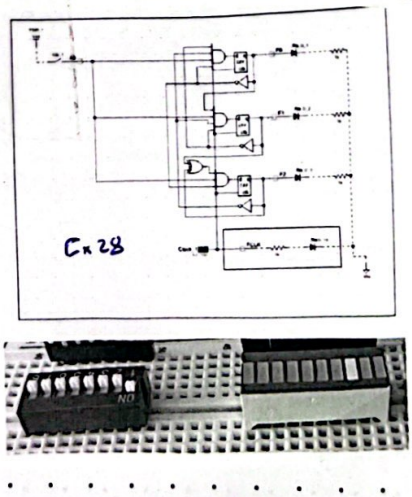


Ex 27



Ex 26





Ex 28

Title	Date
Name	Partner

End of report

Lab 3

Checkoff: Savannah Gordon

10/10

SGJ

Simon Opsahl
4-2-24

6.115 Lab 4

Goals:

1. Learn how to use peripherals on the R31JP
2. Use an A/D converter and LCD screen
3. Learn about electromagnetic actuators
4. Understand, design, and construct a digital feedback controller.
5. Practice writing C code for the PSoC.

Exercises:

1. The relay (a simple linear actuator)
2. Get to know the DC motor
3. Pulse Width Modulation
4. Model the DC Motor
5. Play with the robot arm
6. Feedback Control for a 1st order plant
7. Stepper motors
8. PSoC
9. SpinDude Image Reconstruction

Name	Partner

Note:

I rewired my kit to allow for better wire management. Attached is...

① Keypad and 74C922N:

- Output connected to P1.0-3
- OA connected to —

② ADC0804

- ~~located at FE10h~~ located @ FE10h
- Intr → P3.5

③ DAC (AD558JN + LM358N)

- Connected to FE00 and adjustable gain block.

④ 8255

- PA0-7 : LCD D0-D7
- PB : open
- PC0-3 : 1A-4A on Driver
- PC4-7 : open, ~~R/W~~ E, RS on LCD

⑤ DM74LS138N (multiplexor)

FE00 : DAC

FE10 : ADC

FE20 : 8255

Lab B

Name | Partner

Note (cont.):

⑥ L293DNE (Driver)

- 1A-4A connected to 8255 PC0-3

⑦ LCD screen

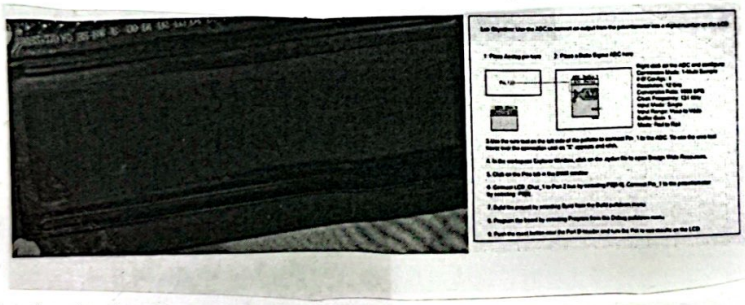
- D0-7 connected to 8255 PA0-7

- RS, ~~RW~~, E connected to 8255 PC7-~~5~~6

~~RW~~ → GND.

Exercise 8: PSoC II 4/12/24

Exercise 2: (I ~~connected~~ shorted R70 ~~+3V/0~~ to A1K) to see the result for a photo) (ADC output from POT)

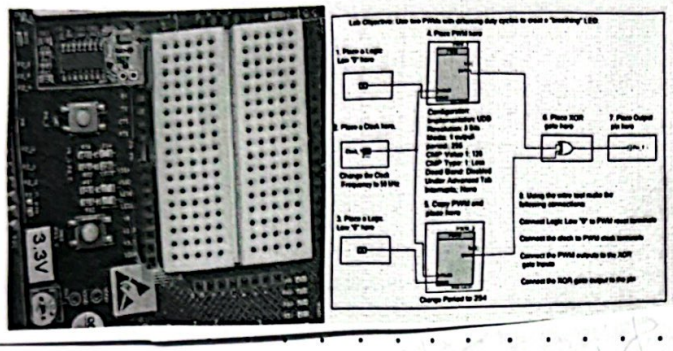


Exercise 3: "Breathing" LED

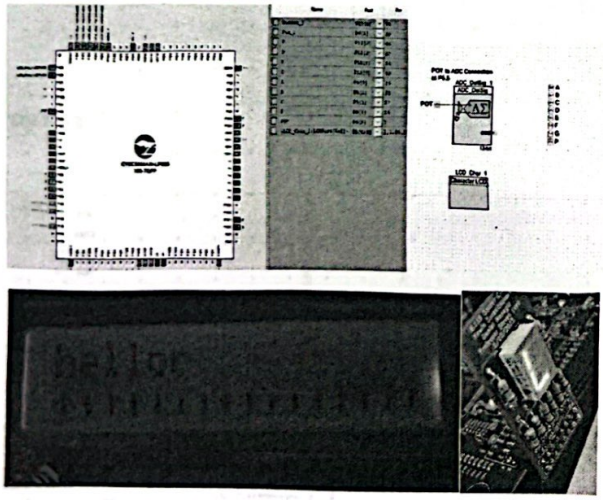
- ~~I made PWM 1 have a duty cycle of 0.5 and PWM 2 a duty cycle of 0.5. I connected the pin to~~ PWM 1 & 2 have duty of 128, but 2 has period of 255. Pin 1 is connected to P6.2.

Title	Lab 4 report (p4)	
Name		Partner

Exercise 8 (cont.):



Custom Design:



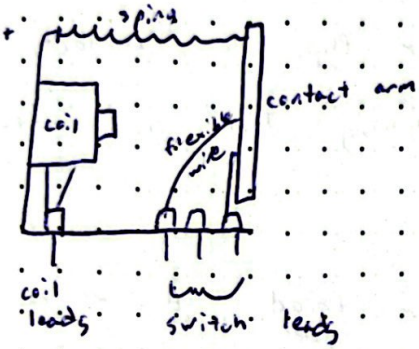
Lab 5 4 report (p 1)

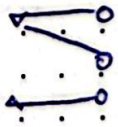
Date 4/08/24

Partner

Exercise 1: The Relay!

Relays: (From sites and instructions)

- A current sent through a coil magnet pulls a flexible, spring-loaded conductive-plate from switch contacts (for mechanical relays).
  - + Allow for high currents and slow switching.
  - + 

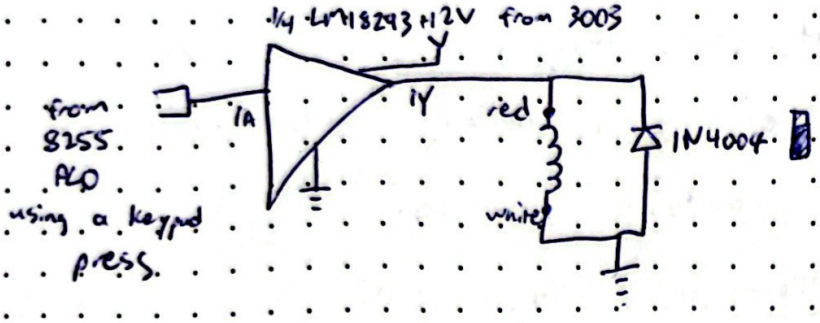
- Ours is a subminiature relay
  - + rated for 12 V, ~160 Ω coil resistance
  - + The coil acts as an inductor, meaning - high voltage spike when current cut
  - + As a result, we need a diode (1N4001)
  - + Our relay is SPDT 

Turning on power supply should flip from one switch to another

Title Lab 4 report (p6)	Date
Name	Partner

Exercise 1 (cont.):

This is the schematic:



Toggling the power flips the contacts from one two the other. The leads switching does not affect the behavior of the relay.

Note: I believe I fried 2A w/ a 12V supply, so I will avoid using it. *I replaced the chip

---

4/9/24

Exercise 2 ~~(cont.)~~: DC Motor

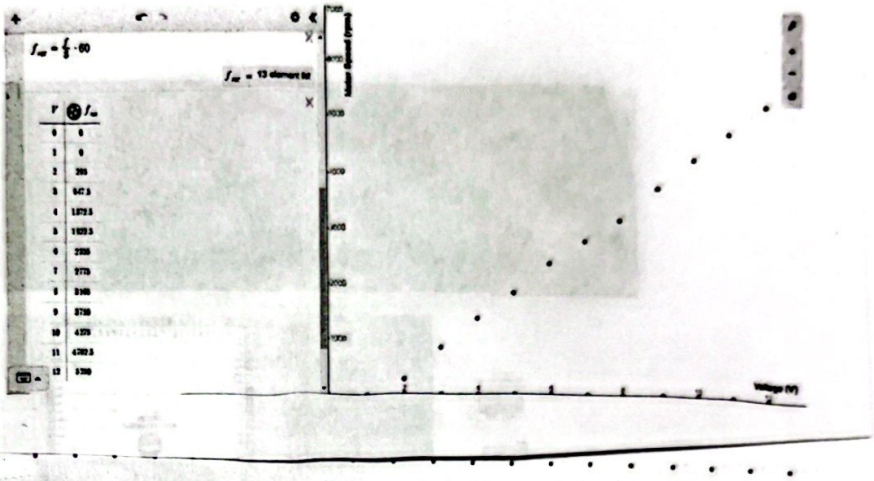
Our motor has a brushed DC motor and an AC generator from a spinning magnet disk.

- The AC generator frequency is 8x the rotation frequency.
- The AC generator is used to measure rotation frequency, important for feedback systems
- We will connect to lab power, and connect the AC generator to the oscilloscope.

Exercise 2 (cont.):

Pinout
- AC Generator → to oscilloscope
- V+, V- → 3005 power supply

V	f	I	I_{load}	f_{load}	V_{drive}
0	0	—	—	—	0
1	0	—	—	—	0
2	38	—	—	—	1
3	113	—	—	—	2.6
4	163	—	—	—	4.26
5	243	—	—	—	5.63
6	314	—	—	—	7
7	370	—	—	—	8.8
8	422	—	—	—	10
9	500	—	—	—	11.8
10	570	—	—	—	12.9
11	635	—	—	—	13.7
12	700	.22	.4	630	15.3



When we apply a load, the shaft speed goes down, and current goes up. This implies that voltage drop affects the speed of the motor, while current affects the amount of force/torque the motor can apply.

Exercise 3: PWM

Instead of using the DAC for variable drive voltages, we can use PWM with different duty cycles and a binary waveform.

Name | Partner

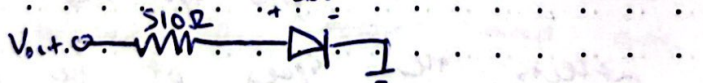
Exercise 3 (cont.):

The average V is DV_s, where D is the proportion of the period spent at V_s: $f_{sw} = \frac{1}{T}$. The waveform should repeat every T seconds, spending DT high and (1-D)T low. By adding a low-pass filter to the load, we can extract the DC component of the average voltage. This is a Buck converter.

To test, we can make a low frequency system (0.5 Hz) with D=.5 and high frequency (5000 Hz) with D=0.5.

~~I will disconnect 4A from PC3, and instead connect it to PC4 so that I can use a direct~~

Timing: ~~0.5~~ 5k Hz, can use 8 bit timer to get a square wave



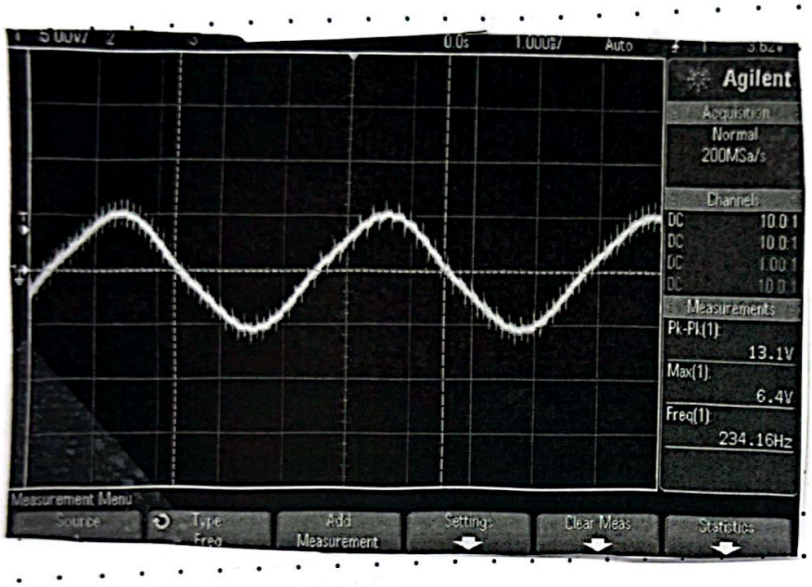
In the high freq case, we have the light appearing consistently lit. The diode is the low pass filter. To make the 0.5 Hz square, I used a 10,000x decrease in period by nested counters. The led flashes every other second.

Partner

Exercise 3 (cont.)

When we instead connect V_{out} to the DC motor, we get the following responses:

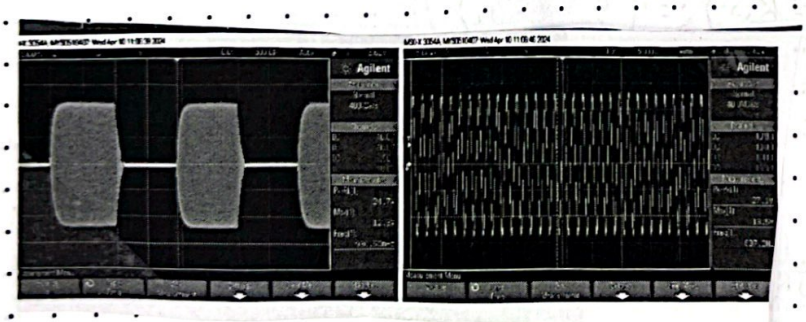
High frequency:



freq = 250 Hz

$|V| = 6.8$ V

Low frequency:



freq = 0 Hz or 637 Hz

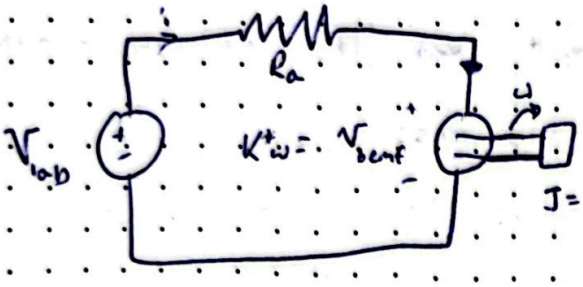
$|V| = 0$ V or 12 V

The motor is on/off in the 0.5 Hz situation, reaching SS in both before the next step. In the high freq case, we have a voltage output and frequency roughly equivalent to 5-6 V, half of our drive. The coil is our low pass filter.

Title Lab 4 report (p10) | Date

Name | Partner

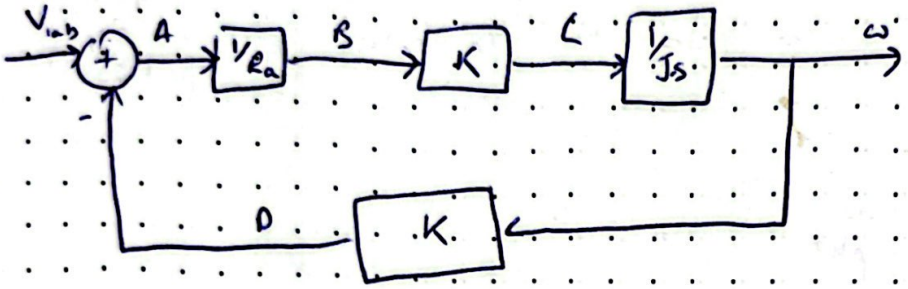
Exercise 4: DC Motor model



Torque = K · i

J = inertia

* ignoring inductance and friction



$A = V_{iab} - K\omega$ (= voltage across resistor)

$D = K\omega$ ($\in V_{bemf}$)

$B = \frac{V_{iab} - K\omega}{R_a}$ (= current)

$C = \frac{K}{R_a}(V_{iab} - K\omega)$ (= torque)

$\omega = \frac{K}{R_a(J \cdot s)}(V_{iab} - K\omega) = \left(\int \frac{\text{torque}}{J}\, dt\right)$

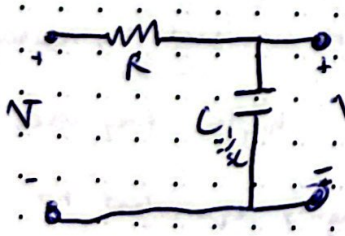
$\omega\left(1 + \frac{K^2}{R_a J s}\right) = \frac{K}{R_a(Js)} V_{iab}$

$\omega = \frac{K}{1 + \frac{K^2}{R_a J s}} \cdot \frac{1}{JR_a s} V_{iab}$

$\omega = \frac{K}{JR_a s + K^2} V_{iab}$

$\omega = \frac{1}{K + \frac{JR_a}{K} s} V_{ab}$

$\omega = \frac{K}{JR_a s(1+K)} V_{iab}$



$V_{out} = \frac{1/sC}{R + 1/sC} V$

$V_{out} = \frac{1}{1 + sRC} V$

Exercise 4 (cont.):

The two circuits (RL and DL motor) differ in a number of ways. First off, at low frequencies (DC), the angular speed is $\frac{1}{K} V_{tach}$, where V_{out} is just V. They are both proportional to input voltage however. Both circuits attenuate higher frequencies.

4/11/24

Exercise 5: Robot Arm

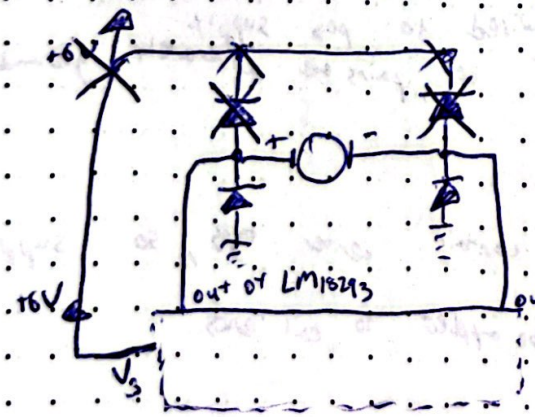
-we have 5 joints, maximally driven at 6 V

Wrist +	1	9	Gripper +
-			-
Shoulder +			NC
-			NC
Base +			NC
-			NC
Elbow +			NC
-	8	16	NC

We will control two of these joints with the L293

I will use the Base + ~~gripper~~ ~~forearm~~ elbow

Per the wiring, the circuit per each motor should be



~~Gripper~~ ~~forearm~~ elbow

+ −

1A 2A

3A 4A

+ −

Base

-For the keypad, A-D will be each of the motors

Title Lab 4 report (p12)	Date 4/11/24
Name	Partner

Exercise 5 (cont.):

- 0 will be 1 ~~0~~ Duty cycle, 1 is 0.1, 2 is 0.2, and so on and so forth.
- Closed loops are better in industrial arms, as we don't have to worry about predicting the exact location of the ~~ser~~ arm after x amount of input, we can just respond to the way the arm is behaving in the system as a whole.

Exercise 7: Stepper Motors

4/15/24

Stepper Motors respond to square waveforms, useful because position can be controlled.

- our stepper motor steps 15° each step

Unipolar:

- Center taps are wired to pos supply.
- The two ends of coil pairs are alternately grounded

Bipolar:

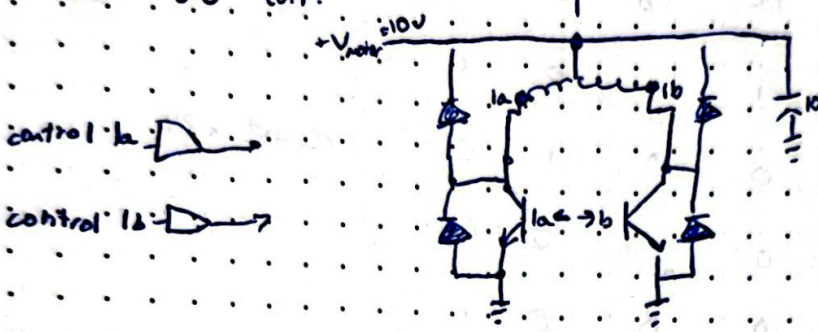
- They do not contain center taps, so supply voltage must be applied to coil ends.

Partner

Exercise 7. (cont.):

To drive stepper motors, you need freewheeling diodes and bypass capacitors.

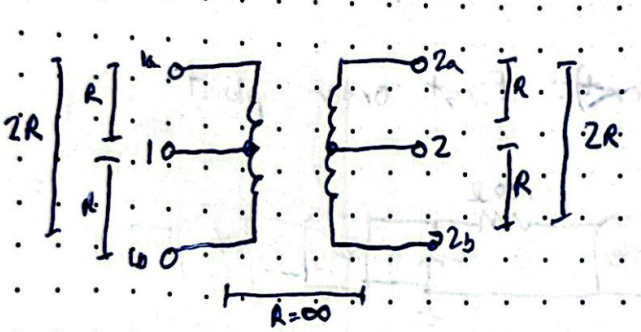
For one coil:



6 total diodes needed for the unipolar motor

We have a unipolar motor (6 leads)

To identify, we have



1 →
1a →
1b →
2 → Gr
2a → Br
2b → B

Spin dude:

	A_0 1	24	S_0	
	2	23		
	3	22		
	A_3 4	21	S_3	
	$\bar{E}_1$ 5	20	X	
	GND 6	19	GND	
	V_{cc} 7	18	V_{cc}	
	GND 8	17	GND	
	GND 9	16	GND	
1 —	R_D 10	15	B_r	— 2a
1a —	Y_E 11	14	B_g	— 2b
2b —	O_r 12	13	Gr	— 2

will only connect pins 10-15 for this exercise

Title Lab 4 report (p14)	Date
Name	Partner

Exercise 5 (cont.):

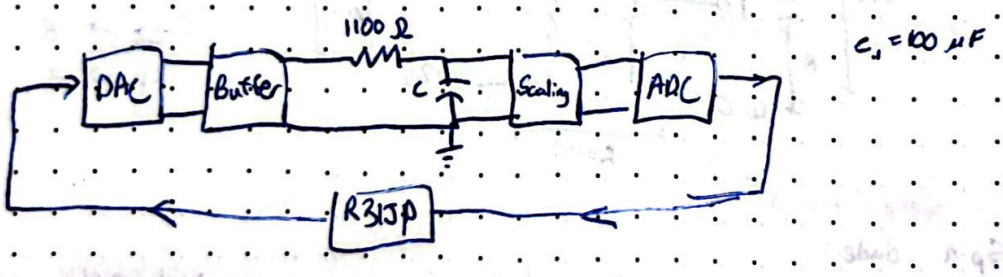
In order to rotate spin dude, we must follow the following step routine:

1a	1b	2a	2b
1	0	0	0
0	0	1	0
0	1	0	0
0	0	0	1

repeat x24

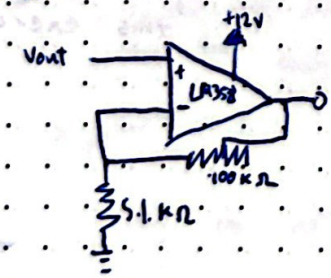
Delay

Exercise 6 (cont.): First order plant



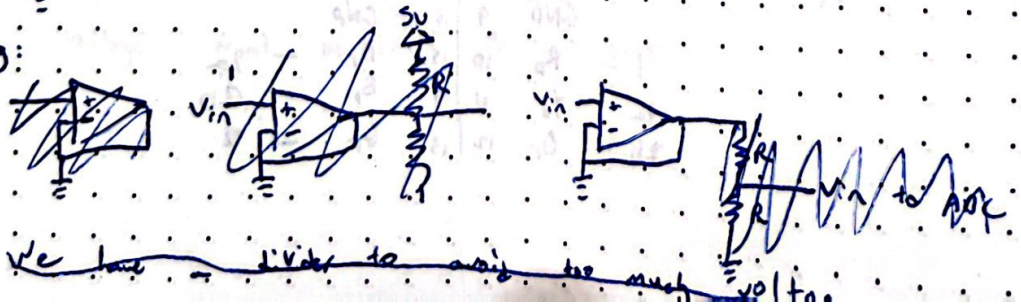
$C_1 = 100\ \mu F$

Buffer:



Tuned for +5 V

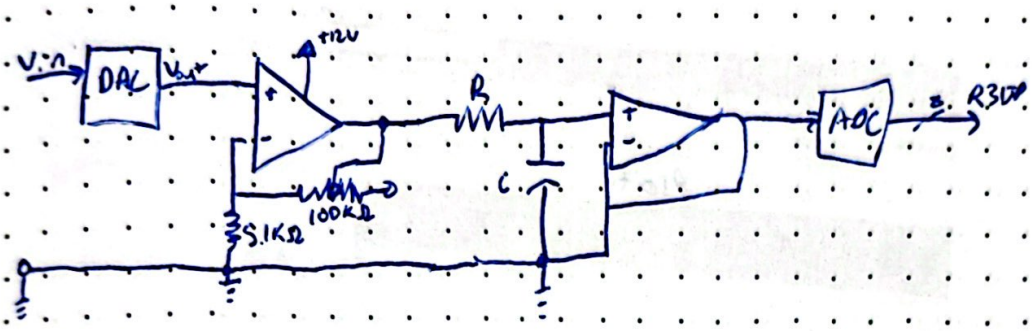
Scaling:

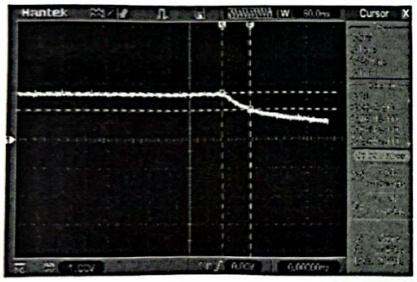


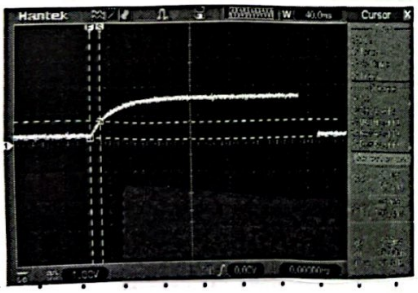
We have a divider to avoid too much voltage

Exercise 6 (cont.):

Total schematic:







There appears to be a faster rise time than fall: (more resolution in positive error)

$V_{fall} = 1.64\,V \cdot (1-36.8\%) = 1.04\,V$

~~1.64V · 36.8 = .6V~~ 1.4V · .368 = .5 + .2V = .7V

$t_{rise} = $ ~~85~~ 18 ms

~~We expect~~

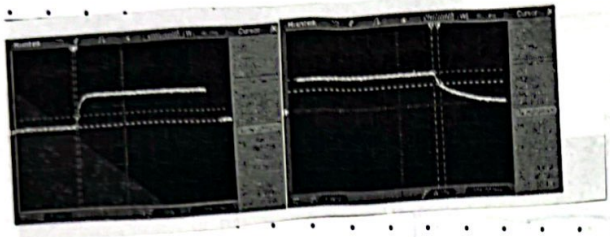
We expect convergence to $\frac{1}{\sqrt{1+\omega^2 R^2 C^2}} \cdot V_{DAC} = \frac{1}{\sqrt{1+(2\pi \cdot 100 \cdot 10\mu)^2}} \cdot 2.5 = 0.822 \cdot 2.5\,V = 2.06\,V$

Difference come from resolution in devices and internal capacitance.

Title Lab 4 report (p16)	Date
Name	Partner

Exercise 6 (cont):

Now with 33 μF



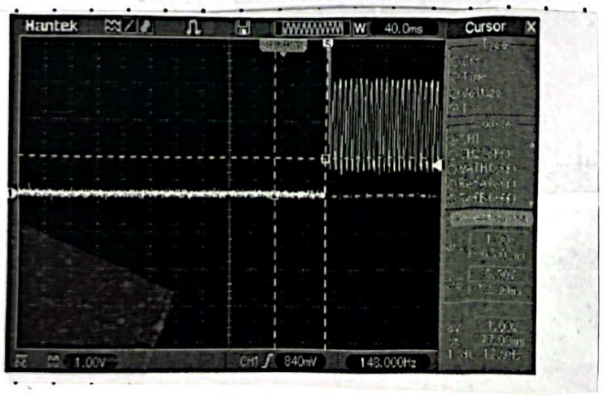
Rise: T=7 ms

Fall: 39 ms

we expect V_o to be $\frac{1}{\sqrt{1+\omega^2 R^2 C^2}} V_{in} = 2.437$ V.

Finally with 1 μF:

Note: we have an oscilatory response at the top



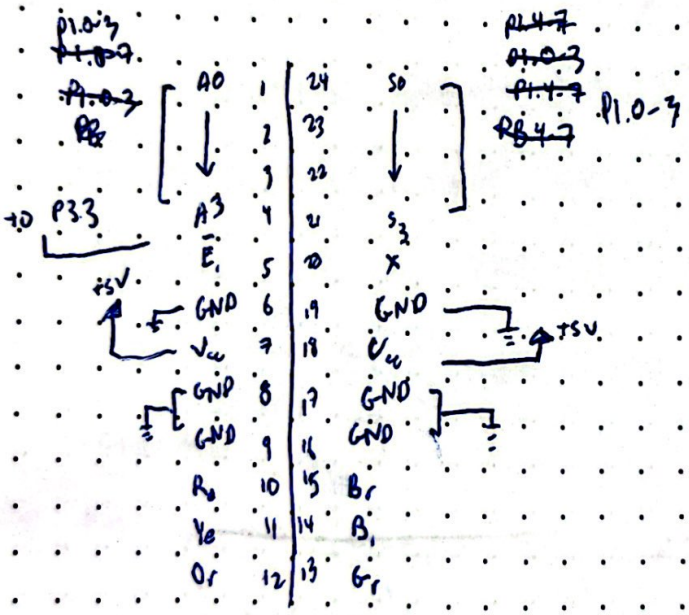
Rise: ~2 ms

Fall: ~3 ms

Exercise 6 (cont.):

We have $V_{out} = \frac{K}{1+sRC+K} V_{in}$ so in all cases, $V_{out}(t=0)$ should be $\frac{K}{1+K} V_{in}$, or $\frac{2}{3} V_{in} = 1.67$ V. Differences come from the small contribution of the sRC and the internal R/C of the circuits.

Exercise 9: Image Reconstruction 4/16/24



9.5/10 EF

Simon Opsahl | Partner

6.115 Final Project

Goals:

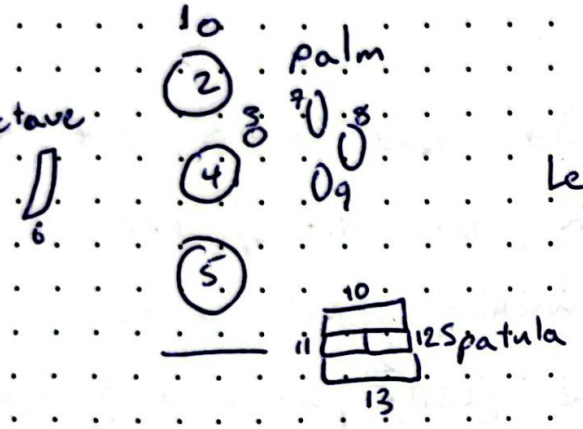
1. Develop a close to fully functional saxophone synthesizer.
   - 1a. Buttons pressed lead to a note that is played on a lab kit speaker.
   - 1b. A sewing machine pedal controls volume.
   - 1c. A mouthpiece (LED + photoresistor) detect intonation.
2. Make a karaoke ~~feautone~~ feature.
   - 2a. Desired buttons pressed and note will be displayed on a TFT/LED Display.
   - 2b. Some measure of quality of tracking is recorded.
3. Make the sound resemple a saxophone.
   - 3a. Use matlab to get a start of the intonation, the middle, and the end of a note.

Title Final Project Report (p2) | Date

Name | Partner

Part I: Buttons to note

The tenor saxophone has 23 buttons, but not all are necessary. The markings are below:



note	keys
Bb	2, 4, 5, 14, 15, 16, 23, 13
B	2, 4, 5, 14, 15, 16, 23, 11
C	2, 4, 5, 14, 15, 16, 23
C#	2, 4, 5, 14, 15, 16, 23, 12
D	2, 4, 5, 14, 15, 16, ~~9~~
Eb	2, 4, 5, 14, 15, 16, 22
E	2, 4, 5, 14, 15, ~~16~~
F	2, 4, 5, 14
F#	2, 4, 5, 15
G	2, 4, 5
G#	2, 4, 5, 10
A	2, 4
Bb	2, 4, 23

Final Project Report (p3)

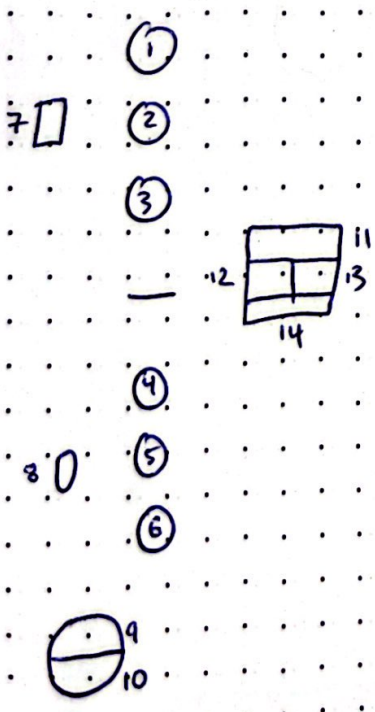
Date

Partner

Part I (cont.):

note	freq	keys
B		2
C		4
C♯		none
D		6, 2, 4, 5, 14, 15, 16
E♭		6, 2, 4, 5, 14, 15, 16, 22
E		6, 2, 4, 5, 14, 15
F		6, 2, 4, 5, 14
F♯		6, 2, 4, 5, 15
G		6, 2, 4, 5
G♯		6, 2, 4, 5, 10
A		6, 2, 4
B♭		6, 2, 4, 20
B		6, 2
C		6, 4
C♯		6,

D → F require palm keys, but are high enough that they can be ignored for the initial iteration. As a result, only 14 keys are used, notated as follows.



Part II: The sound wave

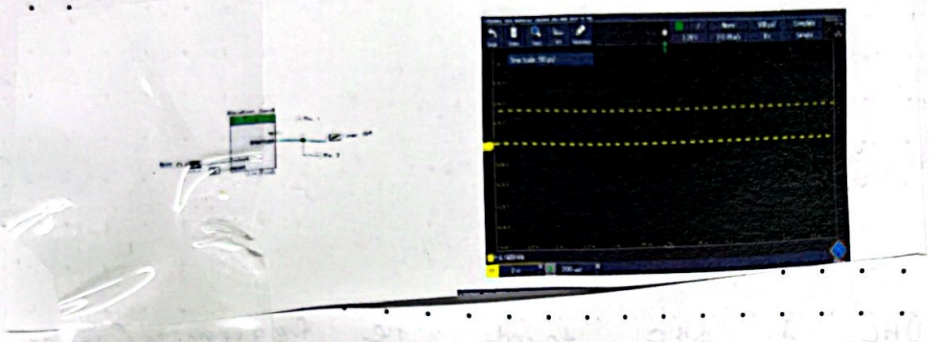
I downloaded a saxophone note waveform online and entered it into matlab. I was able to deconstruct the signal into a beginning, middle, and ending waveform. The beginning waveform is ~~pressed~~ started wherever a note is started, the middle is held for ~~each~~ all of the time the layout is pressed, the ending waveform comes before a new intonation (optional for now). I will start with a repeated waveform in the output DAC. I also found the frequencies/period of a 16 bit counter on a 24 MHz clock for each note.

$B_\flat$	5838.75	4110
B	6187.5	3879
C	6553.125	3662
C#	6946.875	3455
D	7357.5	3262
D#	7796.25	3078
E	8257.5	2906
F	8752.5	2742
F#	9270	2589
G	9821.25	2444
G#	10406.25	2306
A	11025	2177
$B^\flat$	11683.125	2054
B	12375	1939
C	13111.875	1830
C#	13888.125	1728
D	14715	1631
D#	15592.5	1539
E	16520.625	1453
F	17499.375	1371
F#	18540	1294
G	19642.5	1222
G#	20812.5	1153
A	22050	1088
$B^\flat$	23360.625	1027
B	24750	970
C	26223.75	915
C#	27781.875	864

these represent sampling frequencies of the waveform.

Part II (cont.):

When I test with the following circuit, I can get 1/2 pulses of the frequencies needed.

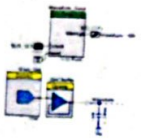


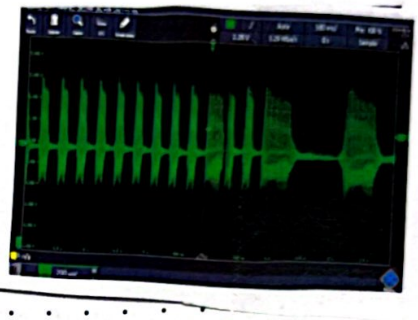
Now I need to send the waveforms out of a DAC pin.

5/8/24

I created a matlab project that downloaded the waveform for a note, split up into beginning, repeating, and ending. I stored those in flash memory, then used a filtered DAC (averages 4 outputs to generate a wave) to make a ~~sin~~ waveform across a 10 nF capacitor. The circuit is as shown on the following page. When sweeping through frequencies, I get varied size waveforms.

Part II (cont.):



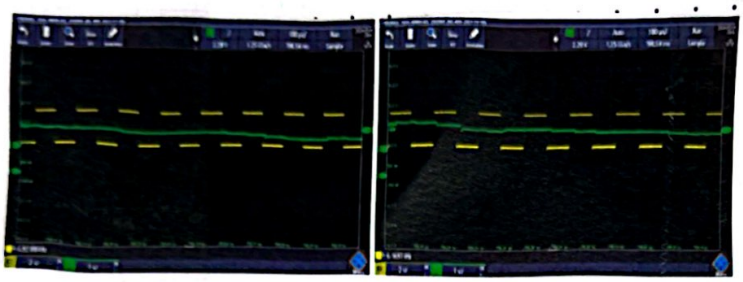


Part III: Buttons to output

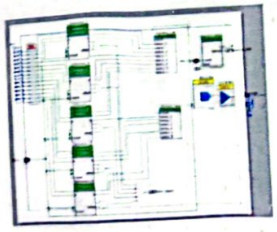
Buttons were bought on Amazon, both with two terminal ~~type~~ type connected to ~~gnd~~ and 3.3V and resistive pull-down. I created a network of 14 buttons (described earlier) that were debounced (at a frequency of 300 Hz) and stored in two status registers. An interrupt is also generated wherever buttons are change configuration. ~~The~~ I began by using each of the buttons to correspond to a different waveform. The configuration will come later. Pictures are on ~~a separate~~ the following page.

Title Final Project Report (p8)	Date
Name	Partner

Part III (cont.):



I tested the ~~voltage press~~ singular button press starting and stopping a note, and it behaved precisely how I expected. The pictures are above, demonstrating a switch from B2 to C#3. The output of the circuit is shown below.



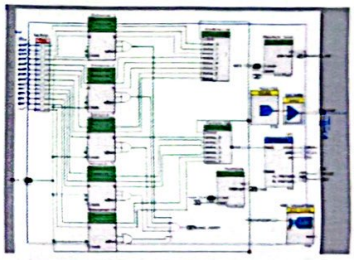
Now the logic needs to be connected to a note. Using the logic table from earlier, we can regulate how button presses affect the note output.

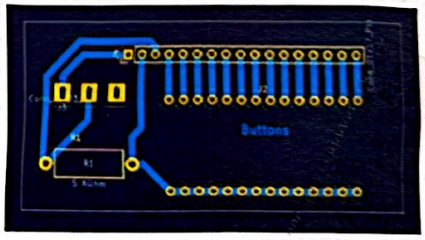
Title Final Project Report (p9) | Date 5/13/24

Name | Partner

Part III (cont.):

I used the following chunk of code and PCB wiring to create a board that took the button inputs to certain ports and the volume pedal to an ADC. I also constructed a physical system for my parts that resembles a saxophone.



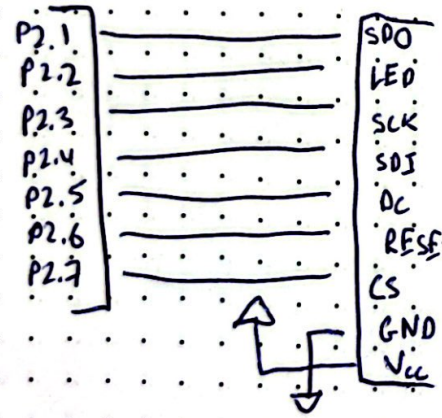


Title Final Project Report (p10)	Date
Name	Partner

Part IV: The TFT Display

Now we need to create the TFT display connection. The course website has a great demonstration of how to use the display with a ILI9341 controller and 8-bit SPI. The demo has functions for writing commands and data. We start with tftStart() to configure the display. We will also use emWin as a graphics library to draw on our display.

We will use the following pinout.



When connected to the PSoC, I made it check if I'm playing along to the song of choice (Ode to Joy) and move the cursor to indicate what note is to be pressed. There is no change in the schematic, but below is the picture of the display.

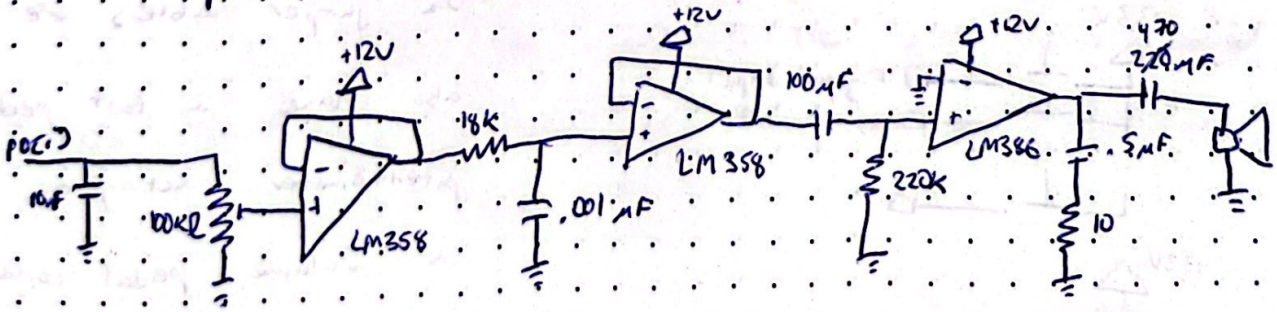
Name | Partner | Date

Part IV (cont.):

I also want to have LEDs indicating which button to press, which can be done with the following display.

Part V: The Sound

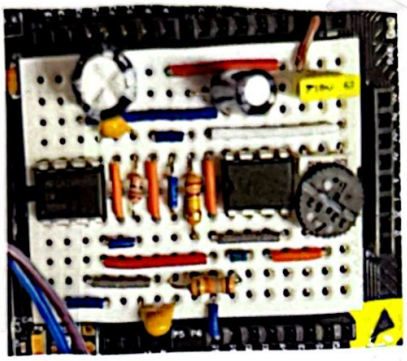
The last thing to do is to connect the sound output to a speaker using the following circuit.



Title Final Project Report (p12)	Date
Name	Partner

Part V (cont.):

The final sound circuit looks like:

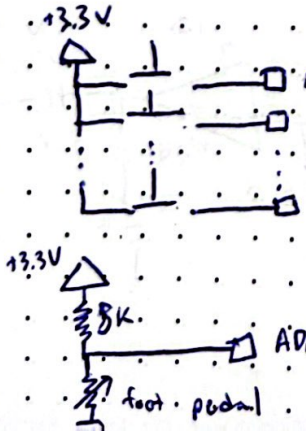


The +12 V is supplied by a variable power supply on the benches, and the speaker is an 8 Ω impedance CE70-30P-8 that was left laying around.

Part VI: The completed system

The final product consists of a number of parts. We have the buttons on the saxophone itself.

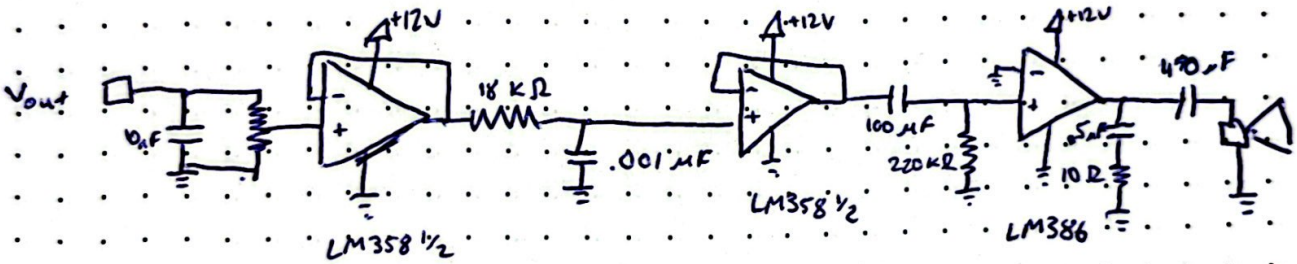
The buttons are connected to a PCB that then connects to the PSoC via jumper cables. We also have a foot pedal potentiometer acting as a volume pedal connected to the same board.



Title: Final Proj. Report (p13) | Date:
Name: | Partner:

Part VII (cont.):

Next is the sound output. I stored the output of processing some .wav files in matlab with the timer periods needed for each note's sampling frequency. When a new note is selected, the code changes the period and sends a new waveform. The waveform is sent through a filtered DAC buffered by a 10 nF capacitor. This is connected in circuit to a series of amplifiers connected to a +12V variable power supply.



We also have a display. The display is comprised of a TFT that is driven by an ILI9341 chip and a series of LEDs driven by digital outputs. The LEDs tell you what note to press; the display shows the song and if you are correct. The plots of general schematics and pictures of the complete system are on the following page.

Title Final Project Report (p14)	Date
Name	Partner

Part VI (cont.):