ECEN 5613 | **Lab #3 Signoff Sheet** | **Spring 2022**

You will need to obtain the signature of your instructor or TA on the following items in order to receive credit for your lab assignment. Print your name below, sign the honor code pledge, circle your course number, and then demonstrate your working hardware & firmware in order to obtain the necessary signatures.

Student Name: Gauriech Ishaan Pilla

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

Student Signature: ______

Signoff Checklist

Part 1 Elements

- ☑ Schematic of acceptable quality (all components shown)
- ☑ Pins and signals labeled, decoupling capacitors, and two 28-pin wire wrap sockets present on board
- ☑ Very good knowledge of a terminal emulator
- ☑ Demonstrates all 32KB of XRAM in memory map are functional, including monitor block fill command
- ☑ Using PAULMON2, demonstrates highest baud rate as: 19200
- ☑ Knows how to use SDCC [IDE or make optional]

SSkharade 03/04/22
TA signature and date

Part 2 Elements

- ☑ Knows how to analyze output files (.RST, .MEM, .MAP) for correct addresses
- ☑ C serial program and virtual debug port functional and code commented
- ☑ Hex display of buffer contents

SSkharade 03/04/22
TA signature and date

Part 3 Required and Supplemental Elements

- ☑ Required ARM code integration and execution
- ☑ 8051 PWM control works correctly, X2 mode
- ☑ Correctly enters Idle mode and exits via external interrupt 1
- ☑ Correctly enters Power Down mode
- ☑ All other PCA software menu items function correctly
- ☑ Good understanding of PCA modes
- ☑ Good user interface; program is easy to use

Maanas MD 03/12/22
TA signature and date

Instructor/TA Comments: ☐ ☐ ☐

FOR INSTRUCTOR USE ONLY Part 1 and 2 Elements	Not Applicable	Below Expectation	Meets Requirements	Exceeds Requirements	Outstanding
Schematics, SPLD code	☐	☐	☐	☑	☐
Hardware physical implementation	☐	☐	☐	☑	☐
Part 1 Required Elements functionality	☐	☐	☐	☑	☐
Sign-off done without excessive retries		☐	☑	☐	
Student understanding and skills	☐	☐	☐	✓ ☐	☐
Overall Demo Quality (Part 2 elements)	☐	☐	☐	✓ ☐	☐

FOR INSTRUCTOR USE ONLY Part 3 Elements	Not Applicable	Below Expectation	Meets Requirements	Exceeds Requirements	Outstanding
Part 3 Required Elements functionality	☐	☐	☐	☑ ✓	☐
Supplemental Elements functionality	☐	☐	☐	☑	☐
Student understanding and skills	☐	☐	☐	☑	☐
Overall Demo Quality (Part 3 elements)	☐	☐	☐	☑	☐

Comments:

- ☑ Optional Challenge: PAULMON2 RUN command
- ☐ Optional Challenge: ISP API calls
- ☐ Optional Challenge: C and Assembly interfacing
- ☐ Optional Challenge: Serial ISR
- ☑ Optional Challenge: SDCC heap memory management analysis

Lab 3 Part 1 & 2 Signoff (03/04/22)

P1

(+) Schematic & hardware complete

(+) Fill command functional

P2

(+) Heap Program working & corner cases handled

(+) Understands output files.

(+) Code well commented.

Challenges

(+) RUN command functional.

Lab 3 Part 3

(+) MSP all required elements completed.

(+) Supplemental MSP

→ Implemented celsius & fahrenheit.

→ Commands to increase & decrease PWM.

→ Period change

→ Spectrum.

(+) Good User Interface for MSP.

(+) 8051 supplemental parts completed.

(+) PCA modes

→ PWM

→ WDT

→ High speed output

(+) Good User Interface for 8051.

(+) SDCC heap analysis completed.

Submission Questions-

a) What operating system (including revision) did you use for your code development?
Answer- I have Used Windows 10 Operating system for my code development.

b) What compiler (including revision) did you use?
Answer- I have used SDCC 4.1.0 compiler for Lab 3.

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

Answer- I have used CodeBlocks IDE for AT89C51 and Code Composer studio for MSP432

d) Did you install and use any other software tools to complete your lab assignment?
Answer- I have used mostly the software tools mentioned in the lab assignments but additional software tool I have used is Tera term to show the UART outputs for my sdcc and msp432 codes.

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

Answer- I was facing issues in sdcc compiler initially to generate the required files and also using the tera term at changed baud rates. But later it was figured and everything worked smoothly.

Thank you.

Things Learnt in Lab 3

1. Memory allocation and error handling.
2. Compiling with SDCC.
3. Writing interrupt handler for AT89C51 in C.
4. TI MSP 432 additional functions and features used to make a user interface program.
5. Designing a user Interface using UART.
6. Paulmon2 and methods to use it
7. Analyze the output files of SDCC
8. X2 mode, PCA, Idle and power down mode in AT89C51.
9. Learn more about logic analyzer and how to debug using it
10. SDCC Heap analysis
11. PAULMON as an on-chip debugger

Challenges Faced

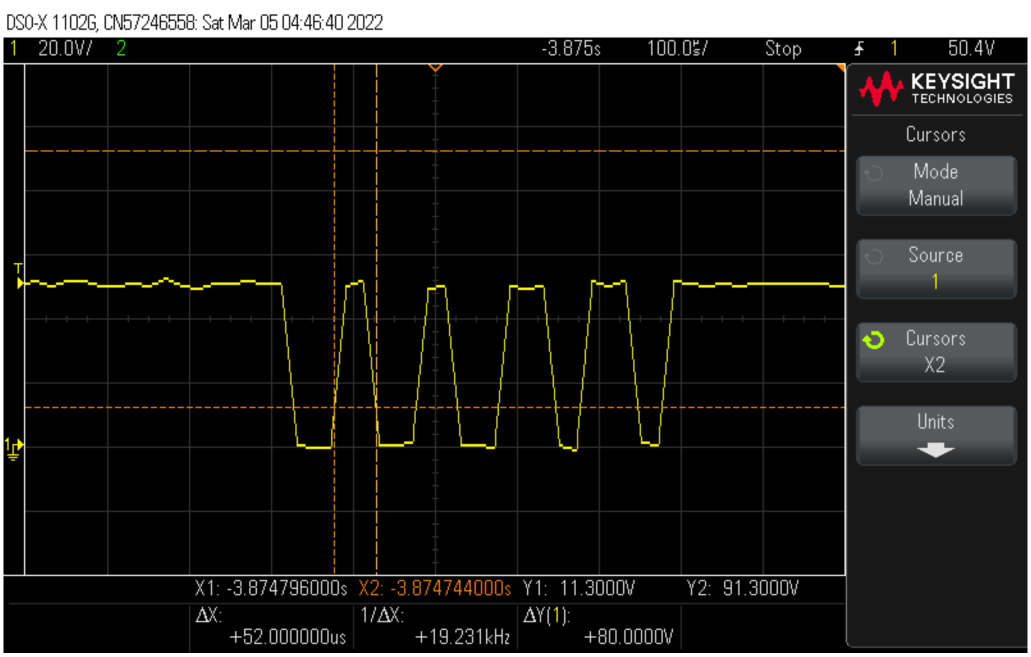
1. Designing a good user interface using UART.
2. Solving Compilation errors.
3. Working with TI MSP 432

Lab 3 part 1

Lab 3 part 1 was a basic checkpoint in which we had to configure hardware to use the NVSRAM as additional XRAM data memory in our system. We used SPLD logic (SPLD Logic file attached in the files) to configure the hardware.

Later we had to configure Paulmon2 and use the AS31 Assembler. To verify if everything was done properly we had to load the paulmon2 and extra file on our 8051 and check the hardware by filling the XRAM space 0x0000 to 0x7FFF with a specific value, like 0x55. If this thing was completed successfully the hardware and Paulmon2 were configured properly. After completing the required element we were given the challenge to use PAULMON as an on-chip debugger. For this, we had to make a few changes in our extra. asm file and use the 's' key to start the single step in paulmon 2. I wrote a basic program to toggle led at p1.1 and demonstrated this using single step.

After configuring paulmon2 and the hardware we were instructed to learn how to use SDCC to generate hex records for our hardware using the large (or small) memory model and then verify that the correct addresses were generated in your code listing file (.rst) and hex record file (.ihx) and also examine the other output files to see how SDCC has allocated space for objects in memory like .mem, .lnk, and .mapWe were given an option to use IDE or makefile. I have chosen to use CodeBlocks Ide. A small program was told to be written to check whether SDCC is giving all the required files. I wrote a small program to toggle the p1.1 led on my board to verify that SDCC was set up properly.



Baud rate verified on oscilloscope for 19200(1/52us)

Lab 3 part 2

The required element of this part was to write a C program to first allocate a heap of size 5000 bytes. Then prompt the user to specify a buffer size between 48 and 4800 bytes, where the buffer size must be evenly divisible by 16. The program would then allocate a buffer (buffer 0) of the requested buffer size in XRAM using the malloc() function and then malloc in XRAM a second buffer (buffer 1) which is also equal to the requested buffer size. If the malloc failed for any buffer, then any successfully allocated space would be freed and the user would be prompted to choose a smaller buffer size. We were instructed to use pointers to external memory to access the buffers.

After the successful creation of the program, the user would then be prompted to enter various lower case characters or choose an option from the menu to perform various functions.

For example - If the '+' character is received, the program must prompt the user to specify a buffer size between 30 and 300 bytes. The program must then try to allocate a new buffer (buffer n) of the requested buffer size in XRAM using the malloc() function.

If the '-' character is received, the program must prompt the user to specify a valid buffer number. If the buffer number is valid, the program must then delete that buffer and use the free() function to free up its space from the heap.

If the character '?' is received, the program must provide a report on the heap, including information about each buffer currently in the heap, including buffer #, buffer start address, buffer end address, total allocated size of the buffer (in bytes), the number of storage characters currently in that buffer, and the number of free spaces remaining in that buffer. If buffer_0 filled completely before a '?' command is received, any excess character subsequently received for that buffer is echoed out the serial port, but is not added to that buffer (it is discarded). Once the '?' command is received, then buffer_0 is emptied

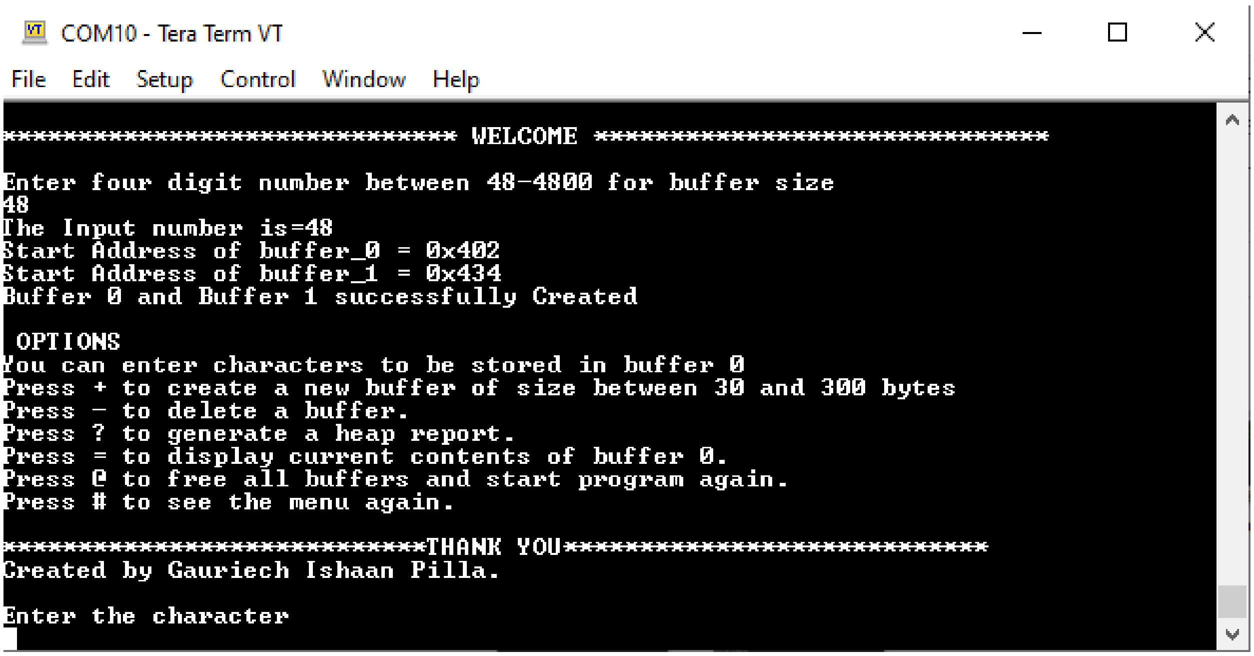
If the '=' character is ever received, the program must display the current contents of buffer_0 in hex, but must not empty the buffer – the data will remain in the buffer until the buffer emptied in response to a '?' command.

If the '@' character is ever received, the program must immediately use the free() function to free the heap space being utilized by all current buffers that have been allocated. The program shall then start over from the beginning and ask the user to specify a new buffer size.

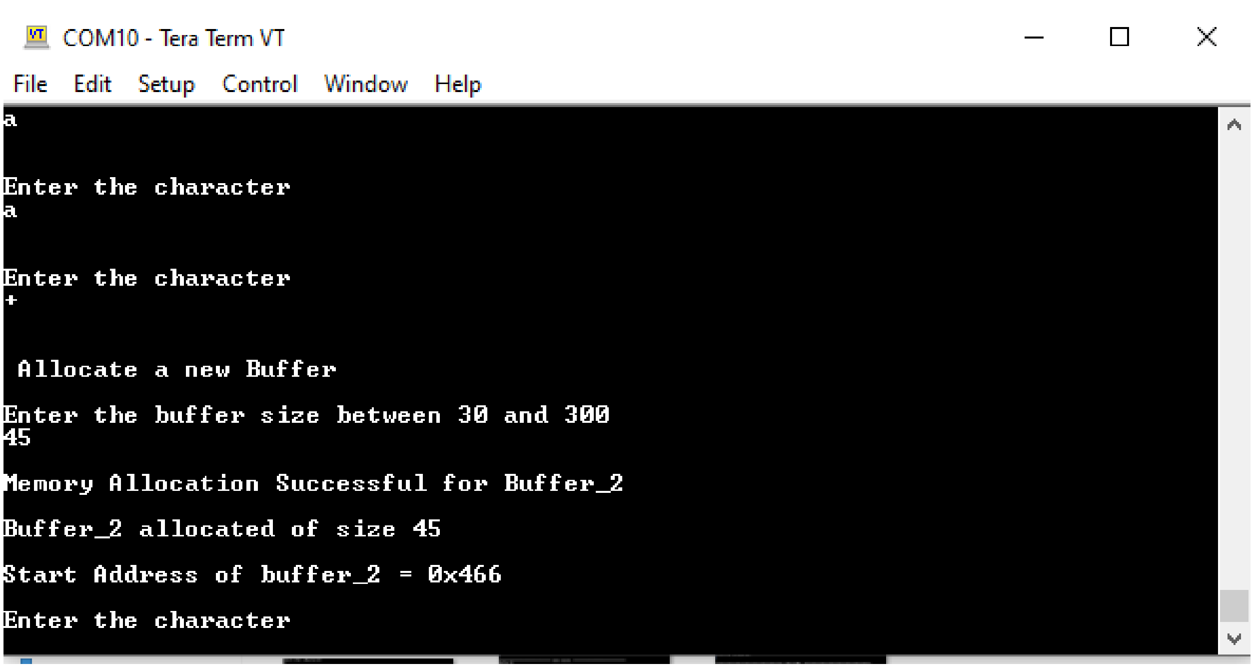
These were some examples to be used. I have implemented an additional function to see the menu again using the '#' key. A menu had to be created and this whole program had to be displayed on UART. Making a good user interface was also one of the requirements of this program and I focused on it.
The Screenshots of working are attached in the lab submission files.

Later we had to make a virtual debug port. This virtual debug port was used to write a value at a specific address and see the value when the command was run using a logic analyzer. Steps were given to set up the debug port and use it in the lab manual.

Introduction Menu



Add Buffer '+' Function.



Delete Buffer '-' Function.

```
Enter the character
-


Enter a valid buffer number
2

Deleting buffer 2

Buffer 2 is Free

Enter the character
```

Heap Report- '?'

```
Enter the character
?

****************************** HEAP REPORT ******************************
Buffer 0

Start Address = 0x402

Ending Address = 0x432

Buffer Size = 48

Storage characters in buffer = 11

Free Spaces in buffer = 37

----------------------------------------------------------------------

Buffer 1

Start Address = 0x434

Ending Address = 0x464

Buffer Size = 48

Storage characters in buffer = 0

Free Spaces in buffer = 48

----------------------------------------------------------------------

Number of storage characters = 11

Total number of characters received = 13

Total number of buffers that were allocated since the start of the program = 2

Total storage characters stored since last '?' = 11
```

Contents of Buffer 0- '='

```
Enter the character
=


Contents of Buffer 0
0x402>> 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61
0x412>> 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61
0x422>> 61 61 61 61 61 61 61 61 61 61 61
Enter the character
```

Free command(@)

```
COM10 - Tera Term VT
File  Edit  Setup  Control  Window  Help
Enter the character
@


Freeing Buffer 0

Buffer 0 is free

Freeing Buffer 1

Buffer 1 is free

Freeing buffer_2

Buffer 2 is free

Freeing buffer_3

Buffer 3 is free

****************************** WELCOME ******************************

Enter four digit number between 48-4800 for buffer size
```

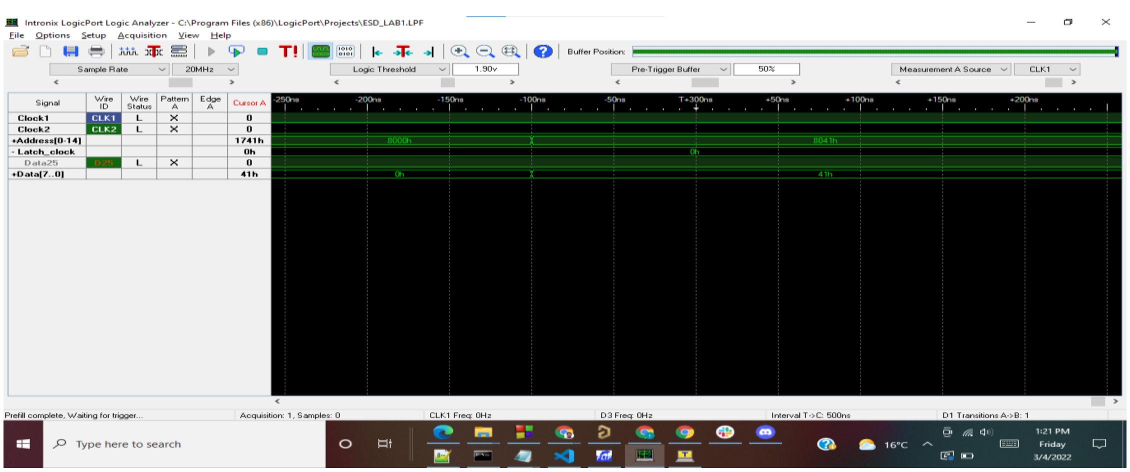
menu again option using ‘#’

```
Enter the character
#


 OPTIONS
You can enter characters to be stored in buffer 0
Press + to create a new buffer of size between 30 and 300 bytes
Press - to delete a buffer.
Press ? to generate a heap report.
Press = to display current contents of buffer 0.
Press @ to free all buffers and start program again.
Press # to see the menu again.

Enter the character
```

Debug port example



Lab 3 part 3

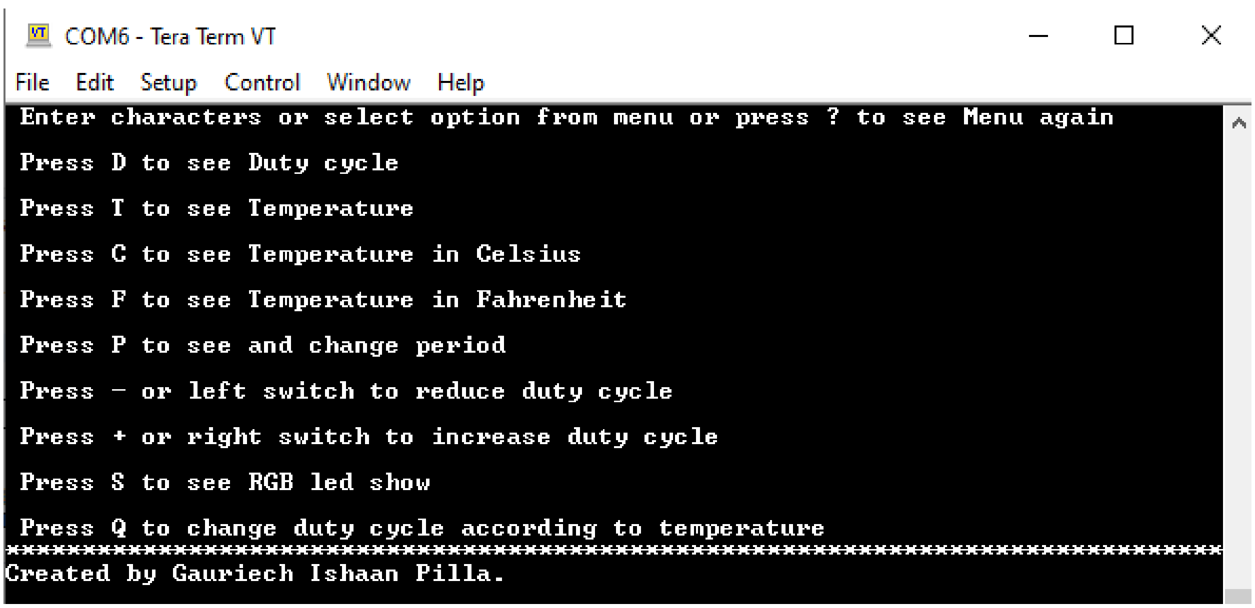
In lab 3 part 3 required element we had to implement a similar user interface program on MSP 432 with different functionality. Similarly, UART had to echo out the input characters and perform certain functions on pressing the keys, For example, if ‘T’ was pressed it had to show the temperature in Celsius and Fahrenheit using the internal Temperature sensor of MSP 432. If ‘F’ was pressed it should display temperature in Fahrenheit and If ‘C’ was pressed it should display temperature in Celsius.

If ‘D’ was pressed it should generate a PWM signal with a default 40% duty cycle using one of the MSP432 GPIO pins. The duty cycle could be altered using the onboard switches, ‘+’ and ‘–‘ key or +/-10% duty cycle if temperature sensor reading varies by +/-0.5 degrees Celsius. Additional functionality of the program was to alter the period of the PWM signal using the key ‘P’ or start an RGB led show using the key ‘S’. This program was also displayed on UART with a user interface containing the menu showing the commands that would be executed if the following key is pressed. Pressing ‘?’ would show the menu again.

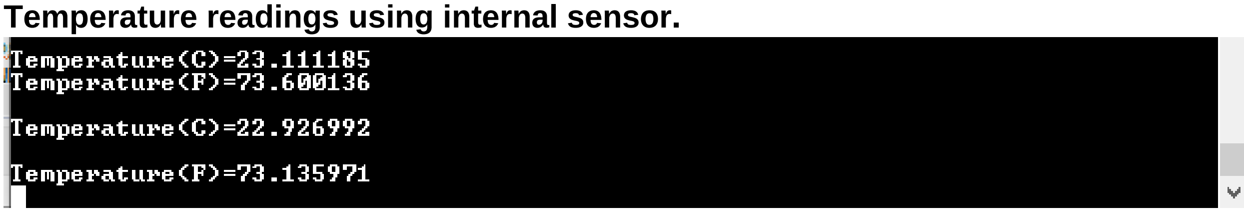
.

The Screenshots and oscilloscope captures of working are attached in the lab submission files.

Menu:



Temperature readings using internal sensor.



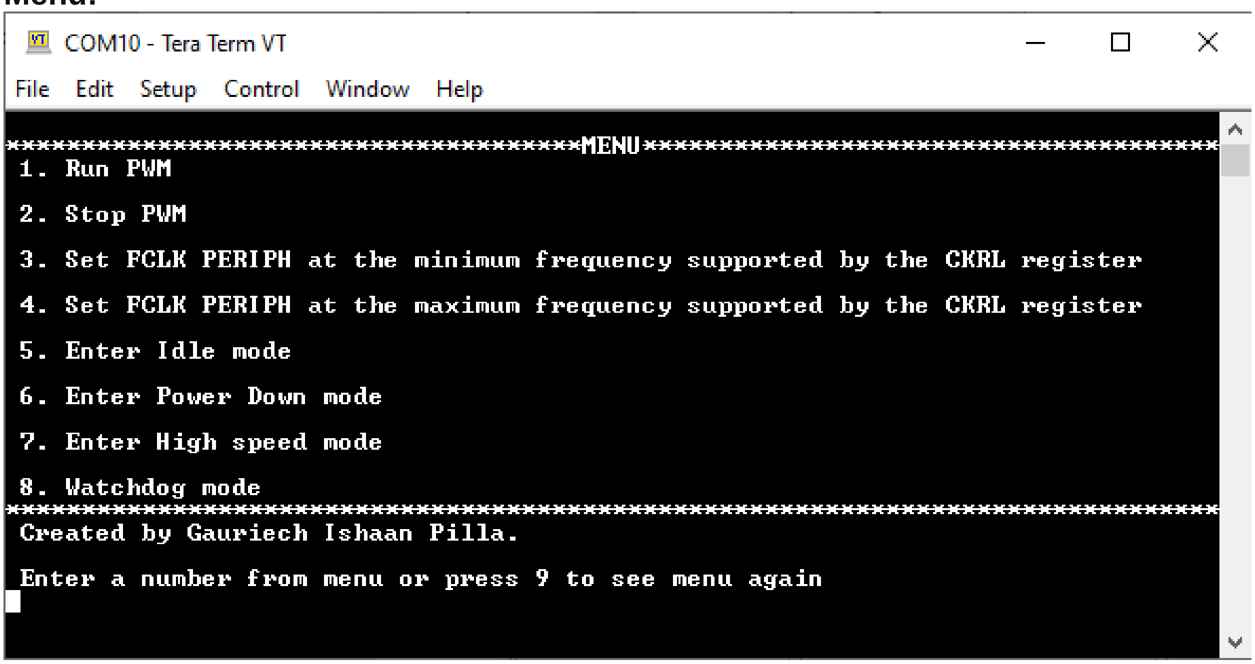
For the supplemental part in part 3, we were told to write another C program on SDCC to demonstrate other features of the Atmel AT89C51RC2 i.e. Programmable Counter Array (PCA). In this element, we had to use at least three of the PCA modes.
PCA modes- 1. Rising and/or falling edge capture 2. Software timer 3. High-speed output 4. Pulse width modulator (PWM) 5. Module 4 can also be programmed as a watchdog timer.
For this element, we also had to configure the system clock in X2 mode. My program had a well-designed user menu that allowed the user to perform the following options:

1. Run PWM (turn on PWM output)
2. Stop PWM (turn off PWM output)
3. Set FCLK PERIPH at the minimum frequency supported by the CKRL register
4. Set FCLK PERIPH at the maximum frequency supported by the CKRL register
5. Enter Idle mode (set IDLE bit in PCON register)
6. Enter Power Down mode (set PDE bit in PCON register)

We had to prove and show how all the modes working and how these modes were set using various registers of 8051.

The Screenshots and oscilloscope captures of working are attached in the lab submission files.

Menu:



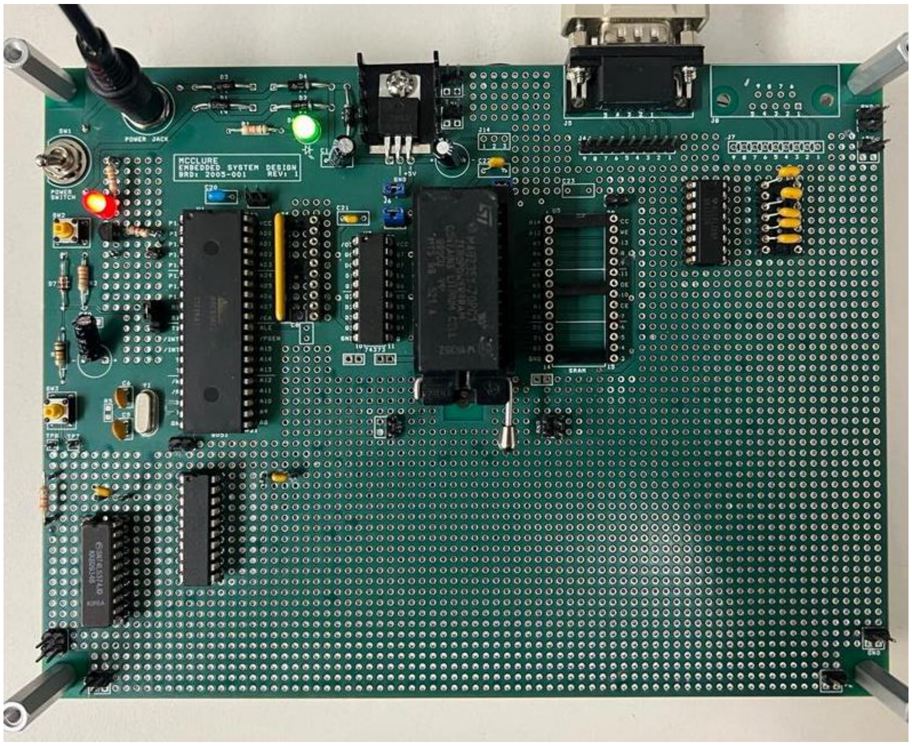
Thank you.

*********************************Challenges*************************************

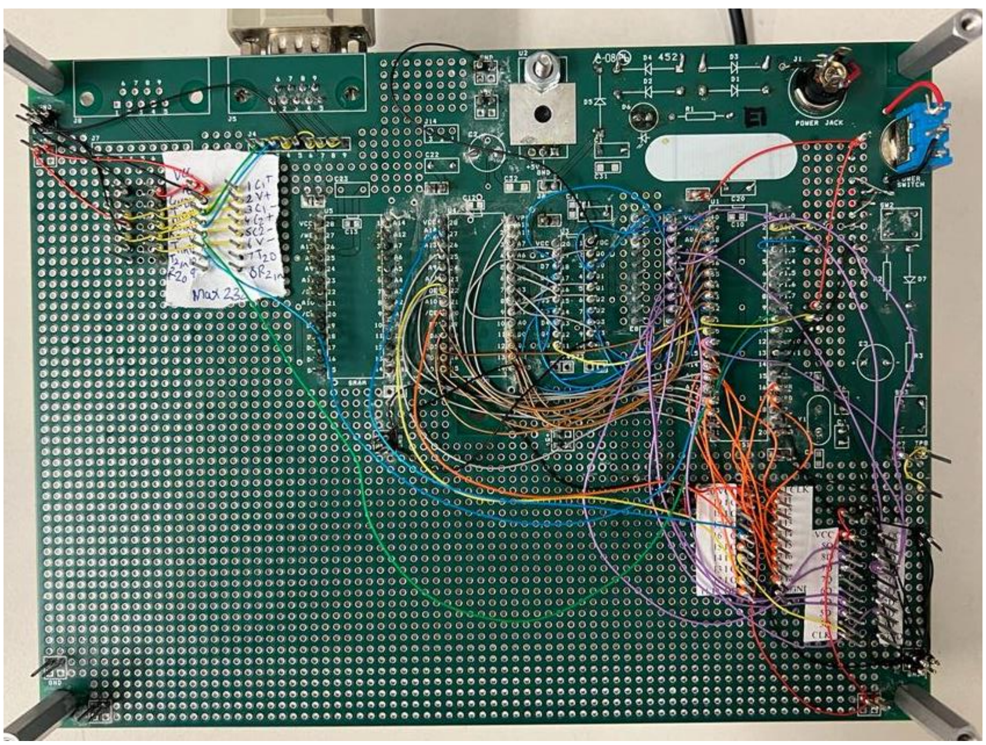
I have submitted another brief report in the challenges folder to explain the challenges I have done in this lab.

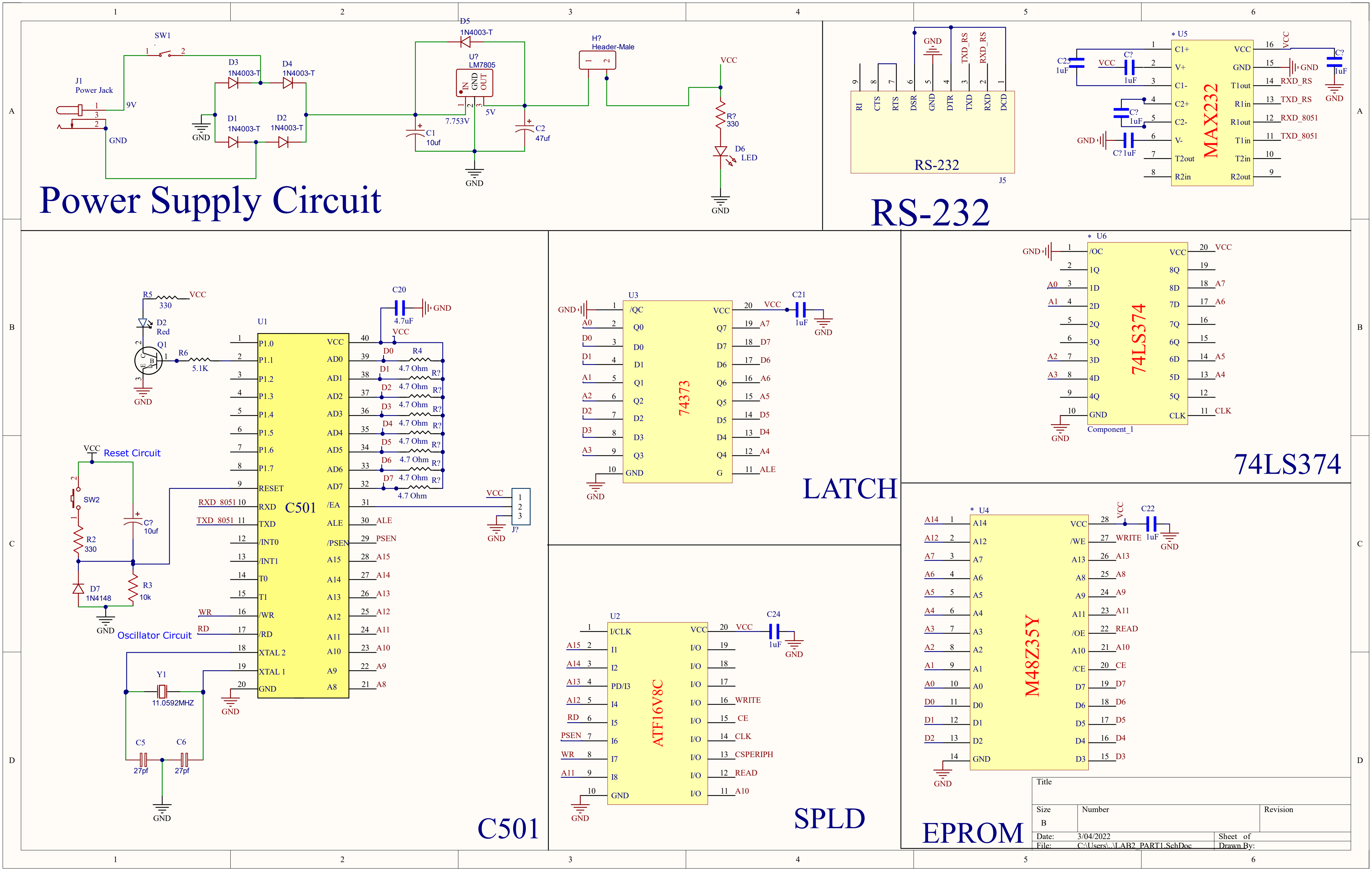
Board Photos

Front



Back





Power Supply Circuit
J1
Power Jack
SW1
D3 1N4003-T
D4 1N4003-T
D1 1N4003-T
D2 1N4003-T
D5 1N4003-T
U? LM7805
IN GND OUT
7.753V
5V
C1 10uf
C2 47uf
H? Header-Male
VCC
R? 330
D6 LED
GND
9V
RS-232
RI CTS RTS DSR GND DTR TXD RXD DCD
TXD_RS
RXD_RS
J5
U5
MAX232
C1+ V+ C1- C2+ C2- V- T2out R2in
VCC GND T1out R1in R1out T1in T2in R2out
TXD_RS
RXD_RS
RXD_8051
TXD_8051
U6
74LS374
/OC 1Q 1D 2D 2Q 3Q 3D 4D 4Q GND
VCC 8Q 8D 7D 7Q 6Q 6D 5D 5Q CLK
Component_1
CLK
U1
C501
P1.0 P1.1 P1.2 P1.3 P1.4 P1.5 P1.6 P1.7 RESET RXD TXD /INT0 /INT1 T0 T1 /WR /RD XTAL 2 XTAL 1 GND
VCC AD0 AD1 AD2 AD3 AD4 AD5 AD6 AD7 /EA ALE /PSEN A15 A14 A13 A12 A11 A10 A9 A8
R5 330
D2 Red
Q1
R6 5.1K
C20 4.7uF
R4 4.7 Ohm
Reset Circuit
SW2
C? 10uf
R2 330
R3 10k
D7 1N4148
Oscillator Circuit
Y1 11.0592MHZ
C5 27pf
C6 27pf
J?
U3
74373
/QC Q0 D0 D1 Q1 Q2 D2 D3 Q3 GND
VCC Q7 D7 D6 Q6 Q5 D5 D4 Q4 G
C21 1uF
ALE
LATCH
U2
ATF16V8C
I/CLK I1 I2 PD/I3 I4 I5 I6 I7 I8 GND
VCC I/O
C24 1uF
WRITE CE CLK CSPERIPH READ
PSEN WR RD
SPLD
U4
M48Z35Y
A14 A12 A7 A6 A5 A4 A3 A2 A1 A0 D0 D1 D2 GND
VCC /WE A13 A8 A9 A11 /OE A10 /CE D7 D6 D5 D4 D3
C22 1uF
WRITE READ CE
EPROM
Title
Size B
Number
Revision
Date: 3/04/2022
Sheet of
File: C:\Users\..\LAB2_PART1.SchDoc
Drawn By: