University of Colorado Boulder

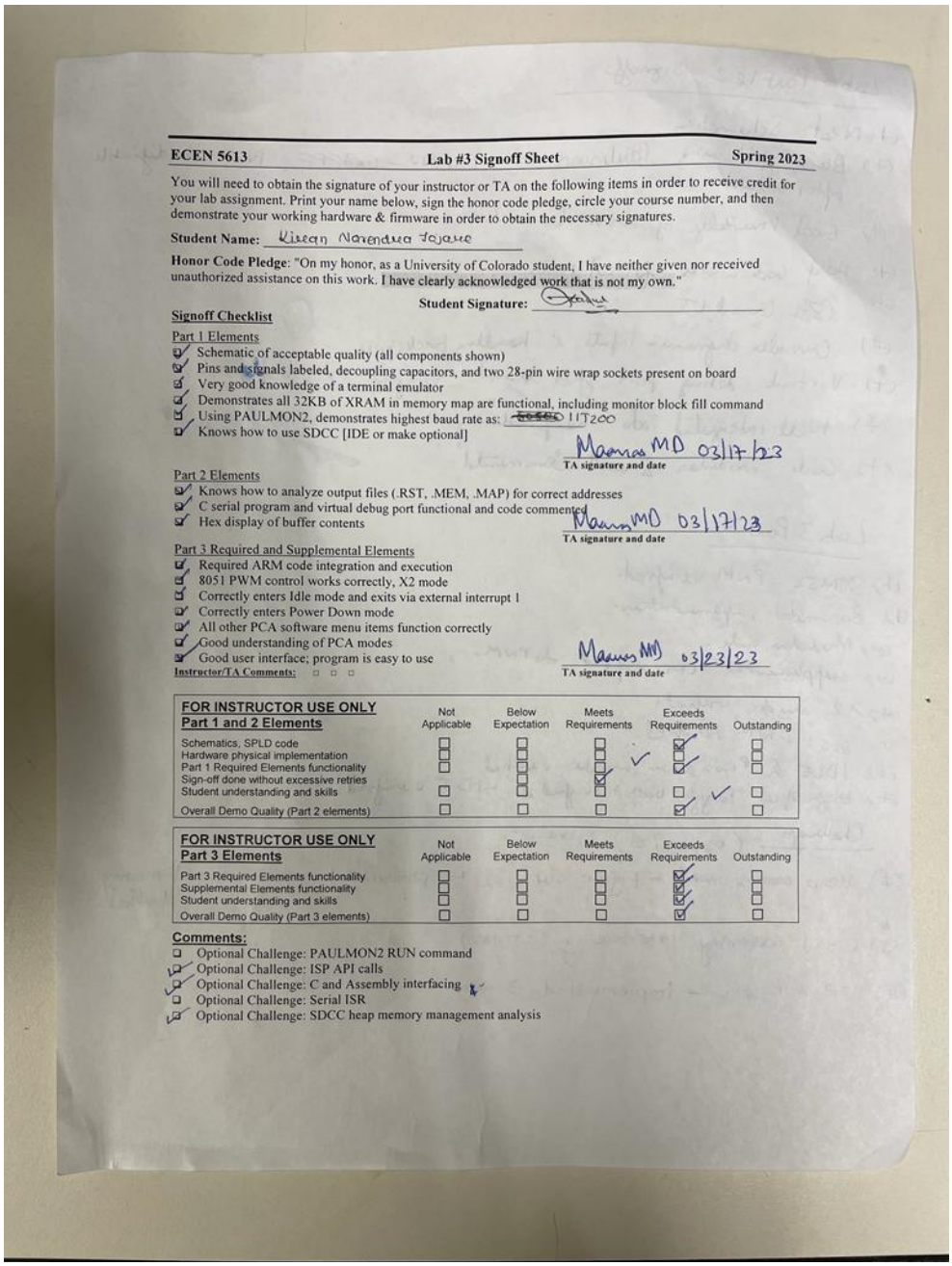
School/College/Department
Center/Office
Building Room Number
000 UCB
Boulder, Colorado 80309-0000

t 303 492 0000
f 303 492 0000
email.email@colorado.edu

LAB REPORT : ESD LAB 3

Signoff Sheet

The figure below shows the sign of sheet for lab 3. Please note that required, supplemental and challenges has been completed for this lab.



ECEN 5613 **Lab #3 Signoff Sheet** **Spring 2023**

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: Kiran Narendra Jojare

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: ~~115200~~ 115200
- ☑ Knows how to use SDCC [IDE or make optional]

Manas MD 03/17/23
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

Manas MD 03/17/23
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

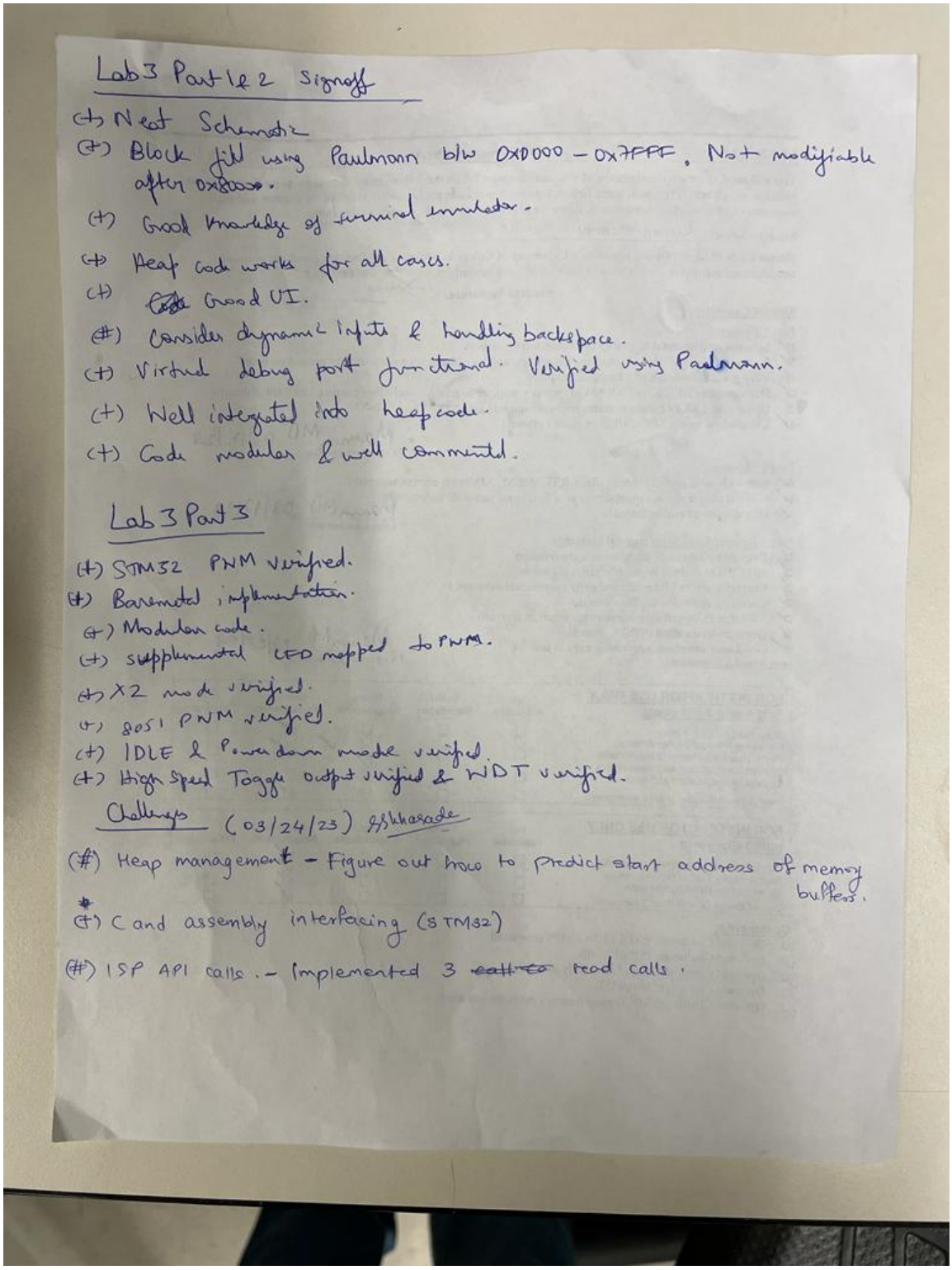
Instructor/TA Comments:

Manas MD 03/23/23
TA signature and date

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



Lab3 Part 1 & 2 Signoff

(+) Neat Schematic
(+) Block fill using Pattern b/w 0x0000 – 0x7FFF. Not modifiable after 0x8000.
(+) Good knowledge of terminal emulator.
(+) Heap code works for all cases.
(+) ~~Code~~ Good UI.
(#) Consider dynamic inputs & handling backspace.
(+) Virtual debug port functional. Verified using Putchar.
(+) Well integrated into heap code.
(+) Code modular & well commented.

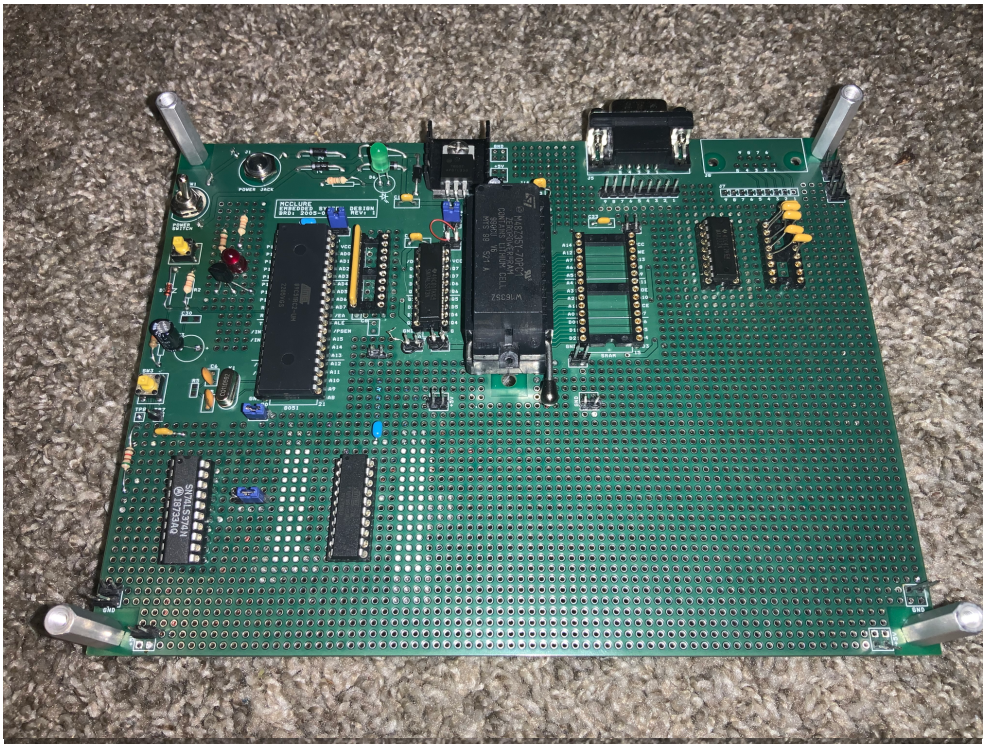
Lab3 Part3

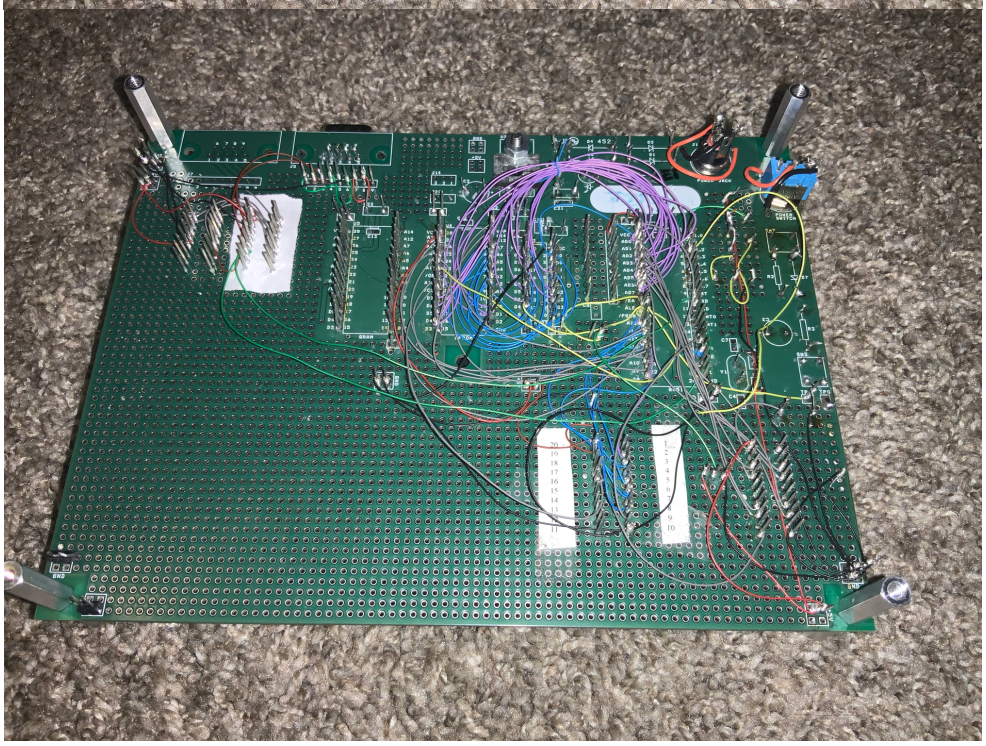
(+) STM32 PWM verified.
(+) Baremetal implementation.
(+) Modular code.
(+) supplemental LED mapped to PWM.
(+) X2 mode verified.
(+) 8051 PWM verified.
(+) IDLE & Powerdown mode verified.
(+) High Speed Toggle output verified & WDT verified.

Challenges (03/24/23) ~~Khasade~~

(#) Heap management – Figure out how to predict start address of memory buffers.
(+) C and assembly interfacing (STM32)
(#) ISP API calls – Implemented 3 ~~callbacks~~ read calls.

Board Pictures





Lab3 Part 1 (Required)

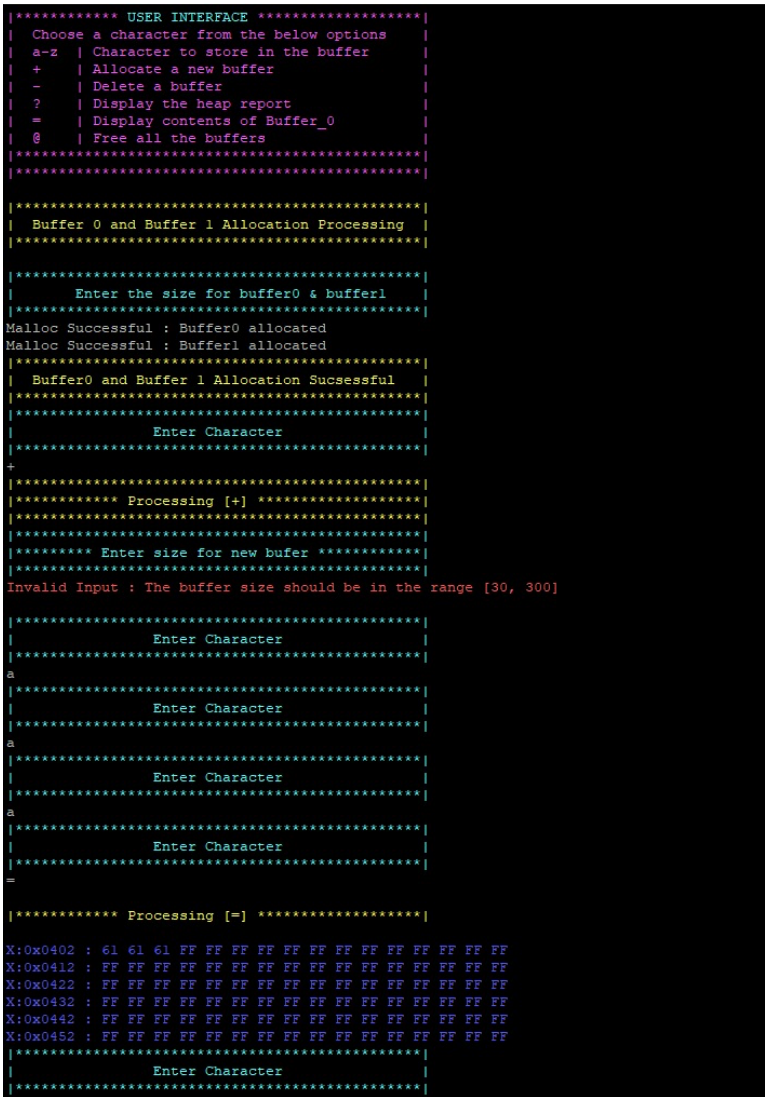
Configured NVSRAM to use as XRAM data memory. Used and verified PAULMONN. Details are available in sign off sheet and code is available in repo. Own version of PAULMONN assembled to work with A31 assembler.

Maximum Baud Rate Achievable under PAULMONN : 57600

Experimental test SDCC project created and verified during sign off. A sample HEX file is modified and analyzed for SDCC project. Sample code was used to toggle port A pin.

Lab 2 Part 2 (Required)

The UI for lab 2 part 2 is as shown below. User takes characters and performs specific task as per the requirement. All parts are verified during sign off.



Lab 2 Part 2 Heap Management Challenge

Click on this file to see the UART log for addresses created during each + and – operation as described by assignment ![UART_LOG]!
Step by step Execution :
Step 1) Heap created with 5600 Bytes
Step 2) Buffer 0 and buffer 1 created with 2400 Bytes
Step 3) + command for 200 Bytes
Step 4) + command for 300 bytes
Step 5) – command buffer 2 deleted

This will create an empty buffer space in place of buffer 3

So if new buffer is created with size of 100 it will occupy that free memory space as seen in the heap report seen below

```
**Buffer [0]**
Buffer [0] : Start Address = [X:0x0402]
Buffer [0] : End Address   = [X:0x0d62]
Buffer [0] : Total Size    = [2400]
Buffer[0] : Number of storage characters since last ?= [0]
Buffer[0] : Number of storage characters since last ?= [5]
**Buffer [1]**
Buffer [1] : Start Address = [X:0x0d64]
Buffer [1] : End Address   = [X:0x16c4]
Buffer [1] : Total Size    = [2400]
Buffer [2] already cleared!!
**Buffer [3]**
Buffer [3] : Start Address = [X:0x1790]
Buffer [3] : End Address   = [X:0x18bc]
Buffer [3] : Total Size    = [300]
**Buffer [4]**
Buffer [4] : Start Address = [X:0x16c6]
Buffer [4] : End Address   = [X:0x172a]
Buffer [4] : Total Size    = [100]
|************ REPORT ENDED  ********************|
```

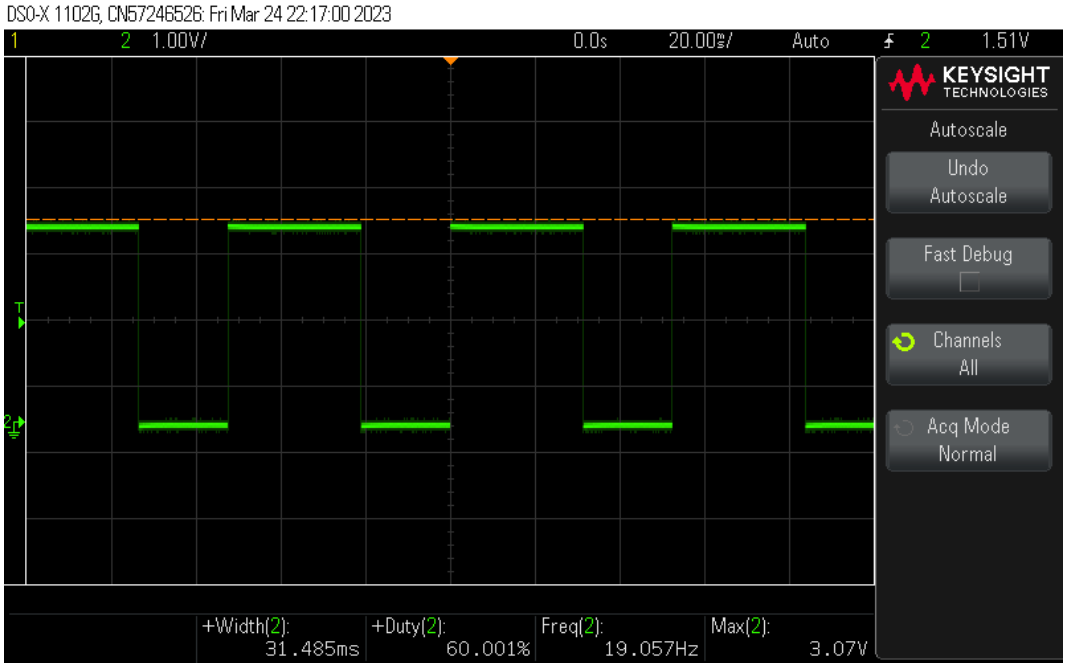
Now if another new buffer is added with 210 bytes length it will be allocated after the after buffer 3 as seen in the figure

```
|***********************************************|
 |************** HEAP REPORT ******************|
 |***********************************************|

**Buffer [0]**

Buffer [0] : Start Address = [X:0x0402]

Buffer [0] : End Address   = [X:0x0d62]

Buffer [0] : Total Size    = [2400]

Buffer[0] : Number of storage characters since last ?=

Buffer[0] : Number of storage characters since last ?=

**Buffer [1]**

Buffer [1] : Start Address = [X:0x0d64]

Buffer [1] : End Address   = [X:0x16c4]

Buffer [1] : Total Size    = [2400]

Buffer [2] already cleared!!

**Buffer [3]**

Buffer [3] : Start Address = [X:0x1790]

Buffer [3] : End Address   = [X:0x18bc]

Buffer [3] : Total Size    = [300]

**Buffer [4]**

Buffer [4] : Start Address = [X:0x16c6]

Buffer [4] : End Address   = [X:0x172a]

Buffer [4] : Total Size    = [100]

**Buffer [5]**

Buffer [5] : Start Address = [X:0x18be]

Buffer [5] : End Address   = [X:0x1990]

Buffer [5] : Total Size    = [210]

|************ REPORT ENDED  *********************|
```

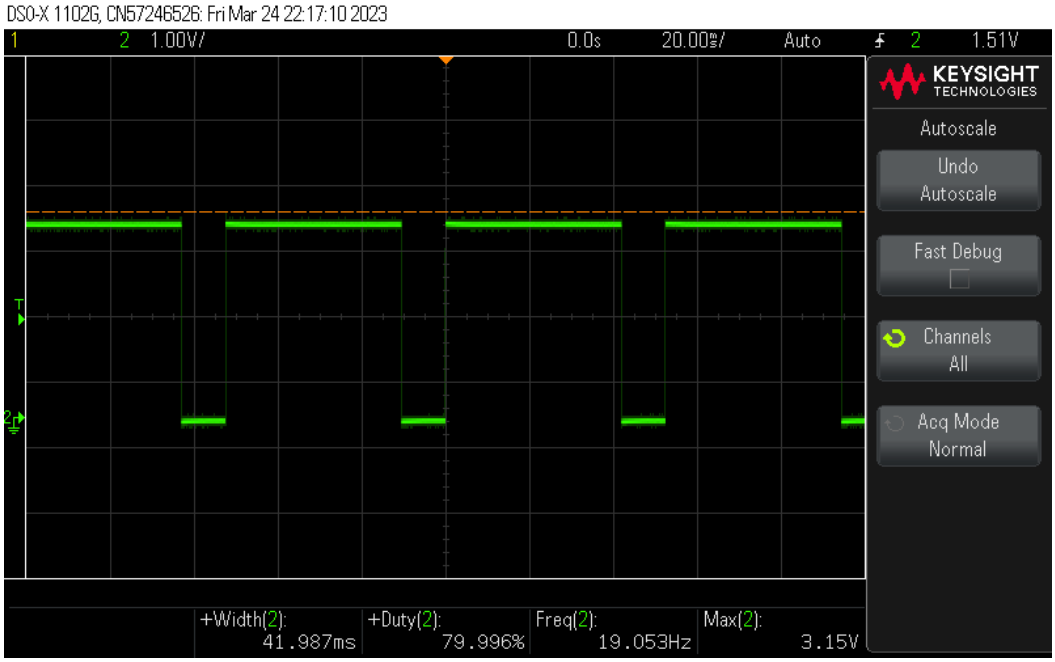
Lastly if one more buffer with the size of 800 bytes is allocated the buffer allocation will fail as malloc will fail to locate free memory in heap as its totally full at this point.

Lab 3 Part 3 (Required)

The STM output from pin PD15 with default duty cycle of 60% is shown below. PD15 is also connected to a BLUE LED. Output verified during signoff.

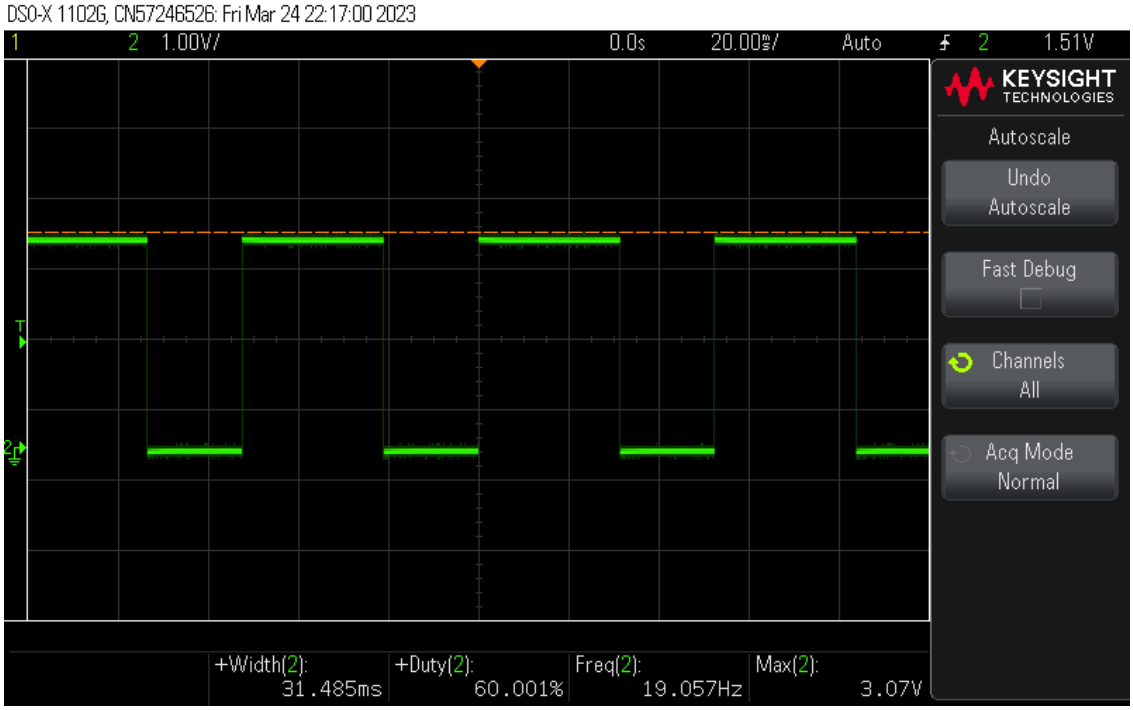


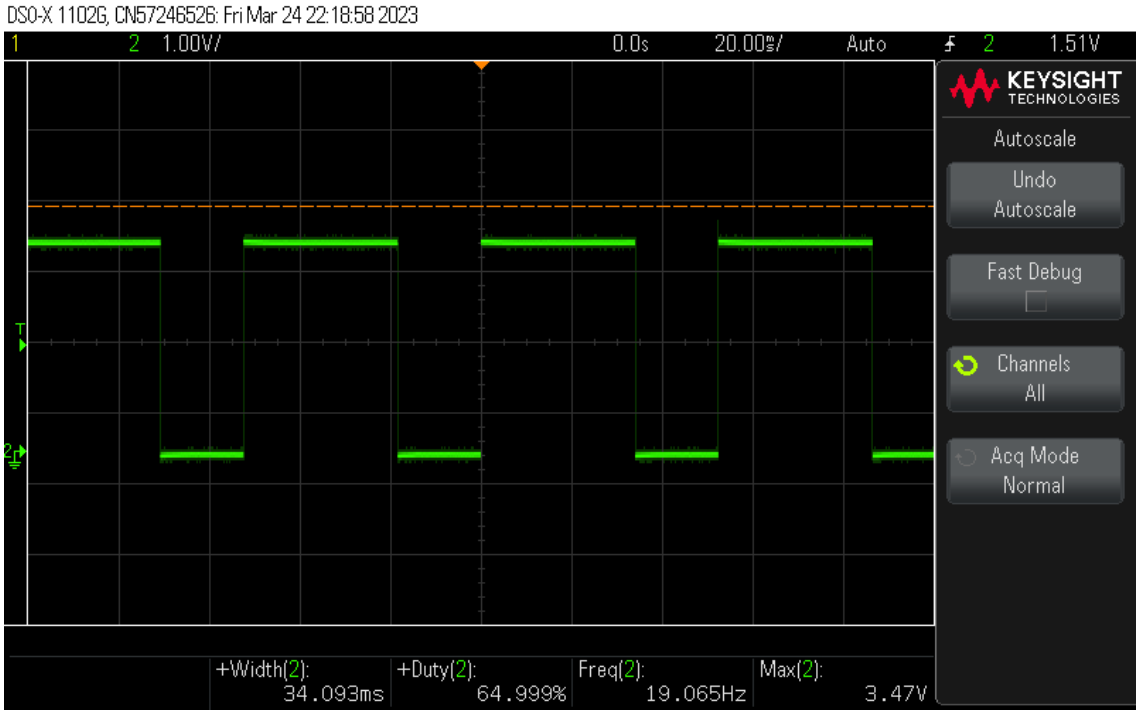
The STM output after switch triggered resulting in duty cycle incrementing by 10% is as follow. The PWM output increments by 10 % after every button press and decrements by 10% once it reaches maximum. Similarly, the PWM output increments by 10 % till it reaches maximum of 100% again. The behavior has been verified and tested during sign off.



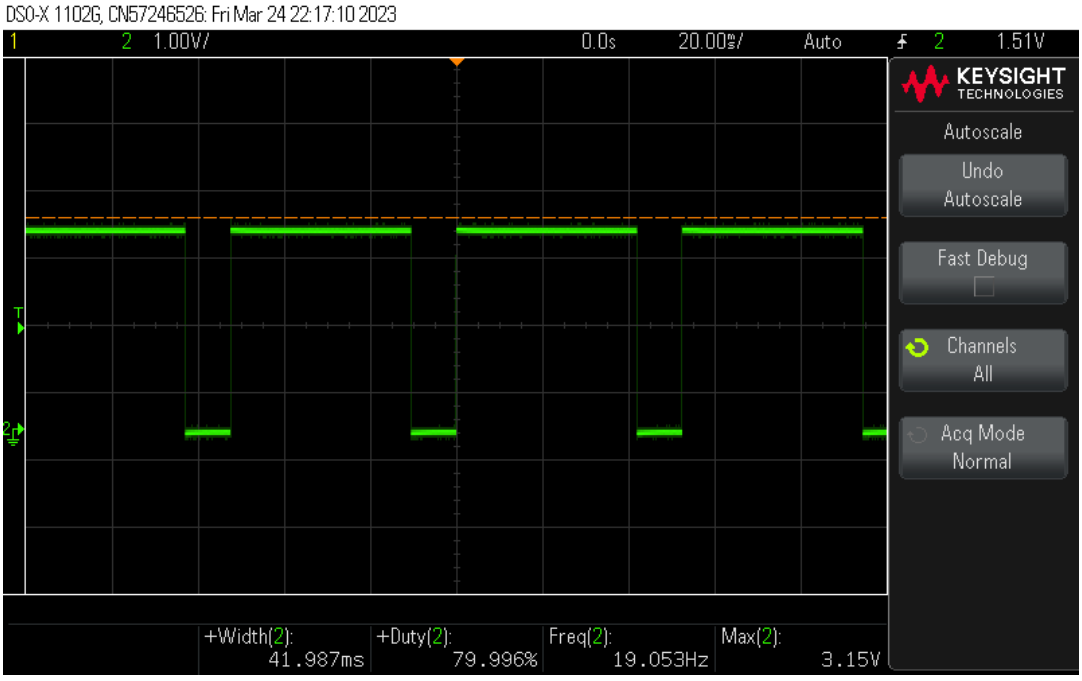
The STM output when character A is pressed over UART UI resulting in duty cycle incrementing by 5% and when B is pressed resulting in decrease in duty cycle by 5% is as follows:

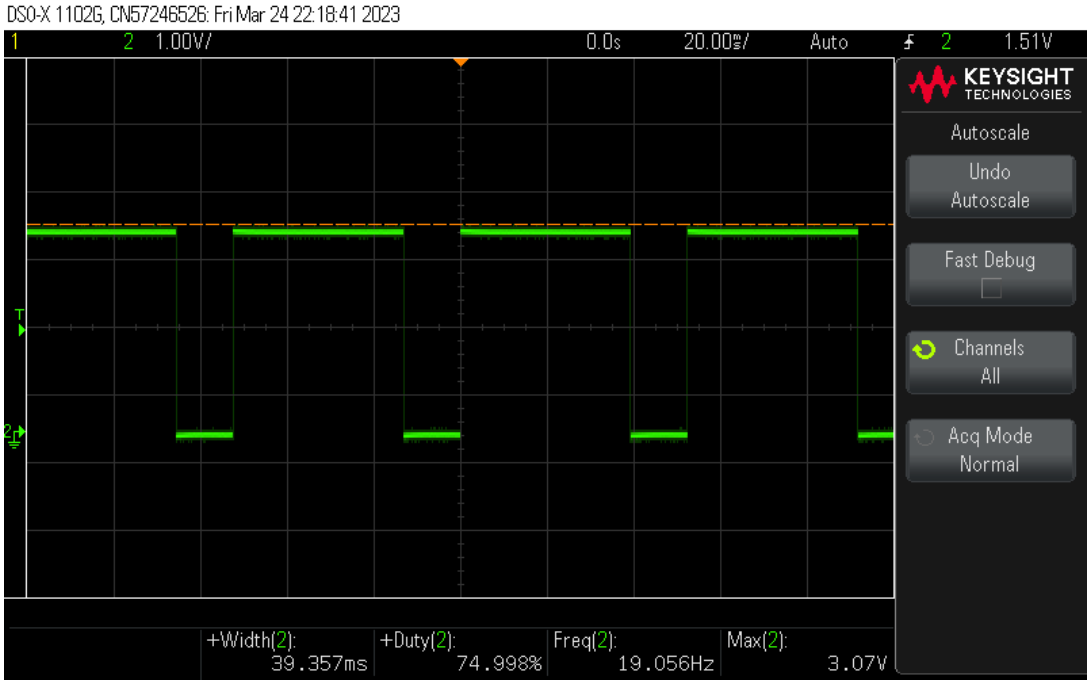
Incrementing by 5%:





Decrementing by 5% :





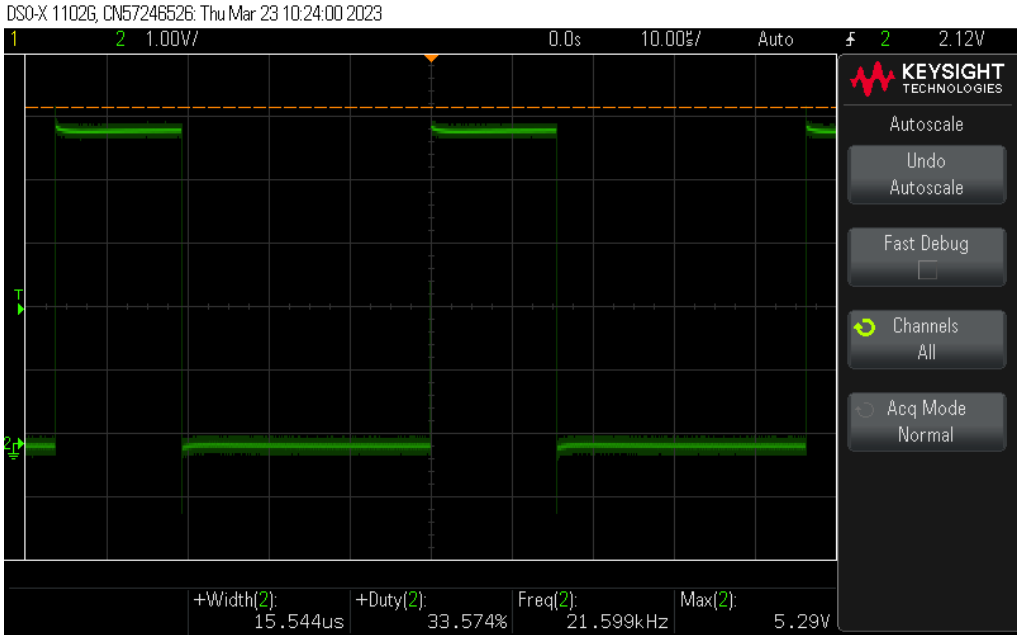
Simple UI repressing the lab 3 part 3 STM part is as below. The sample block is also seen from the picture below.


New
Open
Save
Connect
Disconnect
Options
Clear Data
View
Button Pressed!
Increasing Duty Cycle !
Current Duty Cycle = 700
Button Pressed!
Increasing Duty Cycle !
Current Duty Cycle = 800AA
Character A Detected
Increasing duty cycle by 5%
B
Character B Detected
Decreasing duty cycle by 5%
P
Character P Detected
Printing current cycle !
Current Duty Cycle = 80 %
Button Pressed!
Increasing Duty Cycle !
Current Duty Cycle = 900
Button Pressed!
Increasing Duty Cycle !
Current Duty Cycle = 1000
!! Max Duty Cycle Reached !!
usbserial-0001 / 19200 8-N-1
Connected 00:40:45, 2,558 / 30 bytes

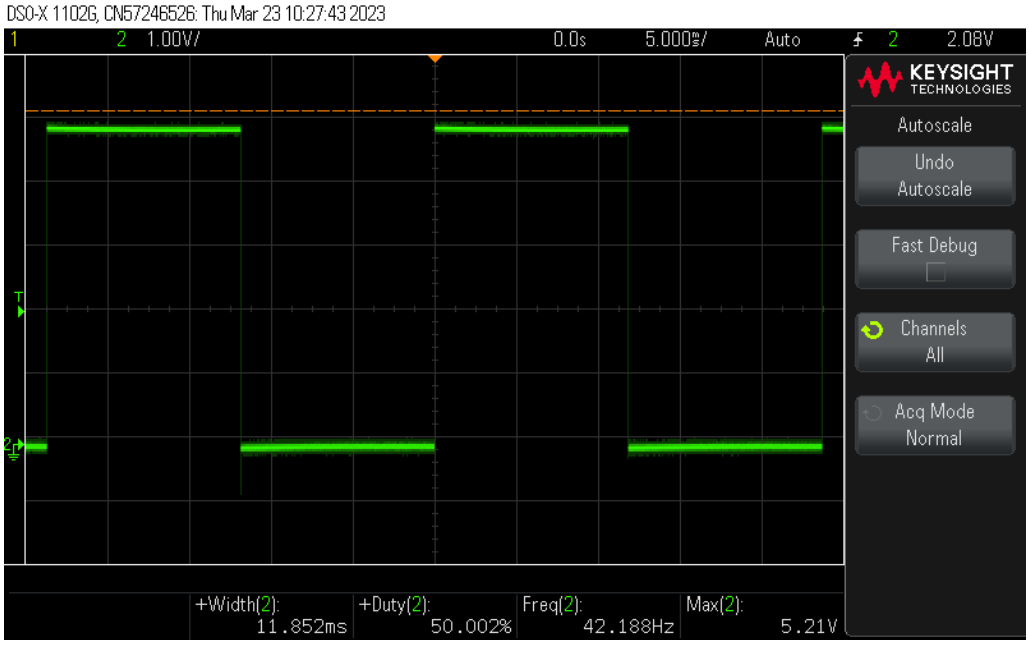
Lab 3 Part 3 (Supplemental)

Output :

The 8051 outputs for PWM on pin 1.3 are as follows. Running at 33% duty cycle and frequency of X2 mode.



The output with High-Speed toggle output enabled is as follows:
Result makes PWM set to 50% duty cycle i.e. ON for 50% time and OFF for 50%.
The frequency will reach upto 42Hz.



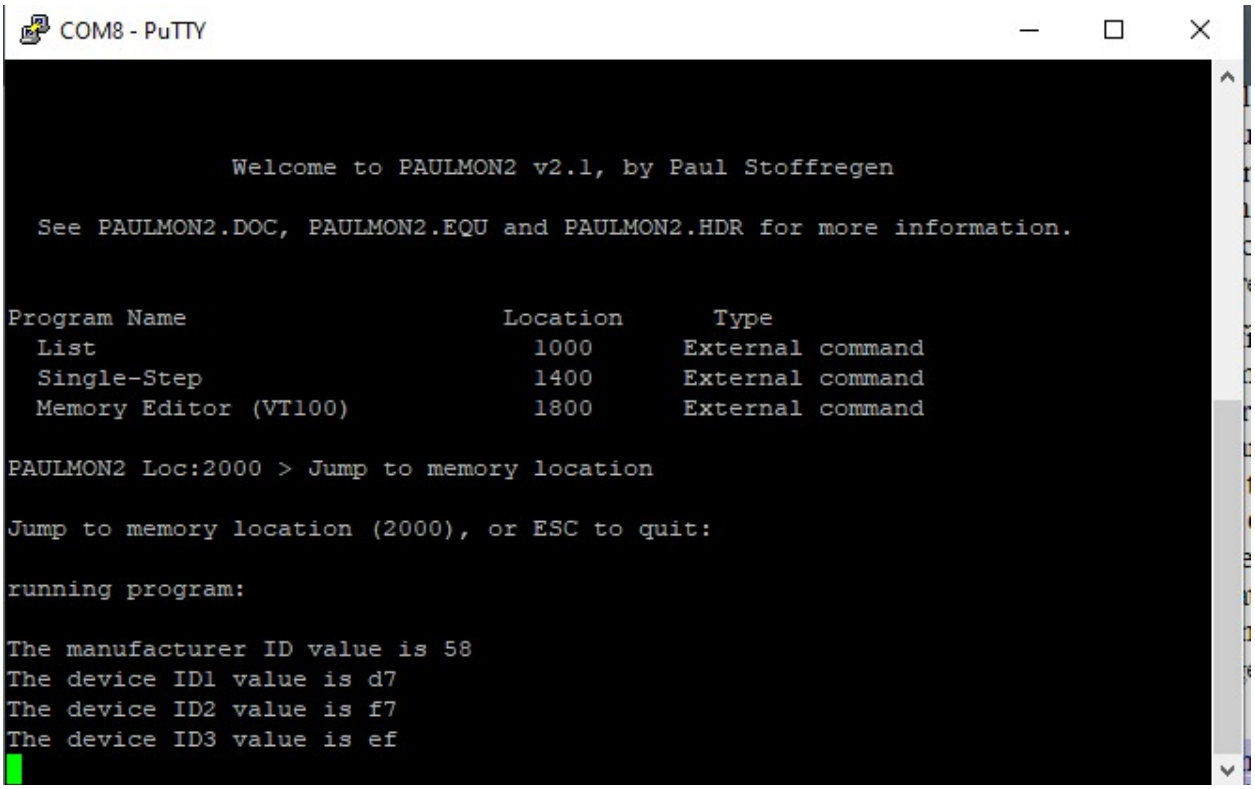
The output with Fclk_periph set to max frequency is as follows. It maintains the frequency of High output mode when triggered. Code also implements power down and Idle mode. The requirements are verified and tested for each one of the outputs along with watch dog timer.

Code also implements the IDLE mode and POWERDOWN mode and the firmware to get out of the interrupt using INT0 interrupt. Setting this pin to LOW will make system come out of power down and idle mode.

Code also implements Fclk_periph in minimum and maximum frequency mode as seen in image below.

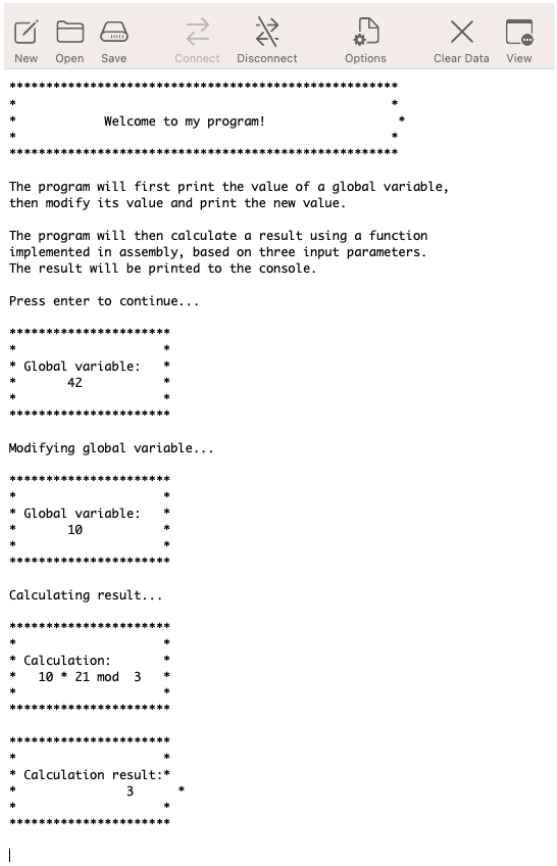
Challenges :

Challenge 2 : ISP Function for AT8951RC2
The code implementing three ISP calls has been implemented to figure out manufacture ID, Device ID1,ID2 and ID3. The UI result is as follows. The values returned matches the values available in data sheet.



Challenge 3: C and Assembly Integration

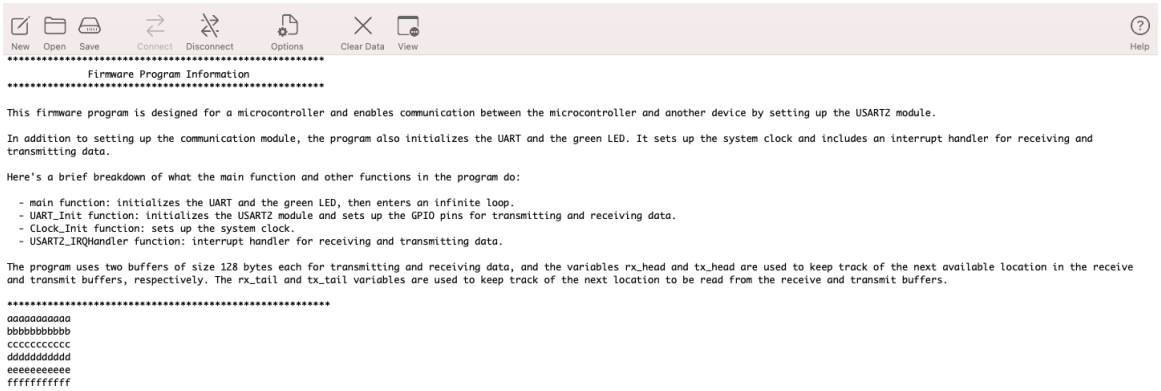
The code available in challenge folder have a UI which shows how to use an assembly function to calculate ((param2 * param3)modparam1). Along with that its changes the value of global variable set in assembly file in C and show the changes.
UI from the code is as follows:



Challenge 4: Interrupt Driven vs Polling Based UART Driver

Based on code uploaded interrupt driver UART driver(using FIFO buffer) is found to be much faster than polling based approach. UI output from interrupt driven firmare is as follows:

UART output below shoes the buffering capacity of interrupt driven approach is significantly higher than polling



Challenge 5: SDCC Heap Management

Malloc's functionality in heap management has been already discussed in above section called "Lab 2 Part 2 Heap Management Challenge".

Submission Questions

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

b) What compiler (including revision) did you use?
SDCC 2.6.2

c) What exactly (include name/revision if appropriate) did you use to build your code (what IDE, make/makefile, or command line)?
Code Blocks for 8051 part of labs
STMCube IDE for ARM part of lab

d) Did you install and use any other software tools to complete your lab assignment?
Yes, I had lot of issues with installing putty for MAC. I used an MAC alternative of putty called "CoolTerm".

e) Did you experience any problems with any of the software tools? If so, describe the problems.
Yes, I had lot of issues with IDE and SDCC environment setup for MAC OS. I was able to install SDCC with MAC environment, but Code Blocks doesn't support latest or even old version of MAC OS.