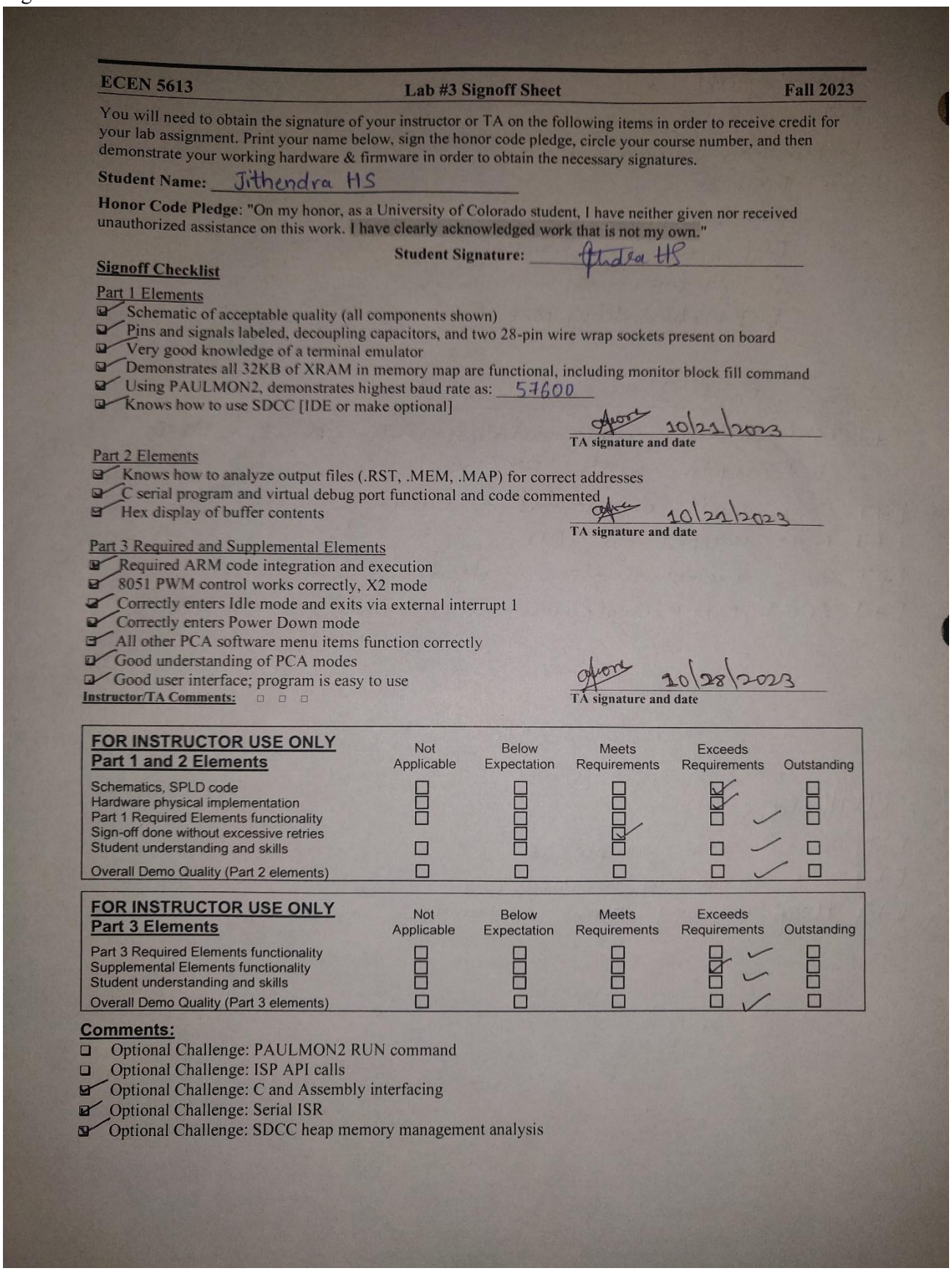
LAB 3 REPORT

Signoff sheet:



ECEN 5613 **Lab #3 Signoff Sheet** **Fall 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: Jithendra HS

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: Jithendra HS

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: 57600
- ☑ Knows how to use SDCC [IDE or make optional]

TA signature and date 10/21/2023

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

TA signature and date 10/21/2023

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:

TA signature and date 10/28/2023

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

[+] Used batchisp.
[+] Paulmon memory editing functional.
[#] Part 2
    [#] Decent UI.
    [+] Corner cases handled.
    [+] All commands functional.
    [+] Virtual debug port functional.

Lab3 Part 3

[+] PWM working implementation with UART & GPIO interrupts with RX/TX buffers
[+] Code well commented & modular.
[+] PWM mapped to LED
[+] Bare-metal implementation
[+] 8051 PCA implementation – PWM, watchdog & high speed o/p

Challenges

[+] Serial ISR completed.
[+] C & assembly completed.
[+] Heap memory mgmt analysis completed.

PART1:

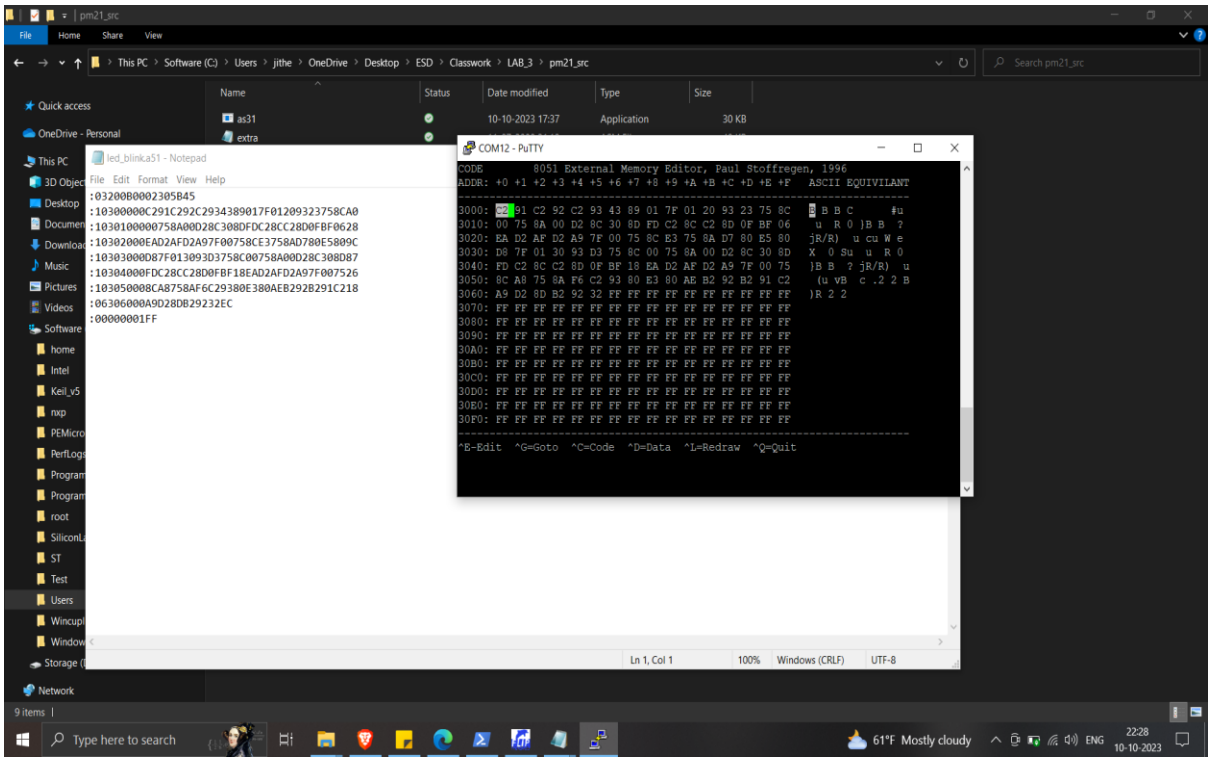


Figure 1.1 Indicates the memory inspection using PAULMON2 monitor.

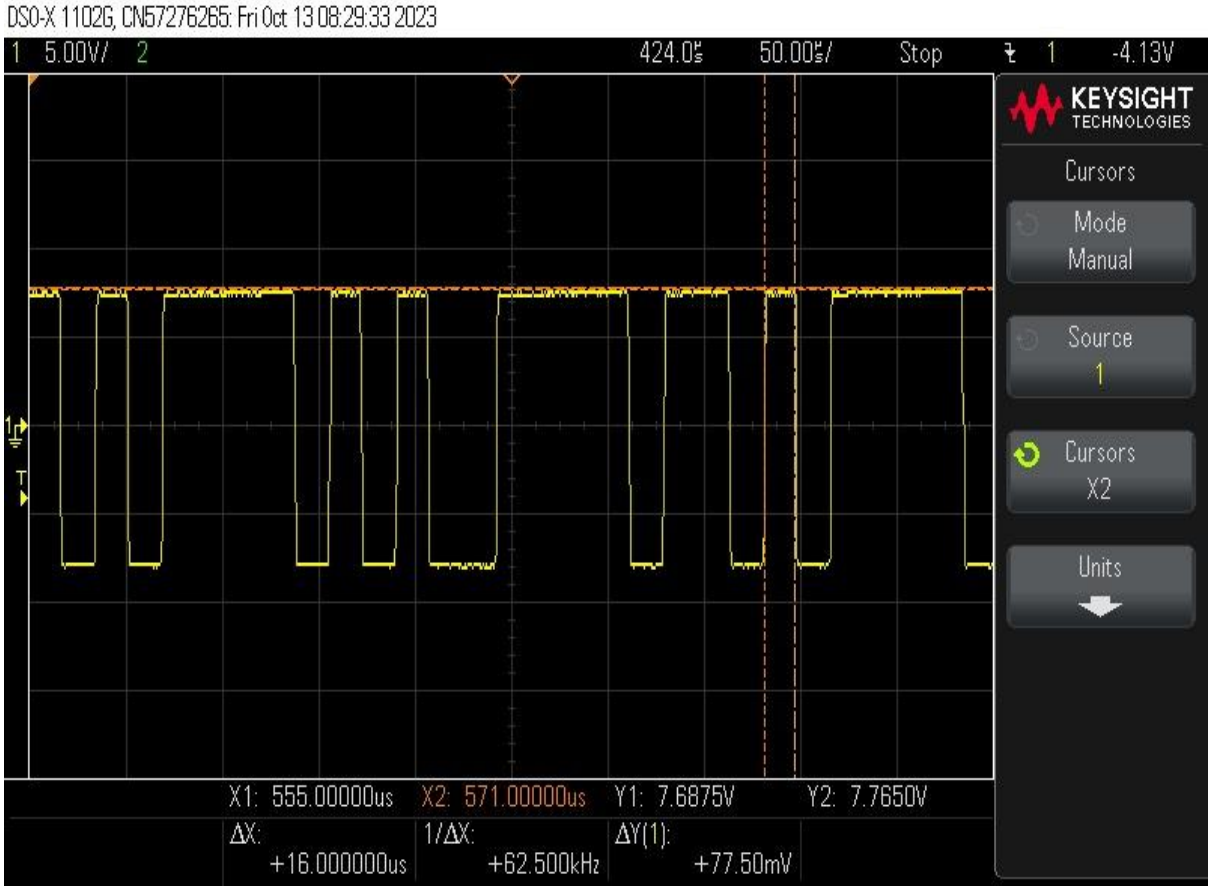


Figure 1.2 Shows the maximum baud rate measured using oscilloscope (logically 57600) 62500.

PART2:

```
               Welcome to PAULMON2 v2.1, by Paul Stoffregen

  See PAULMON2.DOC, PAULMON2.EQU and PAULMON2.HDR for more information.


Program Name                      Location      Type
  List                              1000      External command
  Single-Step                       1400      External command
  Memory Editor (VT100)             1800      External command

PAULMON2 Loc:2000 > Jump to memory location

Jump to memory location (2000), or ESC to quit: 3000

running program:

Specify the buffer size (range 32 to 4800) divisible by 16:
32
* Enter any UPPERCASE letter to append characters into buffer 0
* Press '+' to allocate a new buffer
* Press '-' to remove allocated buffers
* Press '?' to get the heap report
* Press '=' to get buffer 0 content
* Press '@' to restart the program
? mode
Total characters count: 10
Since last '?': 10
Buffer 0 -->
 Start address: X:0x0014
 End address: X:0x0034
 Allocated size: 32
 Storage character counts: 5
 Free space available: 27
Buffer 0 content:

JITHU

Buffer 1 -->
 Start address: X:0x0036
 End address: X:0x0056
 Allocated size: 32
 Storage character counts: 0
 Free space available: 32
Buffer 1 content:


+ mode, specify buffer size (range 20 to 800):
48
Allocated memory for buffer #2
? mode
Total characters count: 15
Since last '?': 5
Buffer 0 -->
 Start address: X:0x0014
 End address: X:0x0034
 Allocated size: 32
 Storage character counts: 0
 Free space available: 32
Buffer 0 content:


Buffer 1 -->
 Start address: X:0x0036
```

```
 Start address: X:0x0036
 End address: X:0x0056
 Allocated size: 32
 Storage character counts: 0
 Free space available: 32
Buffer 1 content:


Buffer 2 -->
 Start address: X:0x0058
 End address: X:0x0088
 Allocated size: 48
 Storage character counts: 0
 Free space available: 48
Buffer 2 content:


- mode, specify buffer number to destroy:
2
Freed buffer #2, try '?' to get info of existing buffers
? mode
Total characters count: 19
Since last '?': 4
Buffer 0 -->
 Start address: X:0x0014
 End address: X:0x0034
 Allocated size: 32
 Storage character counts: 0
 Free space available: 32
Buffer 0 content:


Buffer 1 -->
 Start address: X:0x0036
 End address: X:0x0056
 Allocated size: 32
 Storage character counts: 0
 Free space available: 32
Buffer 1 content:


= mode

X:0x0014 : 4a 49 54 48 55 0 0 0 0 0 0 0 0 0 0 0
X:0x0024 : 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Figure 2.1 Collected from heap memory management program implementation.

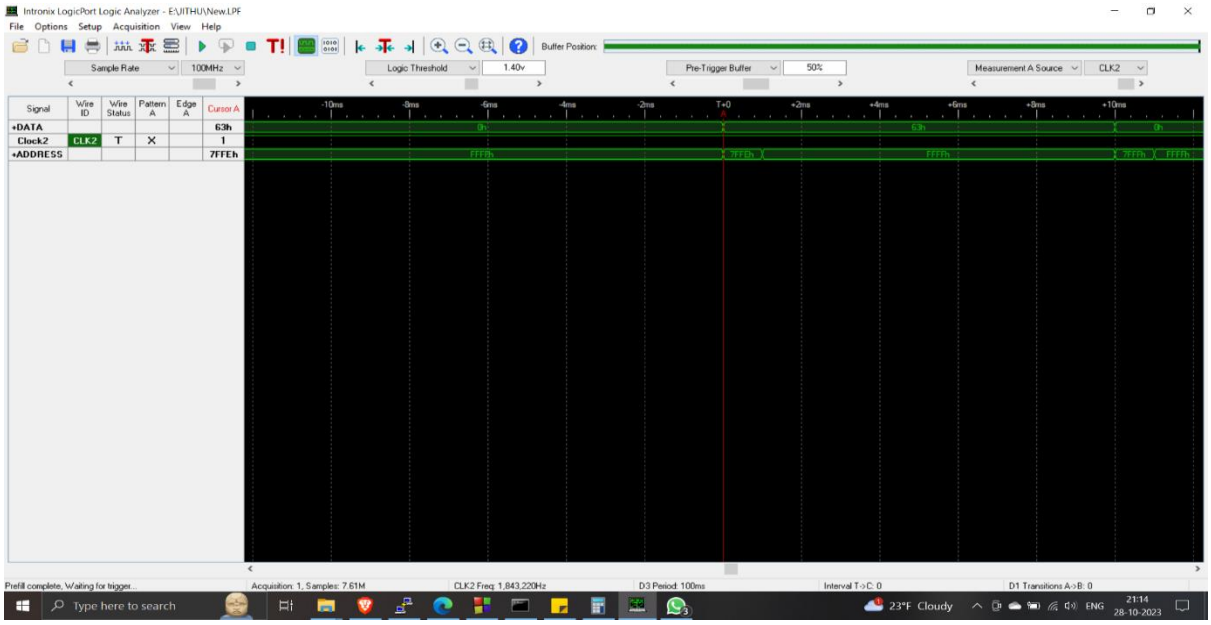


Figure 2.1 shows the value(69h) latched at the buffer due to virtual debug port execution trying to put value to address 0x7FFEh

PART3:

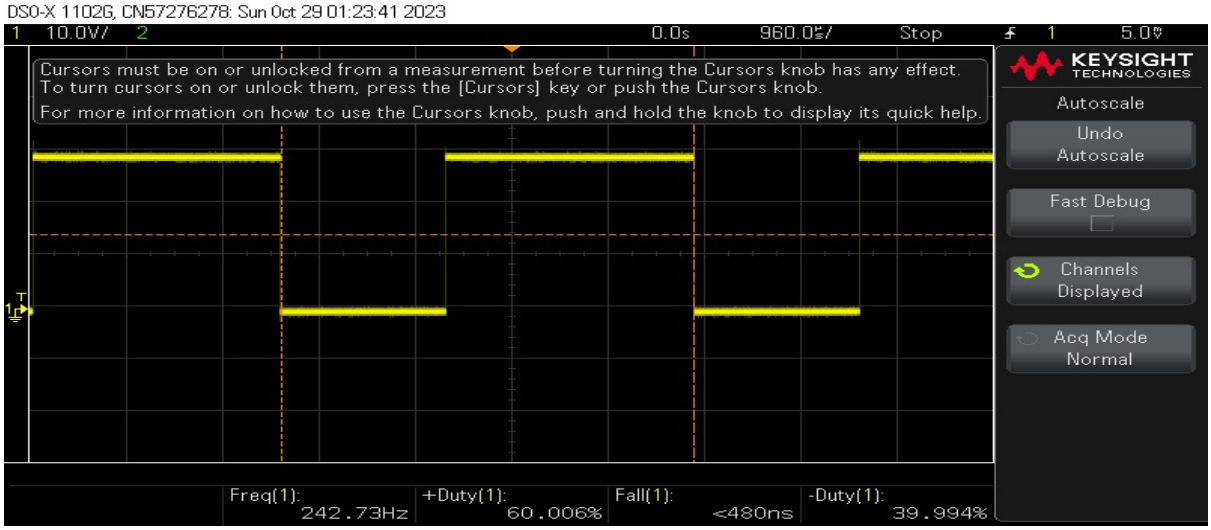


Figure 3.1 Shows the initial 60% duty dycle PWM output from stm32 board

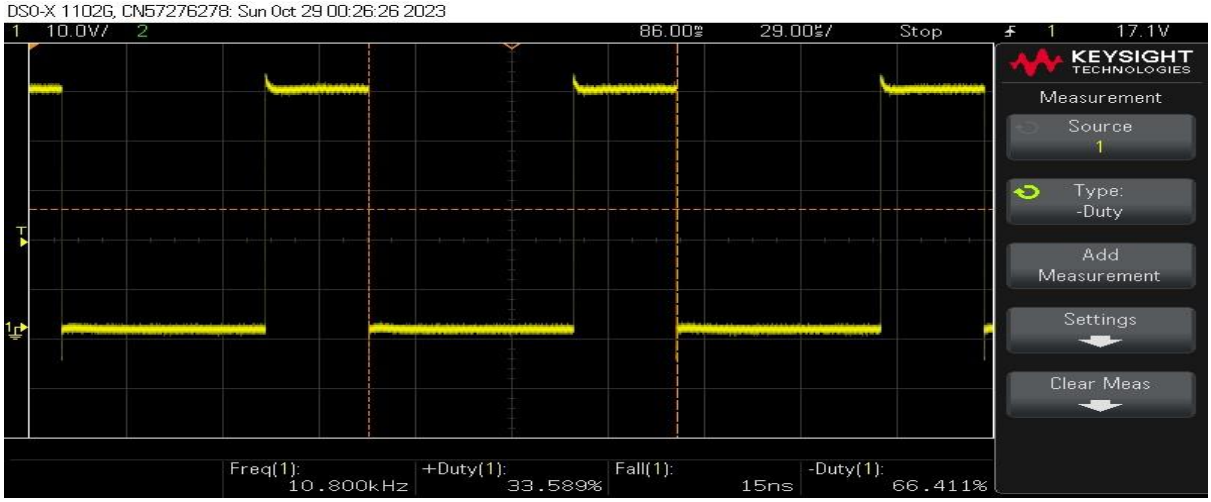


Figure 3.2 shows the duty cycle of PWM output from 8051 board.

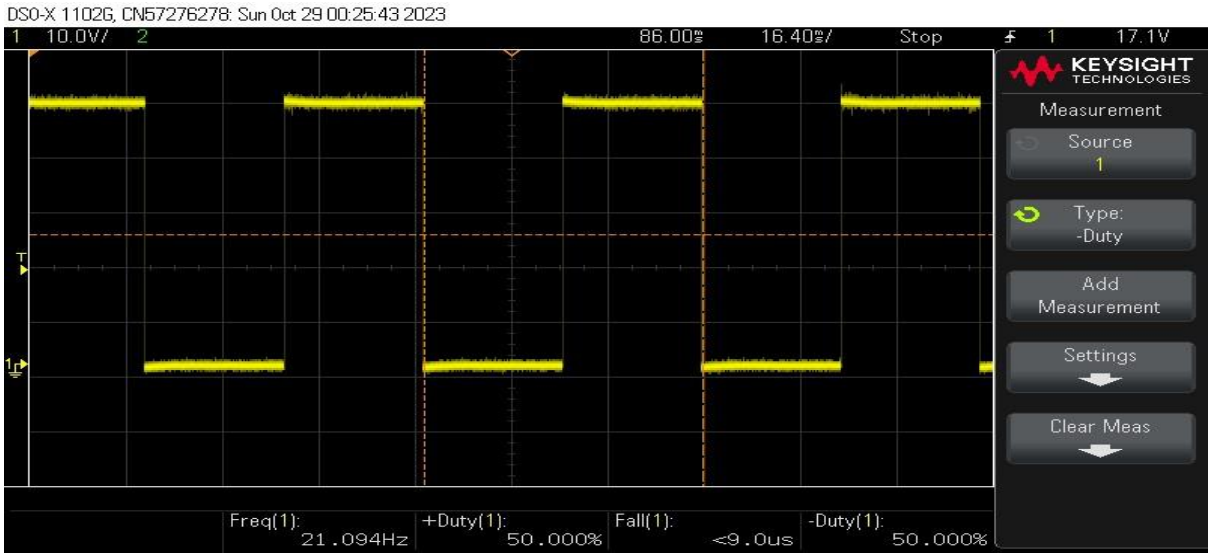


Figure 3.3 shows the duty cycle of High speed output from 8051 board

STM32 baud rate calculation:

Baud rate value to be loaded to register USART_BRR = Fpclk/expected number of data units

USART_BRR= 16Mhz/9600 = 1667 in decimal

Completed most of additional challenges like:

1. Combined assembly and c code.
2. Implemented buffer for UART transmission in ARM code.
3. Heap memory allocation

Answers:

a. OS: Microsoft Windows 10 Home single language edition
   Version: 10.0.19045 Build 19045
b. SDCC version 4.3.2
c. Code blocks 20.3
d. Putty, Flip
e. No problems faced.

Key learnings:

1. Learnt about what is Paulmon2 tool, how to use it and its functionality of memory inspection and execution of code by jumping to the program located space.
2. SDCC compiler, its syntax and how malloc and free works. 8051 PCA counters and its various functionalities
3. ARM architecture related UART, Timer registers and its interrupts.