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: Tharuni Gelli

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: Tharuni

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: 57000
- ☑ Knows how to use SDCC [IDE or make optional]

Reshpande 10/20/23.
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

Reshpande 10/20/23.
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: ☐ ☐ ☐

Reshpande 10/27/23
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

Part 1 and 2

(+) Good schematics.

~~(+) All commands working but are missing/failing~~

(-) (+), (?), (=) functional correctly.

(-) even though buffer is ~~freed~~ free empty still shows buffer size in (?) comment.

(+) Boundary conditions handeled well.

(-) Can improve UI.

Part 3

(+) ARM code functional with PB & UART & PWM mapped to PD12.

(+) PWM, WDT, High-Speed o/p implemented.

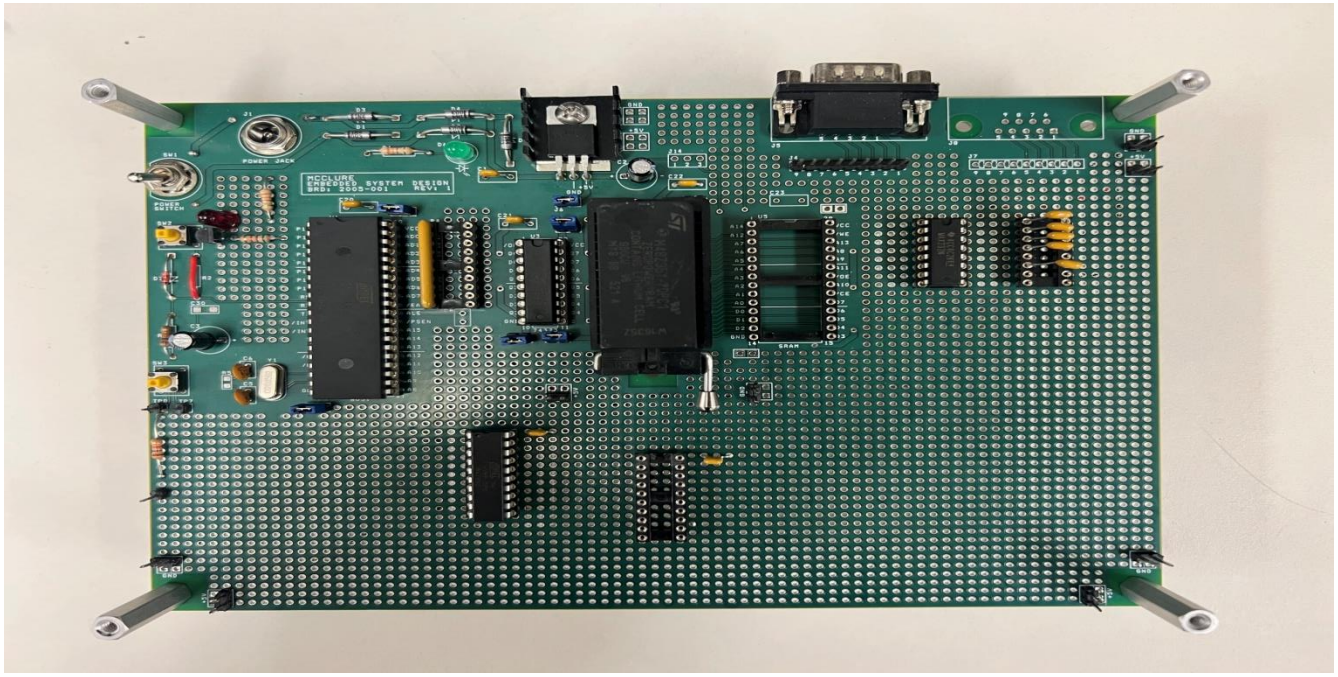
(+) Idle & Power Down funtional.

(+) Good understanding of PCA model.

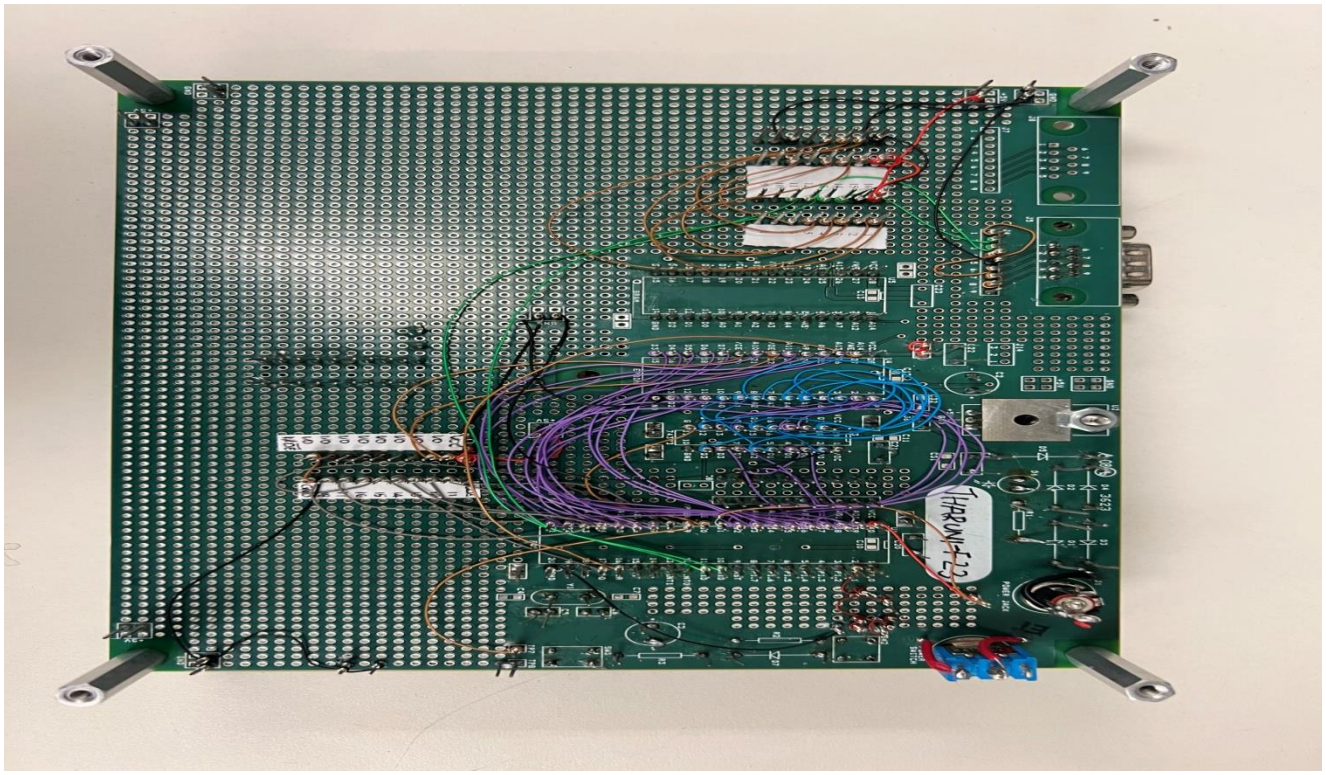
(+) Good understanding of memory management.

ESD LAB-3 WRITE UP – Tharuni Gelli

- 8051 Development board front side

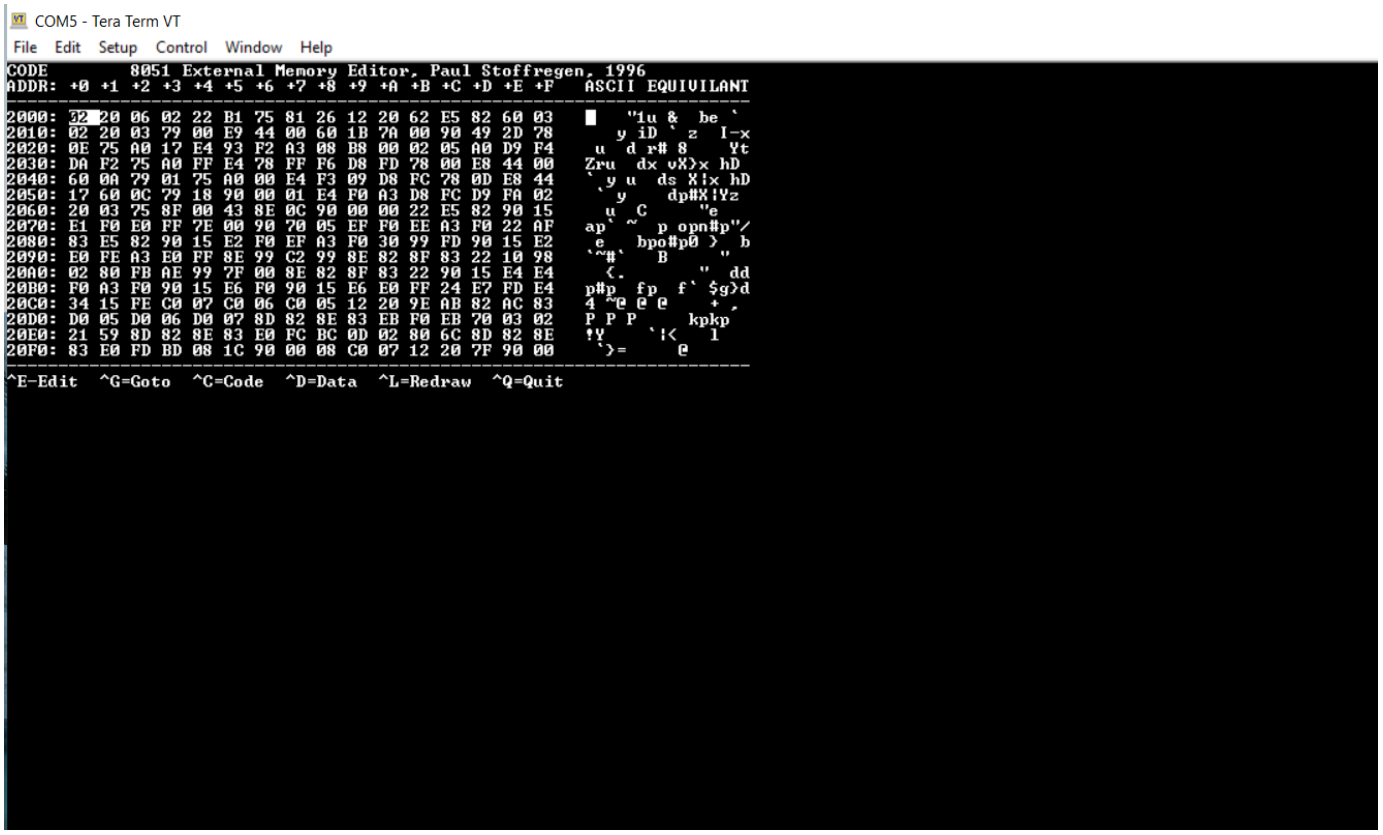


- 8051 Development board back side

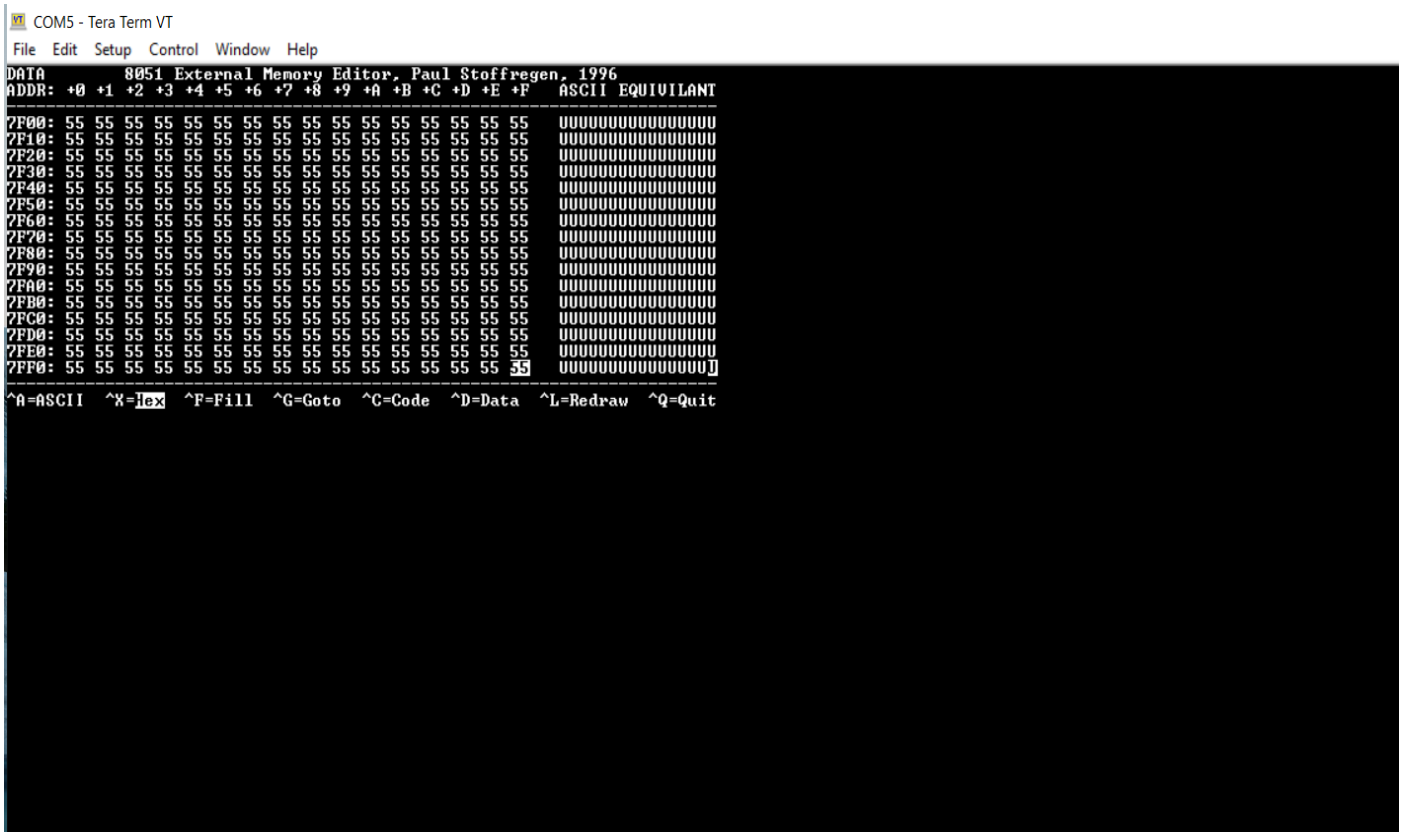


PAULMON COMMANDS:

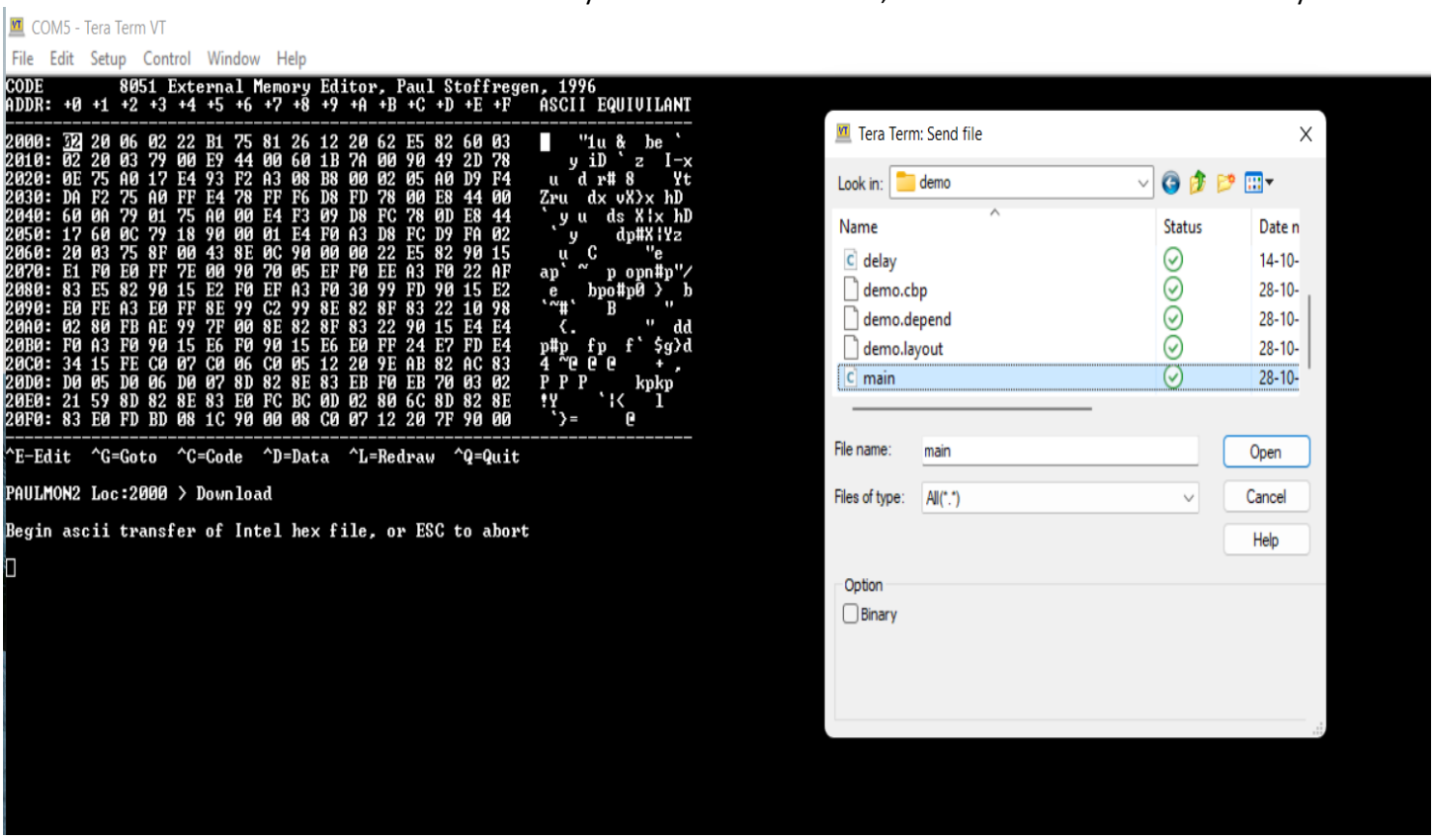
- E Command to see code memory

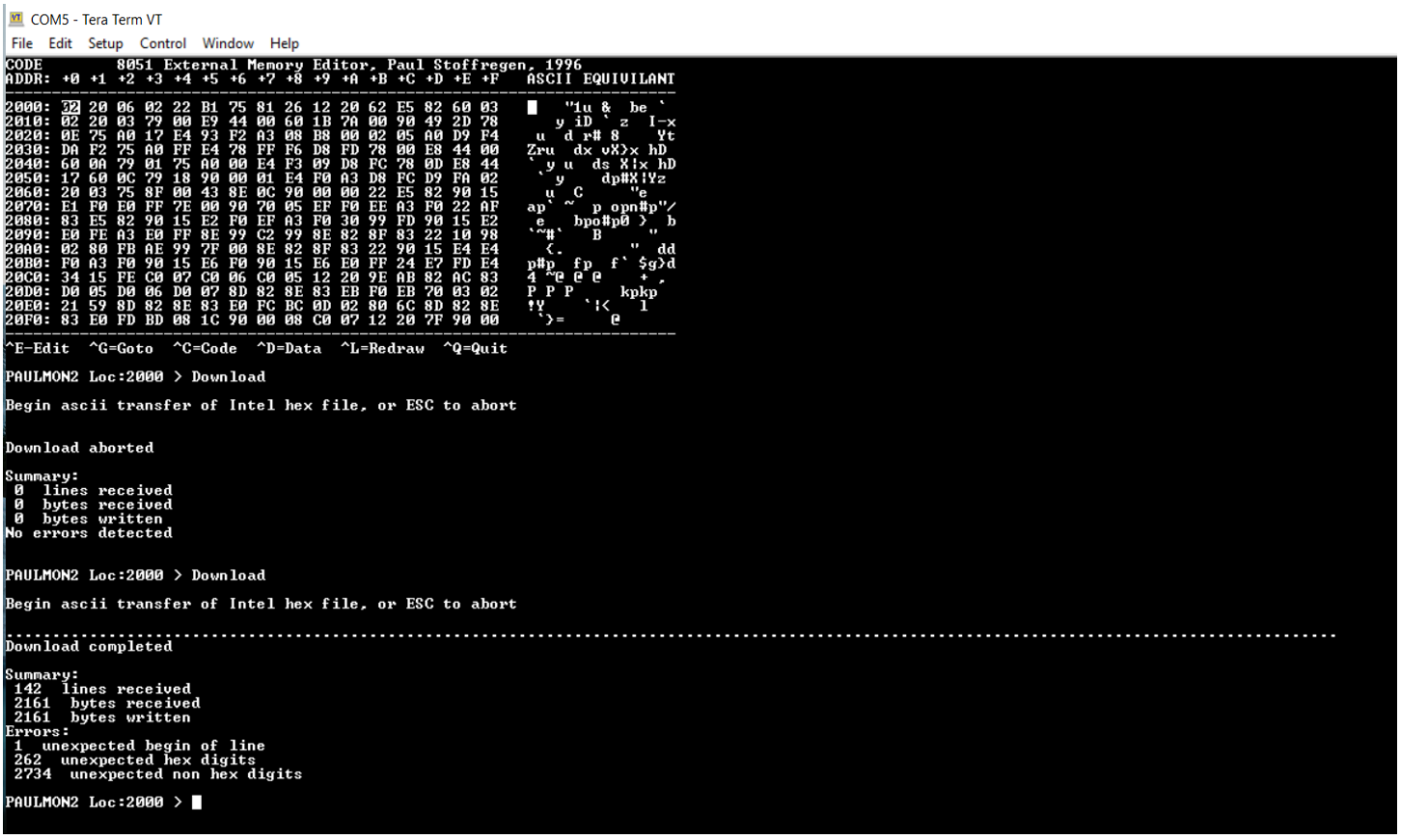


- Ctrl+D command to see data memory- From 0000 to 7FFF with 55



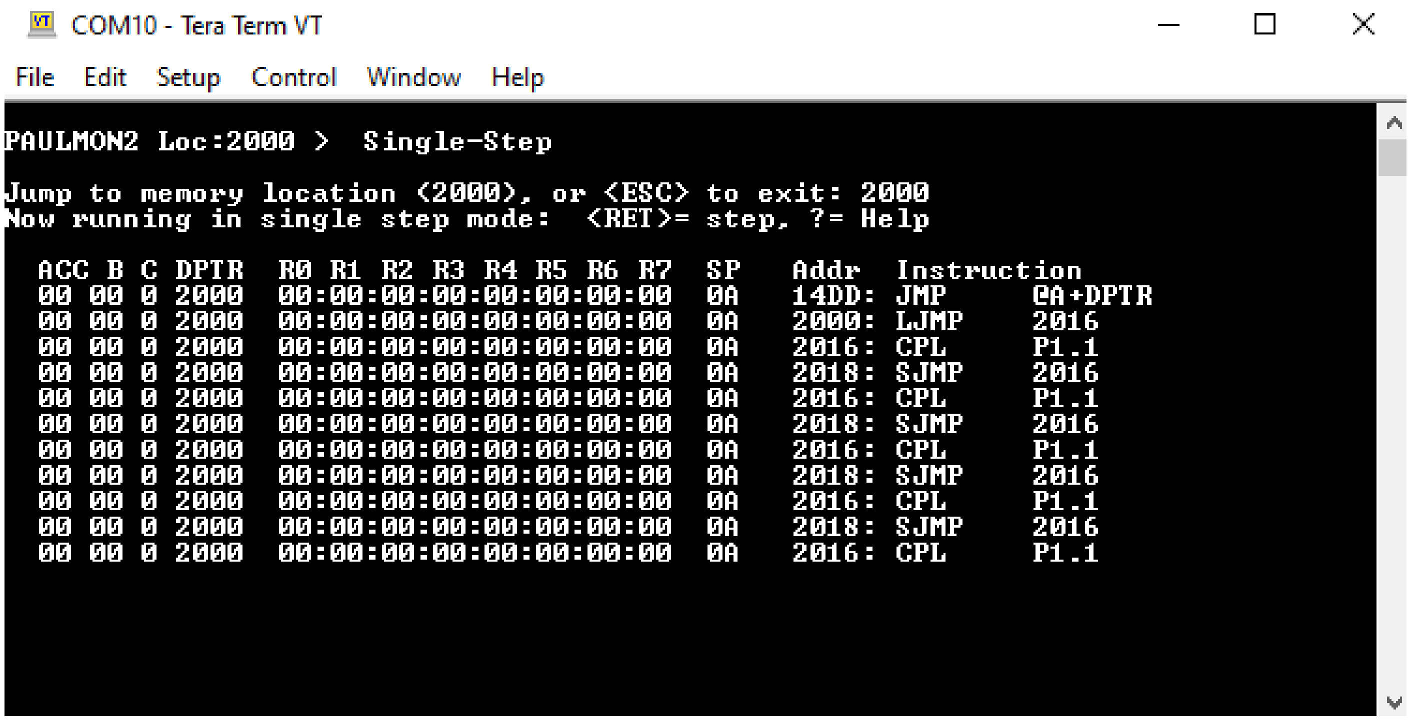
- D command to download HEX file directly from terminal emulator, which overwrites the data memory





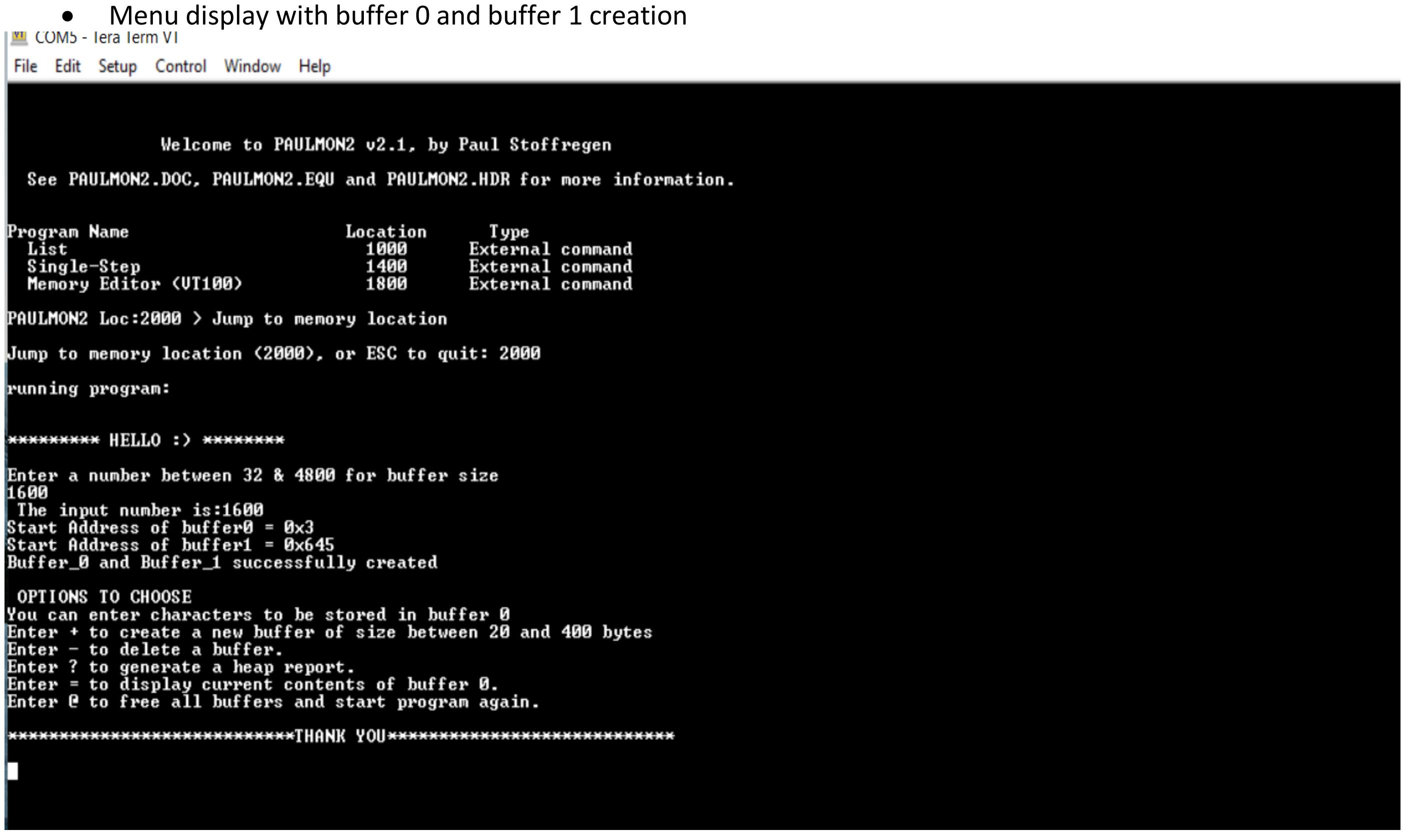
PART1-CHALLENGE – PAULMON as an on-chip debugger –

- For the code I submitted in debugger.asm and .hex file

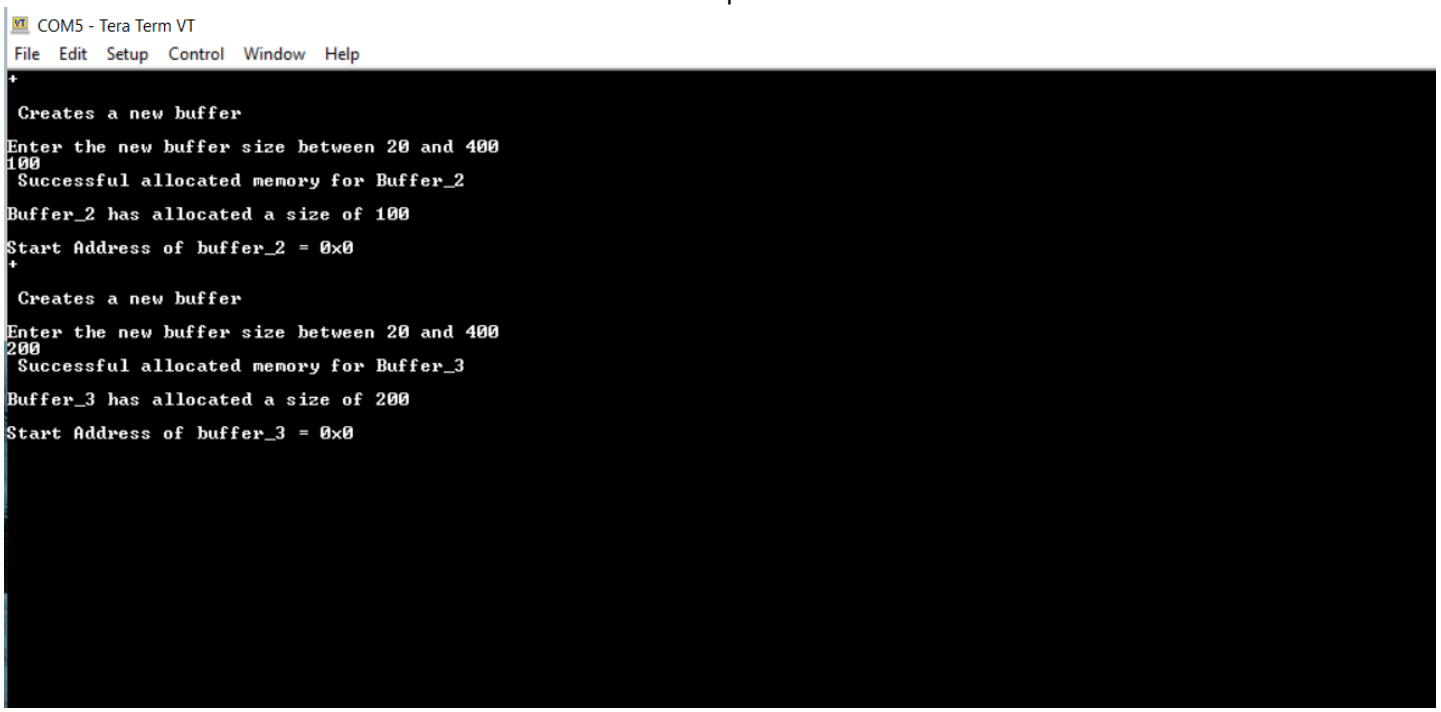


PART2: Buffers allocation using heap of 5000 bytes

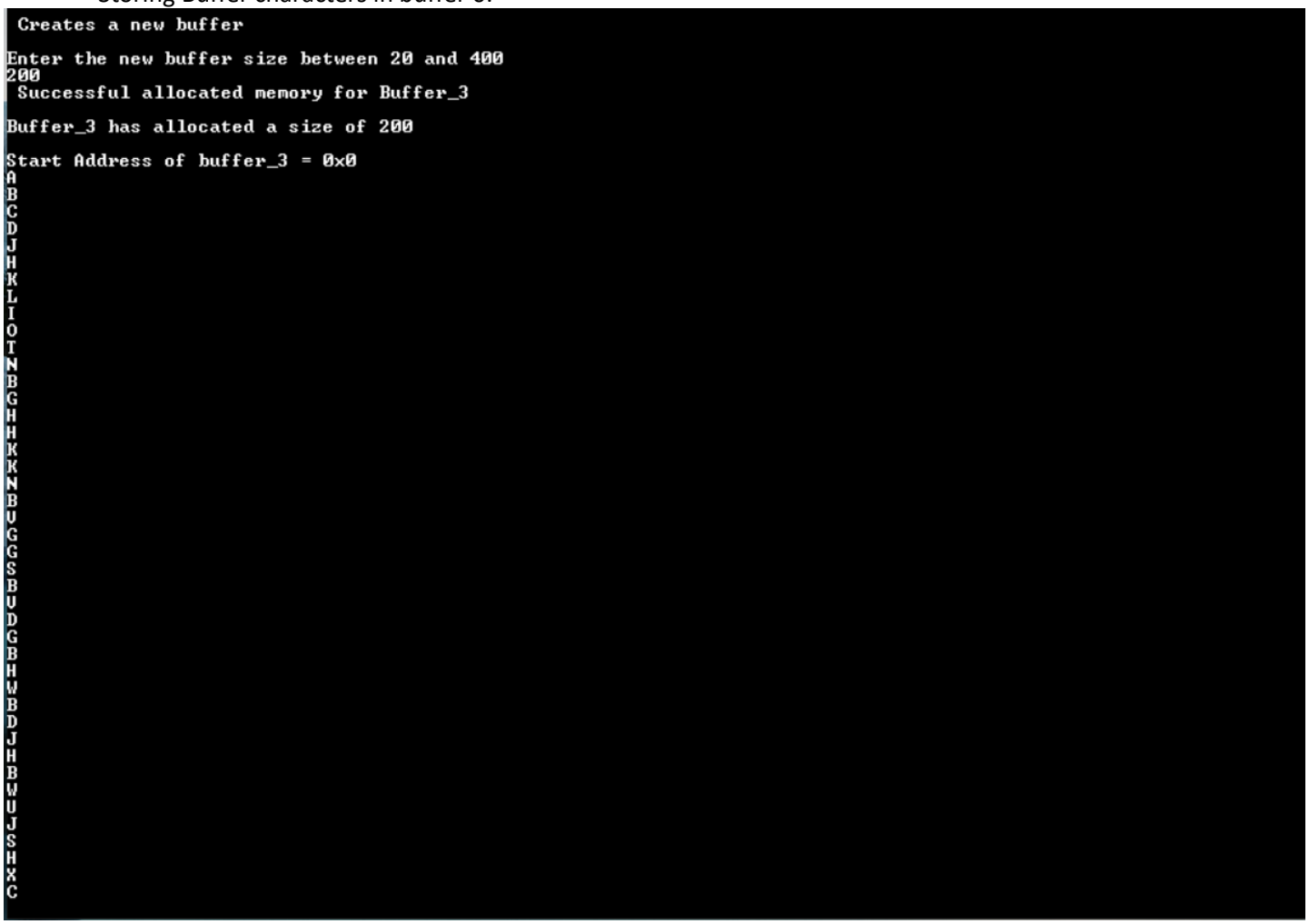
- Menu display with buffer 0 and buffer 1 creation



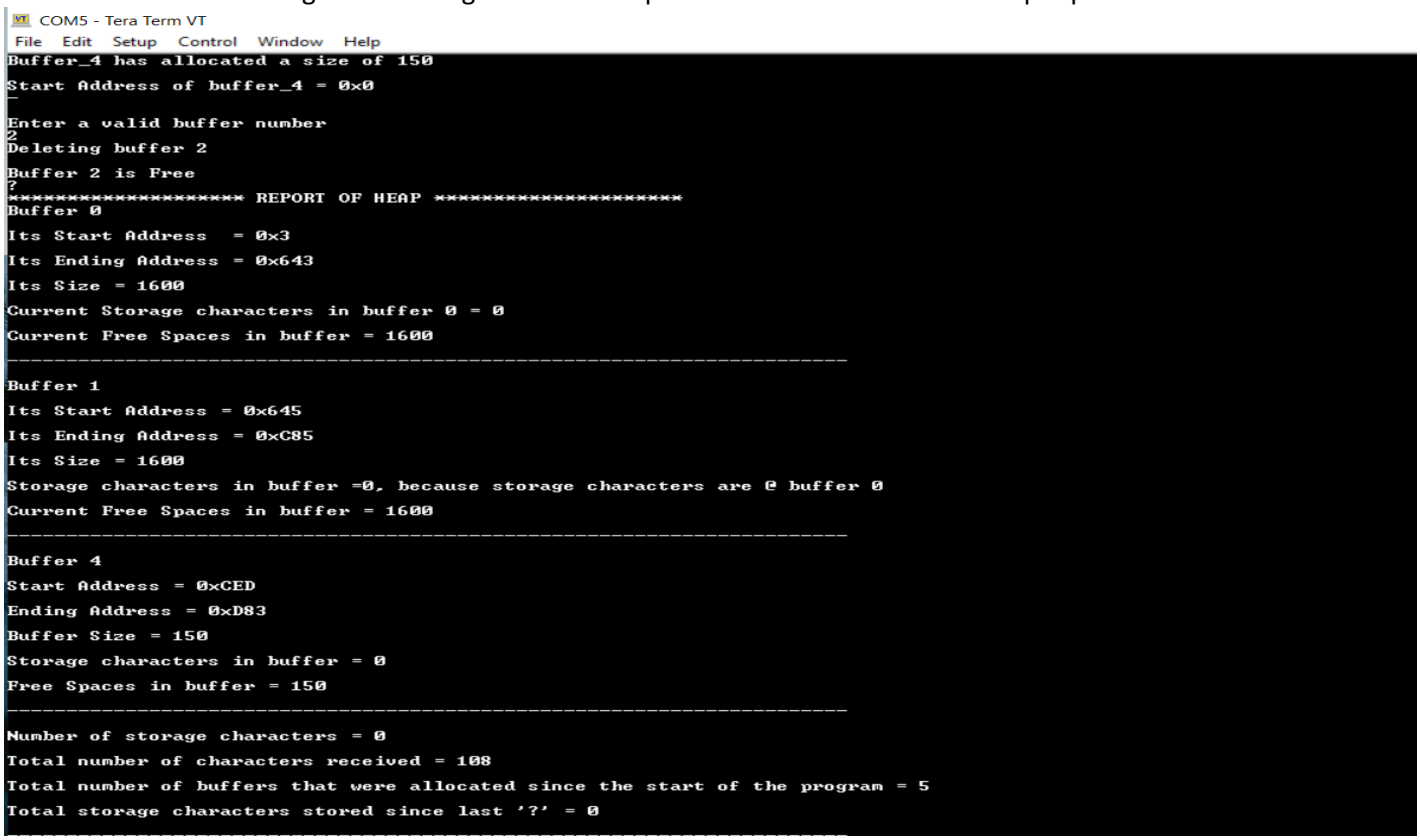
- ‘+’ Character for creation of n buffers with the size specified between 20 to 400



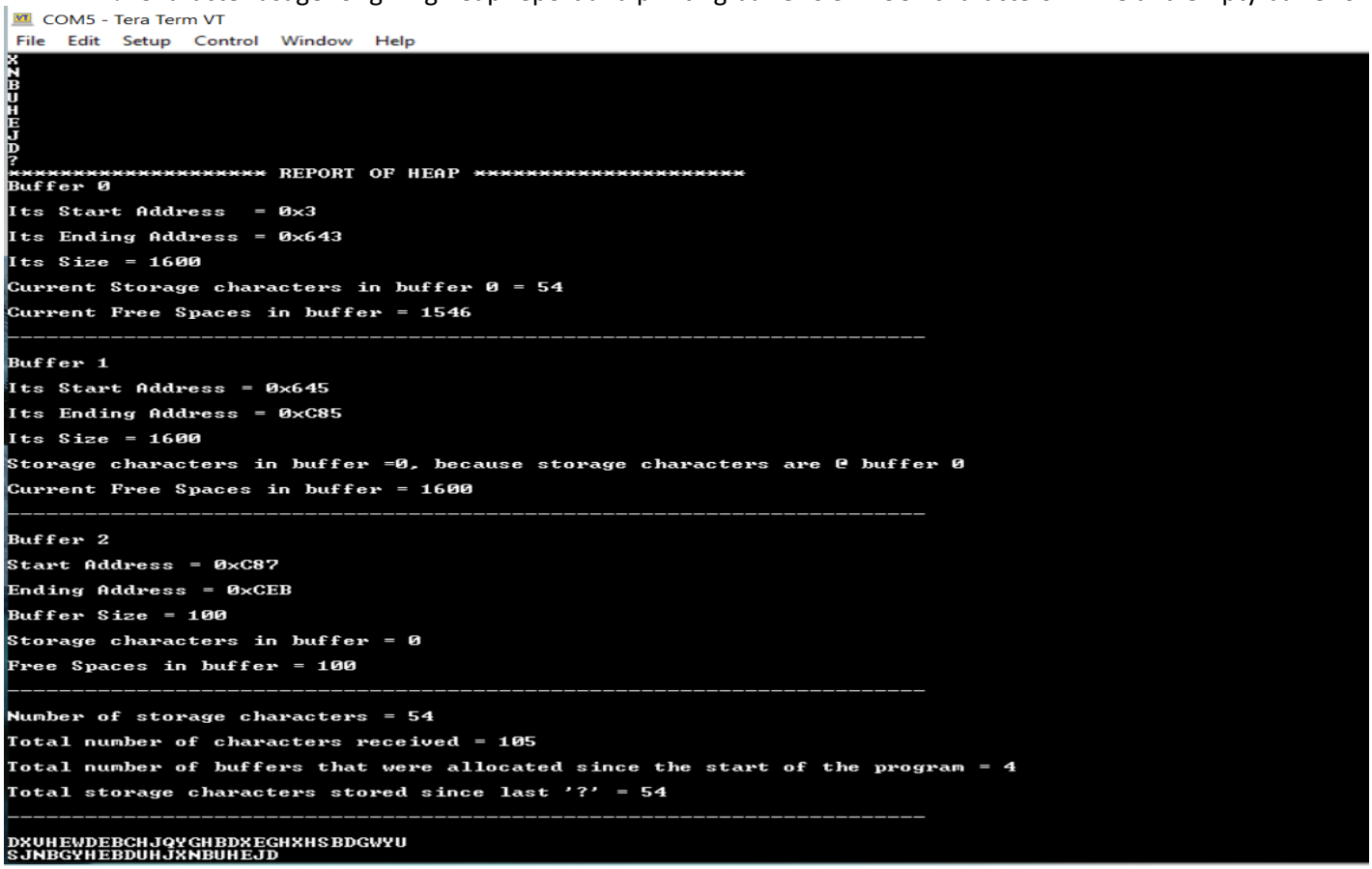
- Storing Buffer characters in buffer 0:



- ‘-‘ Character usage for deleting buffers with specified buffer number and heap report after that ‘-‘ character



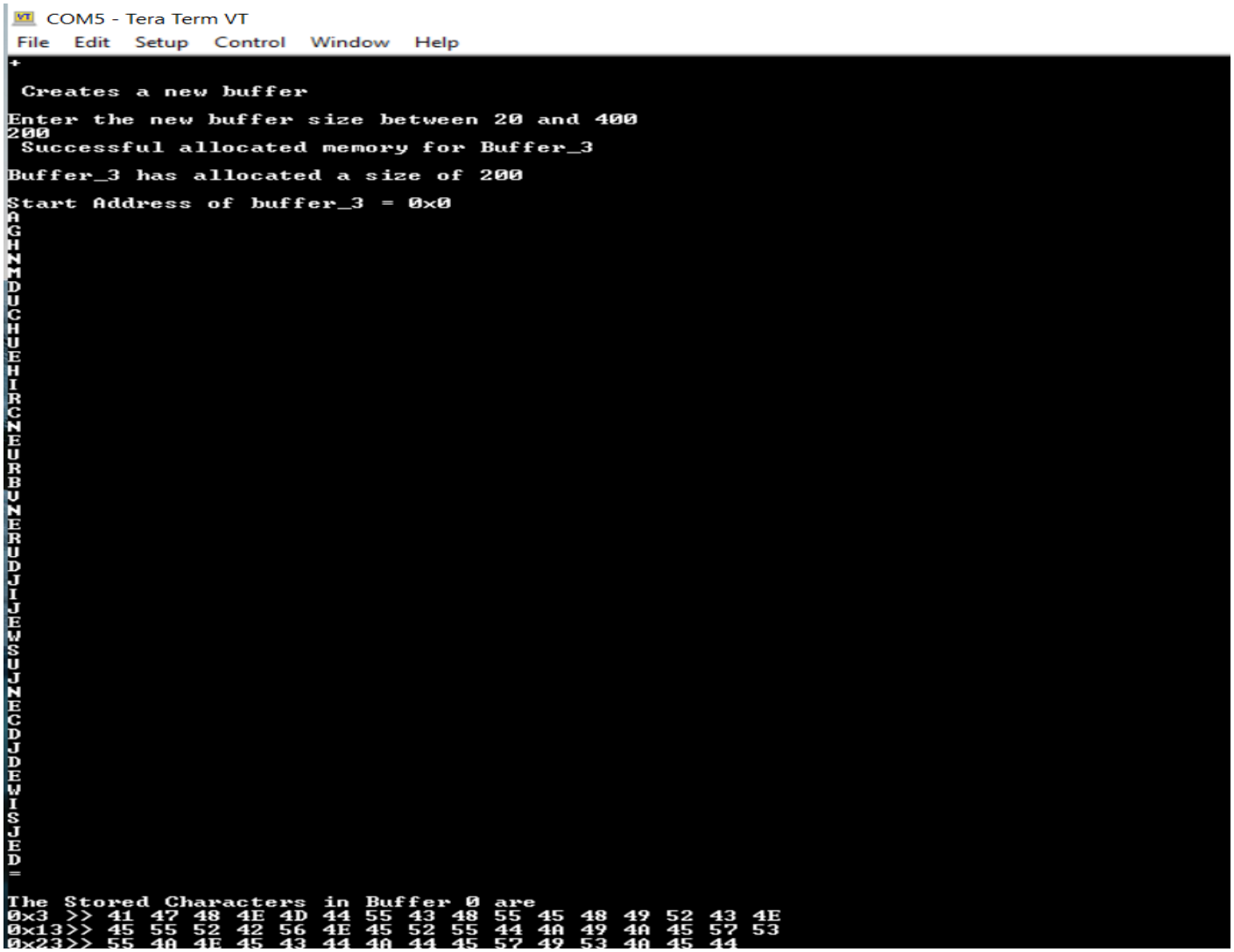
- ‘?’ Character usage for giving heap report and printing buffer 0 32 ASCII characters in line and empty buffer 0



- '@' Character for deleting all the buffers created till now!

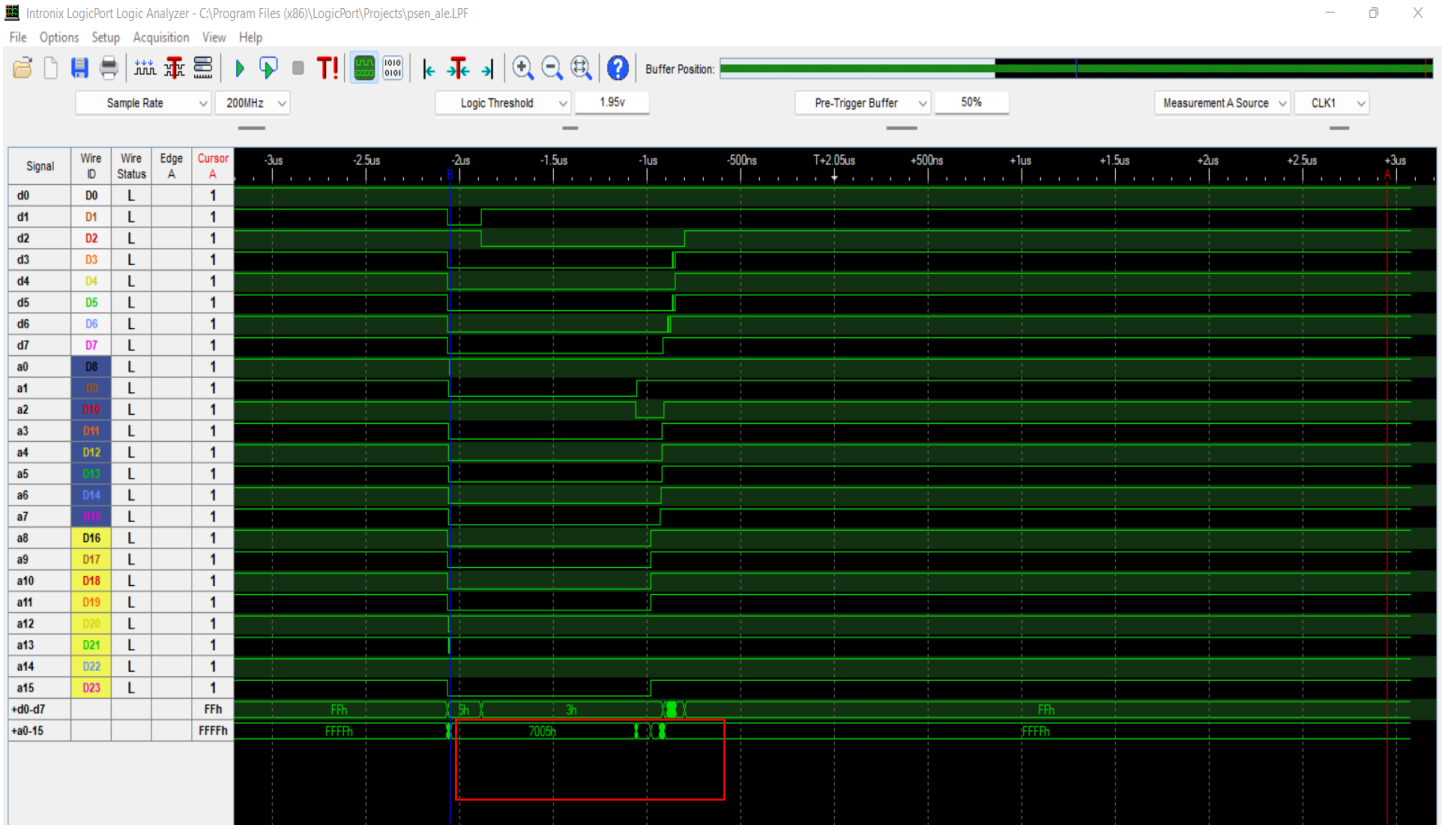
```
------------------------------------------------------------------
@

Deleting Buffer 0

Buffer 0 is free

Deleting Buffer 1

Buffer 1 is free

Deleting buffer_2

Buffer 2 is free

********* HELLO :) ********

Enter a number between 32 & 4800 for buffer size
```

- '=' Character for hex dump with 16bytes in each line with address of each storage character and its hex ASCII value.



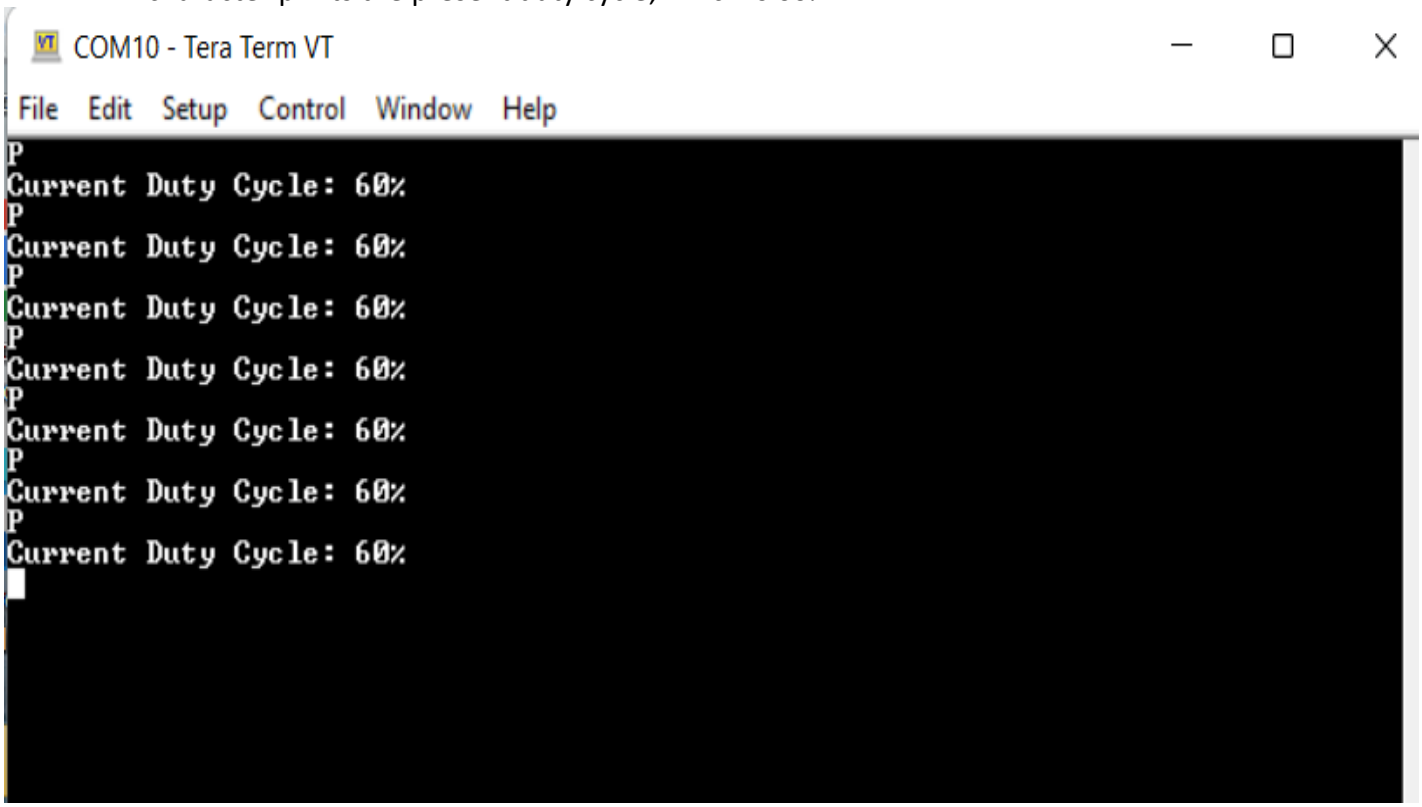
Virtual Debug Port:

- The address location 7005h is used for passing unique characters onto external data memory for each special characters. I am passing values 1,2,3,4 & 5 integers for each special characters in buffer. I am checking for '?' which passes a 3 integer value in 7005h.

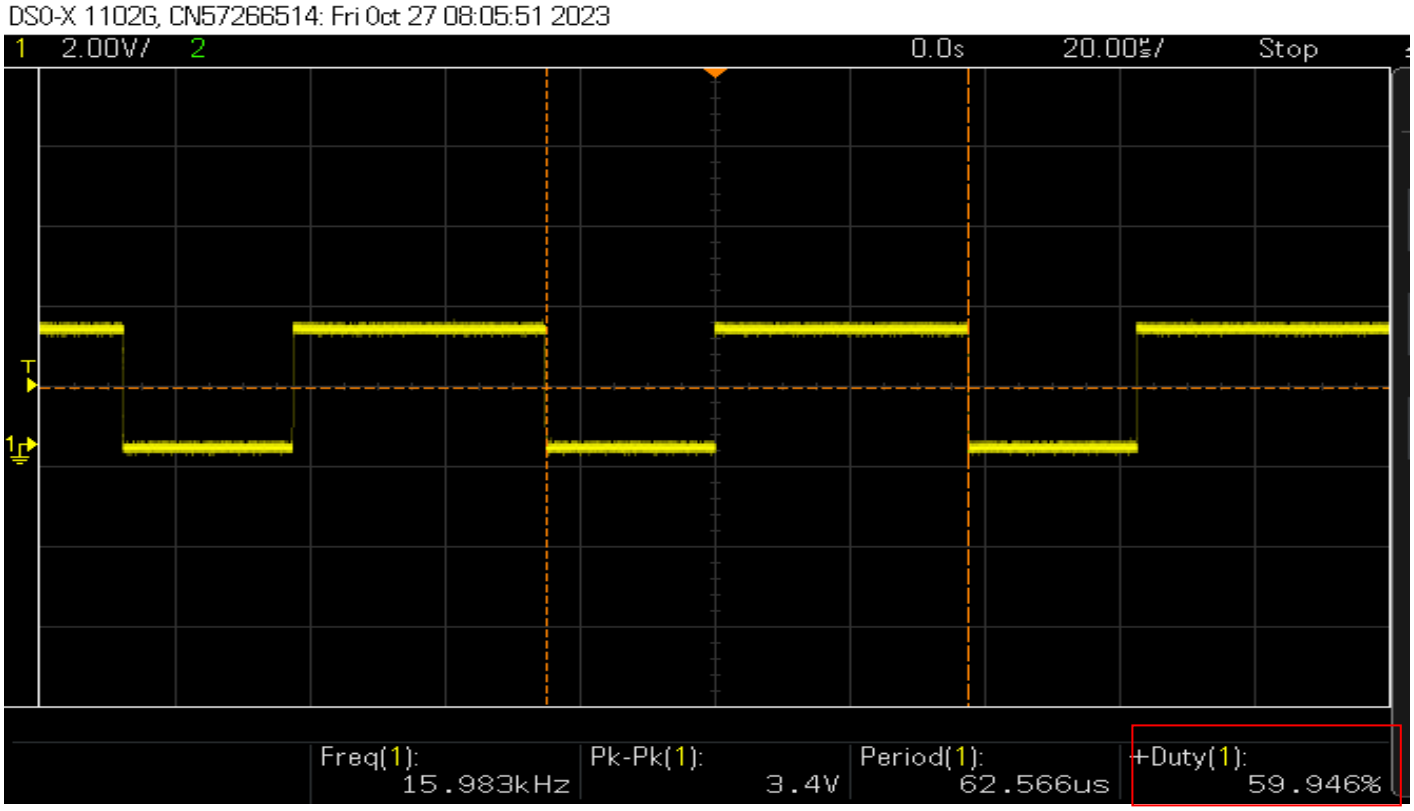


PART3-STM32F411E:

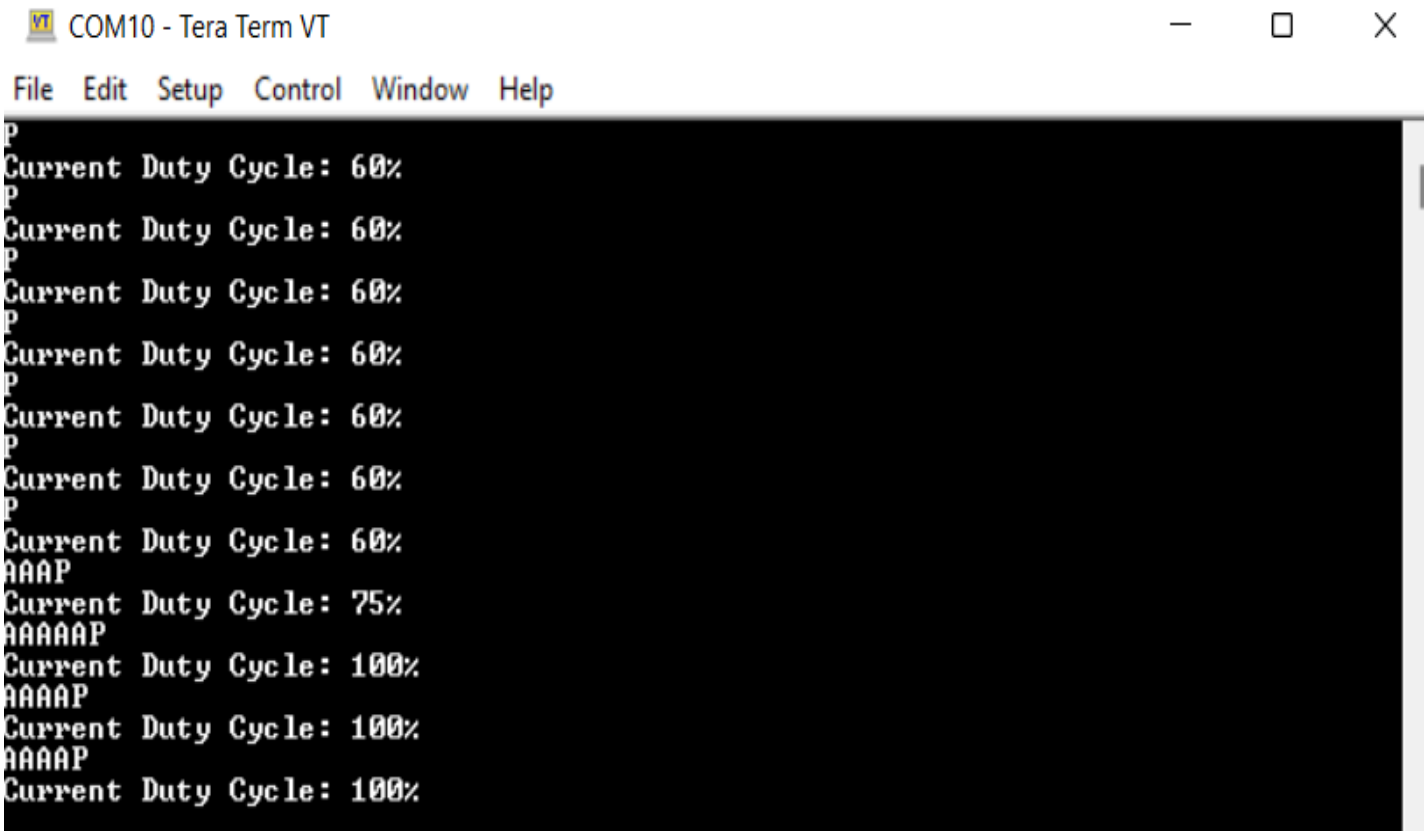
- 'P' character prints the present duty cycle, which is 60%.



- Duty cycle at PORTD12 on board green LED, which has a duty cycle of 60% at the beginning.

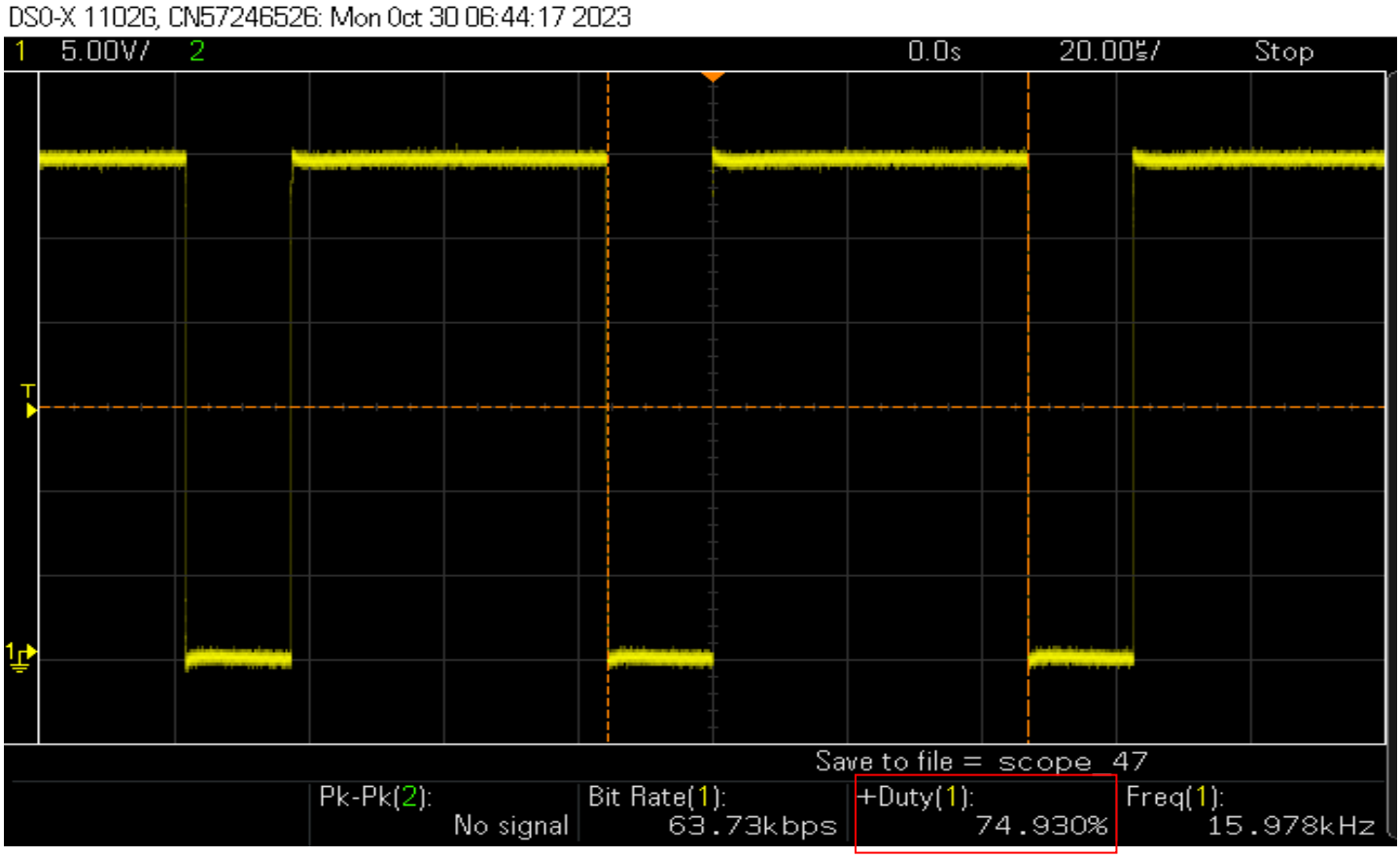


- 'A' Character which increase 5% of duty cycle for every 'A' character into buffer.

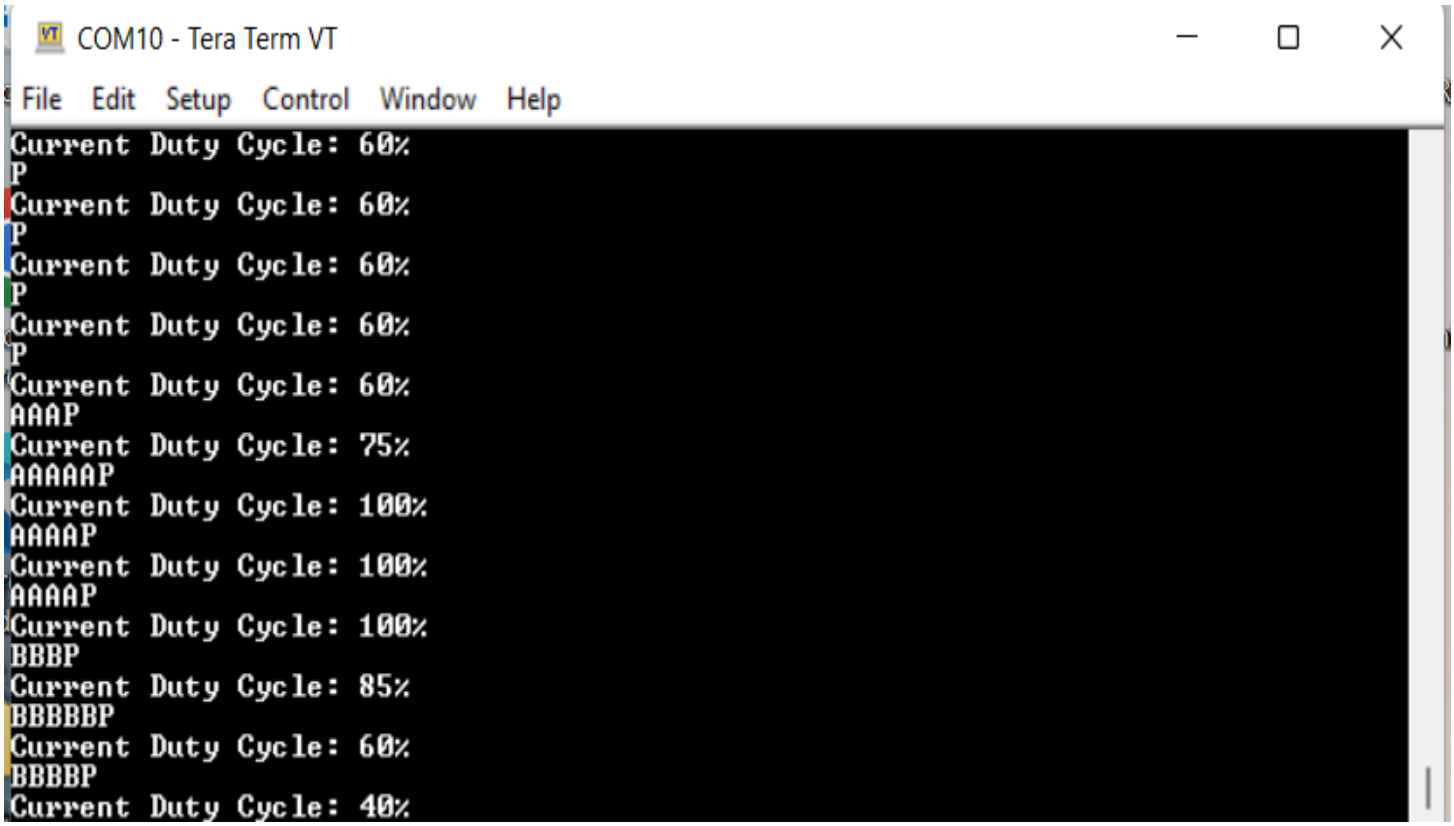


- For every A character input there will be 5% of increment in duty cycle till 100%, there after it will remain at 100%.

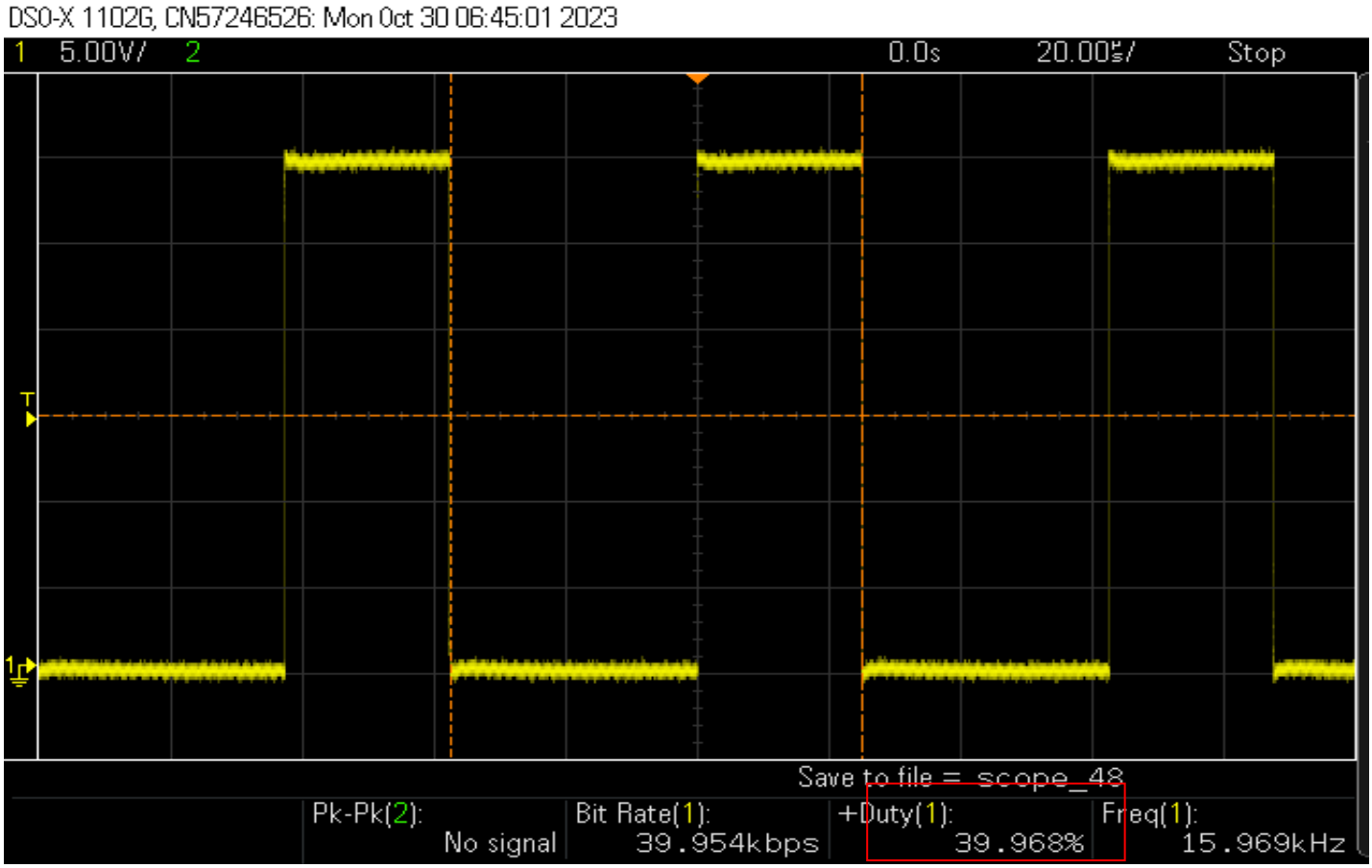
- Oscilloscope output at 75% duty cycle.



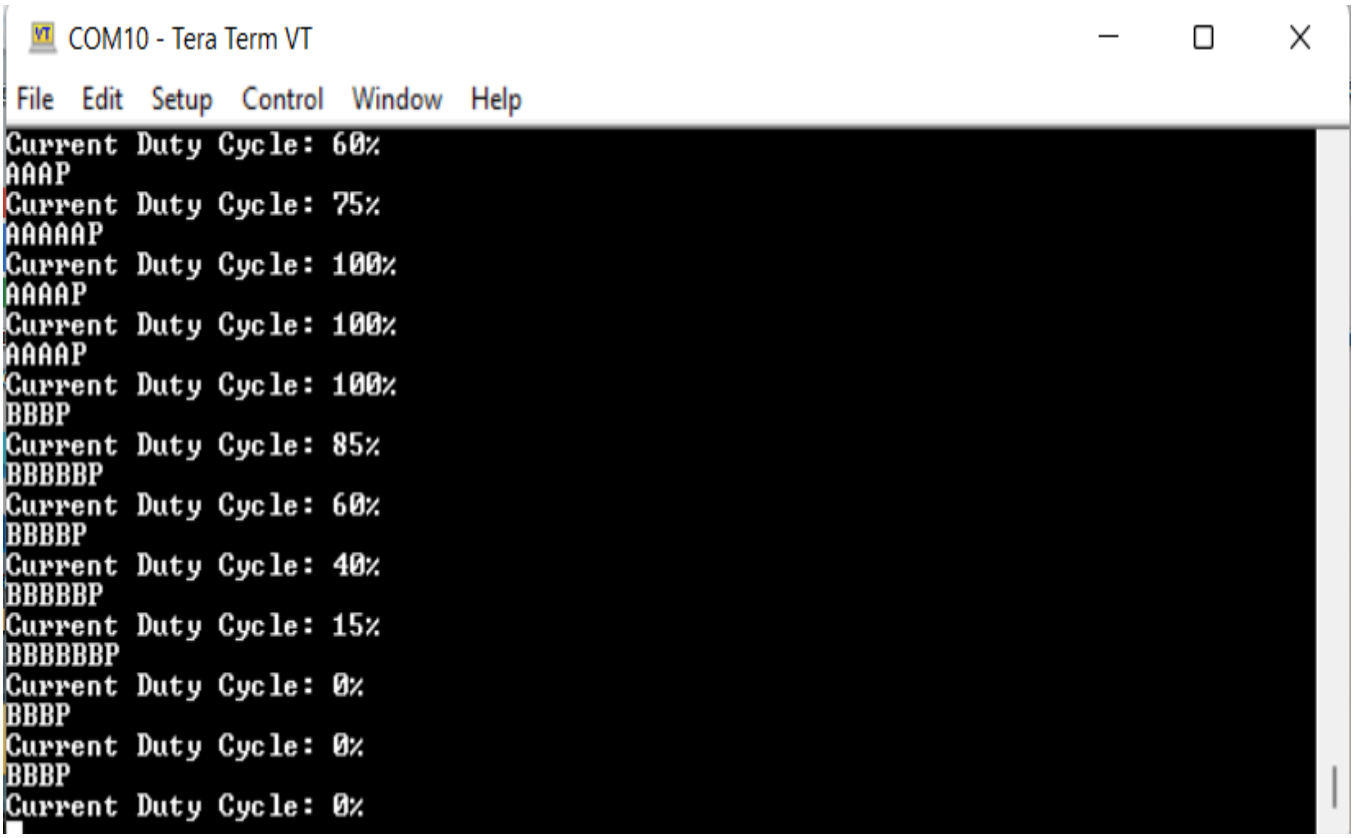
- 'B' Character which decrease 5% of duty cycle for every 'B' character entry



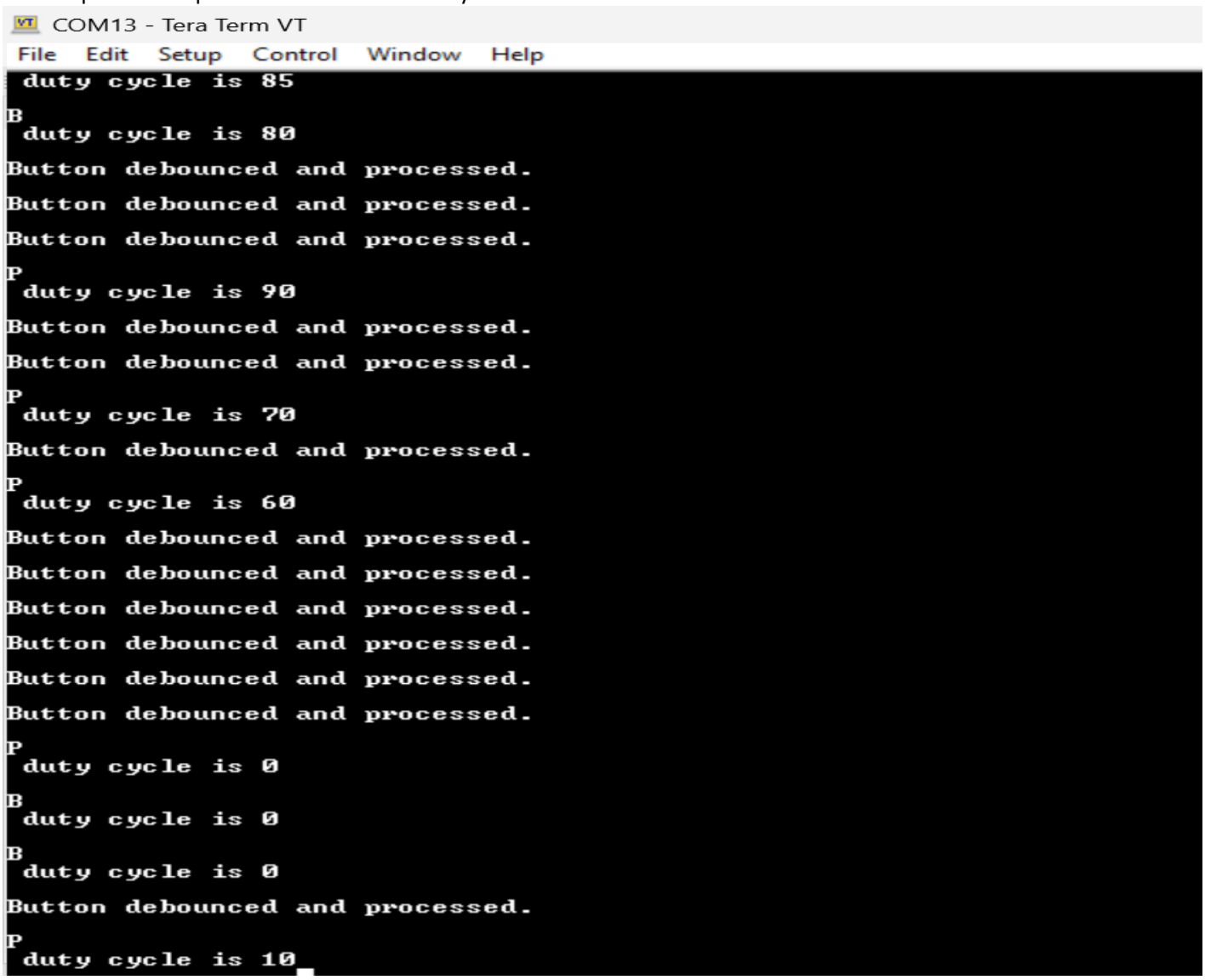
- Oscilloscope output @ 40% duty cycle for ‘B’ character entries



- For every ‘B’ character there will be a 5% of decrement in duty cycle till 0% and it stays there only for further ‘B’ characters as well.



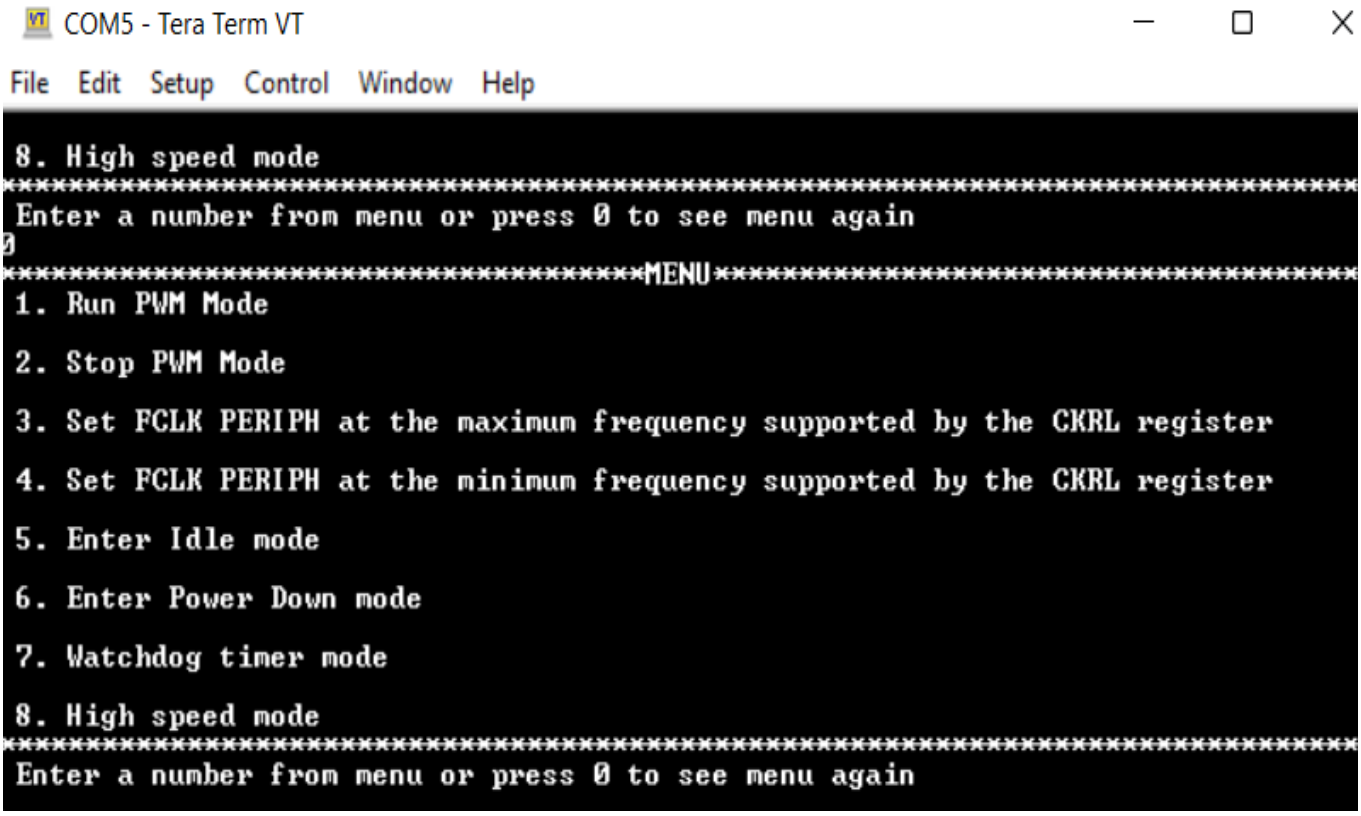
- For user button press there will be a 10% of increment in duty cycle till 100% and then decrement for further press in steps of 10% till 0% and it stays there.



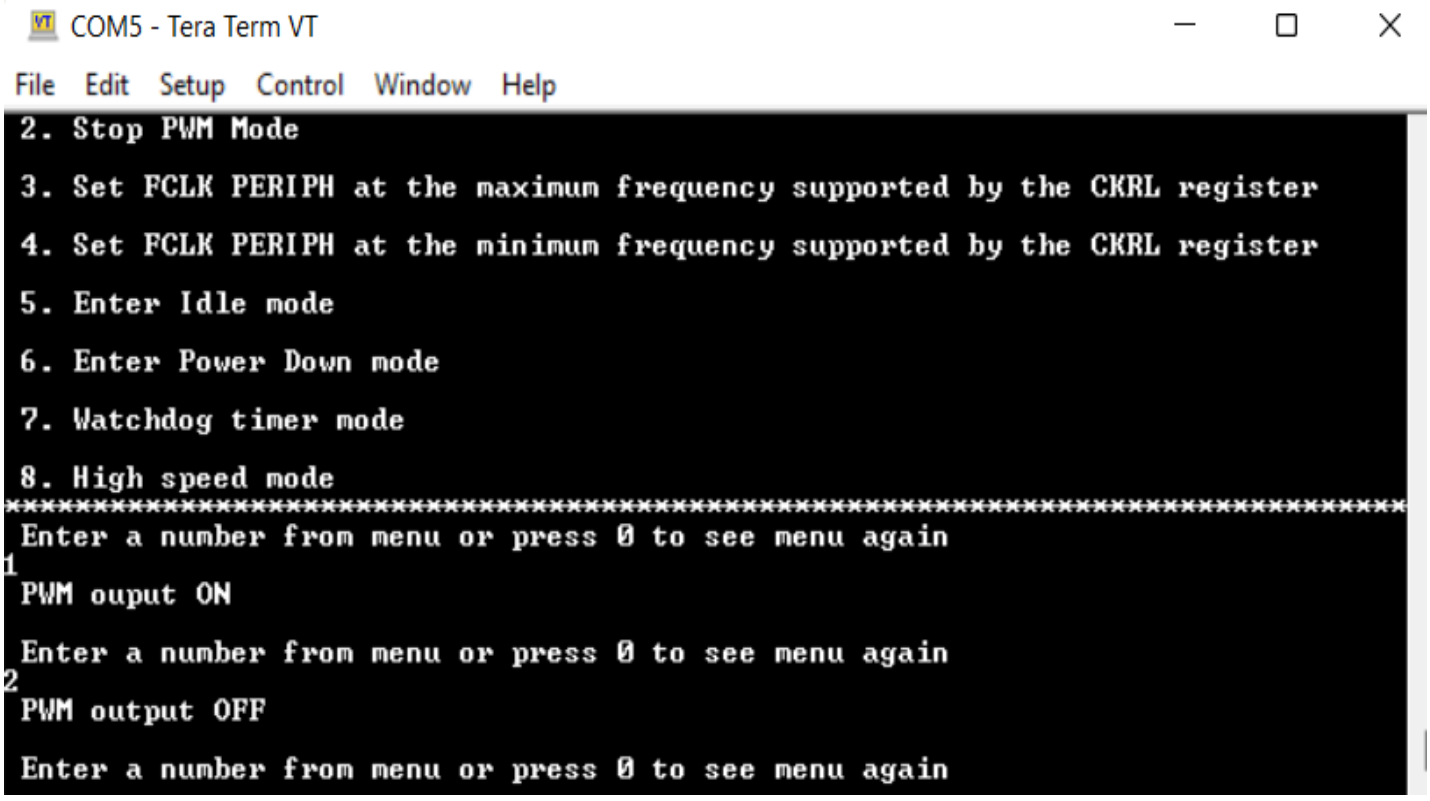
- Here the user button debouncing is avoided by using a timer interrupt which creates a delay of 300ms where it stays in interrupt only without considering further more presses and does not modify the duty cycle.

Supplemental Part 3: PCA MODES OF 8051

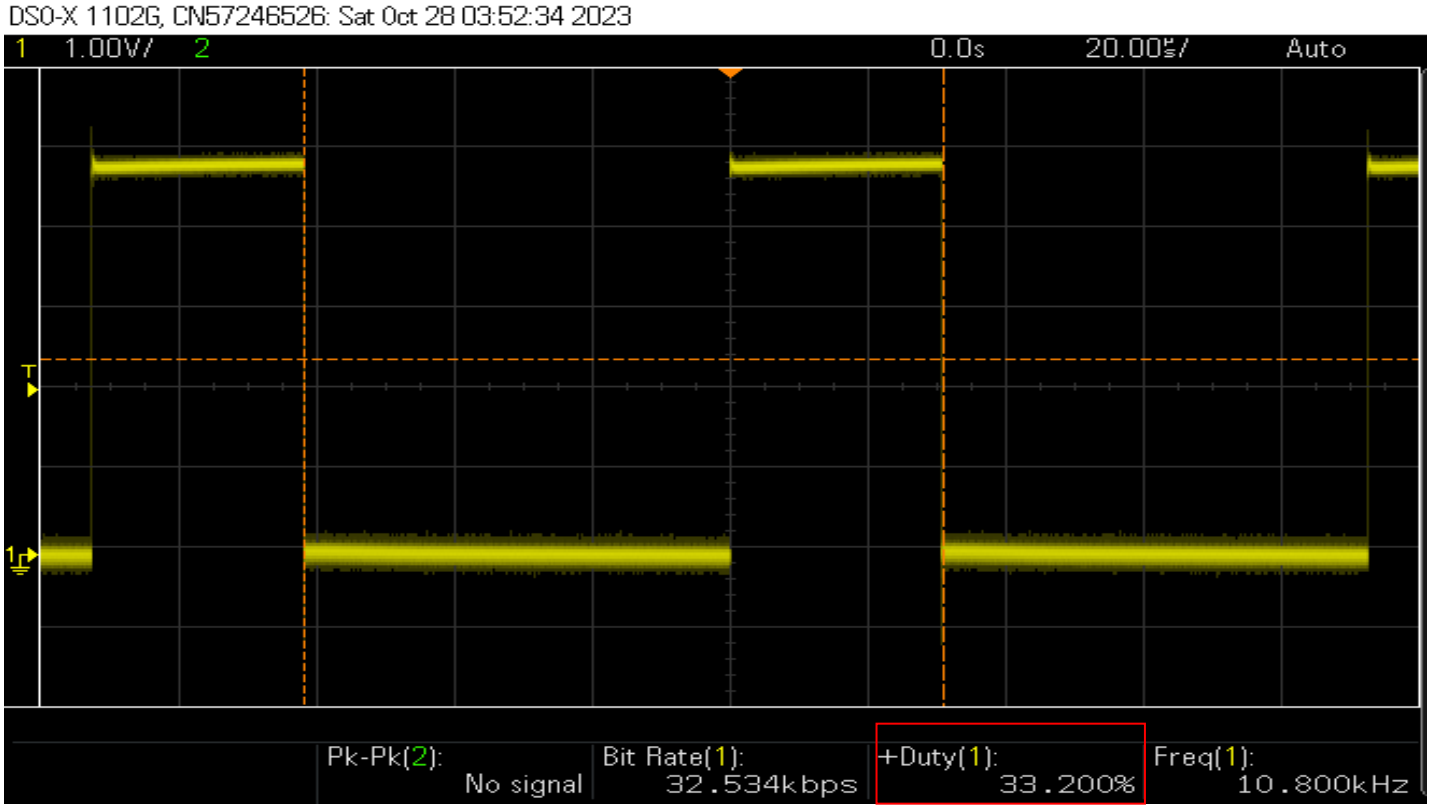
- Menu for choosing PCA modes where it displays the number of available modes that you have implemented for the user to choose. And by giving 0 character as input, It will display the menu again. Here the attached output of terminal emulator.



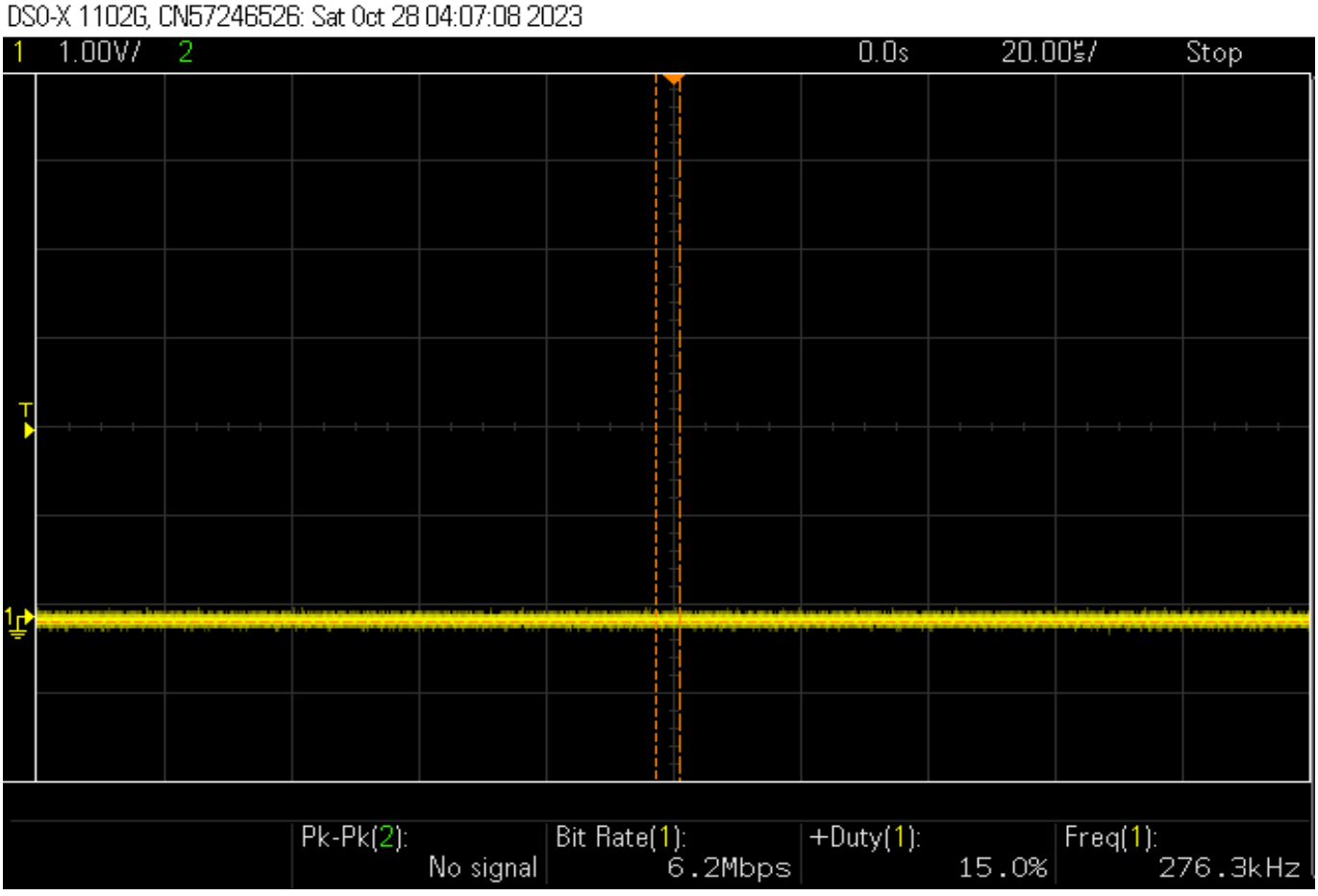
- 1 gives the PWM mode on, which is implemented in module 1 where we have to check for p1.4 pin of 8051 for pwm with a duty cycle of 33%. Here the attached terminal output for it.



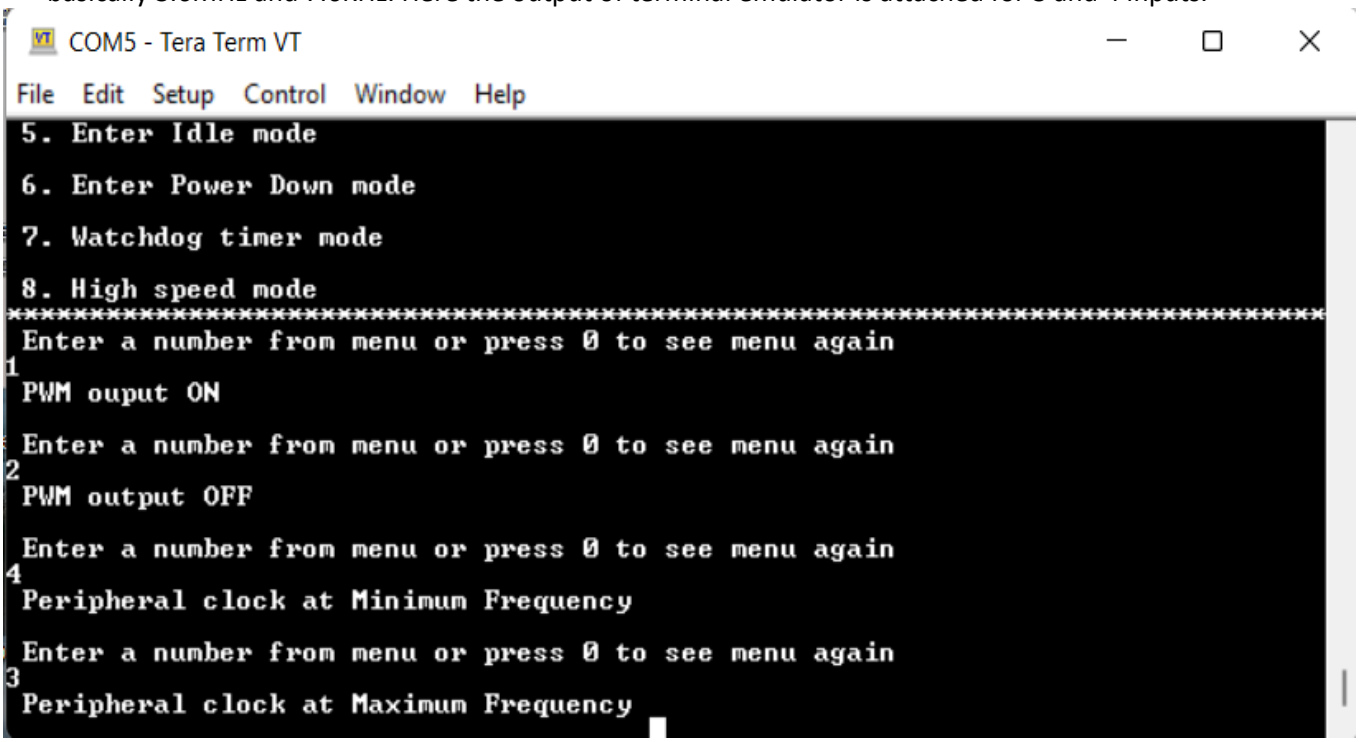
- The oscilloscope output of p1.4 pin of 8051 with a duty of 33% is attached in PWM mode On.



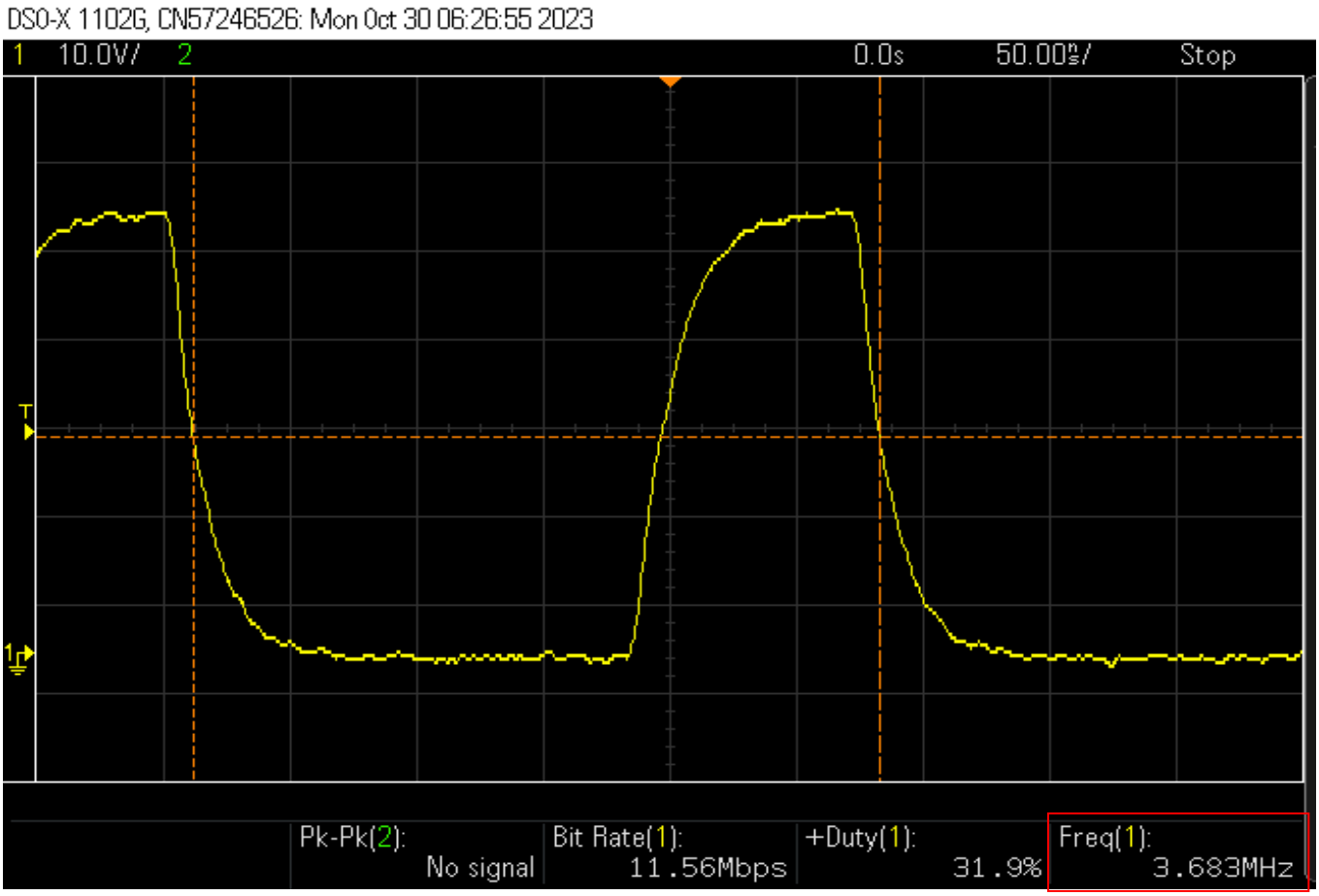
- The oscilloscope output of p1.4 pin of 8051 with a duty of 100% is attached in PWM mode OFF. This gives when you give 2 as input in terminal



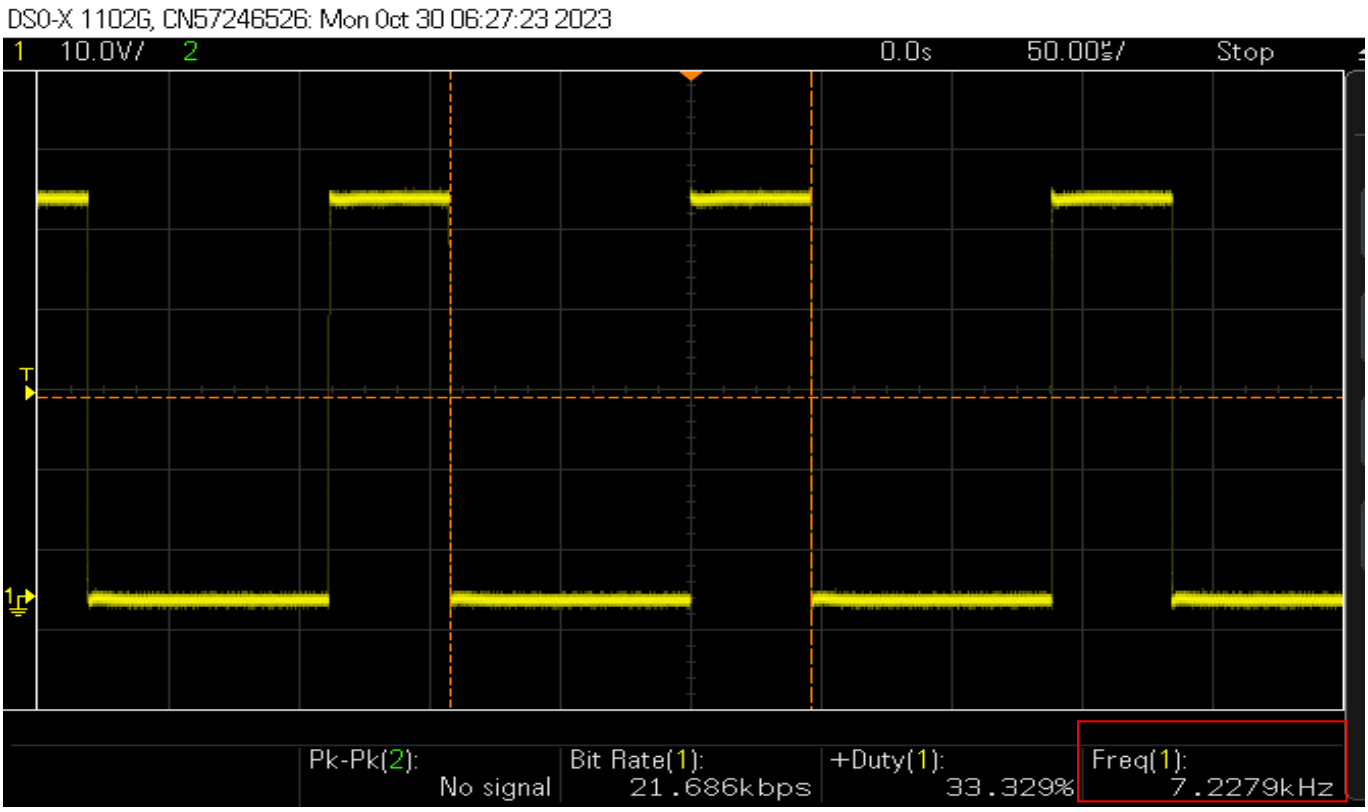
- Input 3 and 4 to the terminal emulator gives runs the ALE at maximum and minimum frequency which is basically 3.6MHz and 7.6KHz. Here the output of terminal emulator is attached for 3 and 4 inputs.



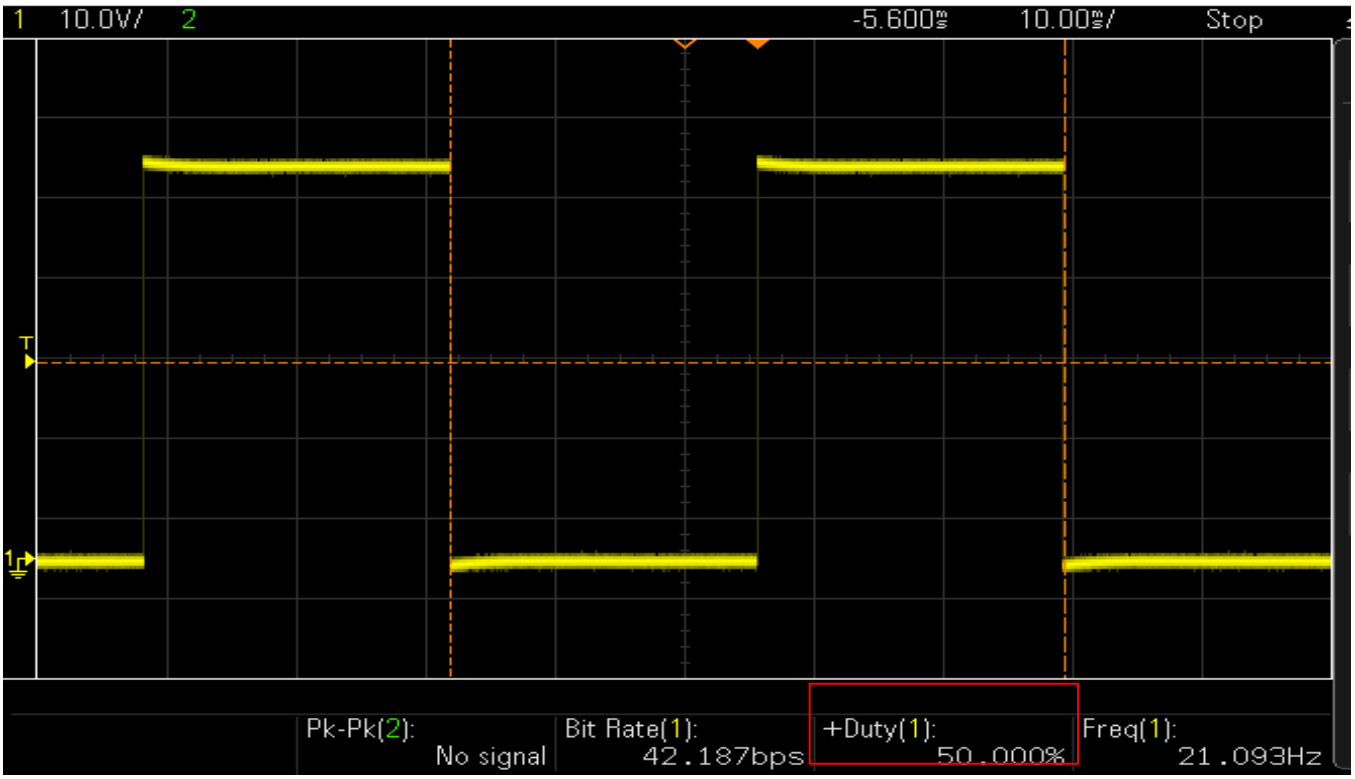
- The oscilloscope output for this 3 and 4 inputs are attached here. The maximum possible frequency seen at ALE is shown below.



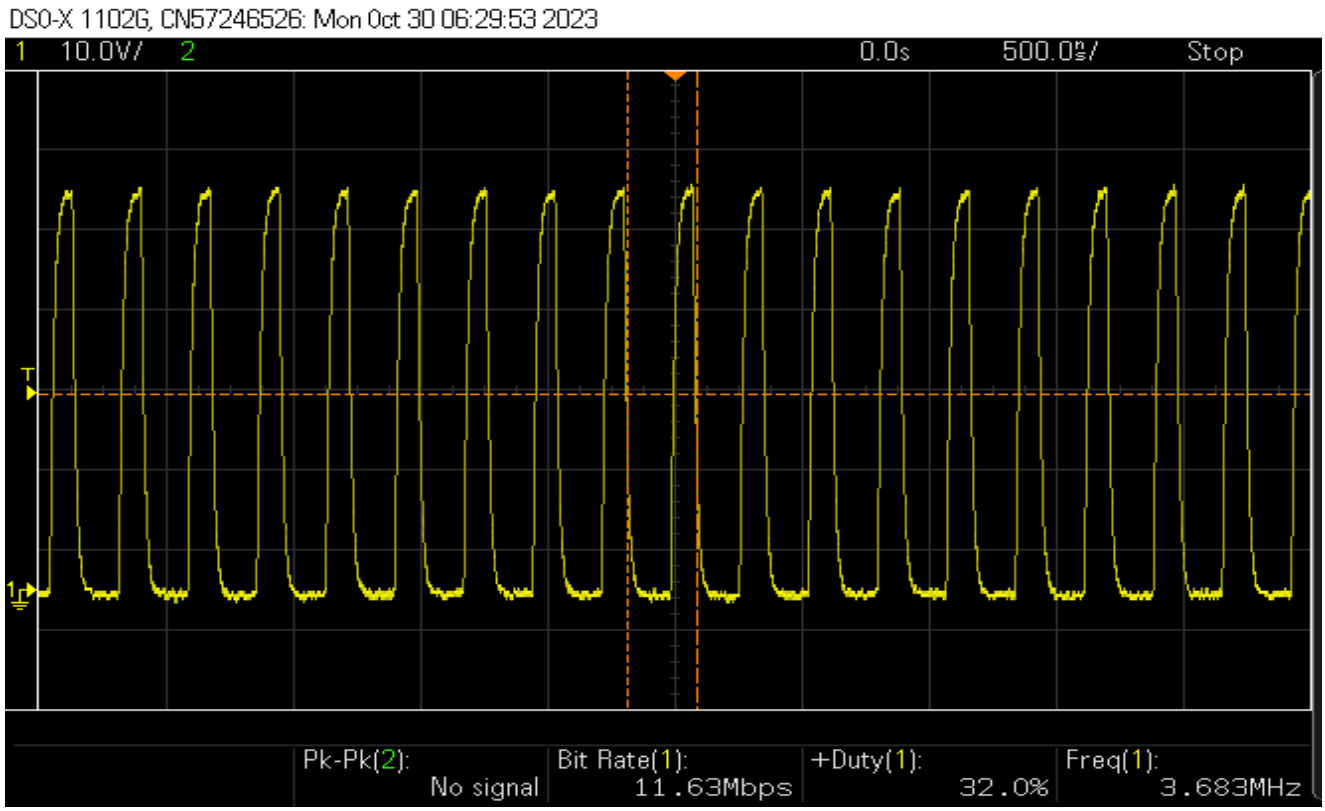
- The minimum possible frequency seen at ALE is shown below.



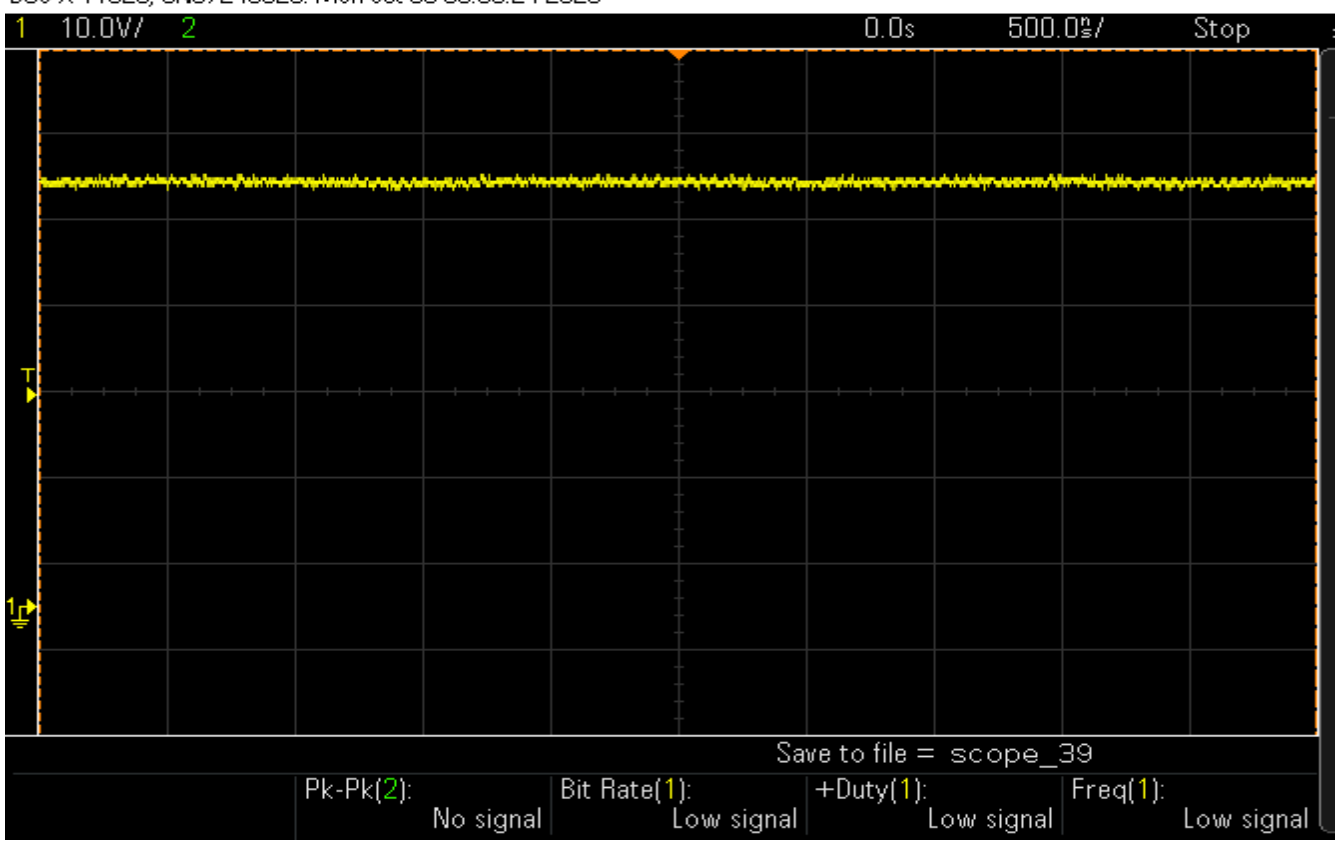
- The output of High speed is give below which is module 0 i.e p1.3 of 8051 pin with a duty cycle of 50% at that pin



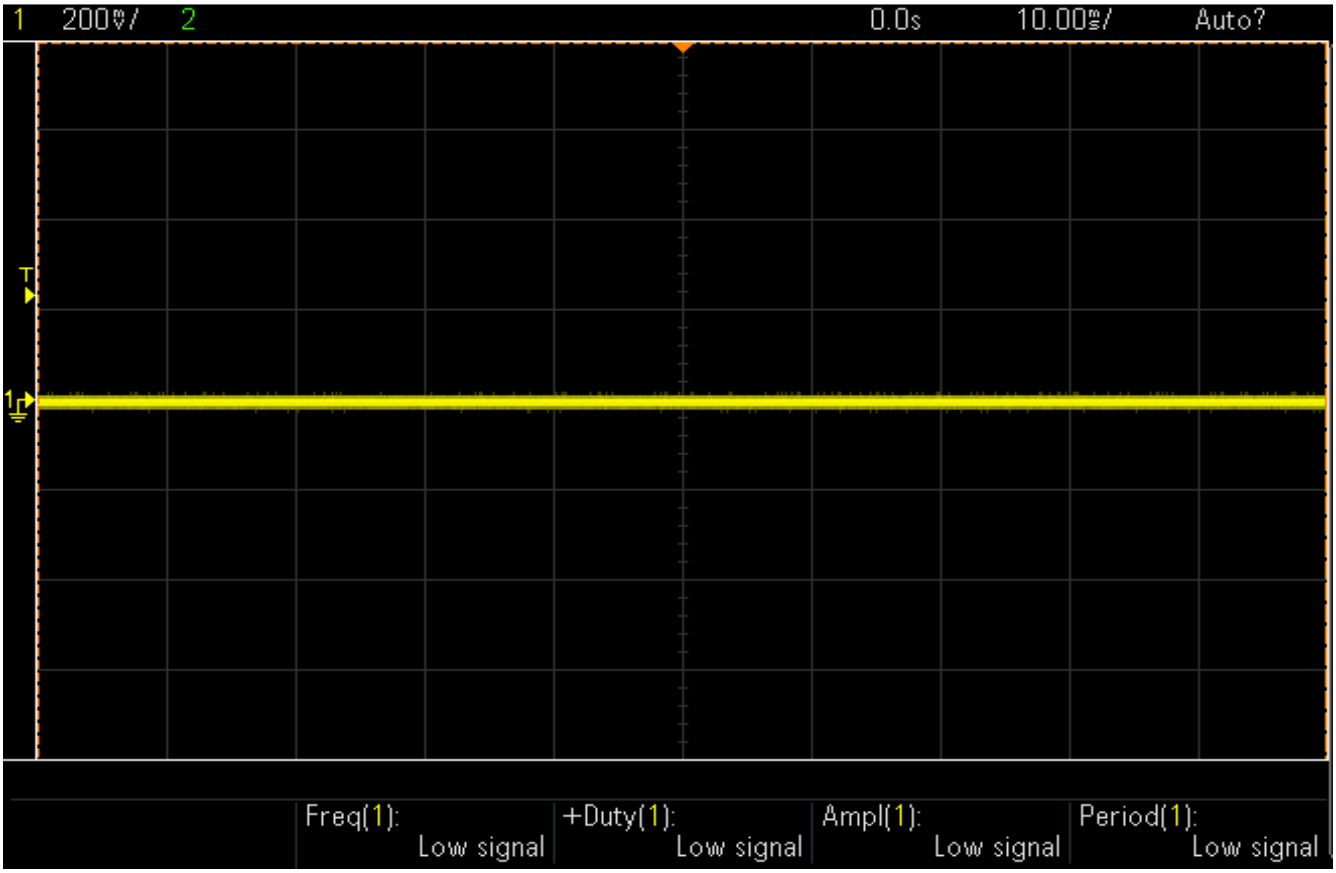
- IDLE mode output is given with oscilloscope which stops the toggling of ALE , to come out of IDLE mode we should raise INT0 which is configured in code. Here the below is before entering IDLE mode.



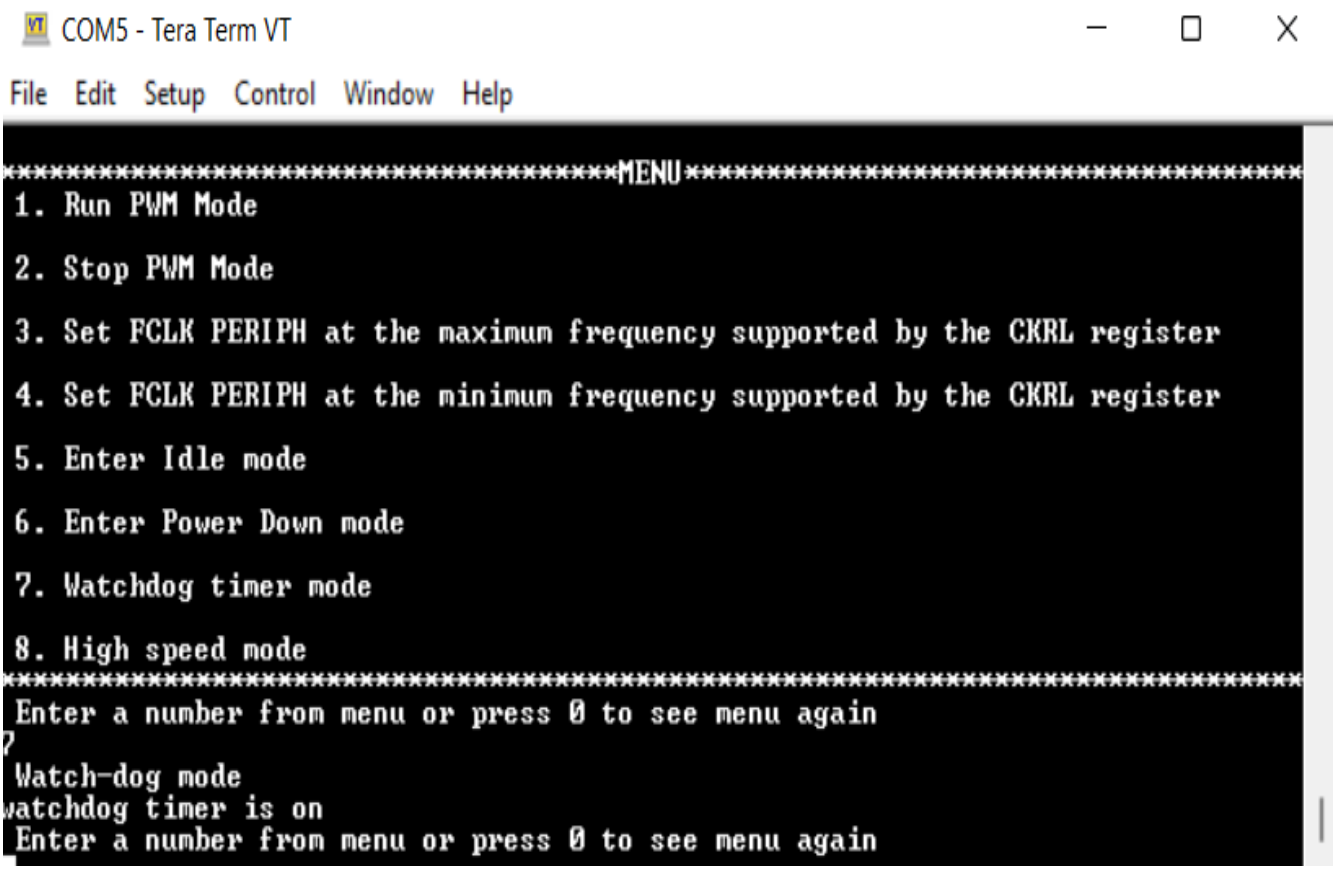
- This is when it entered IDLE mode which stopped toggling the ALE. And then It comes out after touching the INT0 pin of 8051



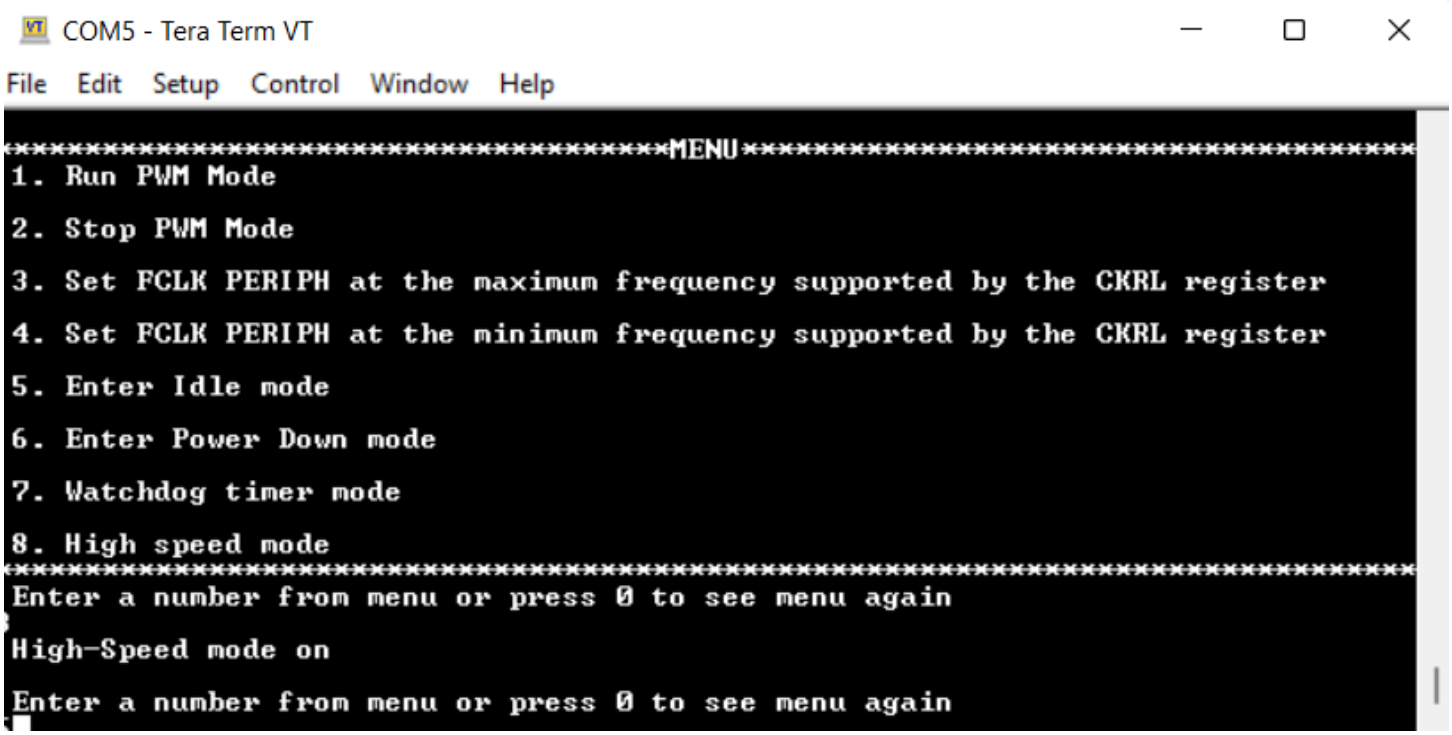
- Output of oscilloscope at power down mode. ALE low in power down mode



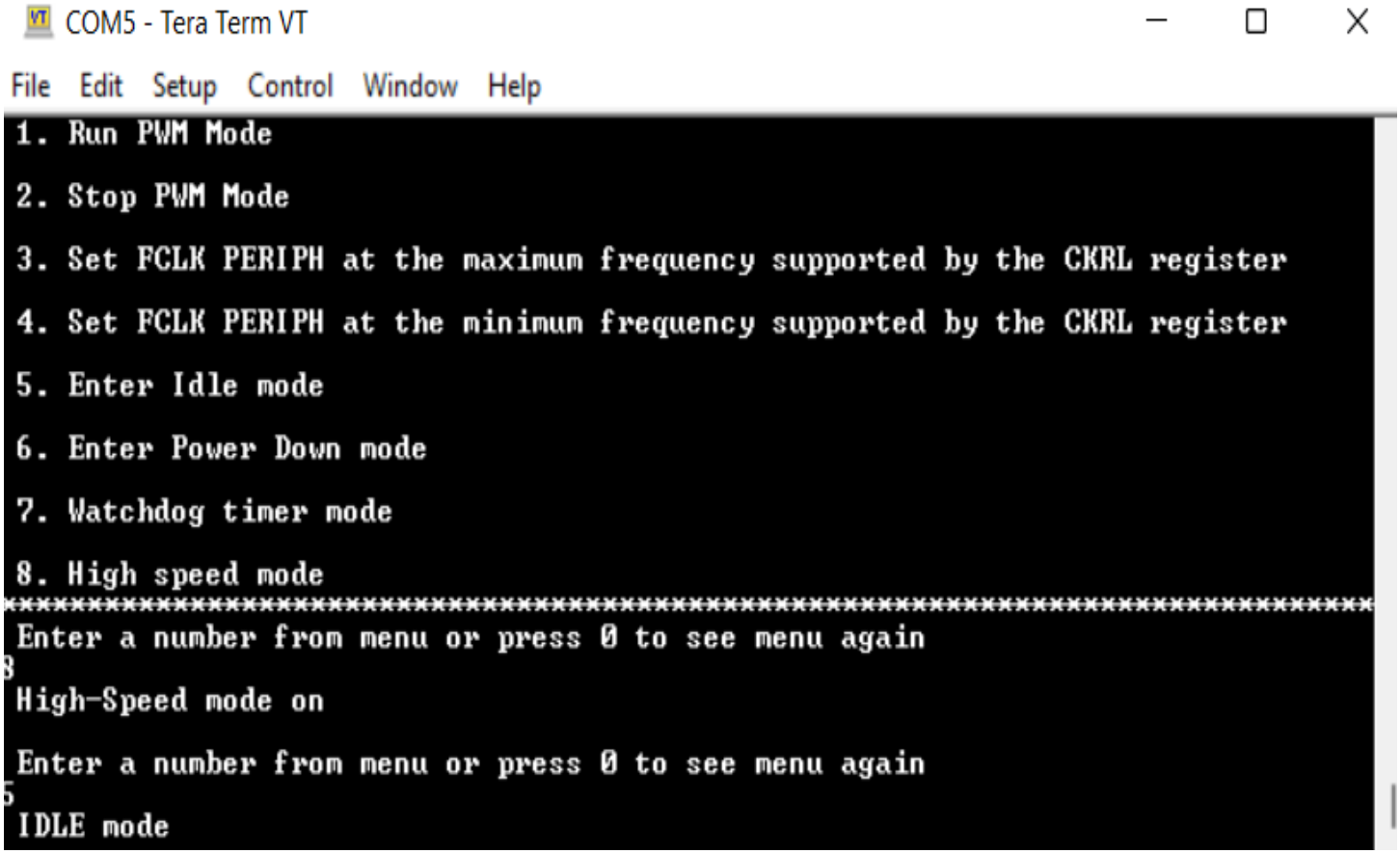
- Output of oscilloscope in watch dog mode. Which is checked by a flag whether watch dog is ON or OFF.



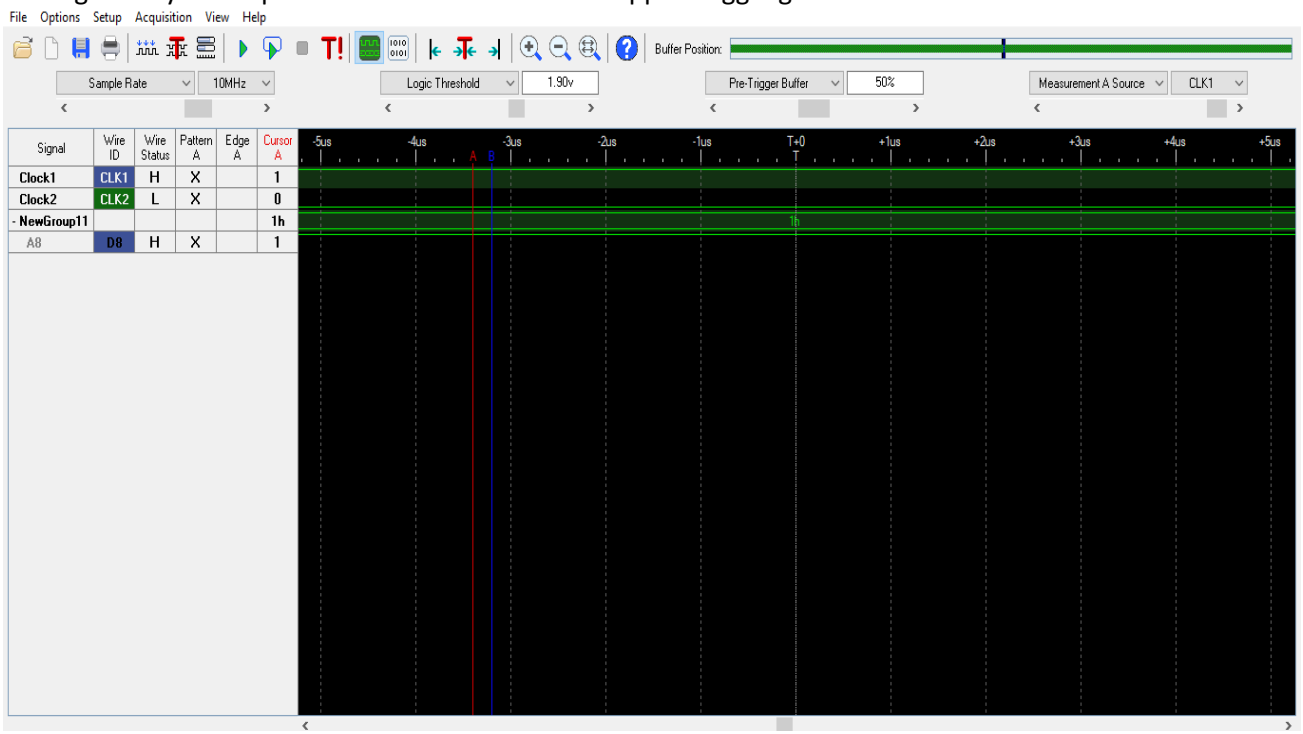
- Outputs of all inputs for all modes in terminal emulator. This is for High speed mode



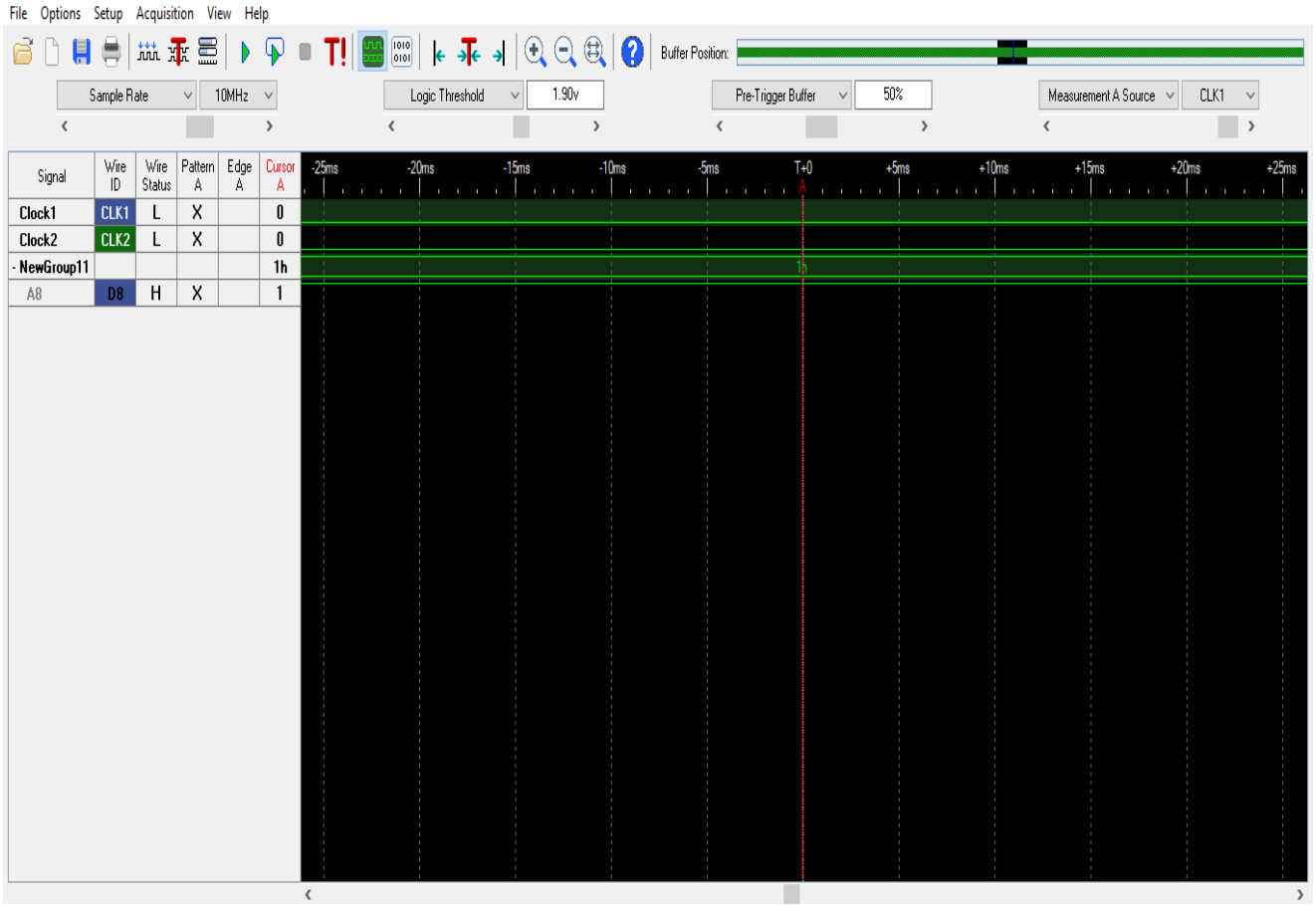
- This is for IDLE mode



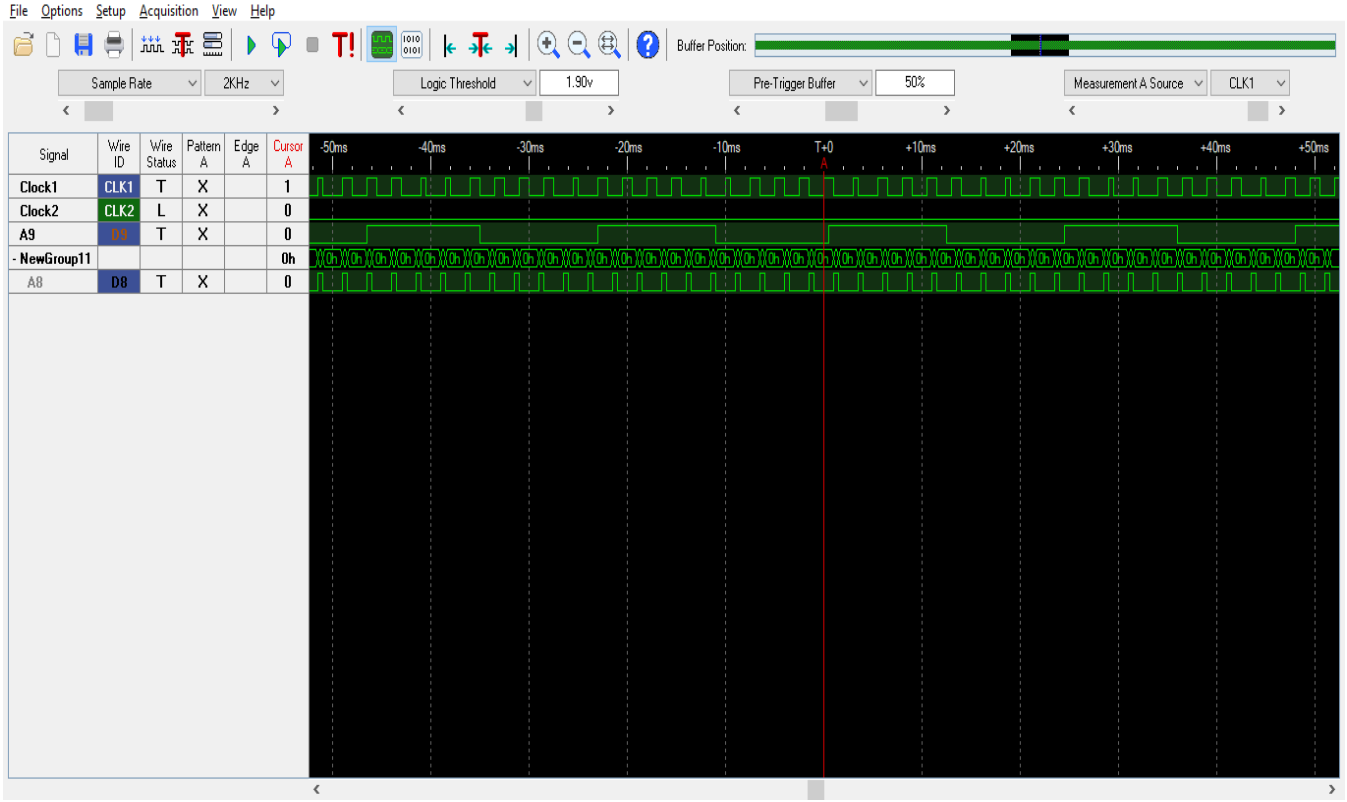
- Logic analyzer outputs for IDLE mode. Which stopped toggling of ALE in this mode.



- Logic analyzer output of power down mode.



- Logic analyzer output of highspeed mode.



- High-Speed Output Mode: In this mode, the PCA module can generate high-speed pulse signals with varying duty cycles, making it useful for tasks such as generating precise timing signals for applications like motor control or frequency synthesis.

- Pulse Width Modulator (PWM) Mode: The PCA module can function as a PWM generator, allowing it to produce a square wave output with a programmable duty cycle. PWM is commonly used for tasks like controlling the speed of motors and regulating the brightness of LEDs.

- Watchdog Timer Mode (Module 4): Module 4 of the PCA can be configured as a watchdog timer. This timer is designed to monitor the operation of the microcontroller and reset it if it fails to provide periodic "petting" signals. It enhances system reliability by preventing lockups or software glitches from causing system failures.

- All these modes are obversed in oscilloscopes and logic analyzers.

OPTIONAL CHALLENGE PART3: Buffer with heap size of 5600

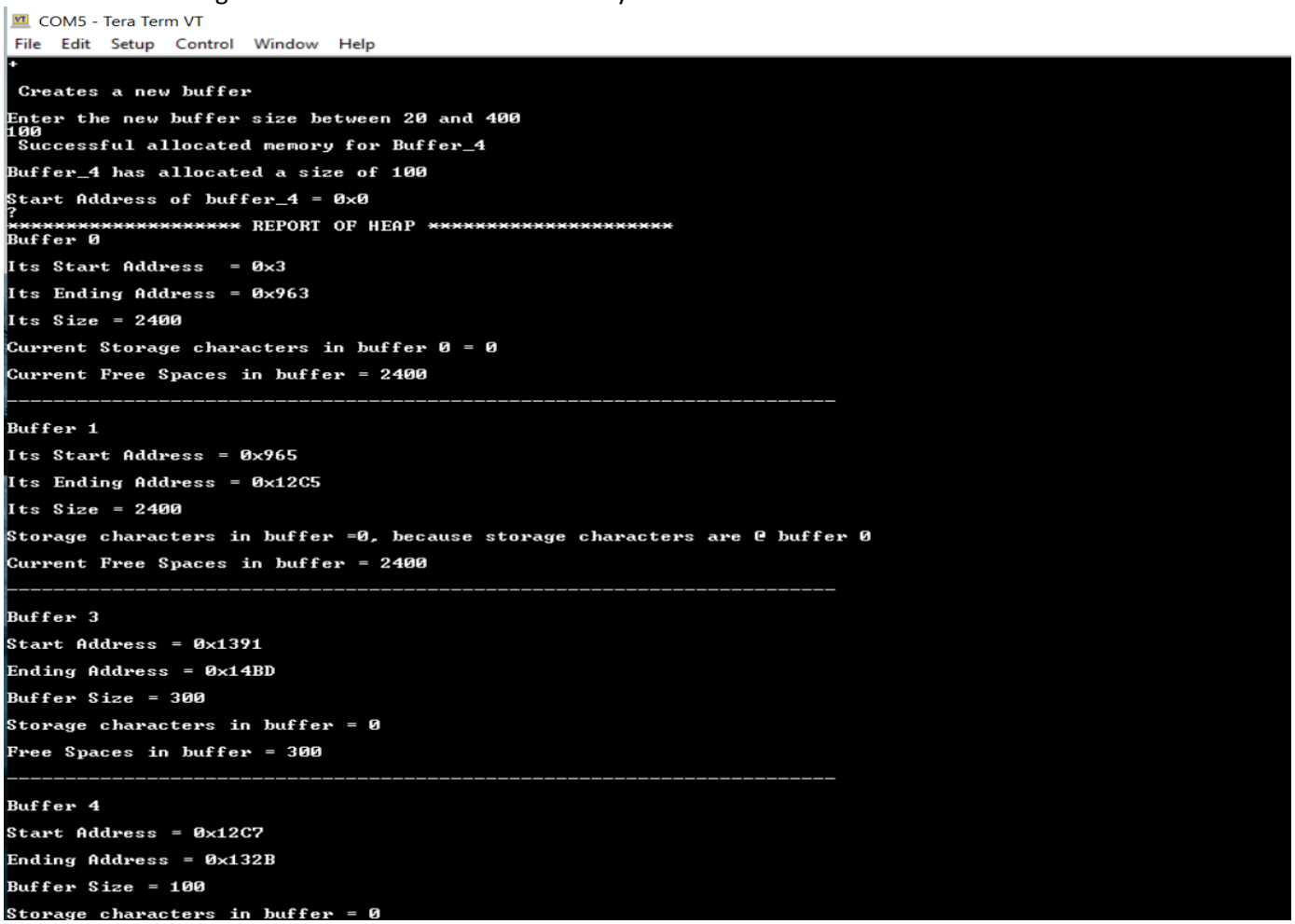
- Creation of buffer 0 and buffer 1 with 2400 bytes. And buffer 2 and buffer 3 with 200 and 300 bytes using + character.

```
PAULMON2 Loc:7FFF > Jump to memory location

Jump to memory location (7FFF), or ESC to quit: 2000

running program:


********* HELLO :) ********

Enter a number between 32 & 4800 for buffer size
2400
 The input number is:2400
Start Address of buffer0 = 0x3
Start Address of buffer1 = 0x965
Buffer_0 and Buffer_1 successfully created

 OPTIONS TO CHOOSE
You can enter characters to be stored in buffer 0
Enter + to create a new buffer of size between 20 and 400 bytes
Enter - to delete a buffer.
Enter ? to generate a heap report.
Enter = to display current contents of buffer 0.
Enter @ to free all buffers and start program again.

**************************THANK YOU**************************
+

 Creates a new buffer

Enter the new buffer size between 20 and 400
200
 Successful allocated memory for Buffer_2

Buffer_2 has allocated a size of 200

Start Address of buffer_2 = 0x0
+

 Creates a new buffer

Enter the new buffer size between 20 and 400
300
 Successful allocated memory for Buffer_3

Buffer_3 has allocated a size of 300

Start Address of buffer_3 = 0x0
```

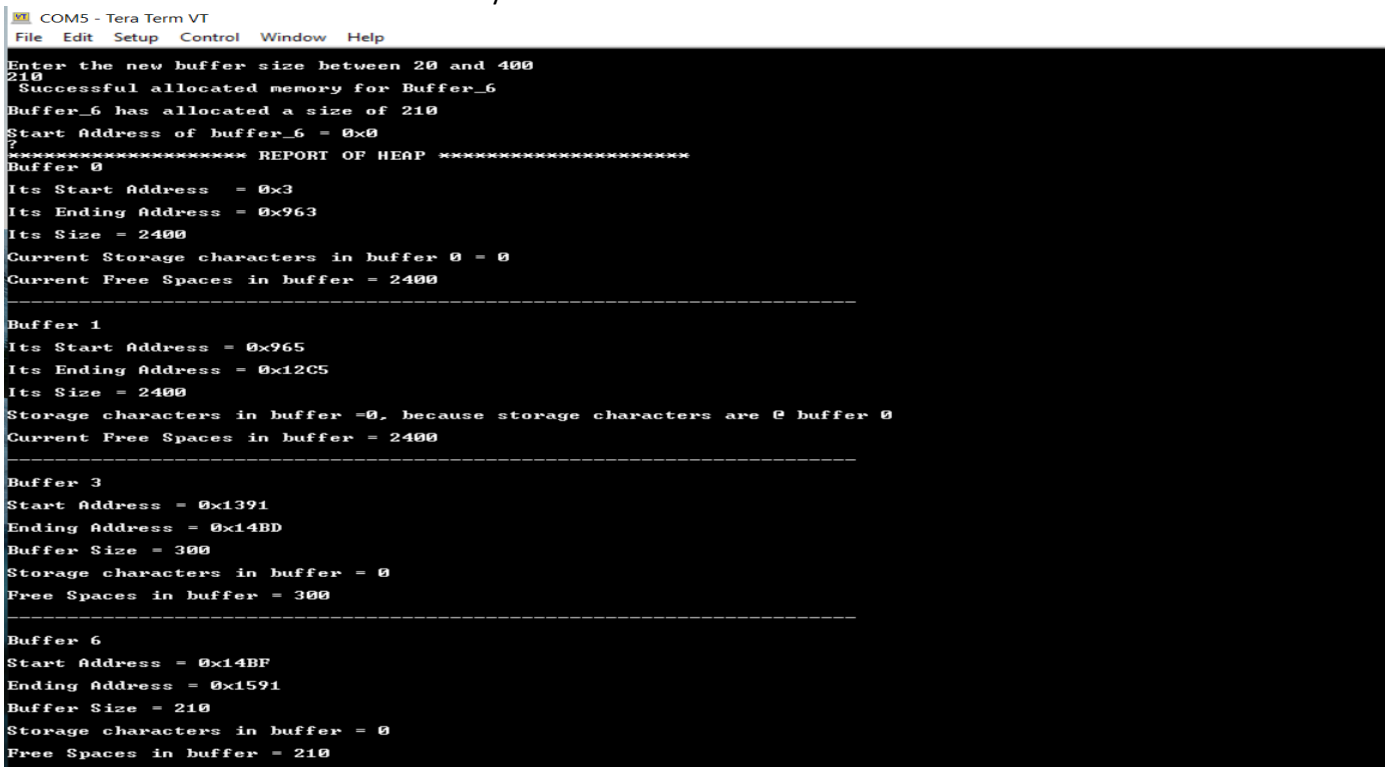
- Deletion of buffer 2 with – character and its heap report.

```
COM5 - Tera Term VT
File  Edit  Setup  Control  Window  Help
300
 Successful allocated memory for Buffer_3
Buffer_3 has allocated a size of 300
Start Address of buffer_3 = 0x0
-
Enter a valid buffer number
2
Deleting buffer 2
Buffer 2 is Free
?
********************* REPORT OF HEAP **********************
Buffer 0
Its Start Address   = 0x3
Its Ending Address  = 0x963
Its Size = 2400
Current Storage characters in buffer 0 = 0
Current Free Spaces in buffer = 2400
------------------------------------------------------------------------

Buffer 1
Its Start Address  = 0x965
Its Ending Address  = 0x12C5
Its Size = 2400
Storage characters in buffer =0, because storage characters are @ buffer 0
Current Free Spaces in buffer = 2400
------------------------------------------------------------------------

Buffer 3
Start Address = 0x1391
Ending Address = 0x14BD
Buffer Size = 300
Storage characters in buffer = 0
Free Spaces in buffer = 300
------------------------------------------------------------------------

Number of storage characters = 0
Total number of characters received = 4
Total number of buffers that were allocated since the start of the program = 4
Total storage characters stored since last '?' = 0
```

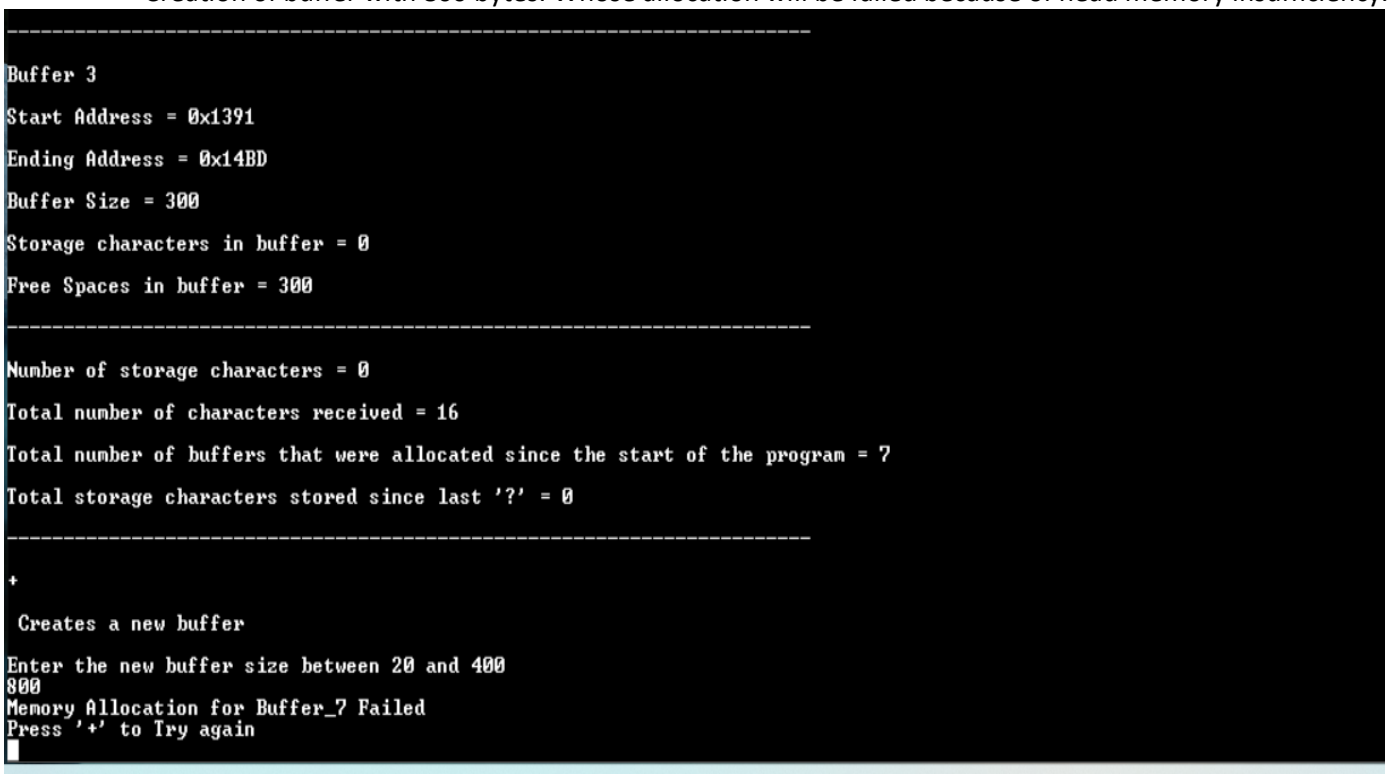
- Creating one more buffer with size of 100 bytes.



- Creation of buffer with 210 bytes of size.



- Creation of buffer with 800 bytes. Whose allocation will be failed because of head memory insufficiency.



The above are required heap buffer with size of 5600 outputs.

a) What operating system (including revision) did you use for your code development?
Windows 11
b) What compiler (including revision) did you use?
SDCC 4.1.0
c) What exactly (include name/revision if appropriate) did you use to build your code (what IDE, make/makefile, or command line)?
Code blocks(SDCC) for AT89C51 along with STM32 IDE
d) Did you install and use any other software tools to complete your lab assignment?
NO
e) Did you experience any problems with any of the software tools? If so, describe the problems
No

LAB 3 Learnings:

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