ECEN5613 ESD

03/25/2012

Lab#3 Report

RS232/ATMEL Flip/SDCC/

Serial Communication using RS-232

Hardware

While doing connections to the hardware I was very careful not to connect t the RS-232 connections to the TTL because the RS-232 operates at a higher voltage and a connecting it to TTL inputs would cause burning out the circuit.

The tip provided by professor during lecture and also in the Lab#3 pdf document informing that making the connections for RX and TX lines for RS-232 should be done by checking the pins 2and 3 on the D-9 connector for the signal using the Terminal emulator was very helpful.

Paulmon2

This is one of the best program which gave me ideas as to what can be done with ASCII characters and creating my own menus and User interface for the labs.

It is also a great tool for debugging.

Editing the .asm file.

The paulmon21.asm file has been matches with the memory map as given in the Handout. i.e.

```
.equ    pgm,  0x2000      ;default location for the user program
.equ    bmem, 0x1000      ;where is the beginning of memory
.equ    emem, 0x7FFF      ;end of the memory
```

The only editing to be done is that every time Paulmon2 gets launched the XRS0 and XRS1 bits have to be set to 1:1

Below is the patch edited in the paulmon21.asm for the power on patch of code

```
poweron:
    orl  8Eh,#00Ch ; Set XRS1:XRS0 bits to 11b
```

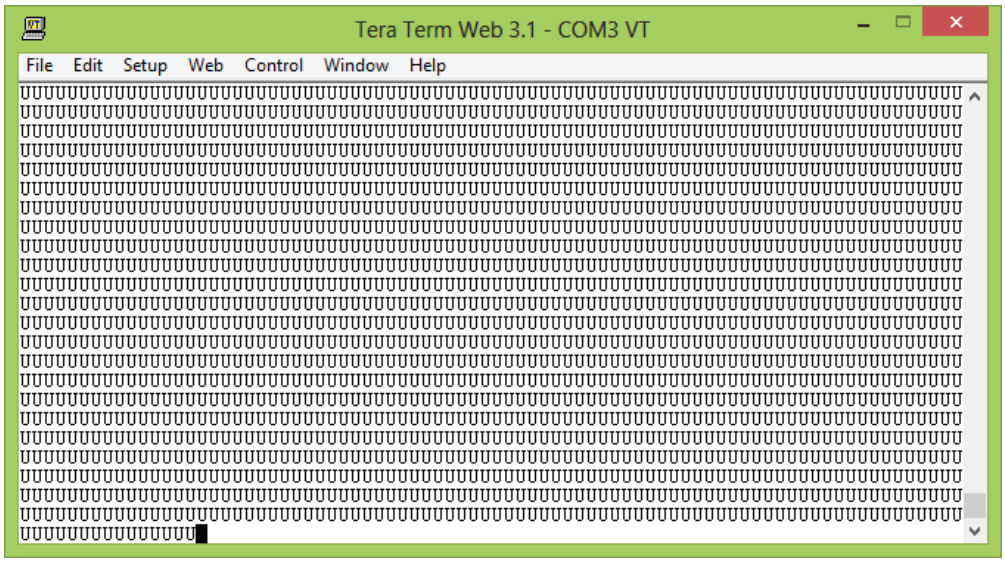
3	XRS1	**XRAM Size** XRS1 XRS0 XRAM size 0 0 256 Bytes (default) 0 1 512 Bytes 1 0 768 Bytes 1 1 1024 Bytes
2	XRS0	

Testing

After confirming all the hardware connetions I tested the RS-232 circuit using Tera Term Pro Serial Emulator, the compiled 5Paulmon21.asm below code which transmits the character 'U' in a loop.

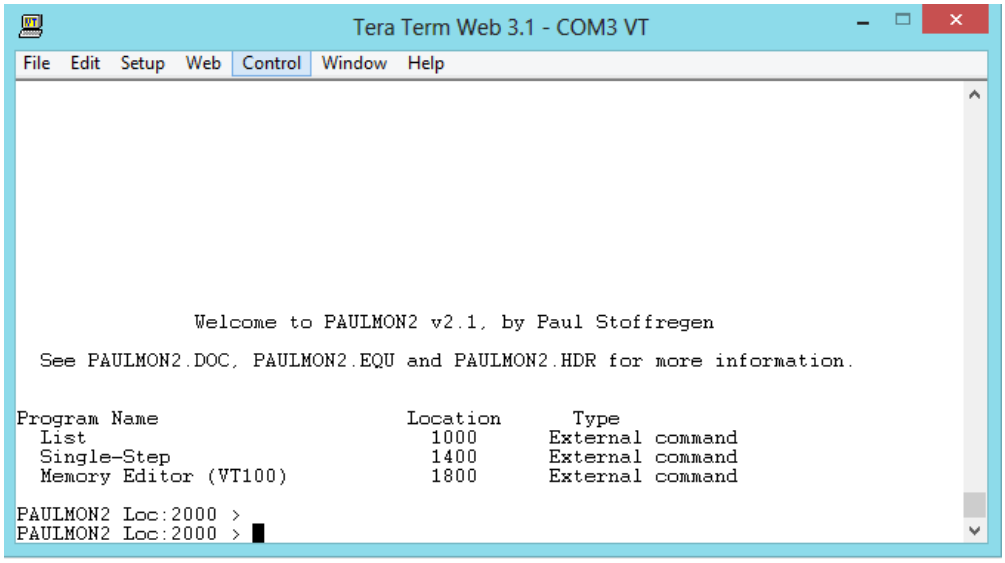
```
        ORG     $0000           ;Stating Location is set as 0x0000
        LJMP    MAIN            ; Jump to MAIN

        ORG     $0100           ; Jump to main Program
MAIN:   MOV     TMOD,#020H      ; Initialize timer in mode 2
        MOV     TH1,#0FDH       ; Load value 0xFDh for 9600 baud rate
        SETB    SCON.6          ; Set Serial port in mode 1
        SETB    TCON.6          ; Start The Timer
LOOP:   MOV     SBUF,#'U'       ; Load the Serial Buffer with the character 'U'
        JNB     SCON.1,$        ; Check the RI flag till set high
        CLR     SCON.1          ; Clear the RI flag
        AJMP    LOOP            ; Loop back to keep sending character 'U'
```

Output of Tera Term Pro



Paulmon Functions

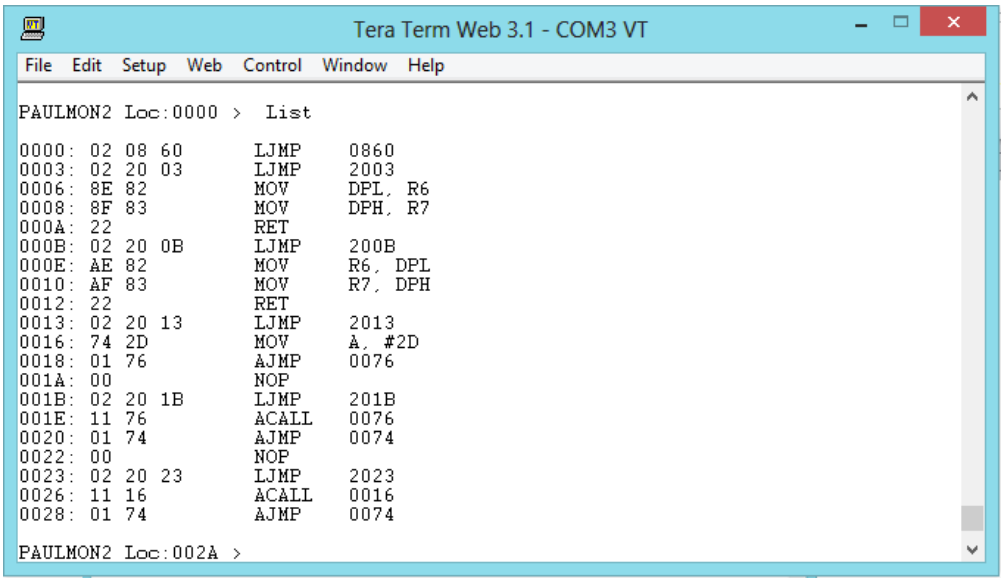
Below is the Welcome Screen for Paulmon



The extra.asm file available with Paulmon contains the 3 programs
List , Single –Step and Memory Editor

The List Function(L)

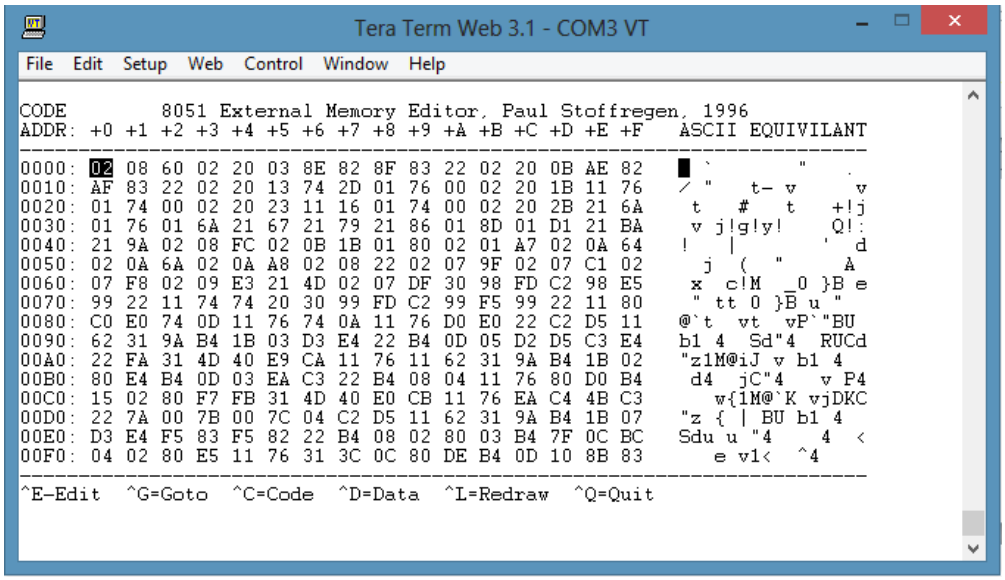
The List function is activated when 'L' key is pressed at the Paulmon prompt.
This function lists the contents of Code/Program Memory as shown below



The list representation shows a combination of memory addresses, hex codes , and mnemonics and operands similar to a .LST file representations printing 20 lines at once.

The Memory Editor(E)

The Memory Editor program can be executed by pressing the 'E' key at the Paulmon prompt. The program "draws" out the memory in a representation as shown below



And provides option to the user to manipulate memory with following options:

CTRL-E - Enable/Disable Editing Mode
CTRL-A - Select ASCII Editing Mode (visible once CTRL-E is Pressed)
CTRL-X - Select HEX Editing Mode (visible once CTRL-E is Pressed)
CTRL-F - Fill a block of memory
CTRL-G - Goto a new memory location
CTRL-C - Display CODE (MOVC) memory
CTRL-D - Display DATA (MOVX) memory
CTRL-L - Redraw Screen
CTRL-Q (or ESC): Quit

Clear Memory (C)
Used to clear the memory because XRAM is usually filled with garbage data.

Hex Dump Internal Memory (H)
Prints the contents of code memory in hex from the address indicated by the user.

Jump (J)
Jumps to a memory location entered by the user

New Location(N)
This is a goto function . Prompts the user for a address and then sets it as a base location for other operations.

Download(D)
Downloads a program (hex file) to the Data memory.

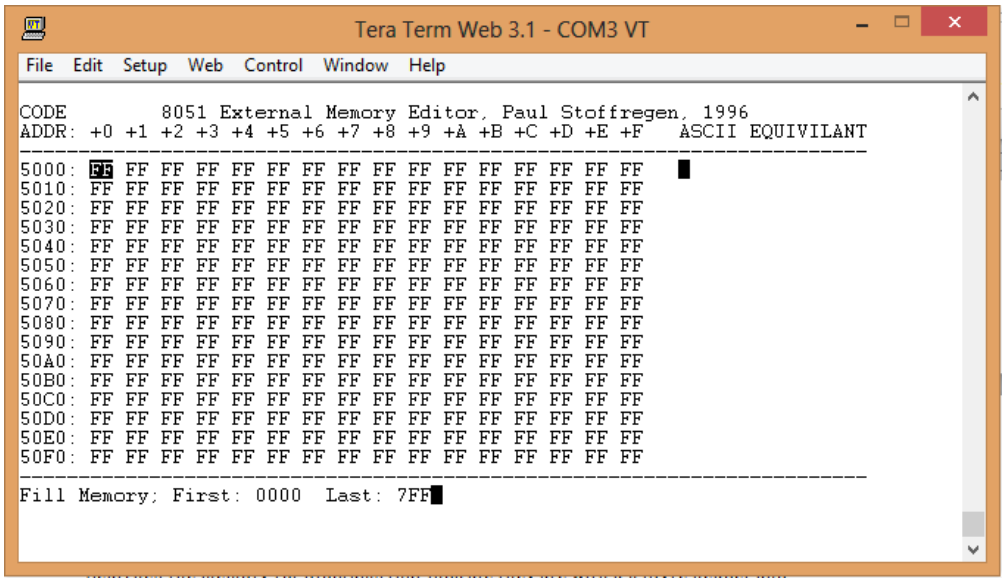
Run Program(R)
Searches the memory for programs that indicate they are with a 64byte header and then executes them.

The **maximum baud rate** at which the Paulmon2 program is operational is **57600**.

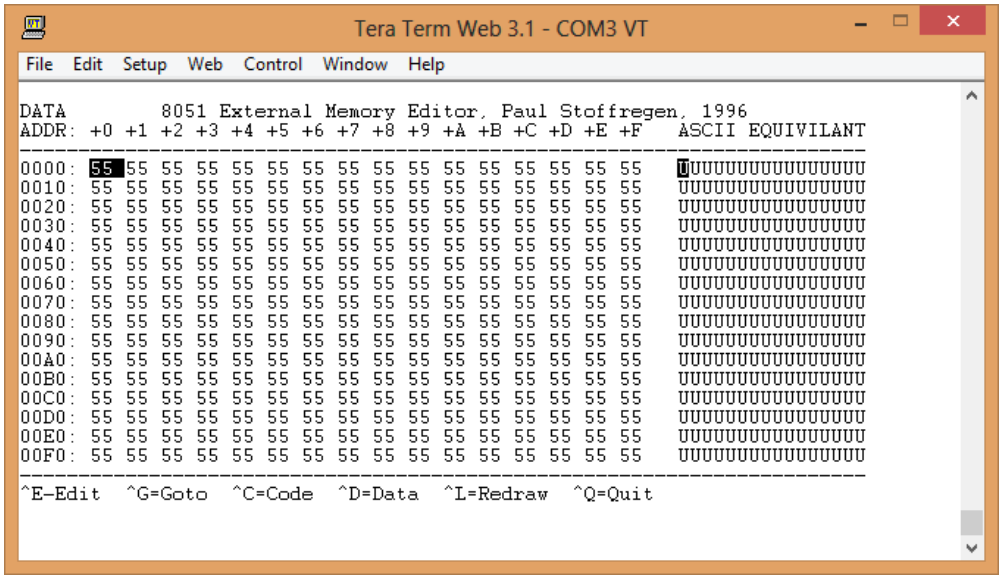
Paulmon Verification

Paulmon Functions were verified as follows:

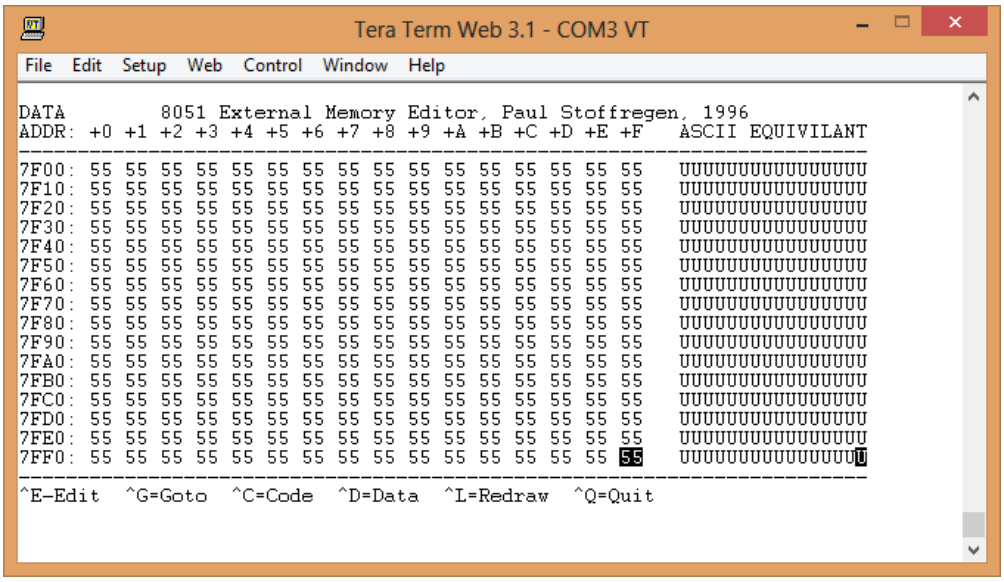
Filling the entire XRAM space 0x0000 to 0x7FFF with 0x55



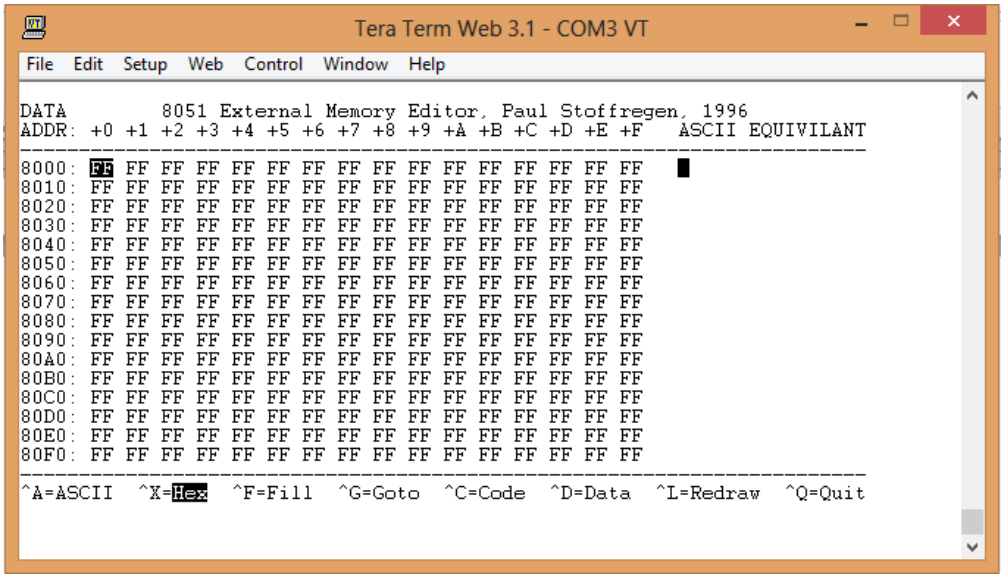
first location 0x000 filled with 0x55



Last Location 0x7FFF filled with 0x55



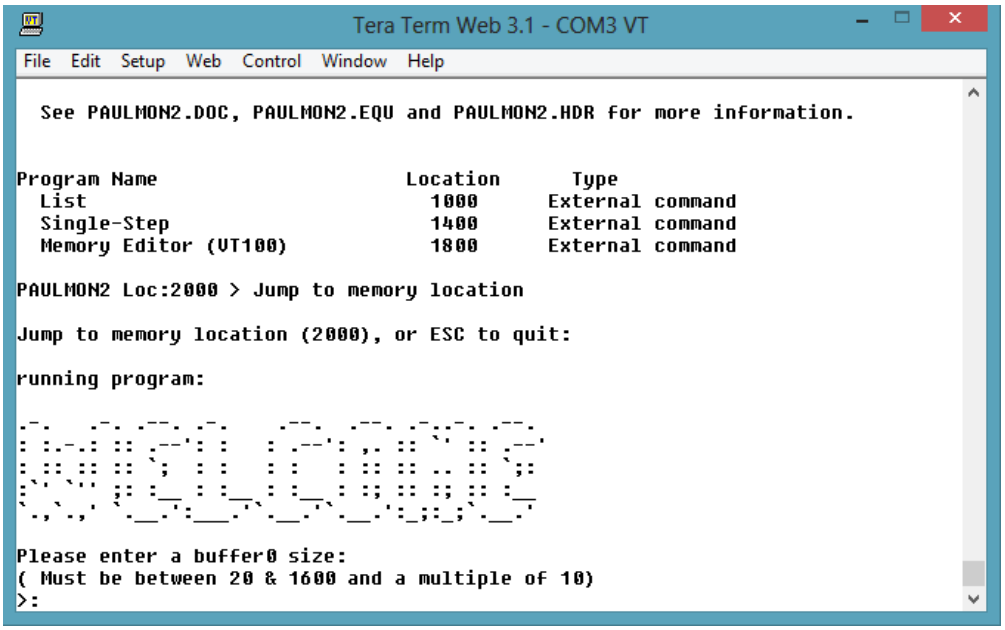
Block filling 0xAA over the memory locations 0x8000 to 0xFFFF does nothing



Program written in SDCC

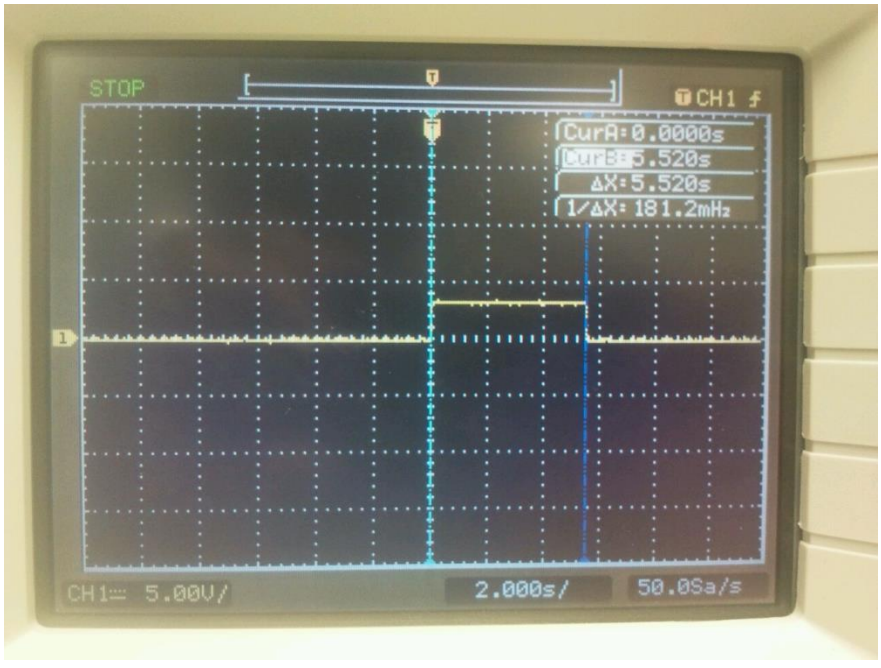
Required Element

(Code is provided in .c file)



The above screenshot shows the welcome screen which prompts the user to input the buffer size.

Toggling port pin 1.7 while consolidation is taking place .The pulse width measured for the toggled port pin is as shown below. (**5.520s in X1 mode)**



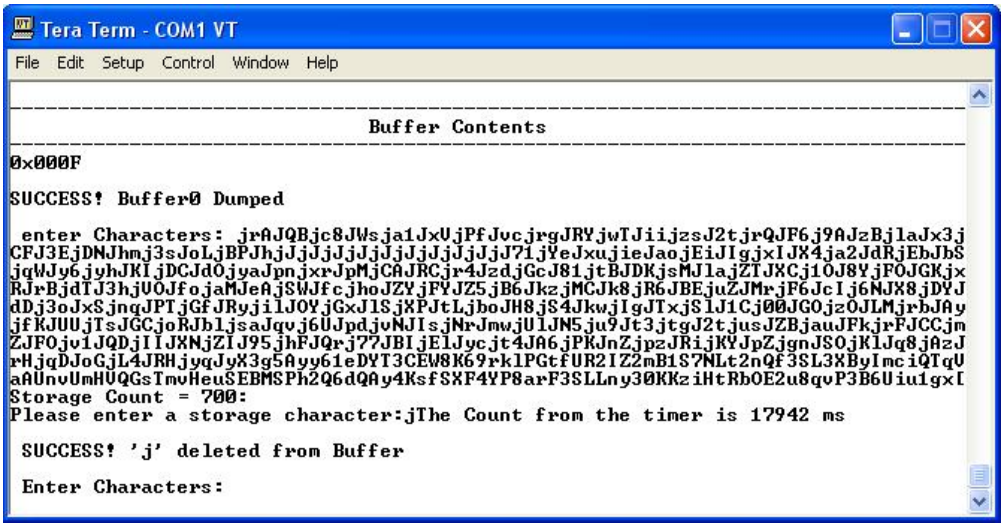
Supplemental Element

For part 17

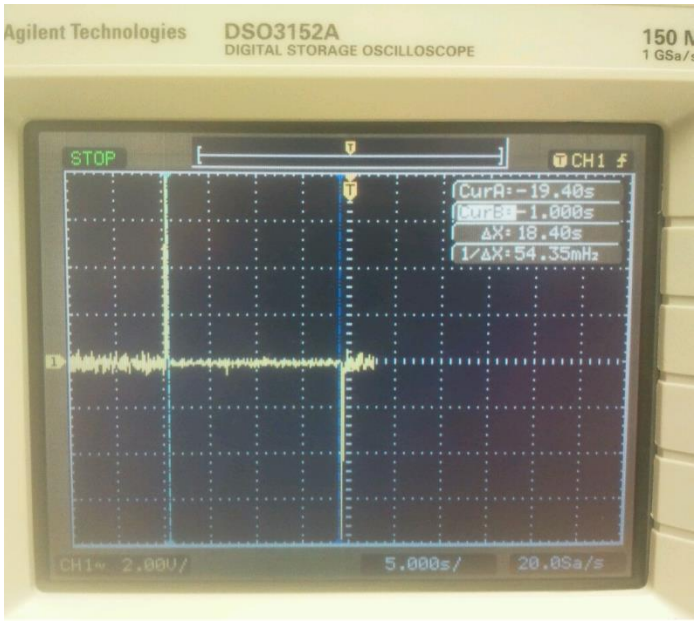
A hardware timer is initialized as follows for measuring the consolidation time.

```
TMOD |= 0x01;  // timer 0 , Mode 1
TH0 = 0xFC;  // time for 1 ms will be from count incrementing from FC65H to FFFFh
TL0 = 0x65;
TR0 = 0;
IE|= 0x82;
timer0cnt=0; //counter that keeps track of roll-overs
TR0=1; // start timer

//consolidation code

TR0=0;

//ISR for Timer 0
void isr_one(void) __interrupt (1) //ISR for Timer 0 overflow
{
    timer0cnt++; // For every 1ms, cnt increments
    TH0 = 0xFC;//Timer is reloaded with the 1ms count
    TL0 = 0x65;
    TF0=0;
}
```

On execution for a different set of test input and consolidation following output was obtained from the timer and the toggle pin as shown below respectively

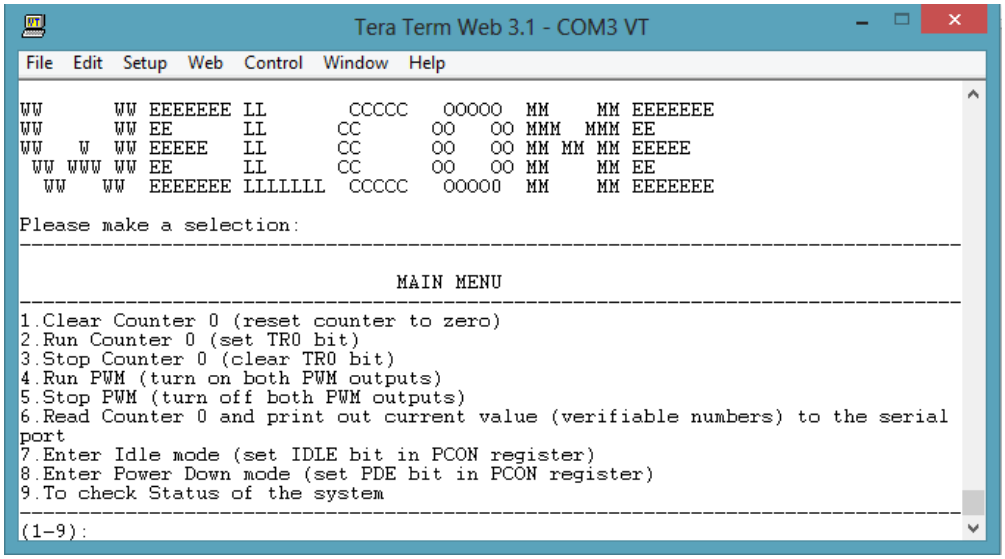


Timer count = **17.942ms**



Toggle pin P1_7 = **18.40s**

For part 18



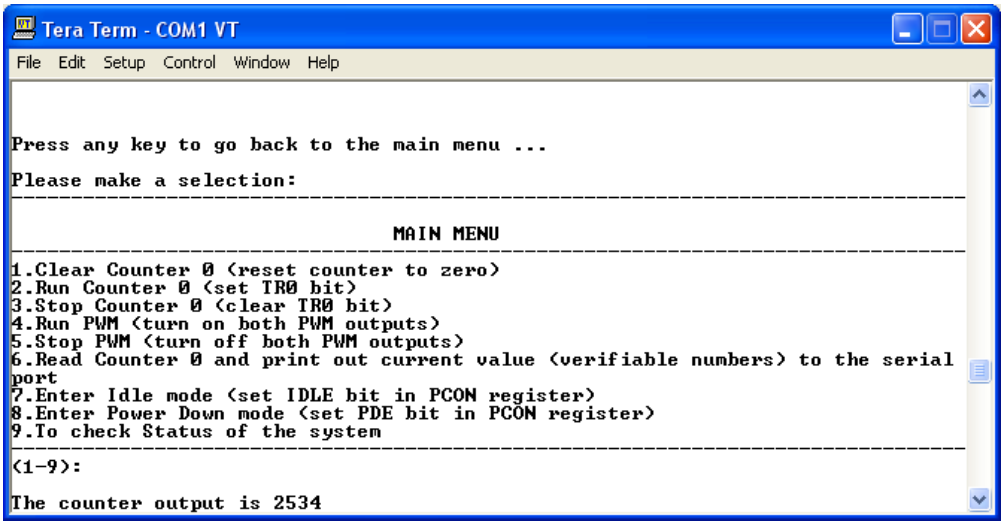
Above is the welcome screen with all the options for the test progam which implements the functions for counter 0 , PWM runs and Power Down and Idle modes.

User can input options from 1-9 for the various functions as listed.

Counter 0

Counter 0 Run **(2)**-> Counter 0 Stop **(3)** -> Read Counter 0 **(6)**

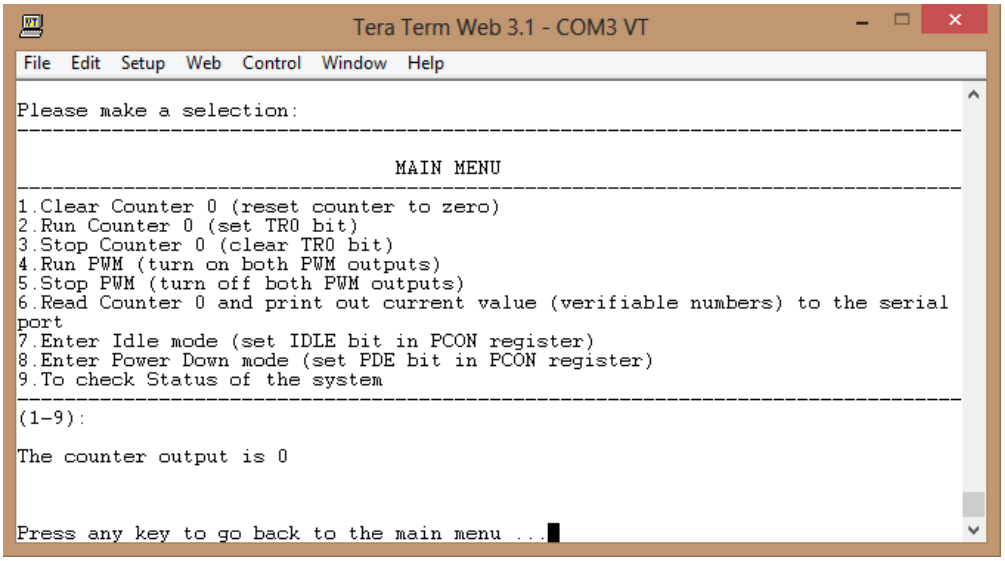
The above options yeilds a the following output



The count was provided from the **trigger signal generated by the CRO** which is a **square wave** of **100KHz** to the P3.3 pin.

The counter counted to 2534 and was stopped and the counter value was read and displayed.

The trigger input when taken out yielded no count, as shown below.



Pulse Width Modulation Outputs

The Pulse width modulation output is obtained by

Setting the following registers with values as below

```
CMOD|=0x02;
//We know that x0-xFF =  256 hence 5% of 256 =12.8 =0xC. Therefore 0xFF-0xC =0xF3
    CCAP0L = 0xF3;
    CCAP0H = 0xF3;
//We know that x0-xFF =  256 hence 5 of 256 =153.6 =0x99.Therfore 0xFF-0xC=0x66
    CCAP1L= 0x66;
    CCAP1H= 0x66;
```

Toggling the ECOM0 & CEX0 pins along with the CR bit (Common Clock Control) from the menus provides the generation of signals with duty cycles of 5% and 60%.
Below are the waveforms of duty cycles obtained at port pins P1_3 and P1_4.

Power Down and Idle Mode

One can enter the IDLE mode by setting the IDL bit(bit 0) in the PCON register. The Power Down Mode can be activated by setting the second bit (bit 1) in the PCON register.

A processor stops operating and pulls the ALE pin low and for both the modes.But the major difference between the two modes is that.

1. The processor can be returned to normal functionality by using any of the hardware interrupts or the serial port. As the clock is still available to the processor and all the timers and counters are still operational while the processor is in Idle Mode

2. In power down mode the Clock to the processor is stopped and hence can only be taken out of the PD mode by the Resetting the processor(using RST) or using the Hardware Interrupts.