

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 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 2 and 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
2	XRS0	0	0	256 Bytes (default)
		0	1	512 Bytes
		1	0	768 Bytes
		1	1	1024 Bytes

Testing

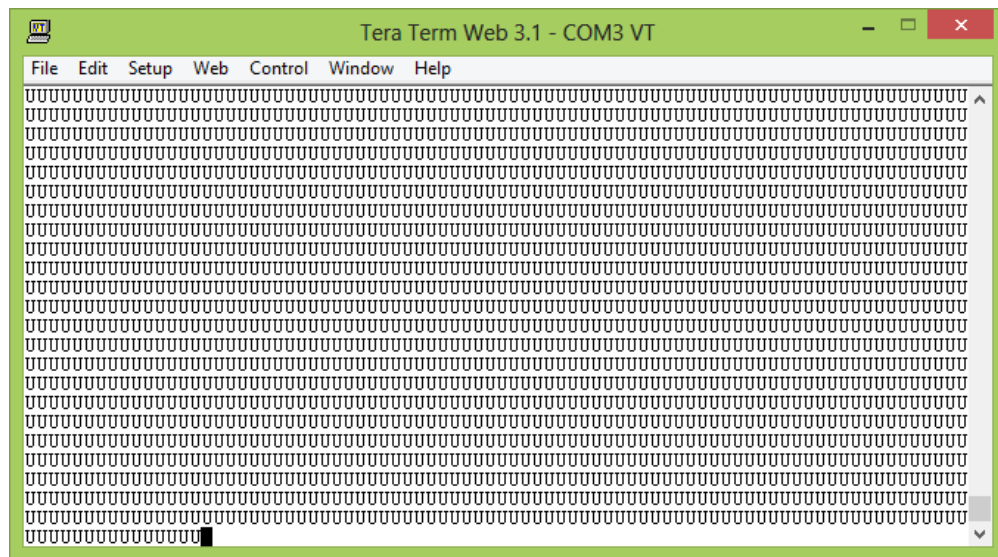
After confirming all the hardware connections I tested the RS-232 circuit using Tera Term Pro Serial Emulator, the compiled 5Paulmon21.asm below code which transmits the character 'U' in a loop.

```

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

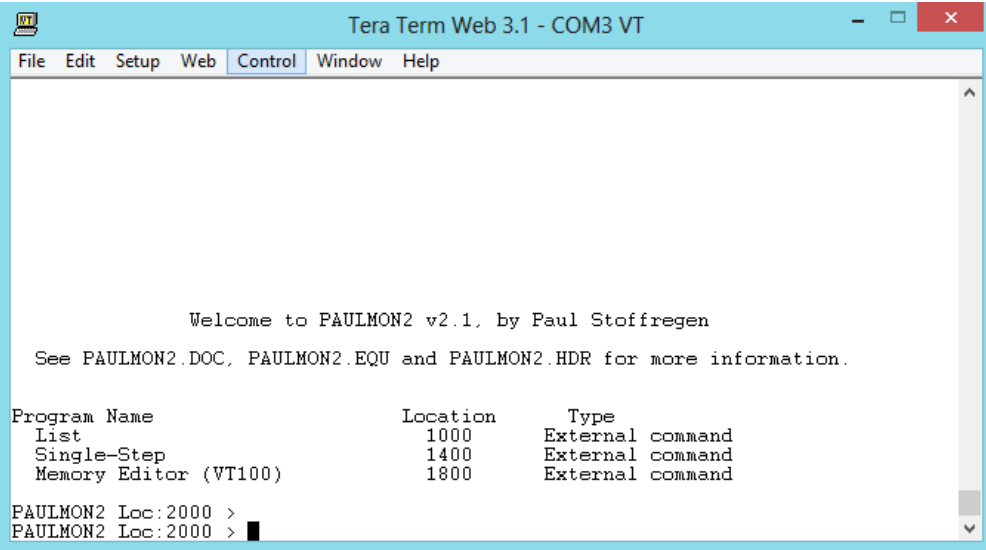
                ORG      $0100                ; Jump to main Program
MAIN:  MOV      TMOD,#020H                ; Initialize timer in mode 2
        MOV      TH1,#0FDH                ; Load value 0xFDh for 9600 baud rate
        SETB     SCON.6                    ; Set Serial port in mode 1
        SETB     TCON.6                    ; Start The Timer
LOOP:  MOV      SBUF,#'U'                ; Load the Serial Buffer with the character 'U'
        JNB      SCON.1,$                  ; 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



```

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

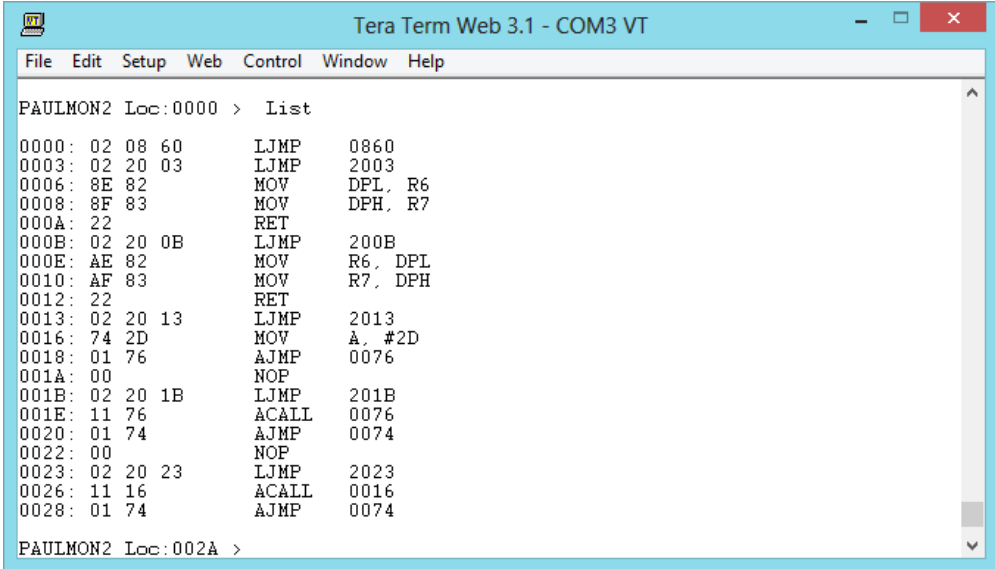
PAULMON2 Loc:2000 >
PAULMON2 Loc:2000 >

```

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



```

PAULMON2 Loc:0000 > List
0000: 02 08 60      LJMP    0860
0003: 02 20 03      LJMP    2003
0006: 8E 82          MOV     DPL, R6
0008: 8F 83          MOV     DPH, R7
000A: 22             RET
000B: 02 20 0B      LJMP    200B
000E: AE 82          MOV     R6, DPL
0010: AF 83          MOV     R7, DPH
0012: 22             RET
0013: 02 20 13      LJMP    2013
0016: 74 2D          MOV     A, #2D
0018: 01 76          AJMP    0076
001A: 00             NOP
001B: 02 20 1B      LJMP    201B
001E: 11 76          ACALL   0076
0020: 01 74          AJMP    0074
0022: 00             NOP
0023: 02 20 23      LJMP    2023
0026: 11 16          ACALL   0016
0028: 01 74          AJMP    0074

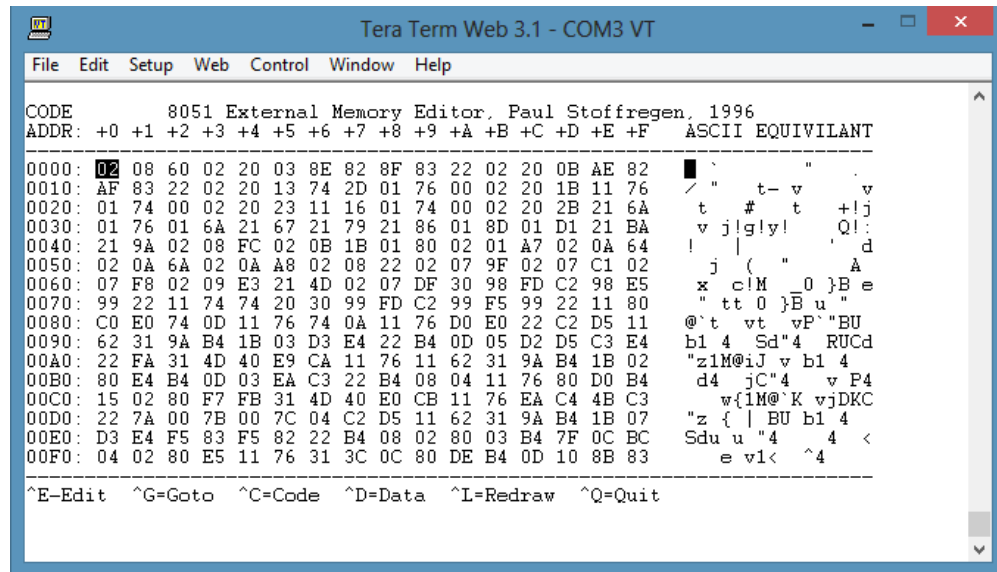
PAULMON2 Loc:002A >

```

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



```
Tera Term Web 3.1 - COM3 VT
File Edit Setup Web Control Window Help
CODE      8051 External Memory Editor. Paul Stoffregen, 1996
ADDR: +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F  ASCII EQUIVILANT
-----
0000: 02 08 60 02 20 03 8E 82 8F 83 22 02 20 0B AE 82  " t- v v
0010: AF 83 22 02 20 13 74 2D 01 76 00 02 20 1B 11 76  / " t- v v
0020: 01 74 00 02 20 23 11 16 01 74 00 02 20 2B 21 6A  t # t +!j
0030: 01 76 01 6A 21 67 21 79 21 86 01 8D 01 D1 21 BA  v j!g!y! Q!
0040: 21 9A 02 08 FC 02 0B 1B 01 80 02 01 A7 02 0A 64  ! j | ' d
0050: 02 0A 6A 02 0A A8 02 08 22 02 07 9F 02 07 C1 02  j ( " A
0060: 07 F8 02 09 E3 21 4D 02 07 DF 30 98 FD C2 98 E5  x c!M _0 }B e
0070: 99 22 11 74 74 20 30 99 FD C2 99 F5 99 22 11 80  " tt 0 }B u "
0080: C0 E0 74 0D 11 76 74 0A 11 76 D0 E0 22 C2 D5 11  @`t vt vP`"BU
0090: 62 31 9A B4 1B 03 D3 E4 22 B4 0D 05 D2 D5 C3 E4  b1 4 Sd"4 RUCd
00A0: 22 FA 31 4D 40 E9 CA 11 76 11 62 31 9A B4 1B 02  "z!M@iJ v b1 4
00B0: 80 E4 B4 0D 03 EA C3 22 B4 08 04 11 76 80 D0 B4  d4 jC"4 v P4
00C0: 15 02 80 F7 FB 31 4D 40 E0 CB 11 76 EA C4 4B C3  w{iM@`K vjDKC
00D0: 22 7A 00 7B 00 7C 04 C2 D5 11 62 31 9A B4 1B 07  "z { | BU b1 4
00E0: D3 E4 F5 83 F5 82 22 B4 08 02 80 03 B4 7F 0C BC  Sdu u "4 ^4 <
00F0: 04 02 80 E5 11 76 31 3C 0C 80 DE B4 0D 10 8B 83  e v!< ^4
-----
^E=Edit ^G=Goto ^C=Code ^D=Data ^L=Redraw ^Q=Quit
```

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)

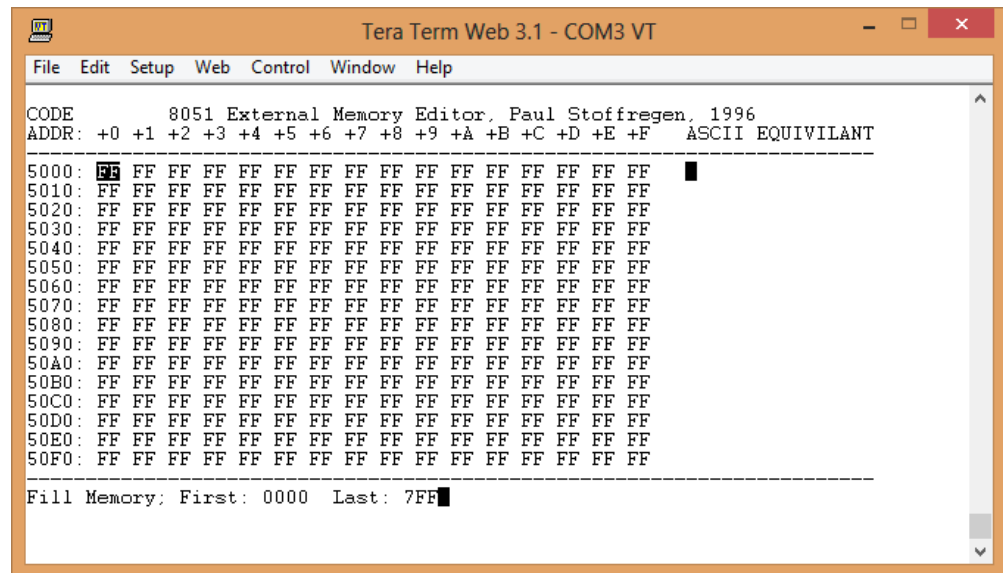
Searches the memory for programs that indicate they are with a 64byte header and then executes them.

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

Paulmon Verification

Paulmon Functions were verified as follows:

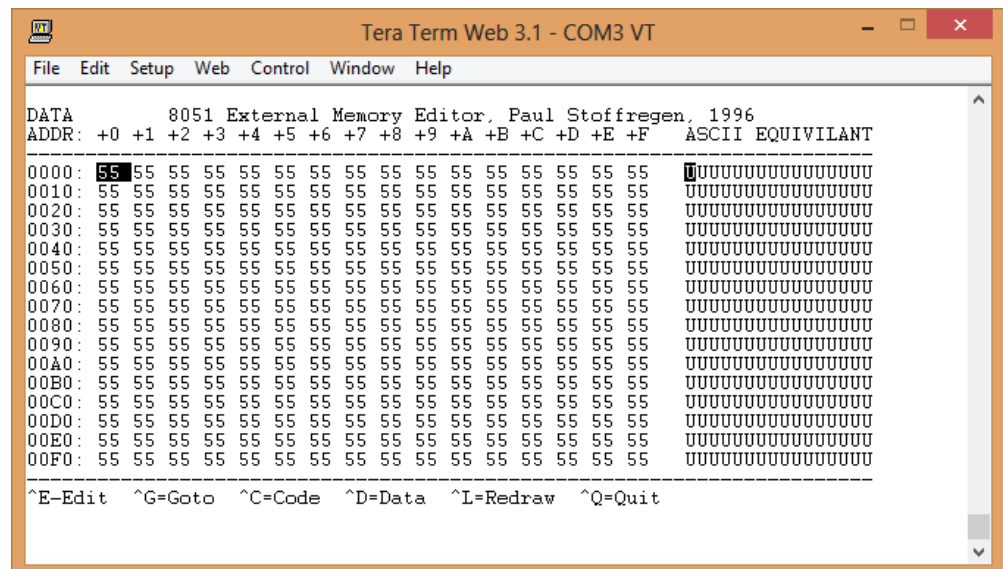
Filling the entire XRAM space 0x0000 to 0x7FFF with 0x55



```
CODE      8051 External Memory Editor, Paul Stoffregen, 1996
ADDR: +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F  ASCII EQUIVILANT
-----
5000: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
5010: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
5020: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
5030: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
5040: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
5050: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
5060: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
5070: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
5080: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
5090: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
50A0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
50B0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
50C0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
50D0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
50E0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
50F0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

Fill Memory; First: 0000 Last: 7FFF
```

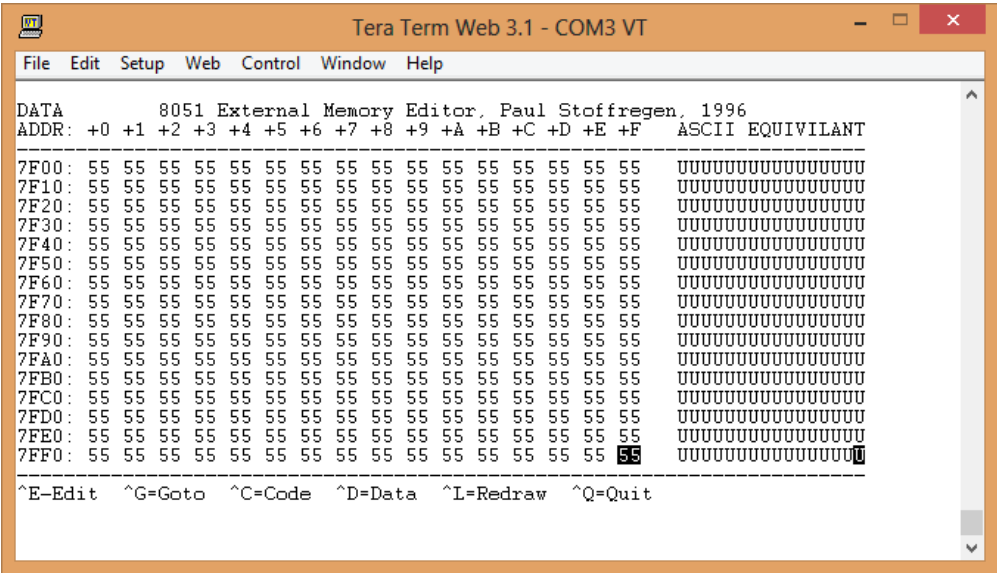
first location 0x000 filled with 0x55



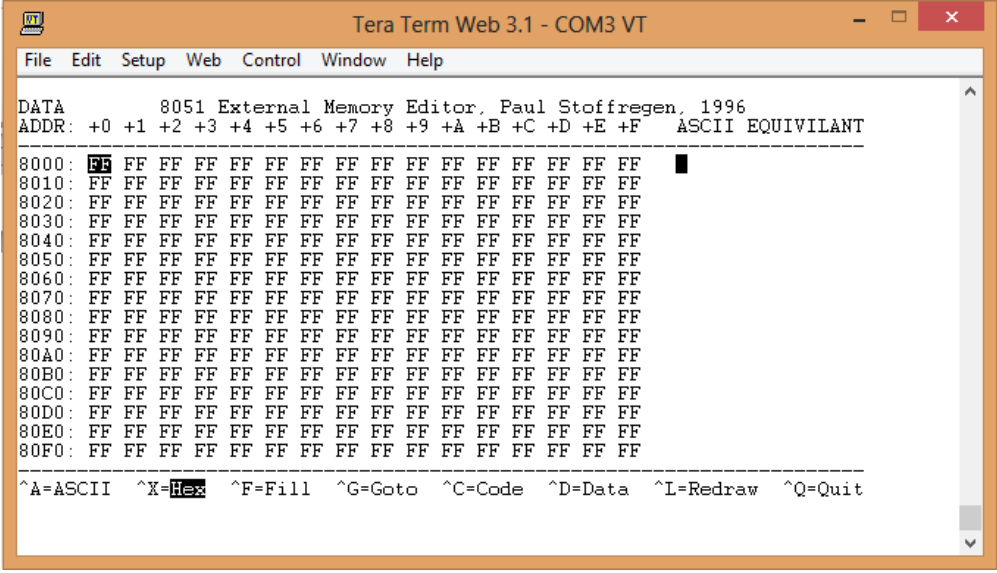
```
DATA      8051 External Memory Editor, Paul Stoffregen, 1996
ADDR: +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F  ASCII EQUIVILANT
-----
0000: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55  UUUUUUUUUUUUUUUUU
0010: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55  UUUUUUUUUUUUUUUUU
0020: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55  UUUUUUUUUUUUUUUUU
0030: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55  UUUUUUUUUUUUUUUUU
0040: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55  UUUUUUUUUUUUUUUUU
0050: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55  UUUUUUUUUUUUUUUUU
0060: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55  UUUUUUUUUUUUUUUUU
0070: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55  UUUUUUUUUUUUUUUUU
0080: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55  UUUUUUUUUUUUUUUUU
0090: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55  UUUUUUUUUUUUUUUUU
00A0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55  UUUUUUUUUUUUUUUUU
00B0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55  UUUUUUUUUUUUUUUUU
00C0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55  UUUUUUUUUUUUUUUUU
00D0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55  UUUUUUUUUUUUUUUUU
00E0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55  UUUUUUUUUUUUUUUUU
00F0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55  UUUUUUUUUUUUUUUUU

^E=Edit ^G=Goto ^C=Code ^D=Data ^L=Redraw ^Q=Quit
```

Last Location 0x7FFF filled with 0x55



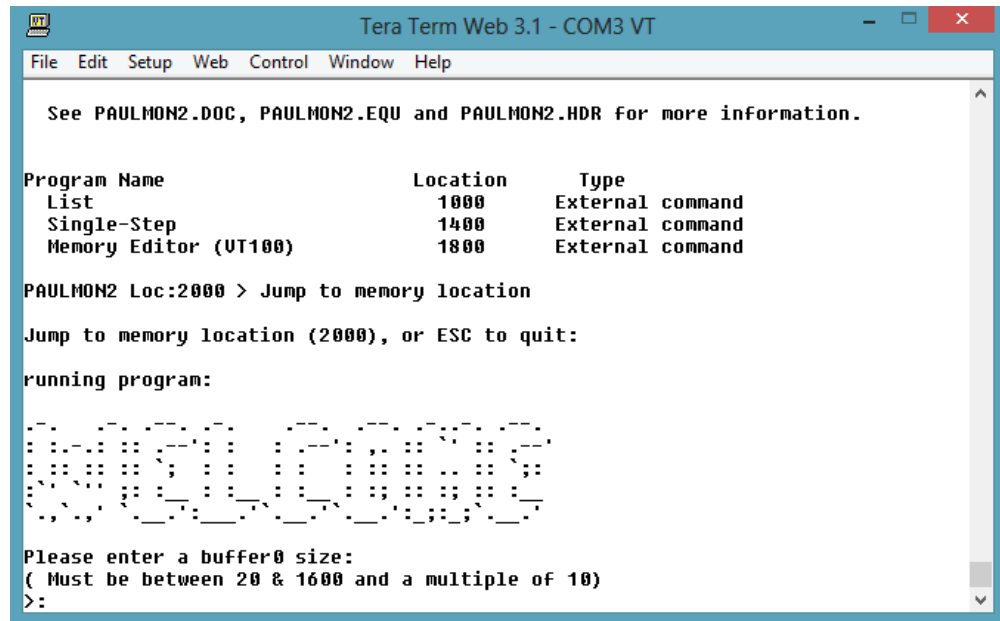
Block filling 0xAA over the memory locations 0x8000 to 0xFFFF does nothing



Program written in SDCC

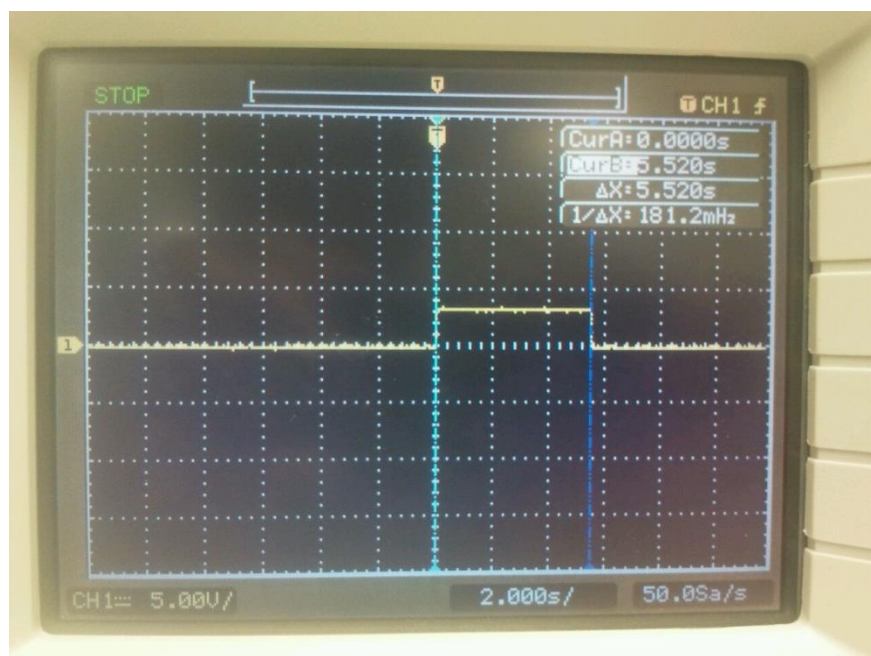
Required Element

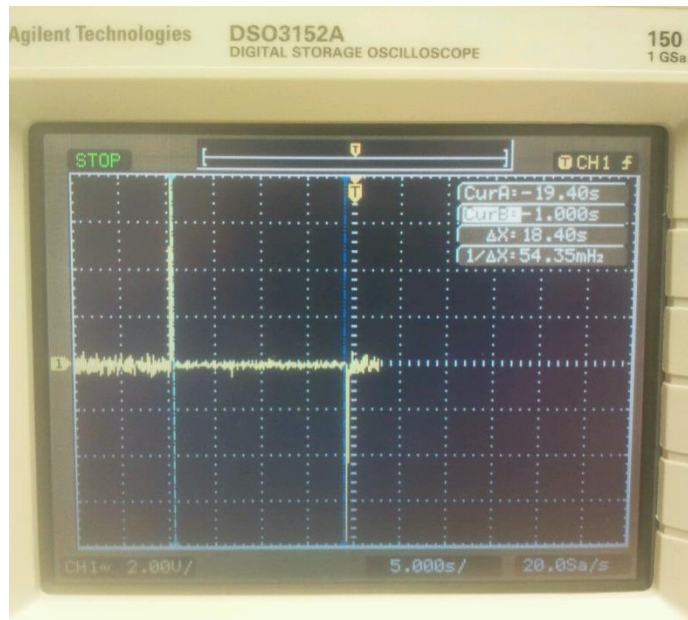
(Code is provided in .c file)



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

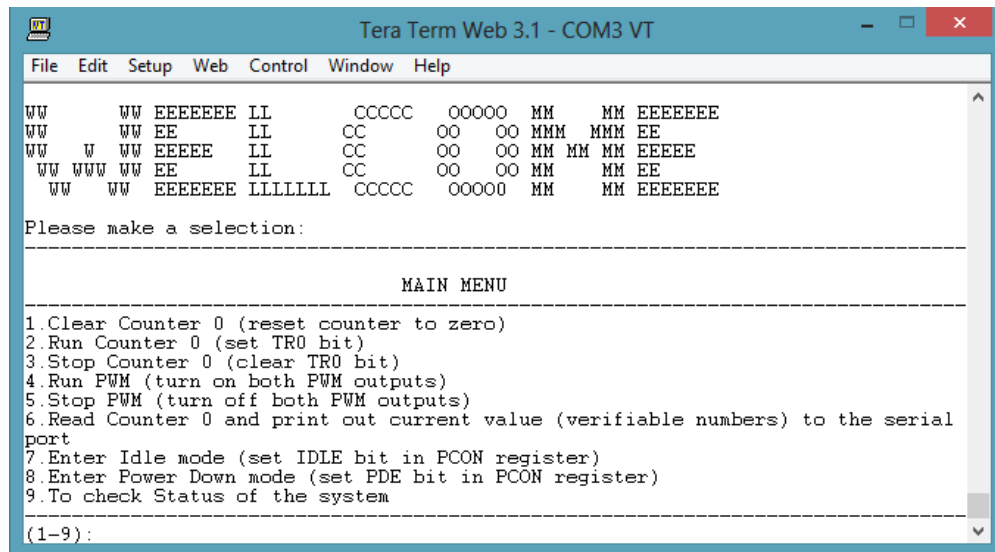
Toggling port pin 1.7 while consolidation is taking place .The pulse width measured for the toggled port pin is as shown below. **(5.520s in X1 mode)**





Toggle pin P1_7 = **18.40s**

For part 18



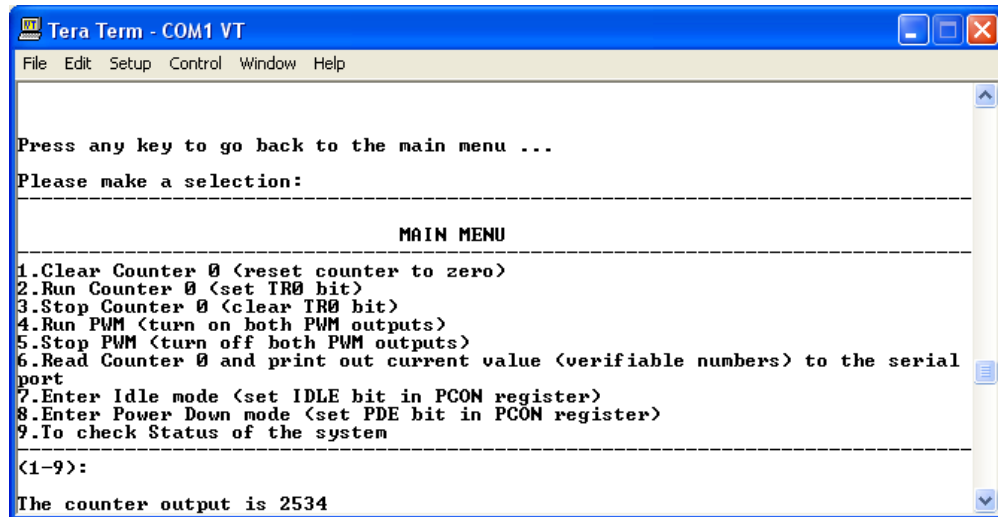
Above is the welcome screen with all the options for the test program which implements the functions for counter 0 , PWM runs and Power Down and Idle modes.

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



```
Tera Term - COM1 VT
File Edit Setup Control Window Help

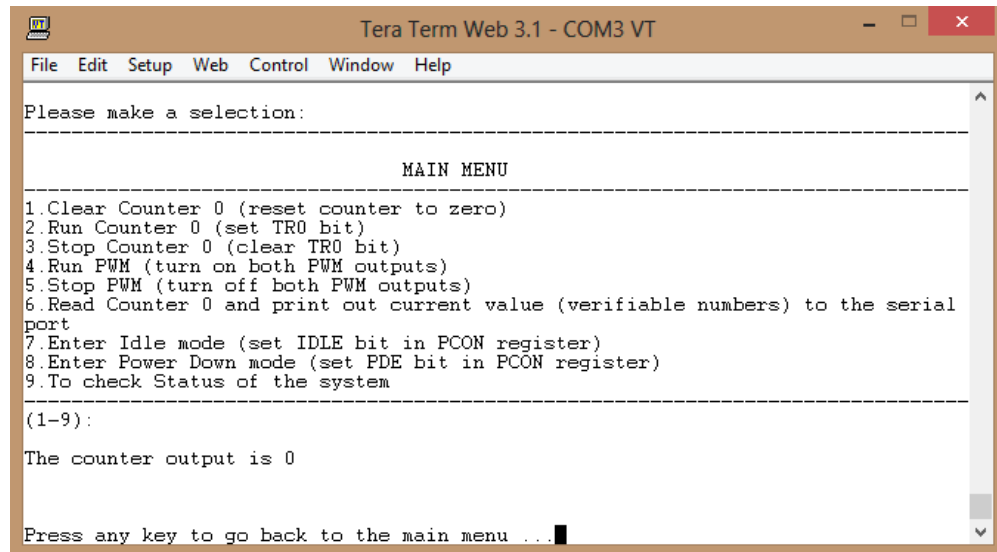
Press any key to go back to the main menu ...
Please make a selection:

-----
MAIN MENU
-----
1.Clear Counter 0 (reset counter to zero)
2.Run Counter 0 (set TR0 bit)
3.Stop Counter 0 (clear TR0 bit)
4.Run PWM (turn on both PWM outputs)
5.Stop PWM (turn off both PWM outputs)
6.Read Counter 0 and print out current value (verifiable numbers) to the serial
port
7.Enter Idle mode (set IDLE bit in PCON register)
8.Enter Power Down mode (set PDE bit in PCON register)
9.To check Status of the system
-----
<1-9>:
The counter output is 2534
```

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.



```
Tera Term Web 3.1 - COM3 VT
File Edit Setup Web Control Window Help

Please make a selection:

-----
MAIN MENU
-----
1.Clear Counter 0 (reset counter to zero)
2.Run Counter 0 (set TR0 bit)
3.Stop Counter 0 (clear TR0 bit)
4.Run PWM (turn on both PWM outputs)
5.Stop PWM (turn off both PWM outputs)
6.Read Counter 0 and print out current value (verifiable numbers) to the serial
port
7.Enter Idle mode (set IDLE bit in PCON register)
8.Enter Power Down mode (set PDE bit in PCON register)
9.To check Status of the system
-----
(1-9):
The counter output is 0

Press any key to go back to the main menu ...
```

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. Therefore 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.