Nathan Tipton

A01207112

Tarrin Rasmussen

Lab 2: Debugging, Commenting, and Logic Analyzers

Overview:

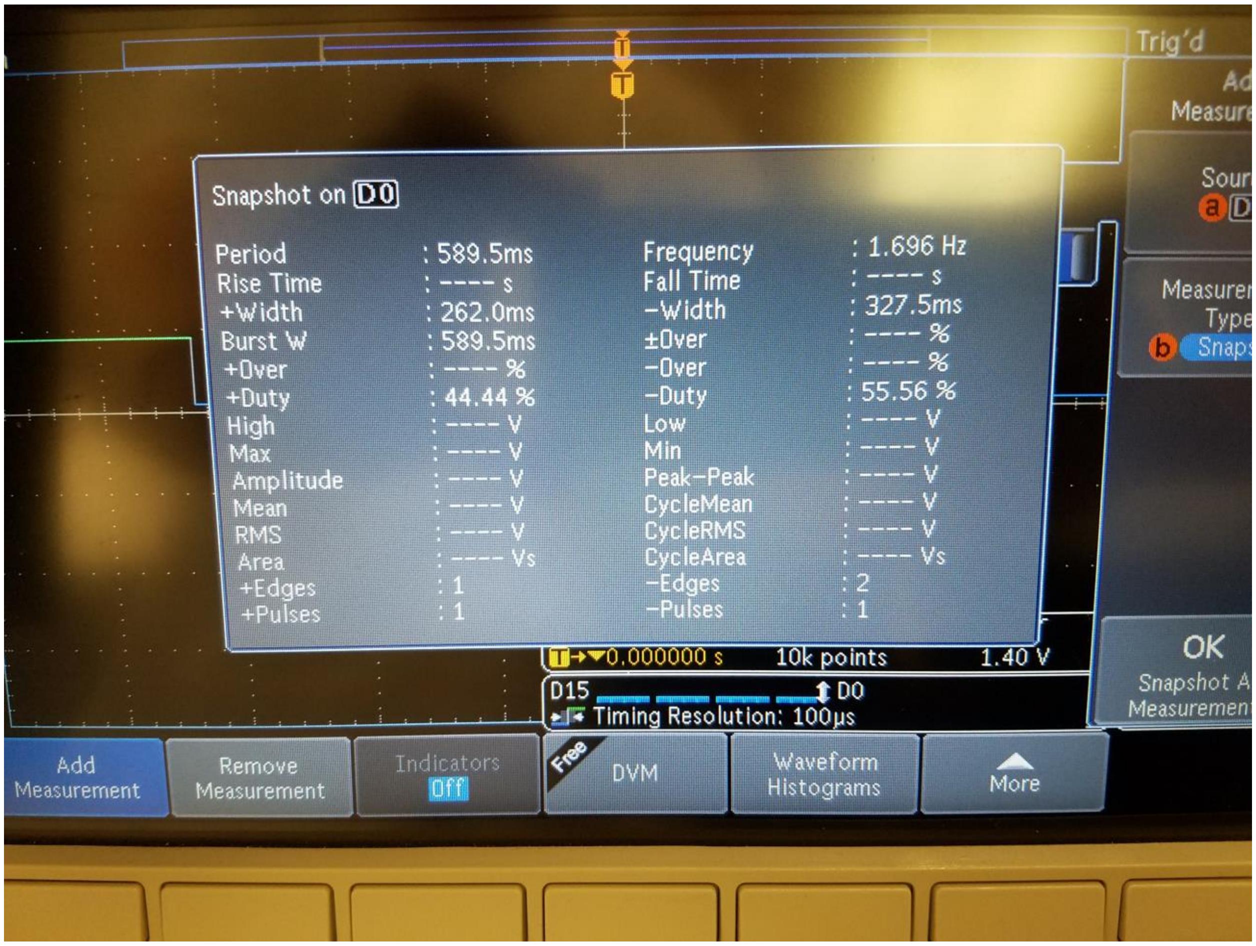
The purpose of this lab was an introduction to debugging.

Background:

Debugging is an important step to writing code. You will need to become proficient at working through the code to find mistakes or problems.

Procedure:

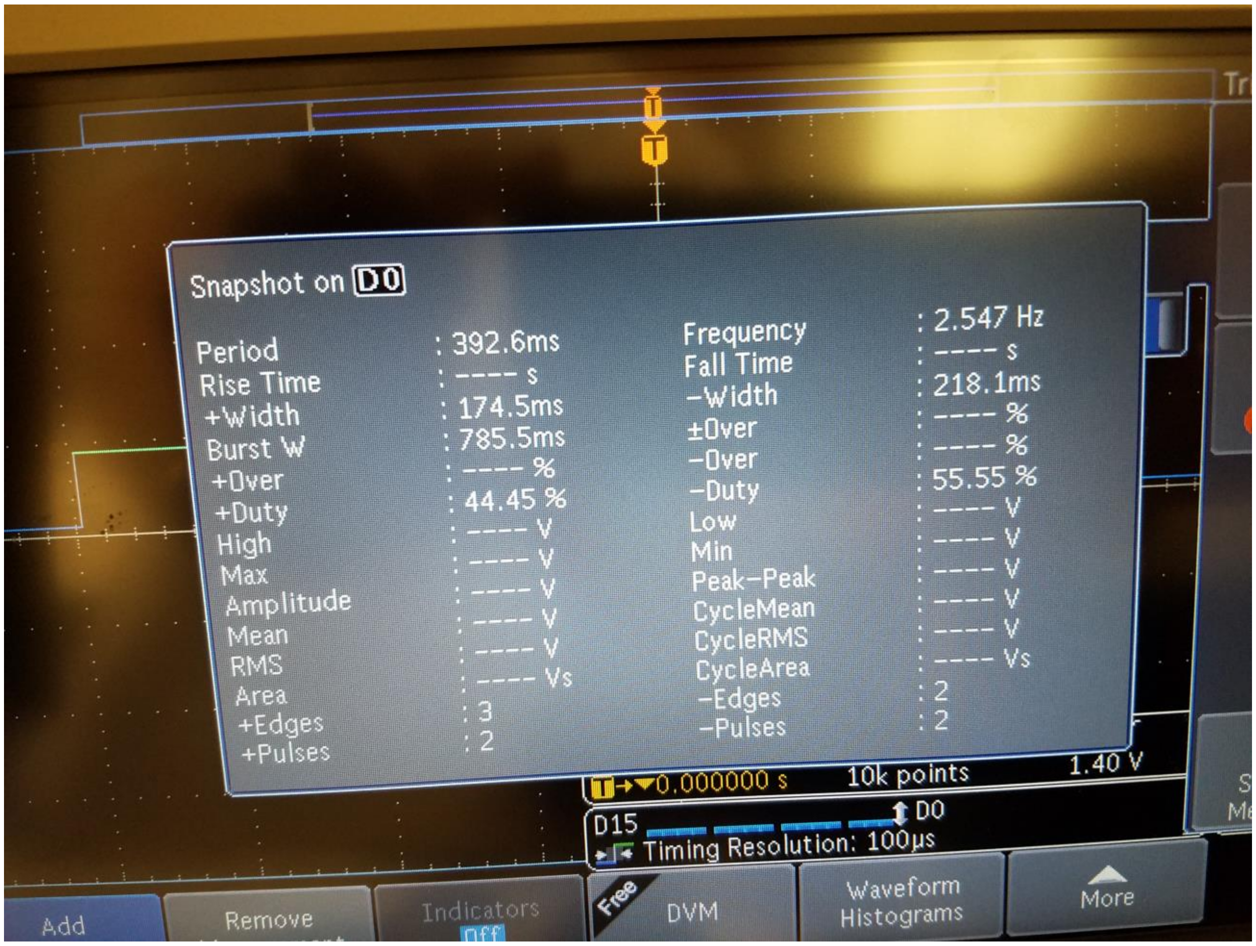
1. We downloaded the blinky.asm file to our board.
2. Using the logic analyzer we measured the frequency of the LED blinking.



```
Snapshot on D0

Period       : 589.5ms      Frequency   : 1.696 Hz
Rise Time    : ---- s       Fall Time   : ---- s
+Width       : 262.0ms      -Width      : 327.5ms
Burst W      : 589.5ms      ±Over       : ---- %
+Over        : ---- %       -Over       : ---- %
+Duty        : 44.44 %      -Duty       : 55.56 %
High         : ---- V       Low         : ---- V
Max          : ---- V       Min         : ---- V
Amplitude    : ---- V       Peak-Peak   : ---- V
Mean         : ---- V       CycleMean   : ---- V
RMS          : ---- V       CycleRMS    : ---- V
Area         : ---- Vs      CycleArea   : ---- Vs
+Edges       : 1            -Edges      : 2
+Pulses      : 1            -Pulses     : 1
```

FIG. 1: Measurement of LED frequency

3. We modified Blinky.asm to make the board blink approximately twice as fast.



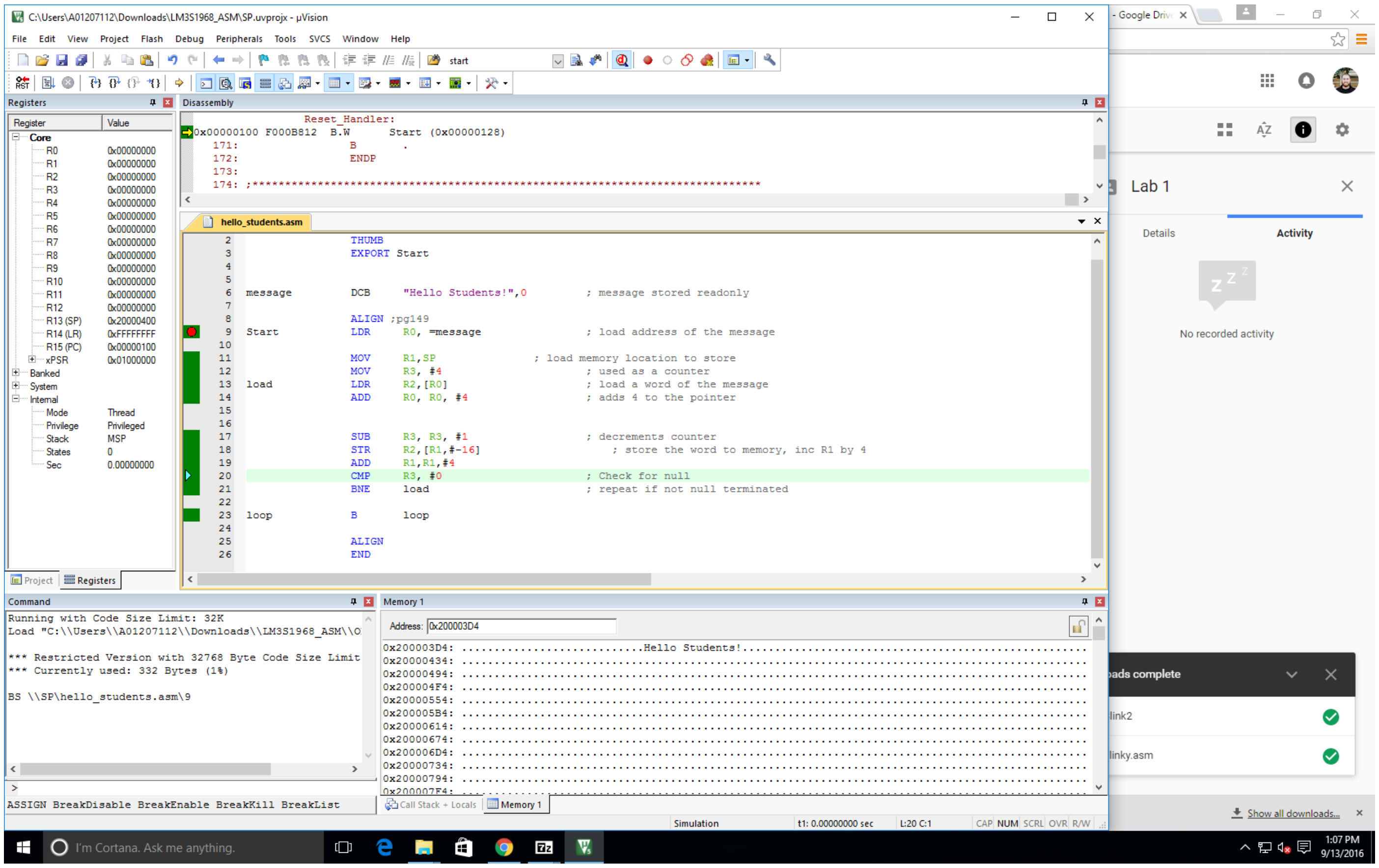
```
Snapshot on D0

Period        : 392.6ms        Frequency    : 2.547 Hz
Rise Time     : ---- s         Fall Time    : ---- s
+Width        : 174.5ms        -Width       : 218.1ms
Burst W       : 785.5ms        ±Over        : ---- %
+Over         : ---- %         -Over        : ---- %
+Duty         : 44.45 %        -Duty        : 55.55 %
High          : ---- V         Low          : ---- V
Max           : ---- V         Min          : ---- V
Amplitude     : ---- V         Peak-Peak    : ---- V
Mean          : ---- V         CycleMean    : ---- V
RMS           : ---- V         CycleRMS     : ---- V
Area          : ---- Vs        CycleArea    : ---- Vs
+Edges        : 3              -Edges       : 2
+Pulses       : 2              -Pulses      : 2

T→0.000000 s     10k points     1.40 V
D15     D0
Timing Resolution: 100µs
```

FIG. 2: Measurement of LED frequency after code edit

4. We commented the code to clarify and improve debugging.

```
*********************************Blinky.asm****************************************
Start
        mov32 R0, #0x400FE108 ; Enable GPIO Clock
        mov R1, #0x20 ; store 0x20 in R1
        str R1, [R0]

        mov32 R0, #0x40025000 ;GPIOF address

        ;unlock GPIOF
        mov32 R1, #0x4C4F434B; GPIO Unlock code.
        str R1, [R0,#0x520]; Store R1 into address R0 + 0x520

        mov R1, #0x1F
        str R1, [R0,#0x524];GPIOCR
        mov R1, #0x11
        str R1, [R0,#0x510]
        mov R1, #0x0F
        str R1, [R0,#0x400] ;GPIODIR
        mov R1, #0x1F
        str R1, [R0,#0x51C] ;digital enable
loop
        MOV32 R1, #0x02          ;load value for turning on LED colore __red__
        STR R1, [R0,#0x38]       ;write the above value to GPIOF ODR register.
        MOV R4, #0                  ;initial value for iteration loop

        MOV32 R5, #0xaaaaaa     ;number of iterations for delay loops

delay1

        ADD R4, #1                  ; ___increment value in R4 each
iteration_________________________

        CMP R4, R5                  ;check number of iterations

        BLE delay1                  ;continue if iterated less than 0xFFFFF + 1 times, otherwise
repeat delay loop


        MOV32 R1, #0x04         ;load value for turning on LED color __blue___

        STR R1, [R0, #0x38]     ;write the above value to GPIOF ODR


        MOV R4, #0              ;initial value for iteration loop

        ;                       ; **** the tm4c123gh6pm has 16 MHz clock.

        ;;                       how long should the loop take with that clock?


delay2

        ADD R4, #1                  ;_____increment value in R4 each iteration____________

        CMP R4, R5                  ;check number of iterations

        BLE delay2                  ;continue if iterated less than 0xFFFFF + 1 times, otherwise
repeat delay loop


        MOV32 R1, #0x08         ;load value for turning on LED _green____

        STR R1, [R0, #0x38]     ;write the above value to GPIOF ODR


        B loop   ;do it all over again, forever

        ALIGN

        END
```
************************************END BLINKY.ASM********************************

5. We simulated the above code in Kiel to become familiar with using the debugger while simulating.
6. We debugged hello_students.asm. We used the debugger in Kiel.



```
Reset_Handler:
0x00000100 F000B812  B.W      Start (0x00000128)
   171:                  B        .
   172:                  ENDP
   173:
   174: ;*****************************************************************************
```

```
                THUMB
                EXPORT Start


message         DCB     "Hello Students!",0         ; message stored readonly

                ALIGN ;pg149
Start           LDR     R0, =message                ; load address of the message

                MOV     R1,SP                   ; load memory location to store
                MOV     R3, #4                      ; used as a counter
load            LDR     R2,[R0]                     ; load a word of the message
                ADD     R0, R0, #4                  ; adds 4 to the pointer


                SUB     R3, R3, #1                  ; decrements counter
                STR     R2,[R1,#-16]                    ; store the word to memory, inc R1 by 4
                ADD     R1,R1,#4
                CMP     R3, #0                      ; Check for null
                BNE     load                        ; repeat if not null terminated

loop            B       loop

                ALIGN
                END
```

Register	Value
Core	
R0	0x00000000
R1	0x00000000
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20000400
R14 (LR)	0xFFFFFFFF
R15 (PC)	0x00000100
xPSR	0x01000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	0
Sec	0.00000000

```
Running with Code Size Limit: 32K
Load "C:\\Users\\A01207112\\Downloads\\LM3S1968_ASM\\O

*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 332 Bytes (1%)

BS \\SP\hello_students.asm\9
```

```
Address: 0x200003D4
0x200003D4: ............................Hello Students!.........................................................
0x20000434: ..............................................................................................................
0x20000494: ..............................................................................................................
0x200004F4: ..............................................................................................................
0x20000554: ..............................................................................................................
0x200005B4: ..............................................................................................................
0x20000614: ..............................................................................................................
0x20000674: ..............................................................................................................
0x200006D4: ..............................................................................................................
0x20000734: ..............................................................................................................
0x20000794: ..............................................................................................................
0x200007F4: ..............................................................................................................
```

C:\Users\A01207112\Downloads\LM3S1968_ASM\SP.uvprojx - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Disassembly
hello_students.asm
Project Registers
Command
Memory 1
ASSIGN BreakDisable BreakEnable BreakKill BreakList
Call Stack + Locals
Simulation
t1: 0.00000000 sec
L:20 C:1
Lab 1
Details
Activity
No recorded activity
Show all downloads...
I'm Cortana. Ask me anything.
1:07 PM
9/13/2016

FIG. 3: Debugging of code

7. Using the logic analyzer, we captured a bit pattern stream. We set it up to begin storing data after receiving ASCII character \n. After deciphering the message using an asci table, we found it to read “HELLO!”.

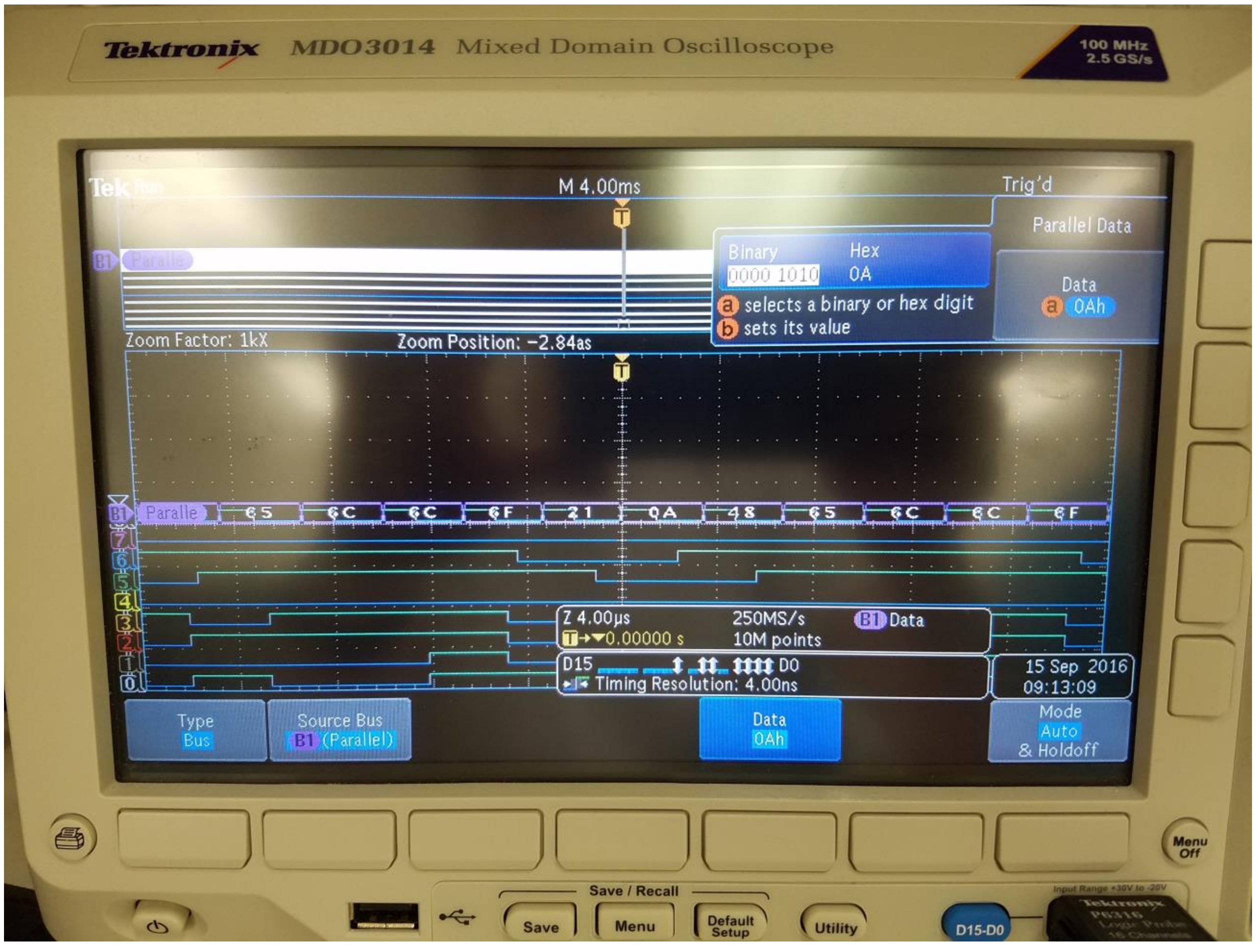


FIG. 4: Deciphering message

Conclusion

This lab was very useful. Using the debugger will be a valuable skill that will save lots of time later on. When we initially used the debugger to debug hello_students.asm we had troubles. We were having a hard time understanding how the stack worked and how the code was using the stack pointer. We came to understand that in order for the code to write the message properly into memory the stack pointer had to be moved back. We did this manually but later found out we could have used the push command. We moved the pointer back to provide the proper number of bytes required to save the message into memory. We also had to increment the register that held the memory address so the different parts of the message weren't just overwriting one another at the same address.

We found it very useful to use the debugger to examine all the registers and watch them as the code ran. You are able to see all the changes made it real time. This allows you a better understanding of what the code is doing. It can be difficult to follow the code when just reading through it.