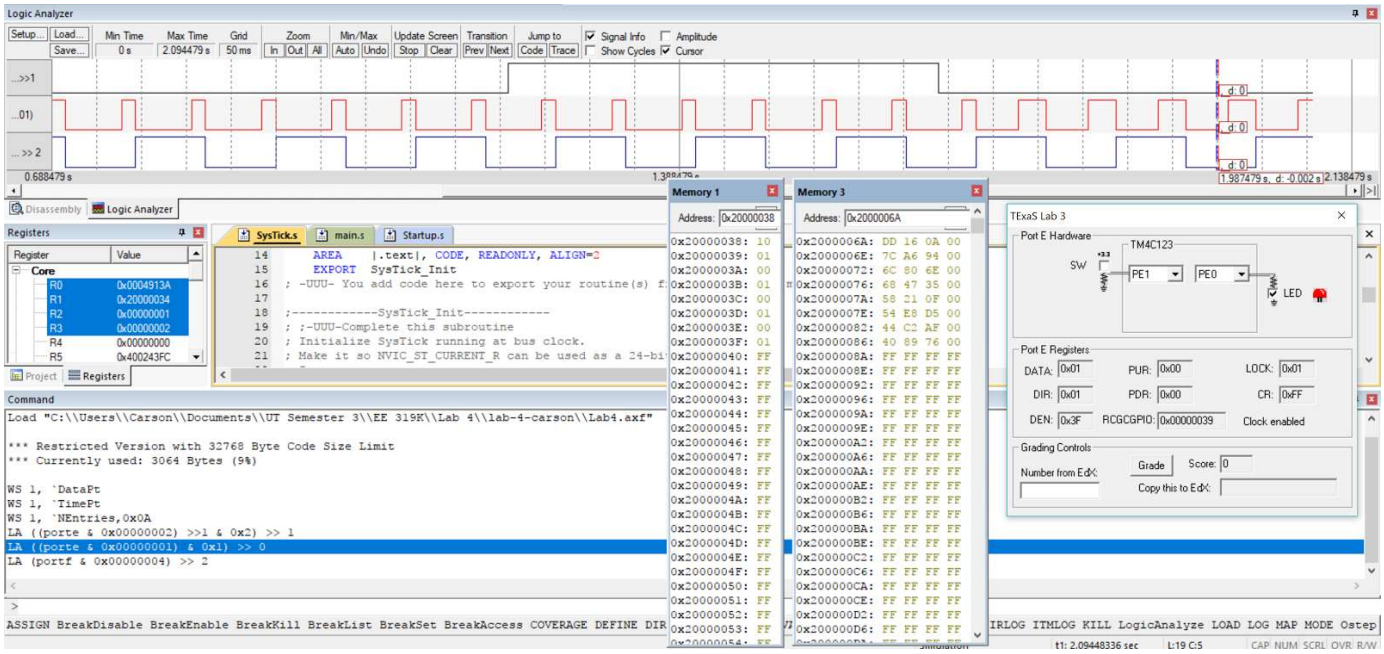
Carson Schubert
Swathi Dochibhotla

# Lab 4: Debugging Methods

**Simulator:**




**Code:**

```
;****************** main.s ***************
; Program written by: Carson Schubert and Swathi Dochibhotla
; Date Created: 2/14/2017
; Last Modified: 10/8/2017
; Brief description of the program
;   The LED toggles at 8 Hz and a varying duty-cycle
;   Repeat the functionality from Lab3 but now we want you to
;   insert debugging instruments which gather data (state and
timing)
;   to verify that the system is functioning as expected.
; Hardware connections (External: One button and one LED)
;  PE1 is Button input  (1 means pressed, 0 means not pressed)
;  PE0 is LED output (1 activates external LED on protoboard)
;  PF2 is Blue LED on Launchpad used as a heartbeat
; Instrumentation data to be gathered is as follows:
; After Button(PE1) press collect one state and time entry.
; After Buttin(PE1) release, collect 7 state and
; time entries on each change in state of the LED(PE0):
```

```
; An entry is one 8-bit entry in the Data Buffer and one
; 32-bit entry in the Time Buffer
;  The Data Buffer entry (byte) content has:
;    Lower nibble is state of LED (PE0)
;    Higher nibble is state of Button (PE1)
;  The Time Buffer entry (32-bit) has:
;    24-bit value of the SysTick's Current register
(NVIC_ST_CURRENT_R)
; Note: The size of both buffers is 50 entries. Once you fill
these
;       entries you should stop collecting data
; The heartbeat is an indicator of the running of the program.
; On each iteration of the main loop of your program toggle the
; LED to indicate that your code(system) is live (not stuck or
dead).

30   31   GPIO_PORTE_DATA_R EQU 0x400243FC 32   GPIO_PORTE_DIR_R
     EQU 0x40024400 33   GPIO_PORTE_AFSEL_R EQU 0x40024420 34
GPIO_PORTE_DEN_R         EQU 0x4002451C 35   36
GPIO_PORTF_DATA_R EQU 0x400253FC 37   GPIO_PORTF_DIR_R
EQU 0x40025400 38   GPIO_PORTF_AFSEL_R EQU 0x40025420 39
GPIO_PORTF_PUR_R         EQU 0x40025510 40   GPIO_PORTF_DEN_R
     EQU 0x4002551C 41   SYSCTL_RCGCGPIO_R EQU 0x400FE608 42
GPIO_PORTF_LOCK_R EQU 0x40025520 43   GPIO_PORTF_CR_R
EQU 0x40025524 44   GPIO_LOCK_KEY            EQU 0x4C4F434B 45
46   NVIC_ST_CTRL_R           EQU 0xE000E010 47
NVIC_ST_RELOAD_R         EQU 0xE000E014 48   NVIC_ST_CURRENT_R
EQU 0xE000E018 49
50   COUNT EQU 0x4C4B4 ;1/40s delay


51
52                                ; RAM Area
53                                THUMB
54                                AREA        DATA, ALIGN=2
55                                ;-UUU-Declare  and allocate
space for your Buffers
56                                ; and any variables (like
pointers and counters) here
```

```
57                                  COUNT_OFF SPACE 4
;global variable for the amount of delay cycles to stay off
58                                  COUNT_ON SPACE 4
;global variable for the amount of delay cycles to stay on 59
DataBuffer SPACE 50


60   TimeBuffer SPACE 50*4 61   DataPt SPACE 4 62   TimePt SPACE
4 63   NEntries SPACE 4

64                                  ; ROM Area
65                                  ;IMPORT  TExaS_Init
66                                  ;IMPORT  SysTick_Init
67                                  ;-UUU-Import routine(s) from
other assembly files (like SysTick.s) here
                          AREA      |.text|, CODE, READONLY,
ALIGN=2
         THUMB
         EXPORT Start
         IMPORT TExaS_Init IMPORT SysTick_Init


 73   74   Start

        75                          ; TExaS_Init sets bus clock
        at 80 MHz
        76                          BL TExaS_Init ; voltmeter,
        scope on PD3
        77                          CPSIE I        ; TExaS
        voltmeter, scope runs on interrupts
        78                          BL PortF_Init 79         BL
        PortE_Init 80         MOV R0, #4 81      LDR R1,
        =COUNT_OFF 82    STR R0, [R1] 83     MOV R0, #1 84
        LDR R1, =COUNT_ON
        85                          STR R0, [R1]
        ;initialize with a 20% duty cycle
        86                          LDR R1, =DataPt 87   LDR R0,
        =DataBuffer
        88                          STR R0, [R1]
        ;initialize pointer to dataBuffer
```

```
        89                          LDR R1, =TimePt 90
          LDR R0, =TimeBuffer
        91                          STR R0, [R1]
        ;initialize pointer to timeBuffer
        92                          BL Debug_Init
        93                          LDR R1, =NEntries   ;set
        NEntries to 0
        94                          MOV R0, #0 95            STR
        R0, [R1] 96      MOV R10, #1 97   loop 98   reg 99
        BL ToggleHB
100                                                  ; button press
indicator

101       LDR R5, =GPIO_PORTE_DATA_R 102   LDR R6, [R5] 103
     AND R6, #0x02 104   CMP R6, #0x02
        105                         BNE Check  ;if the
                button is not currently pressed we check
                for a release case
        106                         MOV R7, #1 ;when the
                button is clicked, a 1 is stored in R7
                107      CMP R10, #1
108       BNE noRecord 109              BL Debug_Capture 110
     SUB R10, #1 111   noRecord
112                B         sts       ;skips the release case
check while the button is held

113   Check 114      CMP R7, #1 115     BNE sts
116       BL CycleChange ;change the duty cycle if the
button was just released 117            MOV R10, #1

118       MOV R9, #7 119   120   sts LDR R1, =COUNT_OFF 121
     LDR R2, [R1]
        122                         MOV R3, R2
                ;store current COUNT_OFF value in R3
                for later use
```

```
        123                             CMP R3, #5 ;check
                if duty cycle is 0% (always off) 124
                    BNE gud
        125                             BL TurnOff  ;make sure
                light is off if duty cycle is 0%
        126                             gud CMP R2, #0
                ;reset condition codes
        127                             test BEQ try  ;check if we
                have more delay cycles to execute and exits loop
                if done 128   do              LDR R0, =COUNT
                ;this loop executes a delay of 1/40s (derived
                from 8Hz toggle) 129   wait SUBS R0, #0x01

130       BNE wait 131        SUBS R2, #1 132   B test 133

134   try   ;check for edge cases and toggle if
necessary 135       CMP R3, #5
        136                             BNE not0     ;if we are at
                a duty cycle of 0%, skip COUNT_ON code and
                toggle
    altogether
        137                             CMP R9, #0 138         BEQ
                wDone 139     BL Debug_Capture 140           SUB
                R9, #1 141    B          loop 142   wDone 143
                    MOV R9, #0

144       B loop 145   not0 146              CMP R3, #0

147       BEQ go   ;if we are at a duty cycle of
100%, skip toggle 148    BL Toggle

149   150   go          LDR R1, =COUNT_ON 151   LDR R2, [R1]
        152                             MOV R3, R2
                ;store current COUNT_ON value in R3
                for later use
        153                             CMP R3, #5 ;check
                if duty cycle is 100% (always on) 154
                    BNE st
```

```
        155                         BL TurnOn   ;make sure
        light is on if duty cycle is 100%
        156                         st   CMP R2, #0
        ;reset condition codes
        157                         tet BEQ try2 ;check if we
        habe more delay cycles to execute and exits loop
        if done
        158                         LDR R0, =COUNT    ;executes
        a delay of 1/40s


159   wait2 SUBS R0, #1 160             BNE wait2 161   SUBS R2,
#1 162   B          tet 163

164   try2 ;check for edge cases and toggle if
neccessary 165      CMP R3, #5
166       BNE not100 ;if we are at a duty cycle
of 100%, skip toggle 167   CMP R9, #0 168
BEQ weDone


169       BL Debug_Capture 170          SUB R9, #1 171   B
     loop 172   weDone 173         MOV R9, #0 174   B
loop 175   not100 176         BL Toggle 177   B             loop
178   179   Debug_Init 180         LDR R0, =DataPt 181     LDR
R5, [R0] 182                  LDR R1, =TimePt 183   LDR R6, [R1]
184                 MOV R2, #0xFF

185                       MOV R3, #0xFFFFFFFF               ; R2
= default value for data buffer, R3 = default for time buffer
186                       MOV R4, #50 ; R4 = loop counter (50 bc
array sizes are 50)
187   loadNulls 188               STRB R2, [R5]
189       STR R3, [R6] ;store the default values
at the current pointer 190   ADD R5, #1
191       ADD R6, #4          ;increment poiner based on data
size (8 bit for data, 32 bit for time) 192        SUB R4, #1
;decrement loop counter
193                       CMP R4, #0
194                       BNE loadNulls               ;loop
again if counter is not 0
```

```
195                          MOV R7, LR
196                          BL SysTick_Init
;initialize sysTick
197                          MOV LR, R7 198           BX LR 199
200   Debug_Capture 201        PUSH{R0-R5} 202   203          LDR
R1, =NEntries 204         LDR R0, [R1] 205               CMP R0,
#50
206                          BEQ Done      ; if there are already
50 entries, skip adding more
207                          ADD R0, #1
208                          STR R0, [R1]  ; increment number of
entries
209       LDR R1, =GPIO_PORTE_DATA_R 210   LDR R0, [R1] 211
     LDR R3, =NVIC_ST_CURRENT_R
212                          LDR R2, [R3] ; read Port E data and
SysTick data
213                          MOV R4, R0  ; copy Port E data into R4
214                  AND R0, #0x01
215                          AND R4, #0x02           ;mask for bit
1 in R4, mask for bit 0 in R0
216                          LSL R4, #3    ;shift bit 1 to bit 4 in
R4
217                          ORR R0, R4   ;put the contents of bit
4 in R4 into R0
218                          LDR R4, =DataPt 219   LDR R5, [R4]
220                          STRB R0, [R5]           ;store this
entry in the data buffer
221                          ADD R5, #1
222                          STR R5, [R4]  ;increment data buffer
pointer
223                          LDR R4, =TimePt 224   LDR R5, [R4]
225                          STR R2, [R5]  ;store time data in time
buffer
226                          ADD R5, #4
227                          STR R5, [R4]  ;increment time buffer
pointer
228                          Done 229      POP{R0-R5} 230   BX LR
231   232   CycleChange
233           ;this subroutine will change the duty cycle by 20%
```

```
234           LDR R1, =COUNT_OFF 235              LDR R2, [R1]
236            CMP R2, #0
        237                           BNE norm     ;checks if
                COUNT_OFF needs to be wrapped around from 0
        238                           MOV R2, #5 239         B
                    mem 240   norm SUB R2, #1 241   mem STR R2,
                [R1] 242                LDR R1, =COUNT_ON 243
                LDR R2, [R1] 244        CMP R2, #5
        245                           BNE nor      ;checks if
                COUNT_ON needs to be wrapped around from 5
        246                           MOV R2, #0 247         B
                    meme 248   nor ADD R2, #1 249   meme STR R2,
                [R1]
        250                           MOV R7, #0 ;resets the
                release case register
        251                           BX LR 252   253   Toggle
254           ;toggles the LED on PE1
255           LDR R1, =GPIO_PORTE_DATA_R 256     LDR R0, [R1]
257                 EOR R0, #0x01 258        STR R0, [R1] 259
     CMP R9, #0 260                BEQ beDone
261                 MOV R11, LR
262                      BL Debug_Capture
263                 MOV LR, R11
264       SUB R9, #1 265     BX LR 266   beDone 267   MOV R9, #0
268   BX LR 269   270   ToggleHB 271   LDR R1,
=GPIO_PORTF_DATA_R 272             LDR R0, [R1] 273         EOR
R0, #0x04 274            STR R0, [R1] 275              BX LR 276
277   TurnOn
278           ;turns PE1 on
279           LDR R1, =GPIO_PORTE_DATA_R 280     LDR R0, [R1]
281                 ORR R0, #0x01 282        STR R0, [R1] 283
     BX LR 284   285   TurnOff
286                      ;turns PE1 off
287                      LDR R1, =GPIO_PORTE_DATA_R
288                      LDR R0, [R1] 289        AND R0, #0x00
290                 STR R0, [R1] 291         BX LR 292   293
PortF_Init
294                      LDR R1, =SYSCTL_RCGCGPIO_R
;enable clock for Port F
```

```
295                          LDR R0, [R1] 296         ORR R0, R0,
#0x20 297       STR R0, [R1]
298                          NOP                ;wait for
stabilized
299                          NOP 300       LDR R1,
=GPIO_PORTF_LOCK_R             ;unlock lock register 301
LDR R0, =GPIO_LOCK_KEY 302               STR R0, [R1]
303                  LDR R1, =GPIO_PORTF_CR_R   ;enable commit
for Port F
304        MOV R0, #0xFF 305             STR R0, [R1] 306   LDR
R1, =GPIO_PORTF_DIR_R    ;set PF2 as output 307         LDR R0,
[R1] 308                  ORR R0, R0, #0x04 309              STR
R0, [R1] 310                    LDR R1, =GPIO_PORTF_DEN_R 311
MOV R0, #0xFF 312         STR R0, [R1] 313                    BX
LR 314   315   PortE_Init 316       LDR R1, =SYSCTL_RCGCGPIO_R
317        LDR R0, [R1] 318              ORR R0, R0, #0x10 319
STR R0, [R1] 320                     NOP 321         NOP 322
LDR R1, =GPIO_PORTE_DIR_R 323                 LDR R0, [R1] 324
ORR R0, R0, #0x01 325    STR R0, [R1] 326   LDR R1,
=GPIO_PORTE_DEN_R 327               MOV R0, #0xFF 328         STR
R0, [R1] 329                   BX LR 330
331                              ALIGN   ; make sure the end of
this section is aligned
332                              END          ; end of file
```

SYSTICK.S

```
1           ; SysTick.s
2 ; Module written by: Carson Schubert and Swathi Dochibhotla
3           ; Date Created: 2/14/2017
4           ; Last Modified: 10/8/2017
5           ; Brief Description: Initializes SysTick 6
 7   NVIC_ST_CTRL_R   EQU
                      0xE000E010
 8                    EQU
 NVIC_ST_RELOAD_R     0xE000E014
 9                    EQU
 NVIC_ST_CURRENT_R    0xE000E018
```

```
10      THUMB 11                AREA       DATA, ALIGN=2
12    ;global variables go here

13      ALIGN 14                AREA       |.text|, CODE, READONLY,
ALIGN=2

   15                           EXPORT SysTick_Init
16    ; -UUU- You add code here to export your routine(s)
from SysTick.s to main.s 17
    18          ;------------SysTick_Init------------
    19          ; ;-UUU-Complete this subroutine
    20          ; Initialize SysTick running at bus clock.
    21          ; Make it so NVIC_ST_CURRENT_R can be used as a
        24-bit time
    22          ; Input: none
    23          ; Output: none
    24          ; Modifies: R1, R0

25    SysTick_Init
       26                       ; **-UUU-**Implement this
              function****
       27                       ;disable during setup
       28                       LDR R1, =NVIC_ST_CTRL_R 29
              MOV R0, #0 30       STR R0, [R1]
       31                       ;set reloard to maximum reload
              value
       32                       LDR R1, =NVIC_ST_RELOAD_R

   33                   LDR R0, =0x00FFFFFF
       34                       STR R0, [R1]
       35                       ;writing any value to CURRENT
              will clear it
       36
```

```
        LDR R1, =NVIC_ST_CURRENT_R 37      MOV R0, #0 38
              STR R0, [R1]
     39                              ;enable SysTick with core
clock

     40                  LDR R1, =NVIC_ST_CTRL_R
          41                         MOV R0, #0x05 ; enable with no
              interrupts
          42                         STR R0, [R1] ; ENABLE and
              CLK_SRC bits set
          43                         BX LR       ; return

  44
          45                         ALIGN       ; make sure the end
              of this section is aligned
          46                         END         ; end of file
```

**Estimated Intrusiveness:**
Instructions in Debug_Capture: 28
Therefore, takes 56 cycles to execute. At 12.5ns per cycle, this comes out to 700ns for execution.
Since the main loop takes ⅛ s to run (since we are blinking at 8hz), the intrusiveness is:
((700ns * 2)/.125s)*100 = **.00112%.**

**Debugging Results:**

Duty Cycle 1:
Time from press to release: 127.03ms
First 6 time differences: 95.782125ms
LED frequency: 7.66 Hz

Duty Cycle 2:
Time from press to release: 159.66405ms
First 6 time differences: 95.363975ms
LED frequency: 7.63 Hz

Duty Cycle 3:
Time from press to release: 47.0692
First 6 time differences: 110.7284ms
LED frequency: 8.86 Hz

Duty Cycle 4:
Time from press to release: 117.6683ms
First 6 time differences: 98.59005ms
LED frequency: 7.89 Hz

Duty Cycle 5:
Time from press to release: 127.03ms
First 6 time differences: 95.782125ms
LED frequency: 7.66 Hz