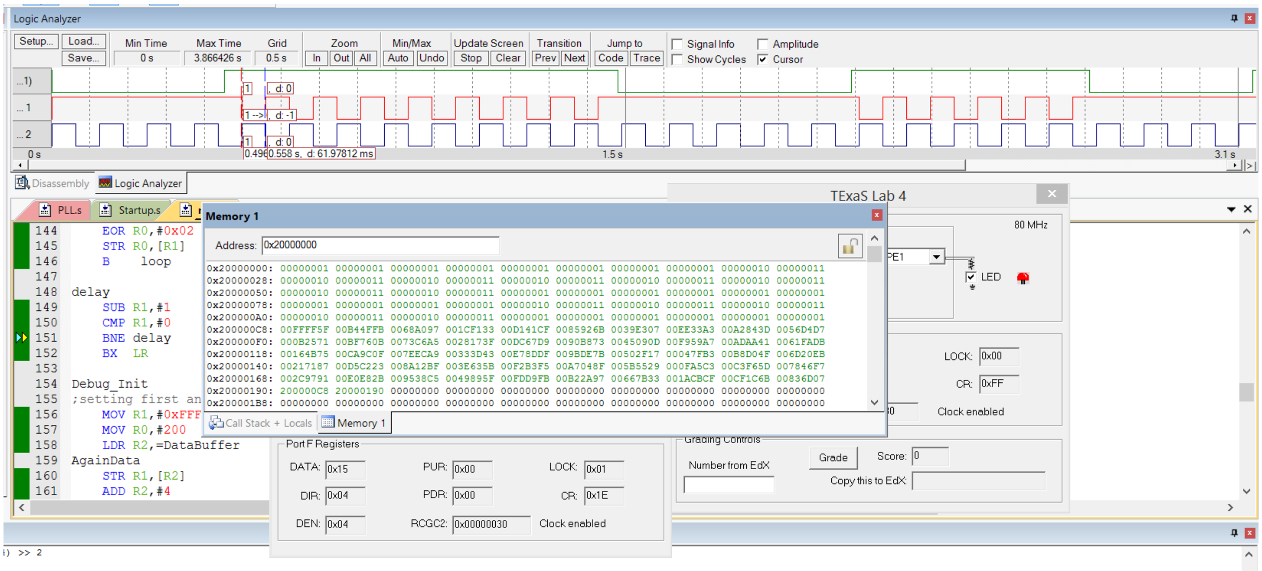

```
    EOR R0,#0x02
    STR R0,[R1]
    B    loop

delay
    SUB R1,#1
    CMP R1,#0
    BNE delay
    BX  LR

Debug_Init
;setting first an
    MOV R1,#0xFFF
    MOV R0,#200
    LDR R2,=DataBuffer
AgainData
    STR R1,[R2]
    ADD R2,#4
```

Memory 1

Address: 0x20000000

```
0x20000000: 00000001 00000001 00000001 00000001 00000001 00000001 00000001 00000001 00000010 00000011
0x20000028: 00000010 00000011 00000010 00000011 00000010 00000011 00000010 00000011 00000010 00000011
0x20000050: 00000010 00000011 00000010 00000011 00000001 00000001 00000001 00000001 00000001 00000001
0x20000078: 00000001 00000001 00000001 00000001 00000010 00000011 00000010 00000011 00000010 00000011
0x200000A0: 00000010 00000011 00000010 00000011 00000001 00000001 00000001 00000001 00000001 00000001
0x200000C8: 00FFFF5F 00B44FFB 0068A097 001CF133 00D141CF 0085926B 0039E307 00EE33A3 00A2843D 0056D4D7
0x200000F0: 000B2571 00BF760B 0073C6A5 0028173F 00DC67D9 0090B873 0045090D 00F959A7 00ADAA41 0061FADB
0x20000118: 00164B75 00CA9C0F 007EECA9 00333D43 00E78DDF 009BDE7B 00502F17 00047FB3 00B8D04F 006D20EB
0x20000140: 00217187 00D5C223 008A12BF 003E635B 00F2B3F5 00A7048F 005B5529 000FA5C3 00C3F65D 007846F7
0x20000168: 002C9791 00E0E82B 009538C5 0049895F 00FDD9FB 00B22A97 00667B33 001ACBCF 00CF1C6B 00836D07
0x20000190: 200000C8 20000190 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x200001B8: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

Port F Registers

DATA: 0x15 PUR: 0x00 LOCK: 0x01

DIR: 0x04 PDR: 0x00 CR: 0x1E

DEN: 0x04 RCGC2: 0x00000030 Clock enabled

LOCK: 0x00

CR: 0xFF

Clock enabled

Score: 0

```
) >> 2
```

```
;****************** main.s ***************
; Program written by: put your names here
; Date Created: 8/25/2013
; Last Modified: 10/6/2013
; Section 1-2pm     TA: Saugata Bhattacharyya
; Lab number: 4
; Brief description of the program
;   If the switch is presses, the LED toggles at 8 Hz
; Hardware connections
;  PE0 is switch input  (1 means pressed, 0 means not pressed)
;  PE1 is LED output (1 activates external LED on protoboard)
;Overall functionality of this system is the similar to Lab 3, with
four changes:
;1-  activate the PLL to run at 80 MHz (12.5ns bus cycle time)
;2-  initialize SysTick with RELOAD 0x00FFFFFF
;3-  add a heartbeat to PF2 that toggles every time through loop,
every 62ms
;4-  add debugging dump of input, output, and time
; Operation
;    1) Make PE1 an output and make PE0 an input.
;    2) The system starts with the LED on (make PE1 =1).
;   3) Wait about 62 ms
;   4) If the switch is pressed (PE0 is 1), then toggle the LED once,
else turn the LED on.
;   5) Steps 3 and 4 are repeated over and over


SYSCTL_RCGC2_R          EQU 0x400FE108
GPIO_PORTE_DATA_R       EQU 0x400243FC
GPIO_PORTE_DIR_R        EQU 0x40024400
GPIO_PORTE_AFSEL_R      EQU 0x40024420
GPIO_PORTE_PUR_R        EQU 0x40024510
GPIO_PORTE_DEN_R        EQU 0x4002451C
GPIO_PORTF_DATA_R       EQU 0x400253FC
GPIO_PORTF_DIR_R        EQU 0x40025400
GPIO_PORTF_AFSEL_R      EQU 0x40025420
GPIO_PORTF_DEN_R        EQU 0x4002551C
NVIC_ST_CURRENT_R       EQU 0xE000E018
NVIC_ST_CTRL_R          EQU 0xE000E010
NVIC_ST_RELOAD_R        EQU 0xE000E014


                AREA   DATA, ALIGN=2
;You MUST use these two buffers and two variables
;You MUST not change their names
DataBuffer      SPACE  200
TimeBuffer      SPACE  200
DataPt          SPACE  4
TimePt          SPACE  4
;These names MUST be exported
     EXPORT DataBuffer
     EXPORT TimeBuffer
```

```
      EXPORT DataPt
      EXPORT TimePt


      ALIGN
      AREA    |.text|, CODE, READONLY, ALIGN=2
      THUMB
      EXPORT  Start
      IMPORT  PLL_Init

Start
      BL   Debug_Init
      BL  PLL_Init    ; running at 80 MHz

;turn port clock on for port E and F
      LDR  R0,=SYSCTL_RCGC2_R
      LDR  R1,[R0]
      ORR  R1,#0x30        ;turn on port E clock bit high
      STR  R1,[R0]
;wait 2 cycles
      NOP
      NOP
;set DIR = 0/1 for E
      LDR  R0,=GPIO_PORTE_DIR_R
      LDR  R1,[R0]
      ORR  R1,#0x02        ;set PE1 as an output bit 1
      AND  R1,#0xFE        ;set PE0 as an input bit 0
      STR  R1,[R0]
;set DIR = 0/1 for F
      LDR  R0,=GPIO_PORTF_DIR_R
      LDR  R1,[R0]
      ORR  R1,#0x04        ;set PF2 as an output bit 1
      STR  R1,[R0]
;turn off AFSEL for E
      LDR  R0,=GPIO_PORTE_AFSEL_R
      LDR  R1,[R0]
      AND  R1,#0xFC        ;set AFSEL bits to 0
      STR  R1,[R0]
;turn off AFSEL for F
      LDR  R0,=GPIO_PORTF_AFSEL_R
      LDR  R1,[R0]
      AND  R1,#0xFB        ;set AFSEL bits to 0
      STR  R1,[R0]
;enable DEN for E
      LDR  R0,=GPIO_PORTE_DEN_R
      LDR  R1,[R0]
      ORR  R1,#0x03        ;set PE1 and PE0 digital enable bits to 1
      STR  R1,[R0]
;enable DEN for F
      LDR  R0,=GPIO_PORTF_DEN_R
      LDR  R1,[R0]
      ORR  R1,#0x04        ;set PF2 digital enable bit to 1
      STR  R1,[R0]
;enable LED for E
      LDR  R1,=GPIO_PORTE_DATA_R
      LDR  R0,[R1]
      ORR  R0,#0x02        ;set PE1 LED bit high to turn on LED
      STR  R0,[R1]
;enable LED for F
      LDR  R1,=GPIO_PORTF_DATA_R
      LDR  R0,[R1]
      ORR  R0,#0x04        ;set PF2 LED bit high to turn on LED
      STR  R0,[R1]



loop
      BL   Debug_Capture
;set delay
      MOV  R1,#16000  ;set R1 for 1ms delay
      MOV  R2,#62          ;set R2 to 62 for a 62ms delay
      MUL  R1,R2
      BL   delay
;enable heartbeat
      LDR  R1,=GPIO_PORTF_DATA_R
      LDR  R0,[R1]
      EOR  R0,#0x04   ;toggle the LED bit on/off using logical EOR
      STR  R0,[R1]
program
      LDR  R1,=GPIO_PORTE_DATA_R ;read PE0
      LDR  R0,[R1]
      AND  R0,#0x01   ;isolate bit 1 so read PE0
      CMP  R0,#0x01   ;see if the switched is pressed or not
      BEQ  pressed         ;if 1, then switch is pressed so branch to
pressed
;not pressed
      LDR  R1,=GPIO_PORTE_DATA_R
      LDR  R0,[R1]
      ORR  R0,#0x02   ;keep the LED bit high
      STR  R0,[R1]
    B    loop

pressed
      LDR  R1,=GPIO_PORTE_DATA_R
      LDR  R0,[R1]
      EOR  R0,#0x02   ;toggle the LED bit on/off using logical EOR
      STR  R0,[R1]
    B    loop

delay
      SUB  R1,#1      ;decrement the counter
      CMP  R1,#0      ;see if counter has hit 0
      BNE  delay      ;if not, retry
```

```
     BX    LR               ;delay has been reached, so branch to main
program

Debug_Init
;setting first and second buffers to 0xFFFF.FFFF
     MOV R1,#0xFFFFFFFF
     MOV   R0,#200                          ;set R0 as a counter
     LDR R2,=DataBuffer                ;R2=DataBuffer pointer
AgainData
     STR R1,[R2]                            ;set DataBuffer to value
0xFFFF.FFFF
     ADD   R2,#4                       ;increment the pointer
     SUB   R0,#4                       ;decrement counter
     CMP   R0,#0
     BNE   AgainData                   ;if counter is not yet 0, repeat

     MOV   R0,#200                          ;set R0 as a counter
     LDR R3,=TimeBuffer                ;R3=TimeBuffer pointer
AgainTime
     STR   R1,[R3]                          ;set TimeBuffer to value
0xFFFF.FFFF
     ADD   R3,#4                       ;increment the pointer
     SUB   R0,#4                       ;decrement counter
     CMP   R0,#0
     BNE   AgainTime                   ;if counter is not yet 0, repeat
;initialize the two pointers to beginning of each buffer
     LDR R0,=DataPt                    ;get current value of pointer
     LDR   R1,=DataBuffer              ;get first address of DataBuffer
     STR R1,[R0]                            ;DataPt now points to first
element of DataBuffer
     LDR R0,=TimePt                    ;get current value of pointer
     LDR   R2,=TimeBuffer              ;get first address of TimeBuffer
     STR R2,[R0]                            ;TimePT now points to first
element of TimeBuffer
;activate SysTick Timer
SysTick_Init
     LDR R1,=NVIC_ST_CTRL_R            ; 1.disable timer, clear ctrl
     MOV R0,#0
     STR R0,[R1]
     LDR R1,=NVIC_ST_RELOAD_R     ; 2. load reload value
     LDR R0,=0x00FFFFFF
     STR R0,[R1]
     LDR R1,=NVIC_ST_CURRENT_R    ; 3. clear current
     MOV R0,#0
     STR R0,[R1]
     LDR R1,=NVIC_ST_CTRL_R            ; 4. enable systick with core
source
     MOV R0,#0x05
     STR R0,[R1]
     BX LR

Debug_Capture
```

```
     PUSH {R0-R3,R12}                 ;save registers
;check if buffer full, if full return to program
     LDR R0, =DataPt                  ;adress of datapt
     LDR R1, [R0]                     ;R1= value datapt
     MOV R2, #0xFFFFFFFF              ;compare to value we wrote into
buffer
     LDR   R3,[R1]
     CMP R3, R2
     BNE goback
;check time buffer
     LDR R0, =TimePt                  ;adress of timept
     LDR R1, [R0]                     ;R1= value timpt
     MOV R2, #0xFFFFFFFF              ;compare to value we wrote into
buffer
     LDR   R3,[R1]
     CMP R3, R2
     BNE goback
;dump PortE data
     LDR R0, =GPIO_PORTE_DATA_R
     LDR R1, [R0]                     ;R1=contents PortE data
     AND R1, #0x03                    ;capture bits 1,0
     MOV R2, R1                       ;R1=R2
     AND R1, #0x01                    ;now R1=bit0
     LSL R1, #4                       ;shift bit0 to bit4 position
     AND R2, #0x02                    ;now R2=bit1
     LSR R2, #1                       ;shift bit1 to bit0 position
     ORR R3, R1, R2                   ;R3=shifted data
     LDR R0, =DataPt                  ;content of datapt is adress of
databuffer
     LDR R1, [R0]                     ;R1= adress within DataBuffer
     STR R3, [R1]                     ;dump portE info into DataBuffer
     ADD R1, #4                       ;increment adress
     STR R1, [R0]                     ;store incremented address to
pointer
;dump time data
     LDR R1, =NVIC_ST_CURRENT_R
     LDR R2, [R1]                     ;R2=current time
     LDR R3, =TimePt
     LDR R12, [R3]                    ;R4=address within TimeBuffer
     STR R2, [R12]                    ;dump time value into TimeBuffer
     ADD R12, #4                           ;increment time pointer
     STR R12, [R3]                    ;store incremented address to
pointer
     POP {R0-R3,R12}
goback
     BX LR                            ;return to program


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

Percent Overhead Calculation

```
36 instructions x 2 = ~72 cycles
72 cycles x 12.5 ns = 900 ns

62 ms (delay) + ((14 instructions x 2)cycles x 12.5 ns)= 62.35 ms

% Overhead = 0.900/62.35 = ~1.44% intrusiveness
```

LED Period Calculation

```
0x00B44FFB - 0x0068A097 = 4,960,100
4960100 x 12.5 ns = 62.00125 ms
```