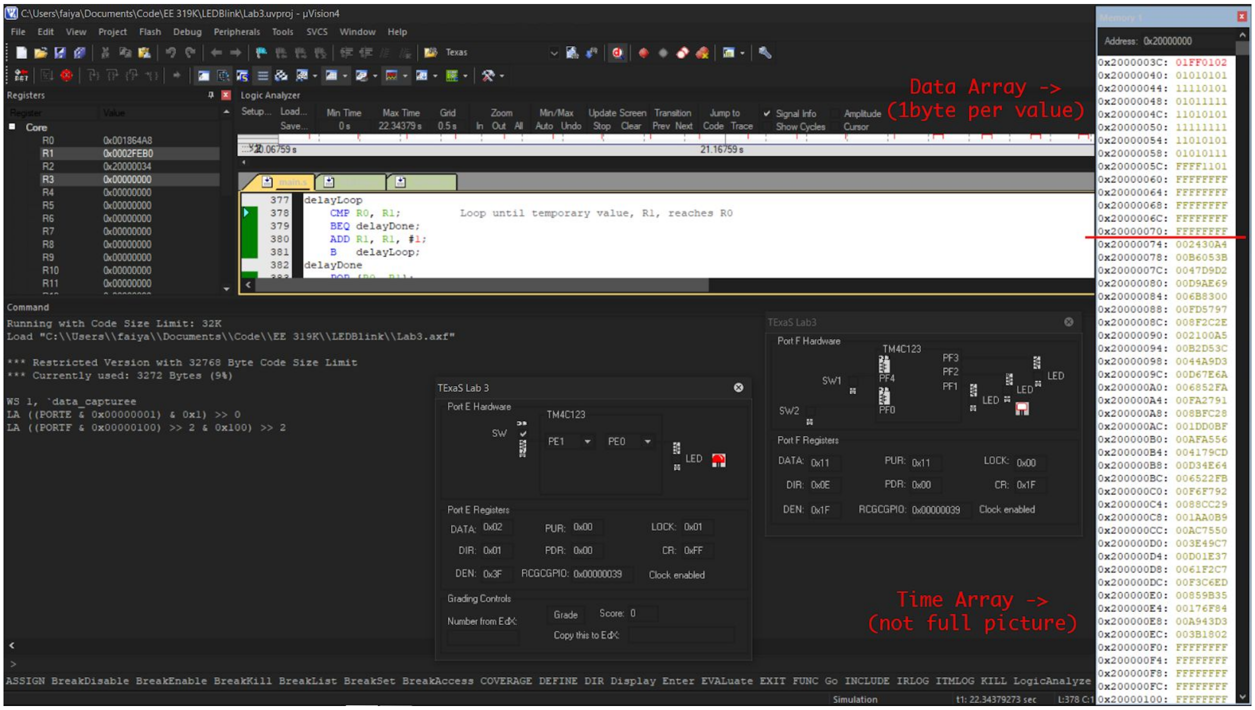

C:\Users\faiya\Documents\Code\EE 319K\LEDBlink\Lab3.uvproj - µVision4

Register	Value
Core	
R0	0x001864A8
R1	0x0002FEB0
R2	0x20000034
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000

```
delayLoop
    CMP R0, R1;         Loop until temporary value, R1, reaches R0
    BEQ delayDone;
    ADD R1, R1, #1;
    B   delayLoop;
delayDone
```

```
Running with Code Size Limit: 32K
Load "C:\\Users\\faiya\\Documents\\Code\\EE 319K\\LEDBlink\\Lab3.axf"

*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 3272 Bytes (9%)

WS 1, `data_capturee
LA ((PORTE & 0x00000001) & 0x1) >> 0
LA ((PORTF & 0x00000100) >> 2 & 0x100) >> 2
```

```
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE DEFINE DIR Display Enter EVALuate EXIT FUNC Go INCLUDE IRLOG ITMLOG KILL LogicAnalyze
```

TExaS Lab 3

Port E Hardware

Port E Registers

DATA: 0x02 PUR: 0x00 LOCK: 0x01
DIR: 0x01 PDR: 0x00 CR: 0xFF
DEN: 0x3F RCGCGPIO: 0x00000039 Clock enabled

Grading Controls

Number from EdX:
Grade Score: 0
Copy this to EdX:

TExaS Lab3

Port F Hardware

Port F Registers

DATA: 0x11 PUR: 0x11 LOCK: 0x00
DIR: 0x0E PDR: 0x00 CR: 0x1F
DEN: 0x1F RCGCGPIO: 0x00000039 Clock enabled

Data Array ->
(1byte per value)

Time Array ->
(not full picture)

Memory 1

Address: 0x20000000

```
0x2000003C: 01FF0102
0x20000040: 01010101
0x20000044: 11110101
0x20000048: 01011111
0x2000004C: 11010101
0x20000050: 11111111
0x20000054: 11010101
0x20000058: 01010111
0x2000005C: FFFF1101
0x20000060: FFFFFFFF
0x20000064: FFFFFFFF
0x20000068: FFFFFFFF
0x2000006C: FFFFFFFF
0x20000070: FFFFFFFF
0x20000074: 002430A4
0x20000078: 00B6053B
0x2000007C: 0047D9D2
0x20000080: 00D9AE69
0x20000084: 006B8300
0x20000088: 00FD5797
0x2000008C: 008F2C2E
0x20000090: 002100A5
0x20000094: 00B2D53C
0x20000098: 0044A9D3
0x2000009C: 00D67E6A
0x200000A0: 006852FA
0x200000A4: 00FA2791
0x200000A8: 008BFC28
0x200000AC: 001DD0BF
0x200000B0: 00AFA556
0x200000B4: 004179CD
0x200000B8: 00D34E64
0x200000BC: 006522FB
0x200000C0: 00F6F792
0x200000C4: 0088CC29
0x200000C8: 001AA0B9
0x200000CC: 00AC7550
0x200000D0: 003E49C7
0x200000D4: 00D01E37
0x200000D8: 0061F2C7
0x200000DC: 00F3C6ED
0x200000E0: 00859B35
0x200000E4: 00176F84
0x200000E8: 00A943D3
0x200000EC: 003B1802
0x200000F0: FFFFFFFF
0x200000F4: FFFFFFFF
0x200000F8: FFFFFFFF
0x200000FC: FFFFFFFF
0x20000100: FFFFFFFF
```

Simulation t1: 22.34379273 sec L:378 C:1

;****************** main.s ***************

; Program written by: Faiyaz Mostofa & Zaine Ahmed

; Date Created: 2/4/2017

; Last Modified: 2/28/2018

; Brief description of the program:

;   A collection of operations are done in this program (at once):

;   - An internal Blue LED blinks repeatedly indicating that the program is running

;   - A Red LED blinks on and off based on different patterns which can be changed by a switch (8Hz).

;   The LED will cycle through different patterns as specified below:

;   :on for (1/40s) -> on for (1/20s) -> on for (3/40s) -> on for (1/10s) -> always on -> always off -> loop:

;   The delays for the blinking is handled by SysTick.s

;   - The Red LED also has a breathing function can be activated by holding the internal button @ PF4 on the micro-controller.

;   The delays for the breathing is handled by the delay function @ the end of this file.

;   - A debugging capture tool that runs 3rd time the loop finishes.

; Hardware connections (External: One button and one LED)

; PE1 is Button input (1 means pressed, 0 means not pressed)

; PE0 is a Red LED output (1 activates external LED on protoboard)

; PF2 is a Green LED output (1 activates external LED on protoboard)

; PF4 is builtin button SW1 on Launchpad (Internal)

;   Negative Logic (0 means pressed, 1 means not pressed)

```
; PortE device registers
GPIO_PORTE_DATA_R  EQU 0x400243FC
GPIO_PORTE_DIR_R   EQU 0x40024400
GPIO_PORTE_AFSEL_R EQU 0x40024420
GPIO_PORTE_DEN_R   EQU 0x4002451C
; PortF device registers
GPIO_PORTF_DATA_R  EQU 0x400253FC
GPIO_PORTF_DIR_R   EQU 0x40025400
GPIO_PORTF_AFSEL_R EQU 0x40025420
GPIO_PORTF_PUR_R   EQU 0x40025510
GPIO_PORTF_DEN_R   EQU 0x4002551C
GPIO_PORTF_LOCK_R  EQU 0x40025520
GPIO_PORTF_CR_R    EQU 0x40025524
GPIO_LOCK_KEY      EQU 0x4C4F434B    ; Unlocks the GPIO_CR register
SYSCTL_RCGCGPIO_R  EQU 0x400FE608
; System Clock reigsters
NVIC_ST_CURRENT_R     EQU 0xE000E018

;Variables that hold the maximum values
MAX_DELAY              EQU 0x1864A8        ; The interval size of the delays (in cycles)
                                                          ;        0x0c (in 10ms)
BREATHE_DELAY_MAX  EQU 0x5E00        ; The delay required

    IMPORT TExaS_Init
          IMPORT SysTick_Init

    THUMB
;------------Global Variables-----------------------------------------------------------------
    AREA    DATA, ALIGN=2

         ;Blinking variables
delay_inc                 SPACE 4      ; how to increment the delays when we need to change
them (1/5 of MAX_DELAY)
delay_off                 SPACE 4      ; how long the LED will stay off (in cycles)
delay_on                  SPACE 4      ; how long the LED will stay on (in cycles)
prev_button_state SPACE     1          ; captures whether a button has been released or pushed
green_counter   SPACE 1      ; it counts everytime the main loop is run and toggles the blue LED
after a certain time is met.

         ;Debuggin variables
data_capture   SPACE 50      ; Array of 50 8-byte numbers
time_capture   SPACE 200    ; Array of 50 32-byte numbers
```

```
debug_capture_counter         SPACE          1         ; it counts everytime the main loop is run
and captures debugging data after a certain amount of loops

NEntries                  SPACE 1                  ; Number of entries in either array

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

;---------Main Code---------------------------------------------------------------------------
Start
 ; TExaS_Init sets bus clock at 80 MHz
   BL  TExaS_Init ; voltmeter, scope on PD3
   BL  Debug_Init ;      Initializes the Debugging Tools
         BL       SysTick_Init;   Initializes the SysTick (method in SysTick.s)
         BL       Ports_Init;                  Initializes Ports E,F
                                                         ; PE0 = Red LED output
                                                         ; PE1 = positive logic Input (Switch)
                                                         ; PF2 = Blue LED output
                                                         ; PF4 = Hold switch for breathing functionality

; Setting up variables
Configure
         LDR     R1, =MAX_DELAY;
         MOV    R2, #5;
         UDIV R2, R1, R2;                split the max delay into 5 equal sections
         LDR     R1, =delay_inc;
         STR     R2, [R1];                         delay_inc = (MAX_DELAY / 5)

         LDR     R1, =delay_inc;
         LDR     R2, [R1];
         MOV    R3, #4;
         MUL    R2, R2, R3;
         LDR     R1, =delay_off;
         STR     R2, [R1];                         Default: the delay_off starts @ 4/5 of the
MAX_DELAY

         LDR     R1, =delay_inc;
         LDR     R2, [R1];
         LDR R1, =delay_on;
         STR     R2, [R1];                         Default: the delay_on on starts @ 1/5 of the
MAX_DELAY
```

```
        LDR    R1, =green_counter;
        MOV    R2, #0;
        STRB R2, [R1];                    Initially set the green_counter to 0
        LDR    R1, =debug_capture_counter;
        MOV    R2, #0;
        STRB R2, [R1];                    Initially set the debug_counter to 0

   CPSIE  I    ; TExaS voltmeter, scope runs on interrupts

; The main loop engine
main_loop

        BL     Check_Debug;          ; Check if we need to record debugging statistics
        BL     Check_Green           ; Check whether to toggle the green LED on or not
        BL     Check_Breathe         ; Check if whether we need to make the LED Breathe

;If a button @ PE1 is pushed, increment the blinking pattern
Blink_ifPushed
        LDR    R1, =GPIO_PORTE_DATA_R;
        LDR R2, =prev_button_state;
        LDRB R2, [R2];
        LDR    R3, [R1];                          <- R3 holds the data from the PortE data
register
        AND    R3, R3, #0x02;                     Check whether the button has been pushed
or not
        CMP R3, R2;                               <- Check if the button is in the same state
as before
        BEQ Blink;
        LDR    R2, =prev_button_state;
        STRB R3, [R2];
        ;BL    Debug_Capture;
; If the button is pushed, set PE4 to 1
        CMP    R3, #0x00;                 If the button is pushed
        BNE    Blink_incrementDuty;
        B Blink;
Blink_incrementDuty
; Incrementing the duty time
        LDR    R2, =delay_inc;
        LDR    R2, [R2];
        LDR    R1, =delay_off;
        LDR    R3, [R1];
        SUB    R3, R3, R2;                        Decrement the off time
        STR    R3, [R1];
```

```
	LDR	R1, =delay_on;
	LDR	R3, [R1];
	ADD	R3, R3, R2;			Increment the on time
	STR R3, [R1];
;Check if the duty time needs to be reset (always on -> always off)
	LDR	R1, =delay_off;
	LDR	R2, [R1];
	CMP	R2, #0;
	BPL	Blink;			If the the off time is < 0 (off < 0%, on > 100%),
reset the values to off = 100%, on = 0%
	LDR	R2, =MAX_DELAY;
	LDR R1, =delay_off;
	STR R2, [R1];
	LDR	R1, =delay_on;
	MOV	R2, #0;			Reset the on time to 0 (light is always off)
	STR	R2, [R1];
Blink
; Turn off the light and wait
	LDR	R1, =GPIO_PORTE_DATA_R;
	LDR	R2, [R1];
	BIC	R2, #0x01;
	STR	R2, [R1];
	LDR R2, =delay_off;
	LDR R0, [R2];
	BL	delay;	;BL	delay;	Delay the program for a amount of time specified in R0
; Turn on the light and wait
	LDR	R1, =GPIO_PORTE_DATA_R;
	LDR	R2, [R1];
	ORR	R2, #0x01;
	STR	R2, [R1];
	LDR R2, =delay_on;
	LDR R0, [R2];
	BL	delay	;BL	delay

   B   main_loop
;-----------------------------------------------------------------------------------------------
Breathe_Start
; a subroutine that handles all the breathing functionality by completely reworking everything
	PUSH {R0-R7};
	PUSH {R8, LR};

	; Setting up variables
	LDR R0, =GPIO_PORTE_DATA_R;
```

```
        LDR     R9, =GPIO_PORTF_DATA_R;
        LDR     R2, =BREATHE_DELAY_MAX;
        MOV     R3, #500;
        UDIV R4, R2, R3;                  The increments of the delay
        ADD     R5, R2, #0;                       Default: off for 4/5 of 80Hz
        MOV     R6, #0;                           Default: on for 1/5 of 80Hz
        ADD     R7, R2, #0;

Breathe_loop
        LDR     R1, [R9];                         <- R1 holds the data from the data register
Breathe_ifPushed
        AND     R3, R1, #0x10;                    Check whether the button has been pushed
or not
        CMP     R3, #0x10;                        Keep Breathing until the button is released.
        BNE     Breathe_incrementDuty;
        B Breathe_Stop;

Breathe_incrementDuty
; Incrementing the duty time
        SUB     R5, R5, R4;                       Decrement the off time
        ADD     R6, R6, R4;                       Increment the on time
        CMP     R5, #0;
        BMI Breathe_Verse;
        BEQ Breathe_Verse;                Check if we've stopped or froze the delay of the
light (either R5 or R6 reach zero)
        CMP     R6, #0;
        BPL Breathe;
Breathe_Verse
        MOV     R3, #-1;
        MUL     R4, R4, R3;                       Once we reach a maximum, down/up or up
depending on the scenario
        SUB     R5, R5, R4;                       Decrement the off time
        ADD     R6, R6, R4;                       Increment the on time
Breathe
; Turn off the light and wait
        BIC     R1, #0x01;
        STR     R1, [R0];
        PUSH {R0, R1};
        ADD     R0, R5, #0;
        BL      delay;                            Delay the program for a amount of time
specified in R7
        POP {R0, R1};
; Turn on the light and wait
```

```
        ORR    R1, #0x01;
        STR    R1, [R0];
        PUSH {R0, R1};
        ADD    R0, R6, #0;
        BL     delay;
        POP {R0, R1};

   B   Breathe_loop

Breathe_Stop
        POP {R8,LR};
        POP {R0-R7};

        BX LR;

;-------CHECK_debug--------------------------------------------------------------------------
; Wait 5 duty cycles, then save the points in the Dubugging arrays
Check_Debug
        PUSH {R0, R1};
        PUSH {R2, LR};
        LDR    R1, =debug_capture_counter;
        LDRB R2, [R1];
        ADD    R2, R2, #1;                    debug_capture_counter++;
        STRB R2, [R1];
        CMP    R2, #6;
        BNE    Check_Debug_Leave;
        BL     Debug_Capture;                 if(debug_capture_counter == 3) capture data
        MOV    R2, #0;
        STRB R2, [R1];
Check_Debug_Leave
        POP    {R2, LR};
        POP {R0, R1};
        BX LR;

;-------CHECK_Green--------------------------------------------------------------------------
; Wait 5 duty cycles, then save the points in the Dubugging arrays
Check_Green
        PUSH {R0, R1};
        PUSH {R2, LR};
        LDR    R1, =green_counter;
        LDRB R2, [R1];
        ADD    R2, R2, #1;                    green_counter++
        STRB R2, [R1];
        CMP   R2, #3;
        BNE   Check_Green_Leave;
        BL    Toggle_Green;                 if(green_counter == 3) toggle Green LED
        MOV   R2, #0;
        STRB R2, [R1];
Check_Green_Leave
        POP   {R2, LR};
        POP {R0, R1};
        BX LR;

;-------CHECK_Breathe--------------------------------------------------------------------------
; If the button @ PF4 is pushed, Start breathing
Check_Breathe
        PUSH {R0, R1};
        PUSH {R2, LR};
        LDR   R1, =GPIO_PORTF_DATA_R;
        LDR   R2, [R1];
        AND   R2, R2, #0x10;                        Check whether the button has been pushed
or not
        CMP   R2, #0x00;
        BNE   Check_Breathe_Leave;          If SW1 is pushed, start the breathing
        BL Breathe_Start;
Check_Breathe_Leave
        POP   {R2, LR};
        POP {R0, R1};
        BX    LR;

;-------DEBUG_Init-------------------------------------------------------------------------------
   ;Initiliazing Debug Dump
Debug_Init
        PUSH {R0, R1}
        PUSH {R2, R3}

        LDR R2, =data_capture;
        LDR R3, =time_capture;              Created pointers
; Fill the data array with 0xFF (signifying empty)
        MOV R0, #50;
setting_data_capture
        SUB R0,R0, #0x01
        MOV R1, #0xFF;
        STRB R1, [R2]
        ADD R2, R2, #1;
        CMP R0, #0x0;
        BNE setting_data_capture
 ; Fill the time array with 0xFFFFFFFF (signifying empty)
        MOV R0, #50;
setting_time_capture
        SUB R0,R0, #1;
        MOV R1, #0xFFFFFFFF;
        STR R1, [R3]
        ADD R3, R3, #4;
        CMP R0, #0x0;
        BNE setting_time_capture

        POP {R2, R3}
        POP {R0, R1}
        BX LR

;-------DEBUG_CAPTURE---------------------------------------------------------------------------
; saves one data point
Debug_Capture
        PUSH {R0,R1}
        PUSH {R2,LR}
        LDR R0 , =NEntries
        LDR R1, [R0]
        CMP R1 , #50
        BHS DONE_C;                             if (the array is not full)
        ADD     R1, R1, #1;                     Add a new entry
        STRB R1, [R0];                          NEntries++;
; Record the current data entries
        LDR R0, =GPIO_PORTE_DATA_R;
        LDR     R0, [R0];
        ADD     R1, R0, #0;
        AND R0, R0, #0x01;              R0 holds the data for PE0
        AND     R1, R1, #0x02;                  R1 holds the data for PE1
        LSL R1, R1, #3;                         Move PE1 to PE4
        ORR     R1, R1, R0;                     Merge the two bits (PE0 | PE4)
        LDR R0, =data_capture;
        LDR     R2, =NEntries;
        LDRB R2, [R2];
        ADD     R0, R0, R2;
        STRB R1, [R0];                          Store the value in the correct spot on the data array
; Record the current time
        MOV     R0, #4;
        MUL R2, R2, R0;                         Increment in the time array by 4 bytes
        LDR     R1, =time_capture;
        ADD     R1, R1, R2;
        LDR R0, =NVIC_ST_CURRENT_R
        LDR R0, [R0];
        STR     R0, [R1];                       Store the current time in the correct spot on the
time array
; Restore the registers and leave
DONE_C
        POP {R2, LR};
        POP {R0,R1}
        BX LR;

;-------Toggle Green LED (PF2)-----------------------------------------------------------------
;Toggles the Green LED on and off (PF2)
Toggle_Green
        PUSH {R0, R1};
        LDR     R0, =GPIO_PORTF_DATA_R;
        LDR     R1, [R0];
        EOR     R1, #0x04;
        STR     R1, [R0];
        POP {R0, R1};
        BX LR;

;----------------------------------------------------------------------------------------------
delay
; a subroutine that loops using the value at R0
        PUSH {R0, R1};
        MOV     R1, #0;
delayLoop
        CMP     R0, R1;                         Loop until temporary value, R1, reaches R0
        BEQ     delayDone;
        ADD     R1, R1, #1;
        B       delayLoop;
delayDone
        POP {R0, R1};
        BX LR;

;----------------------------------------------------------------------------------------------
; Port Initialization
Ports_Init
        PUSH {R0, R1};
        PUSH {R2, LR};
        LDR     R0, =SYSCTL_RCGCGPIO_R;
        LDR     R1, [R0];
        ORR   R1, R1, #0x30;                    Start up Port F and Port E
        STR   R1, [R0];
        NOP;
        NOP;
 ; Configure Port E
        LDR   R0, =GPIO_PORTE_DIR_R;
        LDR   R1, [R0];
        ORR   R1, R1, #0x01;                            PE0 is set to output (LED)
        BIC   R1, R1, #0x12;                            PE1,4 are set to input (buttons)
        STR   R1, [R0];
        LDR   R0, =GPIO_PORTE_AFSEL_R;
        LDR   R1, [R0];
        MOV   R1, #0;                                   Disables the "alternate functions" in
the port
        STR   R1,     [R0];
        LDR   R0, =GPIO_PORTE_DEN_R;
        LDR   R1, [R0];
        MOV   R1, #0xFF;                                1 means enable digital I/O
        STR   R1, [R0];
; Configure Port F
        LDR R1, =GPIO_PORTF_LOCK_R; 2) unlock the lock register
        LDR R0, =GPIO_LOCK_KEY;                         unlock GPIO Port F Commit
Register
        STR R0, [R1];
        LDR R1, =GPIO_PORTF_CR_R;     enable commit for Port F
        MOV R0, #0xFF;                1 means allow access
        STR R0, [R1];
        LDR R1, =GPIO_PORTF_DIR_R;    5) set direction register
        MOV R0,#0x0E;
        STR R0, [R1];
        LDR R1, =GPIO_PORTF_AFSEL_R;          6) regular port function
        MOV R0, #0;                   0 means disable alternate function
        STR R0, [R1];
        LDR R1, =GPIO_PORTF_PUR_R;            pull-up resistors for PF4,PF0
        MOV R0, #0x11;                1)enable for negative logic
        STR R0, [R1];
        LDR R1, =GPIO_PORTF_DEN_R;            7) enable Port F digital port
        MOV R0, #0xFF;                1 means enable digital I/O
        STR R0, [R1];
        POP  {R2, LR};
        POP    {R0, R1};
        BX LR;
```

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

Frequency Calculations:

Time array captures after 6 main loops:

Time_array[0] = 0x02430A4
Time_array[1] = 0x0B604F4

Subtracting the times: 0x0B604F4 - 0x02430A4
= 0x091D450 ← Change in time over 6 main loops (in cycles)
= 9,557,072 (in decimal form)

Divide the value by 6: 0x091D450 / 6
= 1,592,845.33 ← Change in time over 1 main loop (in decimal)

8,000,000 cycles = 1 sec

secs = 1,592,845 / 8* 10^9
Secs = .19910562

Period = 1/ secs
Period = 5 hz