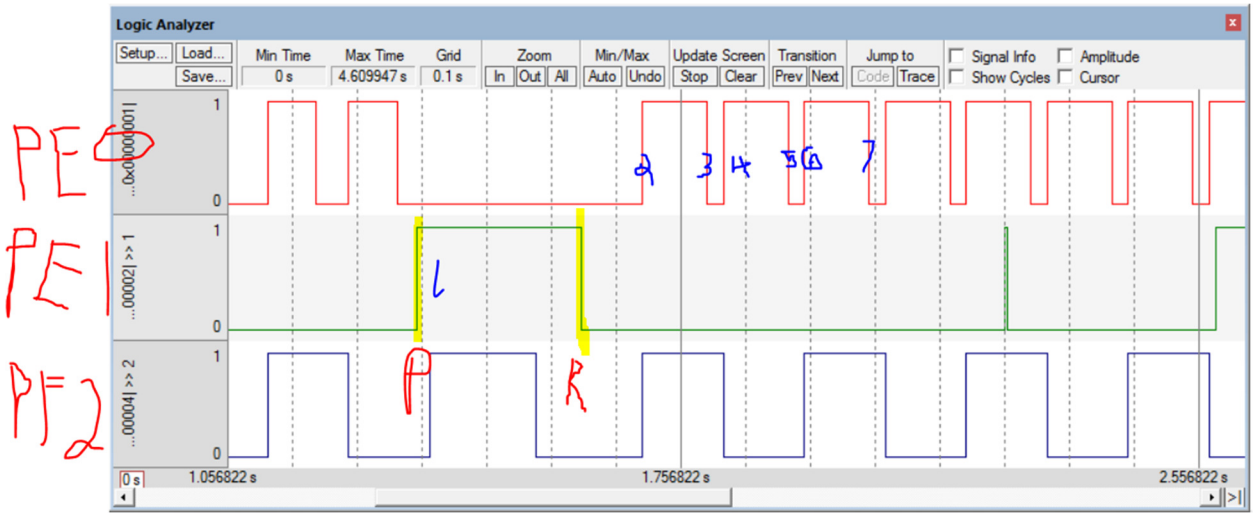
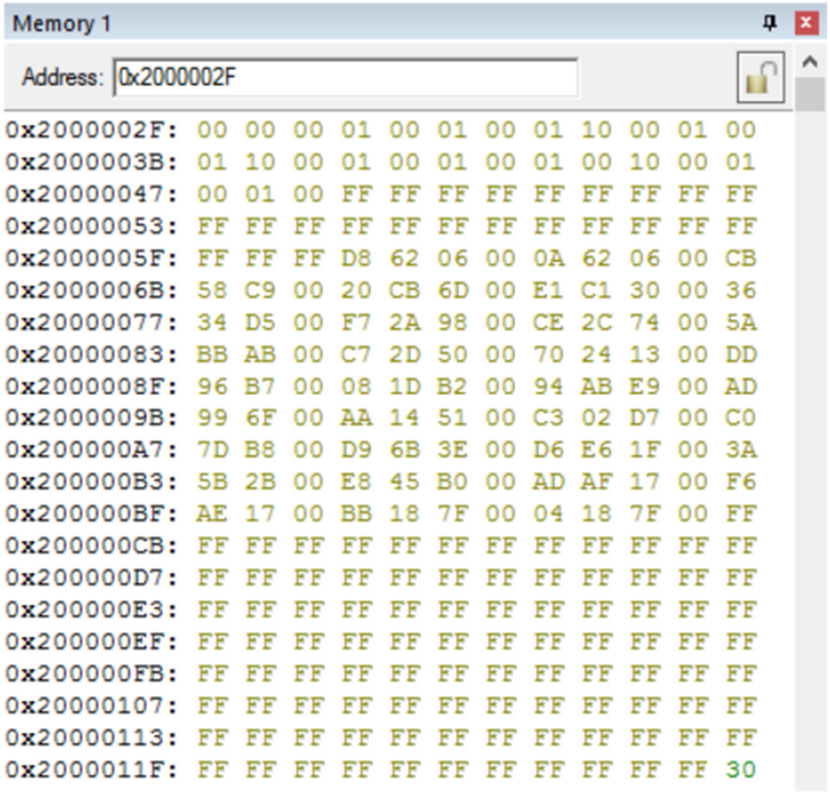
Lab Deliverables for Lab 4





```
;****************** main.s ***************
; Program written by: Jerry Yang/Aditya Tyagi
; Date Created: 2/19/2017
; Last Modified: 2/14/2017
; Brief description of the program
;   The LED toggles at 8 Hz and a varying duty-cycle
;   Repeat the functionality from Lab2-3 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).

GPIO_PORTE_DATA_R  EQU 0x400243FC
GPIO_PORTE_DIR_R   EQU 0x40024400
GPIO_PORTE_AFSEL_R EQU 0x40024420
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_PUR_R   EQU 0x40025510
GPIO_PORTF_DEN_R   EQU 0x4002551C
SYSCTL_RCGCGPIO_R  EQU 0x400FE608
GPIO_PORTF_LOCK_R  EQU 0x40025520
GPIO_PORTF_CR_R    EQU 0x40025524
GPIO_LOCK_KEY      EQU 0x4C4F434B

; RAM Area
```

```
AREA    DATA, ALIGN=2
;-UUU-Declare  and allocate space for your Buffers
;    and any variables (like pointers and counters) here
DataBuffer SPACE 50
TimeBuffer SPACE 200
DataPt SPACE 4
TimePt SPACE 4
NDumps SPACE 1
BlinkingDumps SPACE 1; To check when main loop hits 7 dumps

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

Start
; TExaS_Init sets bus clock at 80 MHz
BL  TExaS_Init ; voltmeter, scope on PD3
CPSIE  I    ; TExaS voltmeter, scope runs on interrupts

 LDR R0, =SYSCTL_RCGCGPIO_R; Clock
```

```
LDR R1, [R0]
ORR R1, #0x30
STR R1,[R0]
NOP
NOP
;2 bus cycles

LDR R1,=GPIO_PORTF_DIR_R ;Set PF0,PF4 as inputs and PF1-3 as outputs
LDR R0,[R1]
AND R0,#0x04
ORR R0,#0x04; PF2 high for heartbeat CHECK THIS, ADD ORR
STR R0,[R1]

LDR R1,=GPIO_LOCK_KEY ;Unlock Port F
LDR R0,=GPIO_PORTF_LOCK_R
STR R1,[R0]

LDR R1,=GPIO_PORTF_CR_R ;Allow changes to PF4
LDR R0,[R1]
ORR R0,R0,#0x10
STR R0,[R1]

LDR R1,=GPIO_PORTF_AFSEL_R ;Clear AFSEL bits to 0 to select regular I/O
LDR R0,[R1]
AND R0,R0,#0x00
```

```
STR R0,[R1]

LDR R1,=GPIO_PORTF_PUR_R ;Enable internal pull-up on PF4
LDR R0,[R1]
ORR R0,R0,#0x10
STR R0,[R1]

LDR R1,=GPIO_PORTF_DEN_R ;Enable data on PF4 bits
LDR R0,[R1]
ORR R0,R0,#0x14; PF2 for heartbeat
STR R0,[R1]

                ; set directon
LDR R0, =GPIO_PORTE_DIR_R
LDR R1, [R0]
ORR R1, #0x01 ;PE0 is the ourput, the rest are o and input?
STR R1, [R0]
;As of right now

                ; AFSEL
LDR R0,=GPIO_PORTE_AFSEL_R
AND R1, #0
STR R1, [R0]    ;check this one //this seems right.
```

```
;set DEN
LDR R0, =GPIO_PORTE_DEN_R
LDR R1, [R0]
ORR R1, #0x1F
STR R1, [R0]

;DEFINE/INITIALIZE DEBUG VARS
Debug_Init
 LDR R1,=NDumps
 MOV R0,#0
 STRB R0,[R1]

 LDR R1,=BlinkingDumps
 MOV R0,#9
 STRB R0,[R1]

 LDR R1,=DataPt; Initialize data pointer
MOV R0, #0x2 ; error: cannot MOV #0x20000030
 LSL R0, #28
 ADD R0, #0x30
 STR R0,[R1]

 LDR R1,=TimePt; Initialize time pointer
MOV R0, #0x2 ; error: cannot MOV #0x20000030 --> meant #0x20000062
```

```
LSL R0, #28
ADD R0, #0x62
STR R0,[R1]

MOV R12,#0xFFFFFFFF; Initialize buffer entries
LDR R1,=DataPt
LDR R1,[R1]
MOV R0,#124; Loop decrementor for do-while loop
L1      STR R12,[R1]
ADD R1,#2
SUBS R0,#1
BNE L1

BL SysTick_Init

;DEFINE BLINK VARS
MOV R4,#0; track state of button, 0-unpressed, not 0-pressed
MOV R5,#140; length of each cycle
MOV R6,#28; length of duty cycle
MOV R7,#28; incrementor
MOV R8,#1; track cycle, 1 if increasing, -1 if decreasing
BL Read_SysTick
;DEFINE BREATHE VARS
MOV R9,#1064; length of each cycle
```

```
MOV R10,#28; duty cycle

MOV R11,#7; incrementor

MOV R12,#1; track cycle, 1 if increasing, -1 if decreasing


;MAIN LOOOOOOOP

loop

LDR R1,=GPIO_PORTE_DATA_R; load PE, PF

LDR R0,[R1]

LDR R3,=GPIO_PORTF_DATA_R

LDR R2,[R3]


EOR R2, #0x4 ;Won't see toggling if breathe on, too fast.

STR R2,[R3]


AND R2,#0x10; checks if PF4 button is pressed

CMP R2,#0

BEQ BREATHELED_ON


; checks if PE1 button is pressed

AND R0,#0x02

CMP R0,#0

BNE SWPRESSED; Turns LED off while pressed, updates duty cycle

B BLINKLED_ON


B loop
```

```
;BLINK SUBROUTINE
BLINKLED_ON
PUSH {R10-R12,LR}
BL CHECKBLINKINGDUMPS
POP {R10-R12,LR}

PUSH{R4,LR}
ORR R0,R0,#0x1; Calculate PE0 & store
STR R0,[R1]
MOV R4,R6
PUSH {R0,R1}
MOV R0,#87 ; To exhibit change in brightness instead of blinking, use 10.
MUL R4,R0   ; Oscilloscope got between 124 (w/61), 125 (w/62)
POP {R0,R1} ; Logic Analyzer got ~>125 (w/87)
BL TIMERLOOP;

BLINKLED_OFF
PUSH {R10-R12,LR}
BL CHECKBLINKINGDUMPS
POP {R10-R12,LR}

EOR R0,#1
STR R0,[R1]
SUB R4,R5,R6
```

```
PUSH {R0,R1}
MOV R0,#87
MUL R4,R0
POP {R0,R1}
BL TIMERLOOP
POP{R4,LR}
MOV R4,#0
B loop

; BREATHE SUBROUTINE
BREATHELED_ON
PUSH {R4,LR}
EOR R0,R2,#0x10; Calculate PE0 & store
LSR R0,#4
STR R0,[R1]
LSL R0,#2; 2 debug instructions (heartbeat)
STR R0,[R3]
MOV R4, R10
BL TIMERLOOP;
BREATHELED_OFF
LSR R0,#2; 3 debug instructions (hearbeat)
EOR R0,#1
STR R0,[R1]
LSL R0,#2
STR R0,[R3]
```

```
BL CHECKDUTYCYCLE

BL UPDATEDUTYCYCLE

SUB R4,R9,R10

BL TIMERLOOP;

POP {R3-R5,LR}

POP {R4,LR}

B loop


; OTHER SUBROUTINES

TIMERLOOP ; delays for (R4)0.01 ms, input: R4, no outputs

CMP R4,#0

BEQ next

SUB R4,#1

PUSH {LR,R4,R5,R6}

BL TIMER

POP {LR,R4,R5,R6}

next CMP R4,#0

BNE TIMERLOOP

BX LR


TIMER; yields 0.01 ms delay

MOV R5, #16

DLoop

MOV R4, #10
```

```
DLoop1
    SUBS R4, #1
    BPL DLoop1
    SUBS R5, #1
    BPL DLoop
    BX LR ;return

SWPRESSED ;Turns off LED indefinitely
    AND R0,#0
    STR R0,[R1]

CMP R4,#1;Checks if button is already pressed, Y-go to main, N-update duty cycle
    BEQ loop

    ;PUSH {R0-R4,LR}; Debug dump
    BL Debug_Capture

    PUSH {R11,R12}; Reset BlinkingDumps to 0
    LDR R12,=BlinkingDumps
    MOV R11,#0
    STR R11,[R12]
    POP {R11,R12}

    ;POP {R0-R4,LR}; End debug dump
```

```
PUSH {LR,R0,R9-R12}; Store R9-R12 on stack, move R5-R8 into R9-R12
MOV R9,R5
MOV R10,R6
MOV R11,R7
MOV R12,R8
BL CHECKDUTYCYCLE
BL UPDATEDUTYCYCLE
POP {R3-R5,LR}
MOV R5,R9
MOV R6,R10
MOV R7,R11
MOV R8,R12
POP {LR,R0,R9-R12} ; Revert original R9-R12

CMP R8,#-1; Resets switch to 0
BNE pass
MOV R8,#1
MOV R6,#0

pass    MOV R4,#1; Sets button state to 'pressed'
B loop

CHECKDUTYCYCLE ;Checks if duty cycle exceeds min/max values
CMP R10,R9 ;Check if duty cycle = cycle length
```

```
        BNE notMax
        MOV R12,#-1
notMax  CMP R10,R11 ;Check if duty cycle = incrementor
        BNE notMin
        MOV R12, #1
notMin  BX LR

UPDATEDUTYCYCLE; Updates duty cycle
        PUSH {R3-R5,LR}
        MUL R5,R12,R11
        ADD R10,R5
        BX LR

Debug_Capture ;ADD CALCULATIONS HERE IN COMMENTS: 100*(NUM INSTRUCTIONS*2)/TIME FOR ONE 8Hz CYCLE
        PUSH {R0-R4,LR}
        LDR R3,=NDumps;Check if buffers full
        LDRB R2,[R3]
        CMP R2,#50;
        POP {R0-R4,LR}
        BEQ back

        ;Dump port E
        PUSH {R0-R4,LR}
        LDRB R0,[R1]; Read port E
        AND R2,R0,#0x3; Mask for PE1-0
```

```
LSL R3,R2,#3; Shift PE1 to bit 4
ADD R2,R3
AND R2,#0x11; Mask for bits to store
LDR R4,=DataPt; Calculate target address
LDR R4,[R4]
LDR R0,=NDumps
LDRB R0,[R0]
ADD R0,R4
STRB R2,[R0] ; Dump

;Dump SysTick
PUSH {R0,LR}
BL Read_SysTick; Output in R0
LDR R4,=TimePt; Calculate target address
LDR R4,[R4]
LDR R3,=NDumps
LDRB R3,[R3]
LSL R3,#2
ADD R3,R4
STR R0,[R3]; Dump
POP {R0,LR}

LDR R0,=NDumps
LDRB R1,[R0]
ADD R1,#1
```

```
        STRB R1,[R0]

        POP {R0-R4,LR}
back    BX LR

CHECKBLINKINGDUMPS; Checks if necessary to dump
    LDR R12,=BlinkingDumps; Check if enough data dumped
    LDRB R11,[R12]
    CMP R11,#7
    BCS skip

    PUSH {R0,LR}
    BL Debug_Capture
    ADD R11,#1
    STR R11,[R12]
    POP {R0,LR}

skip  STR R11,[R12]
    BX LR

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

Part 4: 100*(NUM INSTRUCTIONS*2)/TIME FOR = 880ns