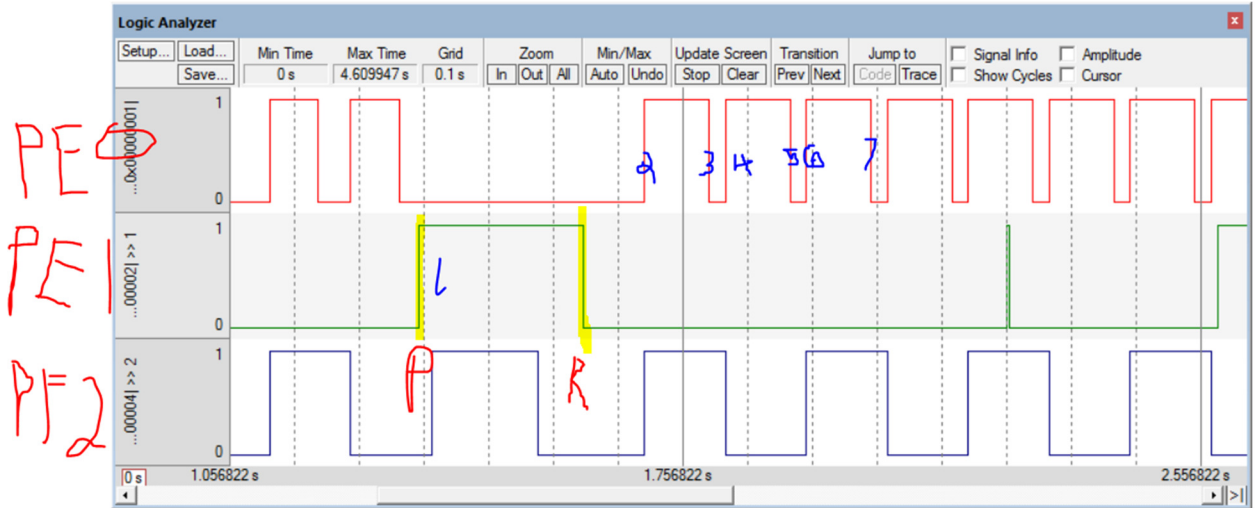


## Lab Deliverables for Lab 4



```
Memory 1
Address: 0x2000002F
0x2000002F: 00 00 00 01 00 01 00 01 10 00 01 00
0x2000003B: 01 10 00 01 00 01 00 01 00 10 00 01
0x20000047: 00 01 00 FF FF FF FF FF FF FF FF
0x20000053: FF FF FF FF FF FF FF FF FF FF FF
0x2000005F: FF FF FF D8 62 06 00 0A 62 06 00 CB
0x2000006B: 58 C9 00 20 CB 6D 00 E1 C1 30 00 36
0x20000077: 34 D5 00 F7 2A 98 00 CE 2C 74 00 5A
0x20000083: BB AB 00 C7 2D 50 00 70 24 13 00 DD
0x2000008F: 96 B7 00 08 1D B2 00 94 AB E9 00 AD
0x2000009B: 99 6F 00 AA 14 51 00 C3 02 D7 00 C0
0x200000A7: 7D B8 00 D9 6B 3E 00 D6 E6 1F 00 3A
0x200000B3: 5B 2B 00 E8 45 B0 00 AD AF 17 00 F6
0x200000BF: AE 17 00 BB 18 7F 00 04 18 7F 00 FF
0x200000CB: FF FF FF FF FF FF FF FF FF FF FF
0x200000D7: FF FF FF FF FF FF FF FF FF FF FF
0x200000E3: FF FF FF FF FF FF FF FF FF FF FF
0x200000EF: FF FF FF FF FF FF FF FF FF FF FF
0x200000FB: FF FF FF FF FF FF FF FF FF FF FF
0x20000107: FF FF FF FF FF FF FF FF FF FF FF
0x20000113: FF FF FF FF FF FF FF FF FF FF FF
0x2000011F: FF FF FF FF FF FF FF FF FF FF 30
```

```
,***** 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 Button(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 LOOOOOOOOP

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 BREAHELED\_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

BRETHELED\_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;

BRETHELED\_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:  $100 * (\text{NUM INSTRUCTIONS} * 2) / \text{TIME FOR ONE 8Hz CYCLE}$

```

        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:  $100 * (\text{NUM INSTRUCTIONS} * 2) / \text{TIME FOR} = 880\text{ns}$

Memory 1																	
Address: 0x2000002F																	
0x2000002F:	00	10	00	01	00	01	00	01	00	01	00	01	00	10			
0x20000039:	00	01	00	01	00	01	00	10	00	01	00	10	00	01			
0x20000043:	00	01	00	01	00	10	00	01	00	01	00	01	00	01			
0x2000004D:	00	01	00	10	00	01	00	01	00	01	00	01	00	01			
0x20000057:	00	10	00	01	00	01	00	01	00	01	00	01	00	10			
0x20000061:	00	67	56	8D	00	7D	03	BF	00	91							
0x2000006B:	7E	64	00	65	B7	DC	00	43	32	82							
0x20000075:	00	17	6B	FA	00	2B	E6	9F	00	FB							
0x2000007F:	1E	18	00	A7	17	A1	00	F3	4B	25							
0x20000089:	00	CB	84	9D	00	A1	FF	42	00	77							
0x20000093:	38	BB	00	9F	B3	60	00	93	EC	D8							
0x2000009D:	00	4D	67	7E	00	47	CA	42	00	BF							
0x200000A7:	33	4B	00	89	2A	96	00	7B	E7	68							
0x200000B1:	00	3B	DE	B3	00	87	9B	86	00	4B							
0x200000BB:	92	D1	00	4B	4F	A4	00	A3	66	86							
0x200000C5:	00	51	57	E1	00	0B	0C	FF	00	35							
0x200000CF:	0B	FF	00	BF	BF	1C	00	E9	BE	1C							
0x200000D9:	00	A7	73	3A	00	D1	72	3A	00	81							
0x200000E3:	6B	C3	00	C7	D4	CB	00	15	D4	CB							
0x200000ED:	00	AF	88	E9	00	FD	87	E9	00	73							
0x200000F7:	3C	07	00	C1	3B	07	00	45	F0	24							
0x20000101:	00	5B	7F	72	00	41	E9	7A	00	5D							
0x2000010B:	A6	4D	00	FD	9C	98	00	51	5A	6B							
0x20000115:	00	E9	50	B6	00	27	0E	89	00	CD							
0x2000011F:	04	D4	00	5F	1C	B6	00	87	42	95							
0x20000129:	00	30	00	00	20	62	00	00	20	32							
0x20000133:	07	00	00	00	8C	00	00	00	00	28							
0x2000013D:	60	65	8C	45	45	33	31	39	4B	20							
0x20000147:	20	04	00	00	00	00	00	00	00	00							
0x20000151:	00	00	00	1B	48	05	60	20	20	20							
0x2000015B:	20	20	20	20	20	04	0B	00	00	DC							
0x20000165:	0A	00	00	B4	0A	00	00	B4	0A	00							
0x2000016F:	00	10	00	01	00	01	00	01	00	10							