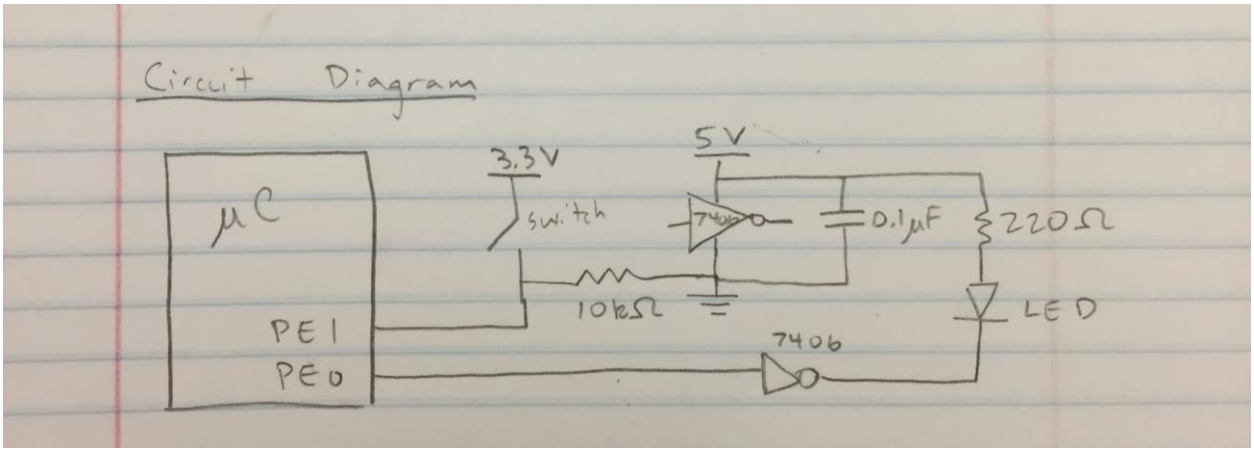
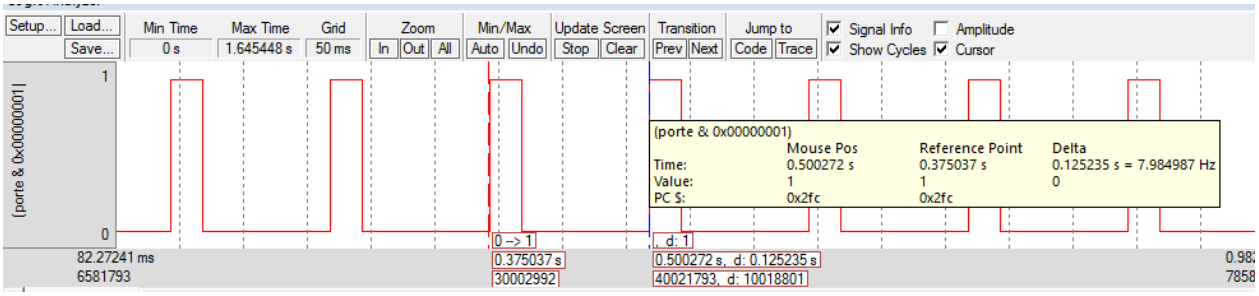
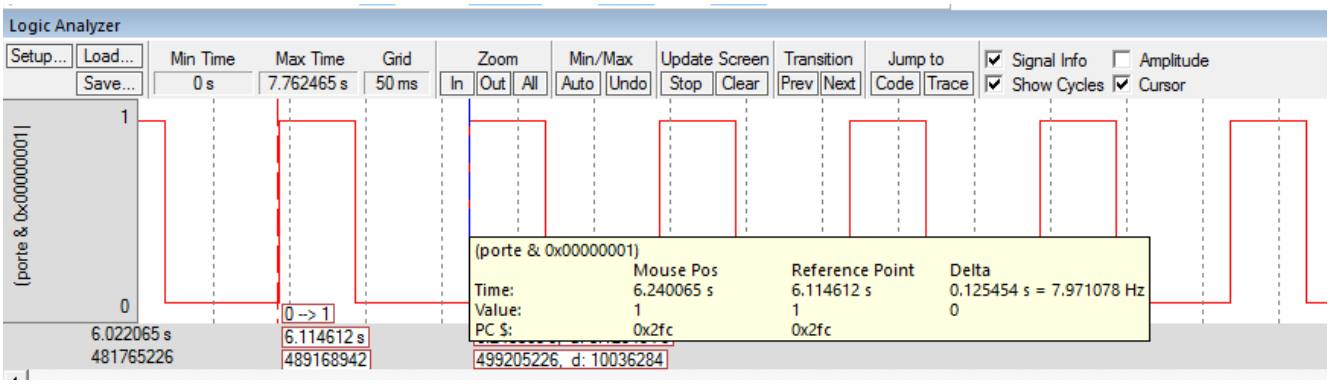
Lab 2-3 Deliverables Brandon Lee, Melanie Feng EE 319K 2/13/17



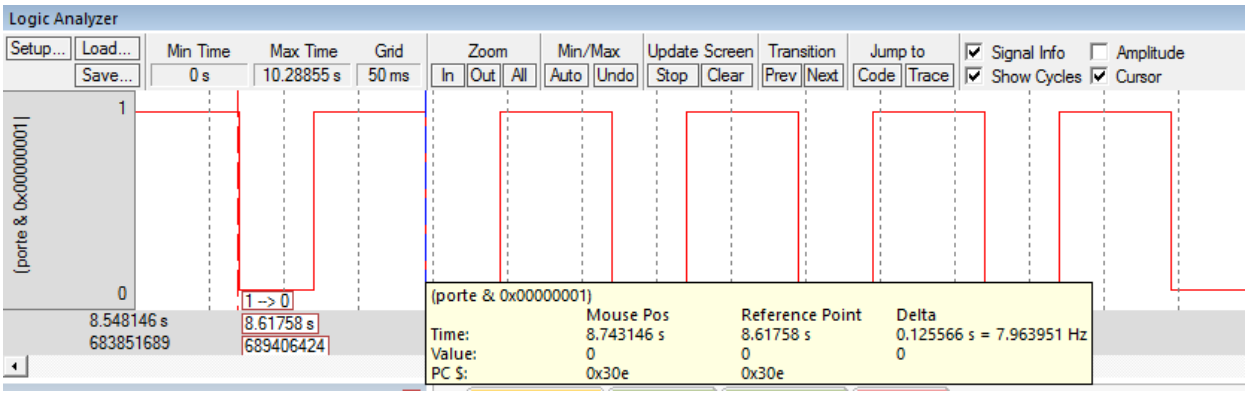
20% duty cycle



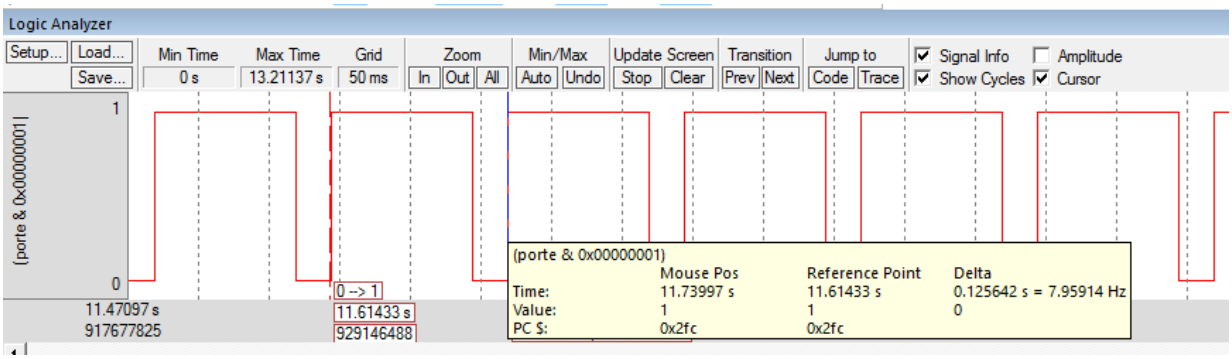
40% duty cycle



60% duty cycle



80% duty cycle

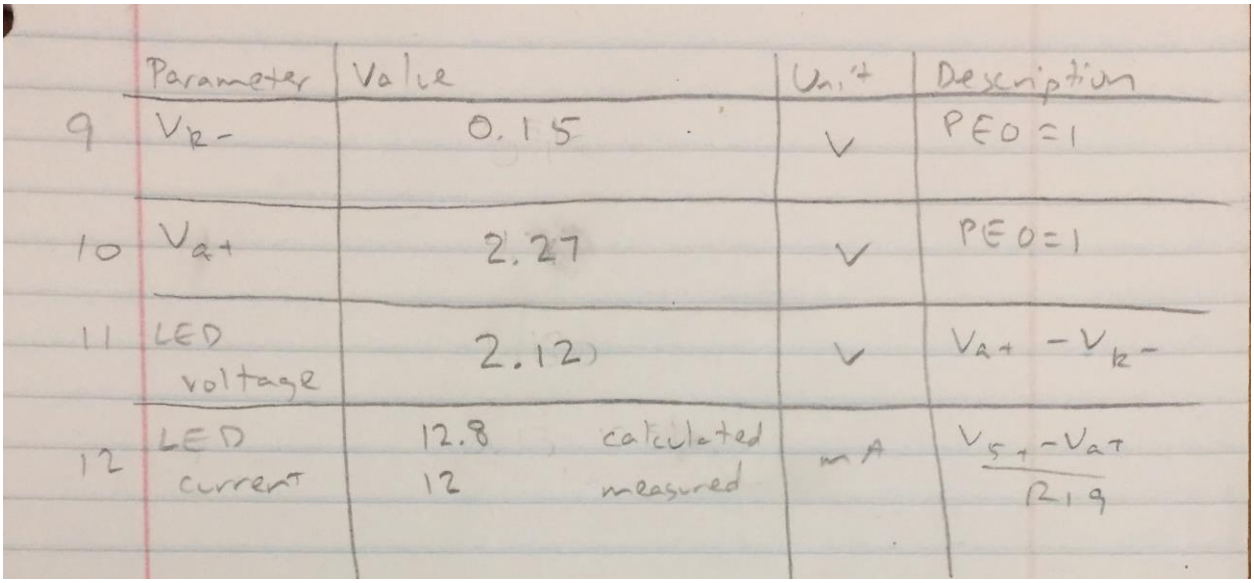


Switch

	Parameter	Value	Units	Conditions
	R1: 10kΩ	9890	Ω	power off, dced
	Supply V $V_{+3.3}$	3,27	V	powered
1	Input V V_{PE1}	0	V	powered, switch not pressed
	Resistor current	0 measured 0 calc	mA	powered, not pressed
	Input V V_{PE1}	3,27	V	powered, switch pressed
	Resistor current	0,32 0,33	mA	powered, switch pressed

LED

	Parameter	Value	Unit	Description
1	R19: 220Ω	215	Ω	power off, dced
2	5V power supply	5,02	V	
3	Output V_{PE0} input to 7406	0	V	PE0 = 0
4	7406 output V_{R-}	3,49	V	PE0 = 0
5	LED at V_{A+}	4,99	V	PE = 0
6	LED voltage	1.5	V	PE:0, calculated
7	LED current	0.13 calculated 0 measurement	mA	PE = 0 $\frac{V_{S+} - V_{A+}}{R19}$
8	V_{PE0} input to 7406	3.25	V	PE0 = 1



	Parameter	Value		Unit	Description
9	V_{K-}	0.15		V	PE0 = 1
10	V_{A+}	2.27		V	PE0 = 1
11	LED voltage	2.12		V	$V_{A+} - V_{K-}$
12	LED current	12.8	calculated	mA	$\frac{V_{5+} - V_{AT}}{R19}$
		12	measured		

Main.s

```
; Program written by: Melanie Feng and Brandon Lee
; Last Modified: 2/14/2017
; Brief description of the program
;   The LED toggles at 8 Hz and a varying duty-cycle
; 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 external9 LED on protoboard)
;  PF4 is builtin button SW1 on Launchpad (Internal)
;        Negative Logic (0 means pressed, 1 means not pressed)
; Overall functionality of this system is to operate like this
;   1) Make PE0 an output and make PE1 and PF4 inputs.
;   2) The system starts with the the LED toggling at 8Hz,
;      which is 8 times per second with a duty-cycle of 20%.
;      Therefore, the LED is ON for (0.2*1/8)th of a second
;      and OFF for (0.8*1/8)th of a second.
;   3) When the button on (PE1) is pressed-and-released increase
;      the duty cycle by 20% (modulo 100%). Therefore for each
;      press-and-release the duty cycle changes from 20% to 40% to 60%
;      to 80% to 100%(ON) to 0%(Off) to 20% to 40% so on
;   4) Implement a "breathing LED" when SW1 (PF4) on the Launchpad is pressed:
;      a) Be creative and play around with what "breathing" means.
;         An example of "breathing" is most computers power LED in sleep mode
;         (e.g., https://www.youtube.com/watch?v=ZT6siXyIjvQ).
;      b) When (PF4) is released while in breathing mode, resume blinking at 8Hz.
```

```
;       The duty cycle can either match the most recent duty-
;       cycle or reset to 20%.
;     TIP: debugging the breathing LED algorithm and feel on the simulator is impossible.
; 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

SYSCTL_RCGCGPIO_R  EQU 0x400FE608
     IMPORT  TExaS_Init
             IMPORT  delaySR
;            IMPORT PE0_toggle
             IMPORT hastoggled
             IMPORT breathing
             EXPORT beginning
     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
             ; turn on port clocks for PE and PF
             LDR R0, = SYSCTL_RCGCGPIO_R
             LDR R1, [R0]
             ORR R1, #0x30
             STR R1, [R0]
             ; wait for 2 cycles
             NOP
             NOP
             ; set pin direction
             LDR R0, = GPIO_PORTF_DIR_R
             LDR R1, [R0]
             AND R1, #0xEF                                          ; PF4 as input (set to 0)
             STR R1, [R0]
             LDR R0, = GPIO_PORTE_DIR_R
             LDR R1, [R0]
```

```
        ORR R1, #0x01                                   ; PE0 as output (set to 1)
        AND R1, #0xFD                                           ; PE1 as input (set to 0)
        STR R1, [R0]
        ; turn off alternate functions
        LDR R0, = GPIO_PORTF_AFSEL_R
        LDR R1, [R0]
        AND R1, #0xEF
        STR R1, [R0]
        LDR R0, = GPIO_PORTE_AFSEL_R
        LDR R1, [R0]
        AND R1, #0xFC
        STR R1, [R0]
        ; set PF4 to negative logic
        LDR R0, = GPIO_PORTF_PUR_R
        LDR R1, [R0]
        ORR R1, #0x10
        STR R1, [R0]
        ; set digital enable
        LDR R0, = GPIO_PORTF_DEN_R
        LDR R1, [R0]
        ORR R1, #0x10
        STR R1, [R0]
        LDR R0, = GPIO_PORTE_DEN_R
        LDR R1, [R0]
        ORR R1, #0x03
        STR R1, [R0]

beginning
        MOV R8, #1
;       MOV R5, #1
        MOV R12, #1


loop
        LDR R0, = GPIO_PORTE_DATA_R
        LDR R1, [R0]

        EOR R1, #0x01

;onLED

        AND R2, R1, #0x01
        SUBS R2, #0x00
        CMP R2, #0x01
```

```
        BNE offSR

        STR R1, [R0]
        EOR R1, #0x01


        MOV R5, #40
        MUL R4, R8, R5
        BL delaySR



offSR STR R1, [R0]
        EOR R1, #0x01


        MOV R5, #5
        SUBS R4, R5, R8
        MOV R5, #40
        MUL R4, R5
    BL  delaySR


        SUBS R12, #1
        CMP R12, #0
        BGE loop        ;decrement R12
        MOV R11, #0
        BL hastoggled ;test

;test if SW1 (PF4) has been pressed
        LDR R0, =GPIO_PORTF_DATA_R
        LDR R1, [R0]            ;breathing
        LSR R9, R1, #4
        AND R9, #1
        CMP R9, #1
        BEQ loop
        BL breathing
;       BL  PE0_toggle
skip7 B loop
```

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

SR.s

```
                AREA |.text|, CODE, READONLY, ALIGN=2
                THUMB
                EXPORT          delaySR
;               EXPORT  PE0_toggle
                EXPORT hastoggled
                EXPORT breathing
            IMPORT beginning
GPIO_PORTE_DATA_R  EQU 0x400243FC
GPIO_PORTF_DATA_R  EQU 0x400253FC


hastoggled
        LDR R0, = GPIO_PORTE_DATA_R
        LDR R1, [R0]
        LSR R9, R1, #1
        AND R9, #1
        CMP R9, #0
        BNE yestog
ret     BX LR
yestog
        LDR R1, [R0]
        LSR R9, R1, #1
        AND R9, #1
        SUBS R4, R9, R8
        CMP R4, #1
        BEQ skip6
        CMP R9, #1
        BEQ yestog

skip6 ADD R8, #1
;       ADD  R4, R8, #0
        CMP R8, #5
        BNE skip4
        MOV R8, #0
ison
        LDR R0, = GPIO_PORTE_DATA_R         ;led remains on without going through toggle
        LDR R1, [R0]
        ORR R1, #1
        STR R1, [R0]
```

```
        LDR R1, [R0]
        LSR R9, R1, #1
        AND R9, #1
        CMP R9, #0
        BNE ison
ison2
        LDR R0, = GPIO_PORTE_DATA_R        ;led remains off w/o toggle
        LDR R1, [R0]
        LSR R9, R1, #1
        ORR R1, #1
        STR R1, [R0]
        AND R9, #1
        CMP R9, #0
        BEQ ison2

isoff
        LDR R1, [R0]
        AND R1, #0xFE
        STR R1, [R0]
        LDR R1, [R0]
        LSR R9, R1, #1
        AND R9, #1
        CMP R9, #0
        BNE isoff

isoff2
    LDR R1, [R0]
        LSR R9, R1, #1
        AND R9, #1
        CMP R9, #0
        BEQ isoff2



skip4 B ret



; R4 holds duty cycle: R4 = 1 -> 20%, R4 = 2 -> 40%
delaySR
        MOV R5, R4
    CMP R5, #0
        BNE delay
        BX LR
```

```
;           MOV R5, #40
delay MOV R6, #12500
            CMP R11, #1
            BNE wait
            MOV R6, #500            ;mess with this
wait  SUBS R6, R6, #0x01
            BNE wait
            SUBS R5, R5, #0x01
            BNE delay                           ;BLE or BNE
            BX LR




; bit = 0 -> SW1 pressed, bit = 1 -> SW2 pressed
breathing
            PUSH {R4, R8}
            MOV R4,#10
            MOV R12, #100
            MOV R11, #1
;           LDR R0, = GPIO_PORTF_DATA_R
checkSW         LDR R1, [R0] ; keeps program in a loop if SW1 is pressed but not released
            LSR R9, R1, #4  ; can also turn LED on here if you want LED to be on when SW1 is pressed (and
held)
            AND R9, #1
            CMP R9, #0
            BLS checkSW             ; if SW1 is 0, aka switch pressed and released,
            AND R1, #0xFE           ; then run this code (make LED breathe)
            LDR R0, = GPIO_PORTE_DATA_R         ; at beginning, set LED off
            LDR R1, [R0]
            AND R1, #0xFE
            STR R1, [R0]
brighter                            ; this works
            PUSH {R4}                           ; saves R4
            LDR R0, = GPIO_PORTE_DATA_R
            LDR R1, [R0]
            ORR R1, #0x01
            STR R1, [R0]                    ; turns on LED
ontime      BL delaySR                      ; keeps LED on for R4 amt of time
            ; check for SW1 here
            LDR R0, = GPIO_PORTF_DATA_R
```

```
        LDR R1, [R0]
        AND R1, #0x10
        LSR R1, #4
        CMP R1, #0
        BEQ poll

        SUBS R4, #10
        CMP R4, #0
        BHI ontime
        POP {R4}
        ADD R8, R4, #0
offtime BL delaySR
        LDR R0, = GPIO_PORTE_DATA_R
        LDR R1, [R0]
        AND R1, #0xFE                ; run this once LED should be turned off
        STR R1, [R0]                 ; turns off LED
        ADDS R8, #10          ; keeps LED off for R8: (100 - R4) amt of time
        CMP R8, #90                  ;MELANIE CHANGE OG 100
        BLO offtime
        ADD R4, #10
        CMP R4, #170                 ;MELANIE CHANGE OG 100
        BLO brighter


dimmer ;SUBS R4, #10
        PUSH {R4}                            ; R4 at beginning is approx 100
        LDR R0, = GPIO_PORTE_DATA_R
        LDR R1, [R0]
        ORR R1, #0x01
        STR R1, [R0]                 ; turns on LED

ontime1
        BL delaySR                   ; keeps LED on for R4 amt of time
        ; check for SW1 here
        LDR R0, = GPIO_PORTF_DATA_R
        LDR R1, [R0]
        AND R1, #0x10
        LSR R1, #4
        CMP R1, #0
        BEQ poll

        SUBS R4, #10
        CMP R4, #0
        BHI ontime1
```

```
        POP {R4}                                ; one instruction: POP {R8}
        ADD R8, R4, #0
offtime1        LDR R0, = GPIO_PORTE_DATA_R
        LDR R1, [R0]
        AND R1, #0xFE
        STR R1, [R0]                    ; turns off LED
        BL delaySR    ; run this once LED should be turned off

        ADDS R8, #10            ; keeps LED off for R8: (100 - R4) amt of time
        CMP R8, #150            ;MELANIE CHANGE OG 100
        BLO offtime1

        ; R4 still contains same value at end of ontime1

        ;  SW1 check here
        LDR R0, = GPIO_PORTF_DATA_R
        LDR R1, [R0]
        AND R1, #0x10
        LSR R1, #4
        CMP R1, #0
        BNE notpressed


poll  LDR R0, = GPIO_PORTF_DATA_R                       ; if SW1 pressed, then loops here (LED
off) until SW1 released
        LDR R1, [R0]
        AND R1, #0x10
        LSR R1, #4
        CMP R1, #1
        BNE poll
        B out

notpressed      SUBS R4, #10                                    ; program goes here if SW1 not
pressed at all
        CMP R4, #0
        BLS prebrighter
        B dimmer


prebrighter
        ADD R4, #10
        B brighter
```

```
out     POP{R4, R8}
        B beginning




        ALIGN
        END
```