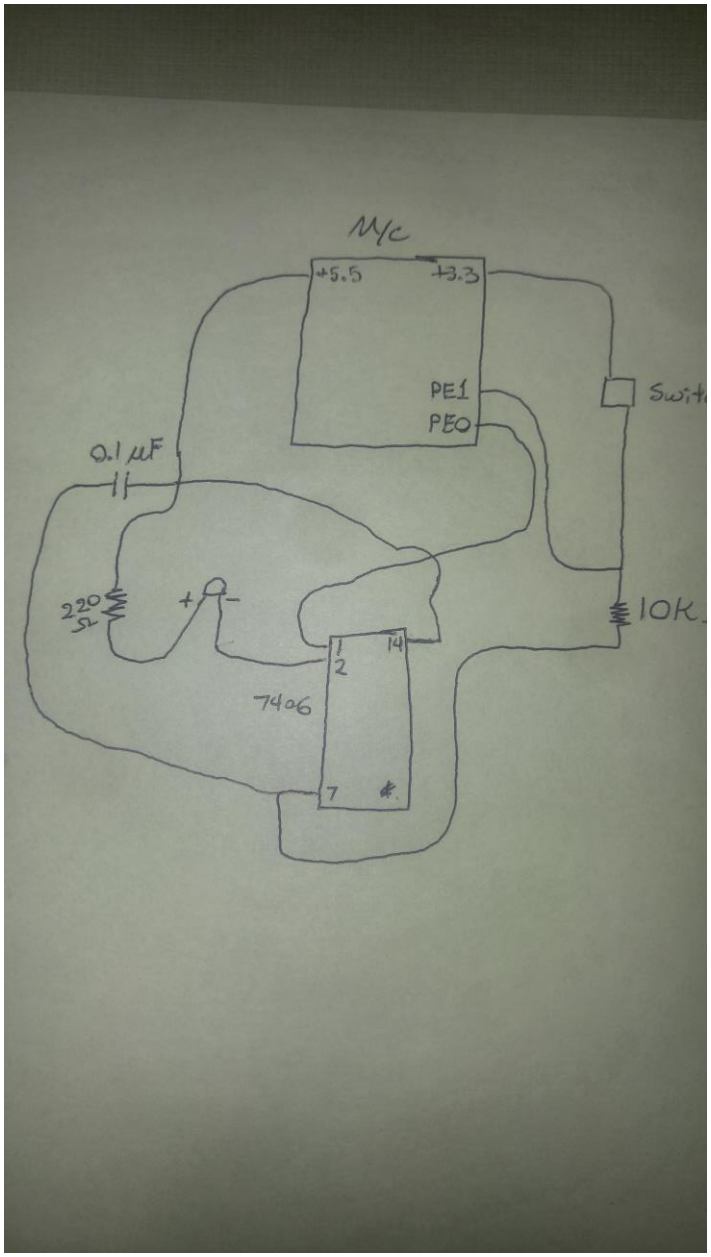
Jason Juliette, Kris Li

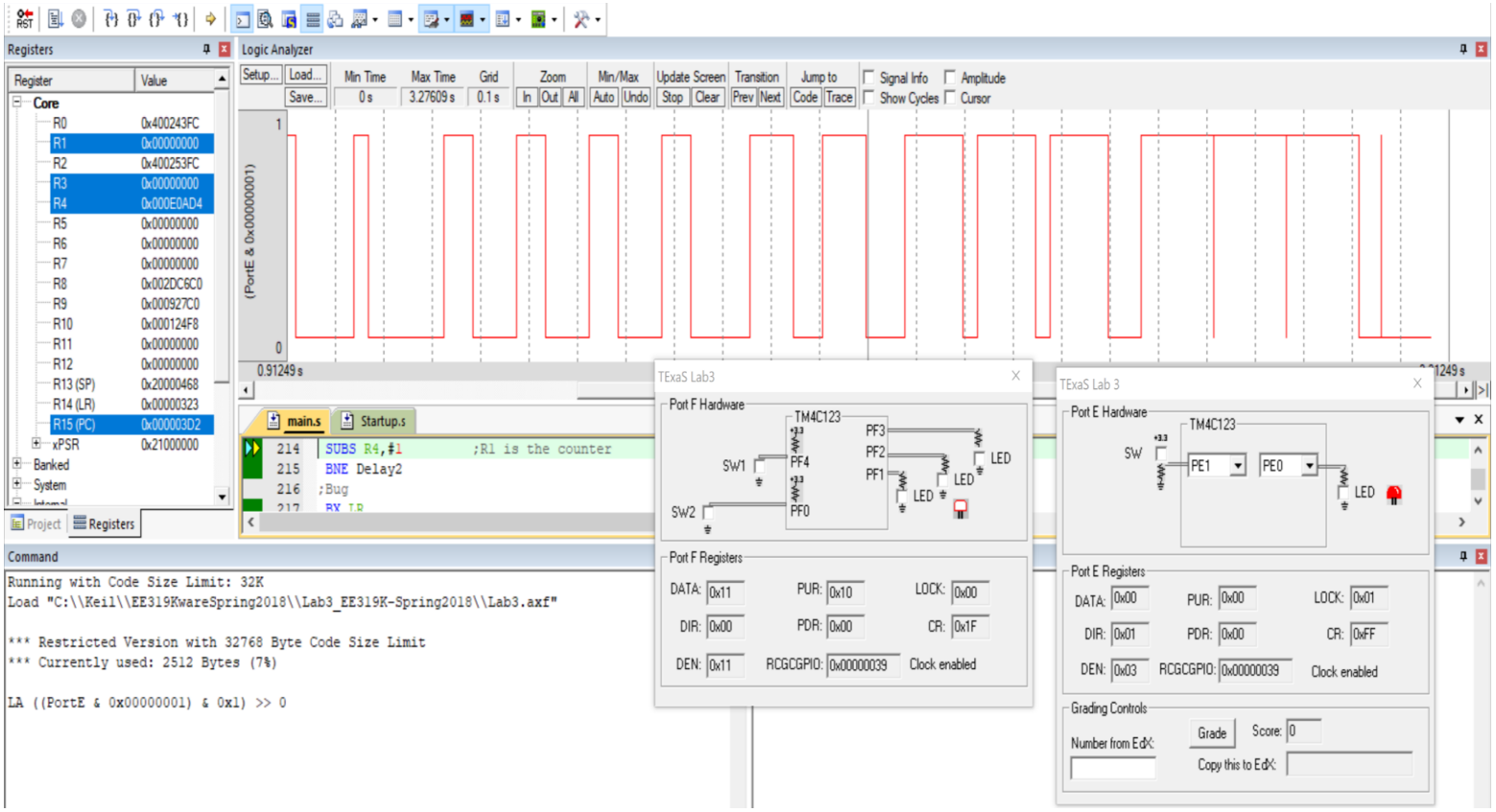
Lab 3 Deliverable

February 15th, 2018

1) Circuit diagram



2) Screenshot like showing your debugging in the simulator



3) Switch measurements (Table 3.1)

Parameter	Value	Units	Conditions
Resistance of the 10kΩ resistor, R1	9.82Kohm 9820ohms	ohms	with power off and disconnected from circuit (measured with ohmmeter)
Supply Voltage, $V_{+3.3}$	3.29V	volts	Powered (measured with voltmeter)
Input Voltage, V_{PE1}	0V	volts	Powered, but with switch not pressed (measured with voltmeter)

Resistor current	0mA	mA	Powered, but switch not pressed $I=V_{PE1}/R1$ (calculated and measured with an ammeter)
Input Voltage, V_{PE1}	3.29V	volts	Powered and with switch pressed (measured with voltmeter)
Resistor current	.335mA	mA	Powered and switch pressed $I=V_{PE1}/R1$ (calculated and measured with an ammeter)

4) LED measurements (Table 3.2)

Row	Parameter	Value	Units	Conditions
1	Resistance of the 220Ω resistor, R19	215 ohms	ohms	with power off and disconnected from circuit (measured with ohmmeter)
2	+5 V power supply V_{+5}	5.03 V	volts	(measured with voltmeter relative to ground, *notice that the +5V power is not exactly +5 volts*)
3	TM4C123 Output, V_{PE0} input to 7406	0.0515 V	volts	with **PE0** = 0 (measured with voltmeter relative to ground)
4	7406 Output, V_{k-} LED k-	3.75 V	volts	with **PE0** = 0 (measured with voltmeter relative to ground)
	LED a+, V_{a+}	5.03 V		with **PE0** = 0

5	Bottom side of R19 (anode side of LED)		volts	(measured with voltmeter relative to ground)
6	LED voltage	1.28 V	volts	calculated as V_{a+} - V_{k-}
7	LED current	0 A And 0 A	mA	calculated as (V_{+5} - V_{a+})/R19 and measured with an ammeter
8	TM4C123 Output, V_{PE0} input to 7406	3.25 V	volts	with **PE0** = 1 (measured with voltmeter relative to ground)
9	7406 Output, V_{k-} LED k-	0.0952 V	volts	with **PE0** = 1 (measured with voltmeter relative to ground)
10	LED a+, V_{a+} Bottom side of R19 (anode side of LED)	2.05 V	volts	with **PE0** = 1 (measured with voltmeter relative to ground)
11	LED voltage	1.95 V	volts	calculated as V_{a+} - V_{k-}
12	LED current	13.86 mA 13 mA	mA	calculated as (V_{+5} - V_{a+})/R19 and measured with an ammeter

5) Assembly source code of your final program

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

; Program written by: Jason Juliette, Kris Li

; Date Created: 2/4/2017

; Last Modified: 1/15/2018

```
; 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 external 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
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

Value EQU 600000                ;this is 20% duty cycle
Half EQU 75000
MAX EQU 3000000                         ;this is 100% duty cycle
                                        ;this is correct for the debugger (possibly not for the
machine)
    IMPORT  TExaS_Init
    THUMB
    AREA    DATA, ALIGN=2
;global variables go 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
```

```
; Initialization goes here
LDR R0, = SYSCTL_RCGCGPIO_R;          i need port e and f
LDR R1,[R0];
ORR R1,#0x30;
STR R1,[R0];
NOP;
NOP;
NOP;
NOP;
LDR R0, = GPIO_PORTF_LOCK_R;          unlock PF4
LDR R1, = GPIO_LOCK_KEY ;
STR R1, [R0];
LDR R0, = GPIO_PORTF_CR_R;
ORR R1, #0xFF;
STR R1, [R0];
LDR R0, = GPIO_PORTF_DIR_R;  port f PF4 as input (0)
LDR R1,[R0];
BIC R1, #0x11;
STR R1,[R0];
LDR R0, = GPIO_PORTF_PUR_R  ; port f is internal so i need to engage the pullup resistor(?)
LDR R1, [R0];
ORR R1, #0x10;
STR R1, [R0];
LDR R0, = GPIO_PORTF_DEN_R ; enable PF4
LDR R1,[R0];
ORR R1,#0x11;
STR R1,[R0];
```

```
LDR R0, = GPIO_PORTE_DIR_R;  port e PE0 equals an output(1) and PE1 equals an input(0)
LDR R1,[R0];
ORR R1,#0x1;
BIC R1,#0x2;
STR R1,[R0];
LDR R0, = GPIO_PORTE_DEN_R ; enable PE0/1
LDR R1,[R0];
ORR R1,#0x3;
STR R1,[R0];



LDR R0, = GPIO_PORTE_DATA_R
LDR R2, = GPIO_PORTF_DATA_R
LDR R8, = MAX
LDR R10, = Half
LDR R9, = Value; value is the incremention, and also the starting point, 1/5 of max duty cycle
MOV R1,R9 ;R1 starts at initial value
    CPSIE  I    ; TExaS voltmeter, scope runs on interrupts
loop                                                    ;////////some of the comments wont make sense cause I changed the program without the comments
                                                                ;////////R4 is generally trash that are consistantly changed
                                                                ;////////R1 is important in that it adjust the delays
                                                                ;////////R2 and R0 hold data registers
                                                                ;////////R3 is typically for moving between Data registers
                                                                ;///////R8 and R9 hold comparison or constants for addition (could we just use immediate mode?) values
                                                                ;///////R10 is important for breathstart, but can be used outside of it and the Occilation subroutine
```

```
                                                ;///////R12 is array
 LDR R3,[R0]                    ;this wil be the register we adjust
 LDR R4,[R2]                    ;i need the real contents of R2
 AND R4,#0x10                           ; checks PF4
 CMP R4,#0
 BEQ breathstart                        ; PF4 is negative logic, so if the button is pressed then
(PF4=0)
 ANDS R4,R3,#0x2                ;checks PE1 value for 1 in bit 1
 BEQ OOF                                ; if zero (button not pressed) continue
 BIC R3,#0x1                    ; not then turn the light off
 STR R3,[R0]                    ;it'll insignificant if the button is slightly pressed
Hold
 LDR R3,[R0]                    ;load a renewed PORTE data register
 ANDS R4,R3,#0x2                ;check bit 1, the button
 BNE Hold                               ;if 1, then it'll contiune the loop an have no light output
 CMP R8,R1                              ;compare count to max
 BNE Nope                               ;if max is greater than R1, skip these next lines
 AND R1,#0                              ;reset R1 (will this actually make the duty cycle 0?)
 B OOF                                  ;skip the addition
                                                ;/////alll of this stuff above works pretty well
except i havent tried the branch to breathstart///
Nope
 ADD R1,R9                              ;duty cycle increase linear?
OOF
 BL Occilate                    ;
 B loop
```

```
breathstart                          ; this is were the breathing subroutine will go
 MOV R1,R9
 B BS
breath
 CMP R1,R8
 BEQ DESCEND
 ADD R1,R10
BS
 LDR R4, = 10
 UDIV R5,R8,R1
 MUL R5,R4
ok
 LDR R4, = 10
 UDIV R1,R4
 UDIV R8,R4
 BL Occilate
 LDR R4, = 10
 MUL R1,R4
 MUL R8,R4
 SUBS R5,#1
 BNE ok
 LDR R6,[R2]                         ;i need the real contents of R2
 AND R4,R6,#0x10        ;to check if it should continue to go through the breathing loop
 CMP R4,#0
 BEQ breath
 MOV R1,R9
    B    loop
DESCEND
 CMP R1,R9
```

```
 BEQ breath
 SUB R1,R10
 LDR R4, = 10
 UDIV R5,R8,R1
 MUL R5,R4
bazzok
 LDR R4, = 10
 UDIV R1,R4
 UDIV R8,R4
 BL Occilate
 LDR R4, = 10
 MUL R1,R4
 MUL R8,R4
 LDR R6,[R2]
 AND R4,R6,#0x10        ;to check if it should continue to go through the breathing loop
 CMP R4,#0
 BNE fin
 SUBS R5,#1
 BNE bazzok
 LDR R6,[R2]
 AND R4,R6,#0x10        ;to check if it should continue to go through the breathing loop
 CMP R4,#0
 BEQ DESCEND
fin
 MOV R1,R9
    B   loop
```

```
Occilate                        ;one subroutine that will shift it up and down
 EOR R3,#0x1                    ; toggle the light, friendly
 STR R3,[R0]                    ; store R3 into Data register
 MOV R4,R1                      ;R4 = R1, the count; change R1 in code to change the duty cycle
 ADD R4,#1                      ;//////there will likely be the same error as in delay2, but when
duty cycle is 0%
Delay
 SUBS R4,#1                     ;R1 is the counter
 BNE Delay              ;PE1 will adjust the value of R1
 EOR R3,#0x1                    ; toggle the light, friendly
 STR R3,[R0]                    ; store R3 into Data register
 SUB R4,R8,R1           ;R4 = difference of R8, and R1 (the rest of the duty cycle)
 ADD R4,#1                      ;///instead of doing the comparison for bug, add 1 to R4 each time.
might change the stuff by a nanosecond but its simple
Delay2

 SUBS R4,#1                     ;R1 is the counter
 BNE Delay2
;Bug
 BX LR


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