main.s | Startup.s

```
45
46      IMPORT  TExaS_Init
47      THUMB
48      AREA    DATA, ALIGN=2
49  ;global variables go here
50  cc  SPACE   4
51
52      AREA    |.text|, CODE, READONLY, ALIGN=2
53      THUMB
54      EXPORT  Start
55
56  Start
57   ; TExaS_Init sets bus clock at 80 MHz
58      BL  TExaS_Init ; voltmeter, scope on PD3
59
60   ; Initialization goes here
61      BL  PortF_E_Init
62      BL  LED_off             ; Make sure LED starts at off
63      MOV R0, #0x00           ; Clear the click_check counter
64      LDR R1, =cc
65      STR R0, [R1]
66
67      CPSIE  I    ; TExaS voltmeter, scope runs on interrupts
68  loop
69  ; main engine goes here
70
71       BL   DC_0
72       BL   DC_20
73       BL   DC_40
74       BL   DC_60
75       BL   DC_80
76       BL   DC_100
77       B    loop
78
79  ; ~~~~~~~~~~~~~~~~~~SUBROUTINES~~~~~~~~~~~~~~~~~ ;
```

main.s | Startup.s

```
79  ; ~~~~~~~~~~~~~~~~~~~SUBROUTINES~~~~~~~~~~~~~~~~~ ;
80
81                                  ; ; ; ; ; ; ; ; ;
82                                  ; Duty Cycle  0 ;
83                                  ; ; ; ; ; ; ; ; ;
84
85  ; Makes sure LED is off at beginning
86  DC_0
87    PUSH {LR,R4}
88        LDR  R1, =GPIO_PORTE_DATA_R  ; Clearing LED Output
89        LDR  R0, [R1]
90        BIC  R0, R0, #0x01
91        STR  R0, [R1]
92  loop_0
93        BL   breathe
94        BL   check_click
95        ADDS R0, R0, #0x00
96        BEQ  loop_0
97
98    POP {LR,R4}
99        BX   LR
100                                 ; ; ; ; ; ; ; ; ;
101                                 ; Duty Cycle 20 ;
102                                 ; ; ; ; ; ; ; ; ;
103
104 ; Creates Duty Cycle of 20% of 8 Hz
105 ; Modifies: R0, R1, R2, (R3)
106
107 DC_20
108   PUSH {LR,R4}                  ; Preserve 8-byte alignment by pushing and popping an even number of registers
109 loop_20
110       BL   breathe
111       BL   check_click
112       ADDS R0, #0               ; register used to check
113       BNE  return_20            ; compare, and if clicked, branch to "return"
```

main.s | Startup.s

```
103
104 ; Creates Duty Cycle of 20% of 8 Hz
105 ; Modifies: R0, R1, R2, (R3)
106
107 DC_20
108   PUSH {LR,R4}                    ; Preserve 8-byte alignment by pushing and popping an even number of registers
109 loop_20
110       BL   breathe
111       BL   check_click
112       ADDS R0, #0                 ; register used to check
113       BNE  return_20              ; compare, and if clicked, branch to "return"
114
115
116       BL   PortE_In
117       ORR  R2, R0, #0xFFFFFFFE    ; Get the output bit
118       MVN  R2, R2                 ; Toggle output
119       BIC  R0, R0, #0x01          ; Clear old output
120       ADD  R0, R0, R2             ; Place new output
121       BL   PortE_Out
122
123       ANDS R1, R0, #0x01          ; decide whether to stall high or low
124       BNE  stall_high20           ; 25 ms
125       B    stall_low20            ; 100 ms
126
127
128 stall_high20            ; Delay for the high portion of 20% Duty Cycle: 25ms
129       MOV  R1, #5
130 stall_a
131       MOV  R0, #65000
132 stall_b
133       SUBS R0, #1
134       BNE  stall_b
135       SUBS R1, R1, #1
136       BNE  stall_a
137
```

main.s | Startup.s

```
138       B    loop_20
139
140 stall_low20          ; Delay for the low portion of 20% Duty Cycle: 100ms
141       MOV  R1, #35
142 stall_c
143       MOV  R0, #65000
144 stall_d
145       SUBS R0, R0, #1              ; Delay the toggle
146       BNE  stall_d
147       SUBS R1, R1, #1              ; Do this 20 times for 1/16 of a second
148       BNE  stall_c
149
150       B    loop_20                 ; Branch back to check if switch is pressed
151
152 return_20
153   POP  {LR,R4}
154       BX   LR
155
156                                    ; ; ; ; ; ; ; ; ;
157                                    ; Duty Cycle 40 ;
158                                    ; ; ; ; ; ; ; ; ;
159
160 ; Creates Duty Cycle of 40% of 8 Hz;
161 DC_40
162   PUSH {LR,R4}
163 loop_40
164       BL   breathe
165       BL   check_click
166       ADDS R0, #0                  ; register used to check
167       BNE  return_40               ; compare, and if clicked, branch to "return"
168
169
170       BL   PortE_In
171       ORR  R2, R0, #0xFFFFFFFE     ; Get the output bit
172       MVN  R2, R2                  ; Toggle output
```

main.s | Startup.s

```
169
170         BL   PortE_In
171         ORR  R2, R0, #0xFFFFFFFE     ; Get the output bit
172         MVN  R2, R2                  ; Toggle output
173         BIC  R0, R0, #0x01           ; Clear old output
174         ADD  R0, R0, R2              ; Place new output
175         BL   PortE_Out
176
177         ANDS R1, R0, #0x01           ; decide whether to stall high or low
178         BNE  stall_high40            ; 50 ms
179         B    stall_low40             ; 75 ms
180
181
182   stall_high40            ; Delay for the high portion of Duty Cycle
183         MOV  R1, #12
184   stall_e
185         MOV  R0, #56000
186   stall_f
187         SUBS R0, #1
188         BNE  stall_f
189         SUBS R1, R1, #1
190         BNE  stall_e
191
192         B    loop_40
193
194   stall_low40             ; Delay for the low portion of Duty Cycle
195         MOV  R1, #18
196   stall_g
197         MOV  R0, #56000
198   stall_h
199         SUBS R0, R0, #1              ; Delay the toggle
200         BNE  stall_h
201         SUBS R1, R1, #1              ; Do this 20 times for 1/16 of a second
202         BNE  stall_g
203
```

Stellaris ICDI

main.s | Startup.s

```
204         B    loop_40                    ; Branch back to check if switch is pressed
205
206 return_40
207   POP  {LR,R4}
208         BX   LR
209
210                                     ; ; ; ; ; ; ; ; ;
211                                     ; Duty Cycle 60 ;
212                                     ; ; ; ; ; ; ; ; ;
213
214 ; Creates Duty Cycle of 60% of 8 Hz;
215 DC_60
216   PUSH {LR,R4}
217 loop_60
218         BL   breathe
219         BL   check_click
220         ADDS R0, #0                   ; register used to check
221         BNE  return_60                ; compare, and if clicked, branch to "return"
222
223
224         BL   PortE_In
225         ORR  R2, R0, #0xFFFFFFFE      ; Get the output bit
226         MVN  R2, R2                   ; Toggle output
227         BIC  R0, R0, #0x01            ; Clear old output
228         ADD  R0, R0, R2               ; Place new output
229         BL   PortE_Out
230
231         ANDS R1, R0, #0x01            ; decide whether to stall high or low
232         BNE  stall_high60             ; 75 ms
233         B    stall_low60              ; 50 ms
234
235
236 stall_high60             ; Delay for the high portion of Duty Cycle
237         MOV  R1, #29
238 stall_i
```

Translate the currently active file | Stellaris ICDI

main.s | Startup.s

```
234
235
236 stall_high60             ; Delay for the high portion of Duty Cycle
237       MOV  R1, #29
238 stall_i
239       MOV  R0, #60000
240 stall_j
241       SUBS R0, #1
242       BNE  stall_j
243       SUBS R1, R1, #1
244       BNE  stall_i
245
246       B    loop_60
247
248 stall_low60              ; Delay for the low portion of Duty Cycle
249       MOV  R1, #11
250 stall_k
251       MOV  R0, #60000
252 stall_l
253       SUBS R0, R0, #1              ; Delay the toggle
254       BNE  stall_l
255       SUBS R1, R1, #1              ; 8 repeats = 25 ms
256       BNE  stall_k
257
258       B    loop_60                 ; Branch back to check if switch is pressed
259
260 return_60
261   POP  {LR,R4}
262       BX   LR
263
264                                    ; ; ; ; ; ; ; ; ;
265                                    ; Duty Cycle 80 ;
266                                    ; ; ; ; ; ; ; ; ;
267
268 ; Creates Duty Cycle of 80% of 8 Hz;
```

Open an existing document | Stellaris ICDI

main.s | Startup.s

```
267
268 ; Creates Duty Cycle of 80% of 8 Hz;
269 DC_80
270   PUSH {LR,R4}
271 loop_80
272       BL   breathe
273       BL   check_click
274       ADDS R0, #0                ; register used to check
275       BNE  return_80             ; compare, and if clicked, branch to "return"
276
277
278       BL   PortE_In
279       ORR  R2, R0, #0xFFFFFFFE   ; Get the output bit
280       MVN  R2, R2                ; Toggle output
281       BIC  R0, R0, #0x01         ; Clear old output
282       ADD  R0, R0, R2            ; Place new output
283       BL   PortE_Out
284
285       ANDS R1, R0, #0x01         ; decide whether to stall high or low
286       BNE  stall_high80
287       B    stall_low80
288
289
290 stall_high80         ; Delay for the high portion of Duty Cycle
291       MOV  R1, #40
292 stall_m
293       MOV  R0, #61800
294 stall_n
295       SUBS R0, #1
296       BNE  stall_n
297       SUBS R1, R1, #1
298       BNE  stall_m
299
300       B    loop_80
301
```

main.s | Startup.s

```
301
302 stall_low80          ; Delay for the low portion of Duty Cycle
303       MOV  R1, #10
304 stall_o
305       MOV  R0, #61800
306 stall_p
307       SUBS R0, R0, #1             ; Delay the toggle
308       BNE  stall_p
309       SUBS R1, R1, #1             ; 8 repeats = 25 ms
310       BNE  stall_o
311
312       B    loop_80                  ; Branch back to check if switch is pressed
313
314 return_80
315   POP  {LR,R4}
316       BX   LR
317
318                                   ; ; ; ; ; ; ; ; ; ;
319                                   ; Duty Cycle 100  ;
320                                   ; ; ; ; ; ; ; ; ; ;
321
322 ; Creates Duty Cycle of 100% of 8 Hz;
323 DC_100
324   PUSH {LR,R4}
325
326       LDR  R1, =GPIO_PORTE_DATA_R
327       LDR  R0, [R1]
328       ORR  R0, R0, #0x01
329       STR  R0, [R1]
330 loop_100
331       BL   breathe
332       BL   check_click
333       ADDS R0, R0, #0x00
334       BEQ  loop_100
335
```

Build target files | Stellaris ICDI

main.s | Startup.s

```
335
336   POP  {LR,R4}
337       BX   LR
338                                   ; ; ; ; ; ; ; ; ; ;
339                                   ;     Breathe     ;
340                                   ; ; ; ; ; ; ; ; ; ;
341
342 ; Activates "breathing" LED for as long as PF4 is low
343 ; PF4 negative logic: 0 if pressed, 1 if not pressed
344 ; Modifies:
345 breathe
346   PUSH {LR,R4,R5,R6,R7,R8}
347
348 ;;;;;;;;;;;;;;;;;;;;
349  BL   check_breathe ;
350  ADDS R0, R0, #0x00 ;
351  BNE  return_breathe;
352 ;;;;;;;;;;;;;;;;;;;;
353                     MOV  R3, #1                 ; Initializing the stalling registers
354                     MOV  R4, #39
355                     MOV  R5, #1
356                     MOV  R6, #-1
357
358 loop_breathe
359
360 ;;;;;;;;;;;;;;;;;;;;
361  BL   check_breathe ;
362  ADDS R0, R0, #0x00 ;
363  BNE  return_breathe;
364 ;;;;;;;;;;;;;;;;;;;;
365
366                  ; Starting new DC ;
367                     MOV  R7, #10                ; Do current DC for X number iterations
368                     ADDS R3, R3, R5             ; If either R3 or R4 become 0,
369                     BEQ  min up                 ; it's time to go backwards
```

main.s | Startup.s

```
366                 ; Starting new DC ;
367                    MOV  R7, #10                 ; Do current DC for X number iterations
368                    ADDS R3, R3, R5              ; If either R3 or R4 become 0,
369                    BEQ  min_up                  ; it's time to go backwards
370                    ADDS R4, R4, R6
371                    BEQ  max_down
372
373 continue_DC       ADDS R7, R7, #0
374                    BEQ  loop_breathe
375
376                    BL   PortE_In
377                    ORR  R2, R0, #0xFFFFFFFE     ; Get the output bit
378                    MVN  R2, R2                  ; Toggle output
379                    BIC  R0, R0, #0x01           ; Clear old output
380                    ADD  R0, R0, R2              ; Place new output
381                    BL   PortE_Out
382
383                    ANDS R1, R0, #0x01           ; decide whether to stall high or low
384                    BNE  stall_high_b
385                    B    stall_low_b
386
387 stall_high_b      MOV  R1, R3
388 stall_y           MOV  R0, #1000 ; <-- decrease for 80 Hz
389 stall_q           SUBS R0, #1
390                    BNE  stall_q
391                    SUBS R1, R1, #1              ; R3 is high stall counter
392                    BNE  stall_y
393                    B    continue_DC
394
395 stall_low_b       MOV  R1, R4
396 stall_x           MOV  R0, #1000
397 stall_r           SUBS R0, #1
398                    BNE  stall_r
399                    SUBS R1, R1, #1              ; R4 is low stall counter
400                    BNE  stall_x
```

main.s | Startup.s

```
401                     SUB  R7, R7, #1              ; a DC iteration after end of every low
402 ;;;;;;;;;;;;;;;;;;;;;
403  BL   check_breathe ;
404  ADDS R0, R0, #0x00 ;
405  BNE  return_breathe;
406 ;;;;;;;;;;;;;;;;;;;;;
407                     B    continue_DC
408
409 min_up             MVN  R5, R5                 ; Now, we add the stall registers in the
410                     ADD  R5, #1                 ; opposite direction by switching the signs
411                     MVN  R6, R6                 ; of their in- and de-crement registers
412                     ADD  R6, #1
413                     BL   LED_off                ; in this case, keep the LED off for a while
414                     ; stalling a little
415                     MOV  R1, #10
416 stall_s            MOV  R0, #1000
417 stall_t            SUBS R0, #1
418                     BNE  stall_t
419                     SUBS R1, R1, #1
420                     BNE  stall_s
421                     ; reinitialize the registers
422                     MOV  R3, #1
423                     MOV  R4, #39
424
425                     B    continue_DC
426
427 max_down            MVN  R5, R5
428                     ADD  R5, #1
429                     MVN  R6, R6
430                     ADD  R6, #1
431                     BL   LED_on
432                     ; stalling a little
433                     MOV  R1, #10
434 stall_u            MOV  R0, #1000
435 stall_v            SUBS R0, #1
```

Build target files | Stellaris ICDI

main.s | Startup.s

```
432                     ; stalling a little
433                     MOV  R1, #10
434 stall_u             MOV  R0, #1000
435 stall_v             SUBS R0, #1
436                     BNE  stall_v
437                     SUBS R1, R1, #1
438                     BNE  stall_u
439                     ; reinitialize the registers, but backwards
440                     MOV  R3, #39
441                     MOV  R4, #1
442
443                     B    continue_DC
444
445 return_breathe
446   POP  {R8,R7,R6,R5,R4,LR}
447       BX   LR
448
449
450 ;------------check_click-------------
451 ; Checks if the button PE1 was pressed AND released too
452 ; Keeps track of a counter that tells if the button was pressed since
453 ; before the beginning of the subroutine
454 ; Modifies: R1, R2, R3
455 ; Inputs: none
456 ; Output: R0, 1 if yes, 0 if no
457 check_click
458
459       LDR  R3, =cc                  ; Address of click counter
460       LDR  R1, =GPIO_PORTE_DATA_R
461
462 ; check for button pressed already (cc=#1)
463       LDR  R0, [R3]
464       ADDS R0, #0
465       BNE  release                  ; If pressed alread, check for released
466
```

Stellaris ICDI

main.s | Startup.s

```
466
467 ; check for new press
468       LDR  R0, [R1]                 ; Read and check PE1
469       ANDS R2, R0, #0x02
470       BEQ  no_click
471       MOV  R2, #0x01
472       STR  R2, [R3]
473
474 release
475       LDR  R0, [R1]                 ; Check if released now
476       ANDS R2, R0, #0x02
477       BNE  no_click
478       MOV  R0, #0x00                ; Clear counter
479       STR  R0, [R3]
480       MOV  R0, #0x01                ; yes signal
481       BX   LR
482
483 no_click
484       MOV  R0, #0x00                ; no signal
485       BX   LR
486
487 ;-----------check_breathe------------
488 ; Checks PF4 and returns whether its button
489 ; (negative logic) is high or low
490 ; If button is pressed (PF4 = 0), R0 = 0
491 ; If button isn't pressed (PF4 = 1), R0 = 1
492 ; Inputs: none
493 ; Outputs: R0 (1 or 0)
494 ; Modifies: R0, R1
495 check_breathe
496       LDR  R1, =GPIO_PORTF_DATA_R
497       LDR  R0, [R1]
498       ANDS R0, R0, #0x10            ; check PF4
499       BEQ  return_check_breathe     ; If PF4 = zero, that will be the output anyway
500       MOV  R0, #0x01
```

Stellaris ICDI

main.s | Startup.s

```
499         BEQ  return_check_breathe      ; If PF4 = zero, that will be the output anyway
500         MOV  R0, #0x01
501 return_check_breathe
502         BX   LR
503
504 ;------------LED_off-----------------
505 ; Turns the LED on Port E off
506 ; Modifies: R0, R1
507 LED_off
508         LDR  R1, =GPIO_PORTE_DATA_R
509         LDR  R0, [R1]
510         BIC  R0, #0x01
511         STR  R0, [R1]
512         BX   LR
513
514 ;------------LED_on-----------------
515 ; Turns the LED on Port E on (full brightness)
516 ; Modifies: R0, R1
517 LED_on
518         LDR  R1, =GPIO_PORTE_DATA_R
519         LDR  R0, [R1]
520         ORR  R0, #0x01
521         STR  R0, [R1]
522         BX   LR
523
524 ;------------PortE_In---------------
525 ; Returns R0 with the value of Port E's Data register
526 PortE_In
527         LDR  R1, =GPIO_PORTE_DATA_R
528         LDR  R0, [R1]
529         BX   LR
530
531 ;------------PortE_Out--------------
532 ; Stores R0 into the Port E Data register
533 PortE_Out
```

Stellaris ICDI

main.s | Startup.s

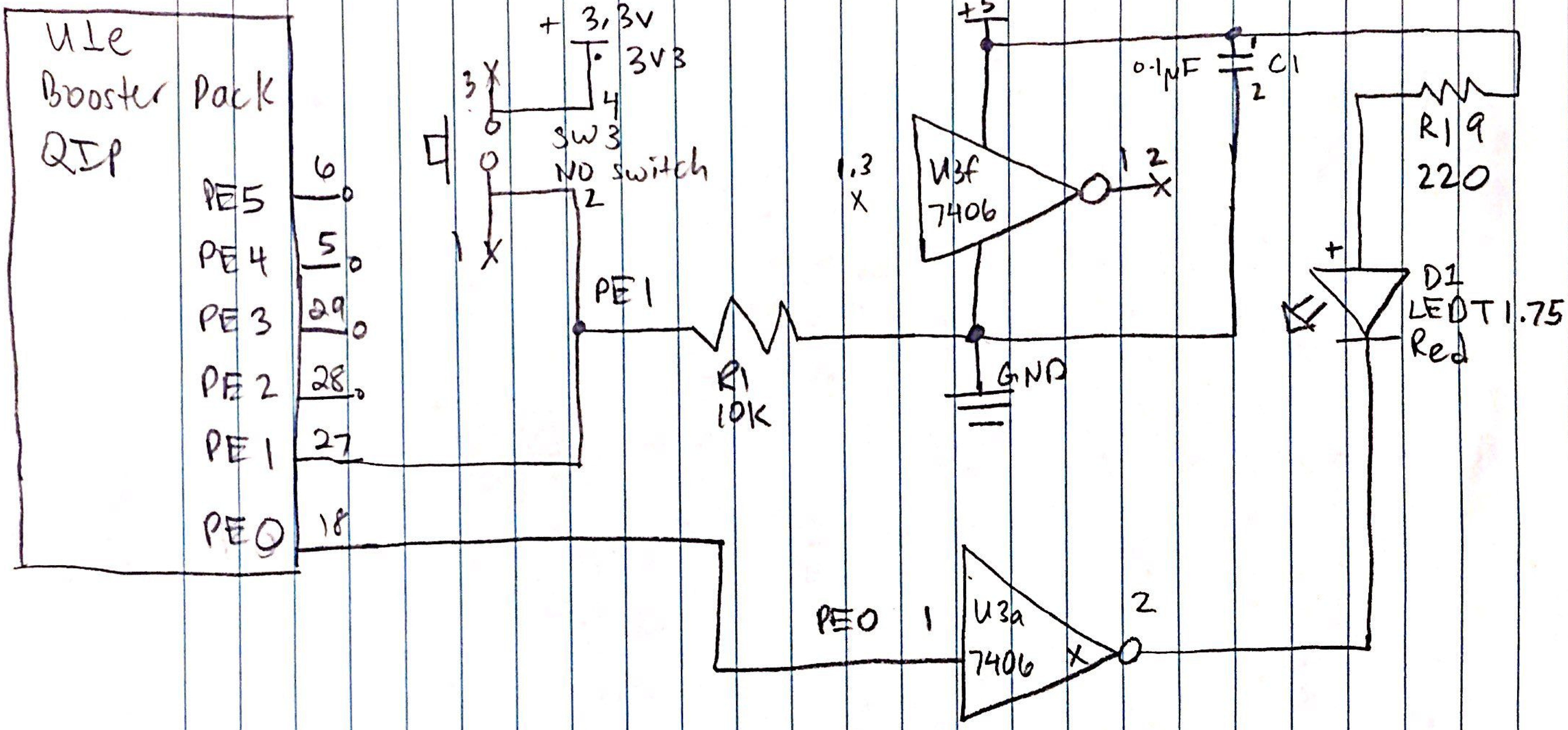
```
530
531 ;------------PortE_Out---------------
532 ; Stores R0 into the Port E Data register
533 PortE_Out
534       LDR  R1, =GPIO_PORTE_DATA_R
535       STR  R0, [R1]
536       BX   LR
537
538 ;------------PortF_E_Init------------
539 ; Initialize GPIO Port F for negative logic switches on PF0 and
540 ; PF4 as the Launchpad is wired.  Weak internal pull-up
541 ; resistors are enabled, and the NMI functionality on PF0 is
542 ; disabled.  Make the RGB LED's pins outputs.
543 ; Initialize GPIO Port E. PE1 is positive logic input.
544 ; and PE0 is positive logic output.
545 ; Input: none
546 ; Output: none
547 ; Modifies: R0, R1, R2
548 PortF_E_Init
549     LDR R1, =SYSCTL_RCGCGPIO_R      ; 1) activate clock for Port F and E
550     LDR R0, [R1]
551     ORR R0, R0, #0x30               ; set bit 5 and 6 to turn on clock
552     STR R0, [R1]
553     NOP
554     NOP                             ; allow time for clock to finish
555     LDR R1, =GPIO_PORTF_LOCK_R      ; 2) unlock the lock register
556     LDR R0, =0x4C4F434B             ; unlock GPIO Port F Commit Register
557     STR R0, [R1]
558     LDR R1, =GPIO_PORTF_CR_R        ; enable commit for Port F
559     MOV R0, #0xFF                   ; 1 means allow access
560     STR R0, [R1]
561    ;LDR R1, =GPIO_PORTF_AMSEL_R     ; 3) disable analog functionality
562    ;MOV R0, #0                      ; 0 means analog is off
563    ;STR R0, [R1]
564    ;LDR R1, =GPIO_PORTF_PCTL_R      ; 4) configure as GPIO
```

Stellaris ICDI

main.s | Startup.s

```
558      LDR R1, =GPIO_PORTF_CR_R        ; enable commit for Port F
559      MOV R0, #0xFF                   ; 1 means allow access
560      STR R0, [R1]
561     ;LDR R1, =GPIO_PORTF_AMSEL_R     ; 3) disable analog functionality
562     ;MOV R0, #0                      ; 0 means analog is off
563     ;STR R0, [R1]
564     ;LDR R1, =GPIO_PORTF_PCTL_R      ; 4) configure as GPIO
565     ;MOV R0, #0x00000000             ; 0 means configure Port F as GPIO
566     ;STR R0, [R1]
567      LDR R1, =GPIO_PORTF_DIR_R       ; 5) set direction register
568      MOV R0,#0x0E                    ; PF0 and PF7-4 input, PF3-1 output
569      STR R0, [R1]
570      LDR R1, =GPIO_PORTF_AFSEL_R     ; 6) regular port function
571      MOV R0, #0                      ; 0 means disable alternate function
572      STR R0, [R1]
573      LDR R1, =GPIO_PORTF_PUR_R       ; pull-up resistors for PF4,PF0
574      MOV R0, #0x11                   ; enable weak pull-up on PF0 and PF4
575      STR R0, [R1]
576      LDR R1, =GPIO_PORTF_DEN_R       ; 7) enable Port F digital port
577      MOV R0, #0xFF                   ; 1 means enable digital I/O
578      STR R0, [R1]
579  ;PortE_Init
580      LDR R1, =GPIO_PORTE_DIR_R       ; 1) set direction register
581      MOV R0,#0x01                    ; PE1 input, PE0 output
582      STR R0, [R1]
583      LDR R1, =GPIO_PORTE_DEN_R       ; 3) enable Port E digital port
584      MOV R0, #0x03                   ; 1 means enable digital I/O
585      STR R0, [R1]
586      BX  LR
587
588
589       ALIGN      ; make sure the end of this section is aligned
590       END        ; end of file
591
592
```

Create an empty document | Stellaris ICDI



Fawadul Rafuel

Hag Henegon

C:\Users\FAWADUL\Desktop\College\UT Fall 2017\EE 319K\Lab 3 Old\lab-3-fawadul-h-and-rafael-h\Lab3.uvproj - µVision4

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

Registers

Register	Value
Core	
R0	0x00006224
R1	0x00000001
R2	0x00000001
R3	0x20000030
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20000468
R14 (LR)	0x0000034D
R15 (PC)	0x0000035C

Logic Analyzer

Setup... Load... Save... | Min Time 0 s | Max Time 1.434893 s | Grid 0.2 s | Zoom In Out All | Min/Max Auto Undo | Update Screen Stop Clear | Transition Prev Next | Jump to Code Trace | Signal Info | Amplitude | Show Cycles | Cursor

(porte & 0x00000001)

1

0

0.626253 s

0 --> 1

0.793409 s

1.466253 s

(porte & 0x00000001)	Mouse Pos	Reference Point	Delta
Time:	0.917453 s	0.793409 s	0.124044 s = 8.06165 Hz
Value:	1	1	0
PC $:	0x58a	0x58a	

Disassembly | Logic Analyzer

main.s | Startup.s

```
184  stall_e
185        MOV  R0, #61800
186  stall_f
     0, #1
     tall_f
     1, R1, #1
     tall_e

     oop_40

                ; Delay for the low portion of Duty Cy
     1, #24
```

TExaS Lab 3

Port E Hardware

TM4C123

SW (+3.3) PE1 PE0 LED

Port E Registers

DATA:	0x01	PUR:	0x00	LOCK:	0x01
DIR:	0x01	PDR:	0x00	CR:	0xFF
DEN:	0x03	RCGCGPIO:	0x00000039	Clock enabled	

Grading Controls

Number from EdX: Grade Score: 0

TExaS Lab3

Port F Hardware

TM4C123

SW1 (+3.3) PF4 PF3 PF2 PF1 PF0 SW2 LED LED LED

Port F Registers

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

Memory 1

Address: 0x20000000

```
0x20000000: 000C3                                                    00000000
0x2000001C: 00000                                                    00000000
0x20000038: 00000000 00000000 31334545 20204B39 00000003 00000000 00000000
```

reakSet BreakAccess COVERAGE

Simulation | t1: 1.43489713 sec | L:187 C:1 | CAP NUM SCRL OVR R/W

Type here to search

2:19 PM 9/27/2017

C:\Users\FAWADUL\Desktop\College\UT Fall 2017\EE 319K\Lab 3 Old\lab-3-fawadul-h-and-rafael-h\Lab3.uvproj - µVision4

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

Registers

Register	Value
Core	
R0	0x000084DE
R1	0x00000005
R2	0x00000000
R3	0x20000030
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20000468
R14 (LR)	0x000003A3
R15 (PC)	0x000003C6

Logic Analyzer

Setup... Load... Save... | Min Time 0 s | Max Time 4.539394 s | Grid 0.2 s | Zoom In Out All | Min/Max Auto Undo | Update Screen Stop Clear | Transition Prev Next | Jump to Code Trace | Signal Info | Amplitude | Show Cycles | Cursor

(porte & 0x00000001)

1

0

3.730754 s

0 -> 1

3.935437 s

4.570754 s

(porte & 0x00000001)	Mouse Pos	Reference Point	Delta
Time:	4.059554 s	3.935437 s	0.124117 s = 8.056922 Hz
Value:	1	1	0
PC $:	0x58a	0x58a	

Disassembly | Logic Analyzer

main.s | Startup.s

```
251         MOV  R0, #61800
252  stall_1
253         SUBS R0, R0, #1          ; Delay the toggle
            tall_1
            1, R1, #1                ; 8 repeats = 25 ms
            tall_k

            oop_60                   ; Branch back to check if s

            4}
            R
```

TExaS Lab 3

Port E Hardware

TM4C123

SW +3.3 PE1 PE0 LED

Port E Registers

DATA:	0x00	PUR:	0x00	LOCK:	0x01
DIR:	0x01	PDR:	0x00	CR:	0xFF
DEN:	0x03	RCGCGPIO:	0x00000039	Clock enabled	

Grading Controls

Number from EdX: Grade Score: 0

TExaS Lab3

Port F Hardware

TM4C123

SW1 +3.3 PF4

SW2 +3.3 PF0

PF3 PF2 PF1 LED LED LED

Port F Registers

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

Memory 1

Address: 0x20000000

```
0x20000000: 000C3                                                    00000000
0x2000001C: 00000                                                    00000000
0x20000038: 00000000 00000000 31334545 20204B39 00000003 00000000 00000000
```

reakSet BreakAccess COVERAGE

Simulation | t1: 4.53939795 sec | L:254 C:1 | CAP NUM SCRL OVR R/W

Type here to search

2:20 PM 9/27/2017