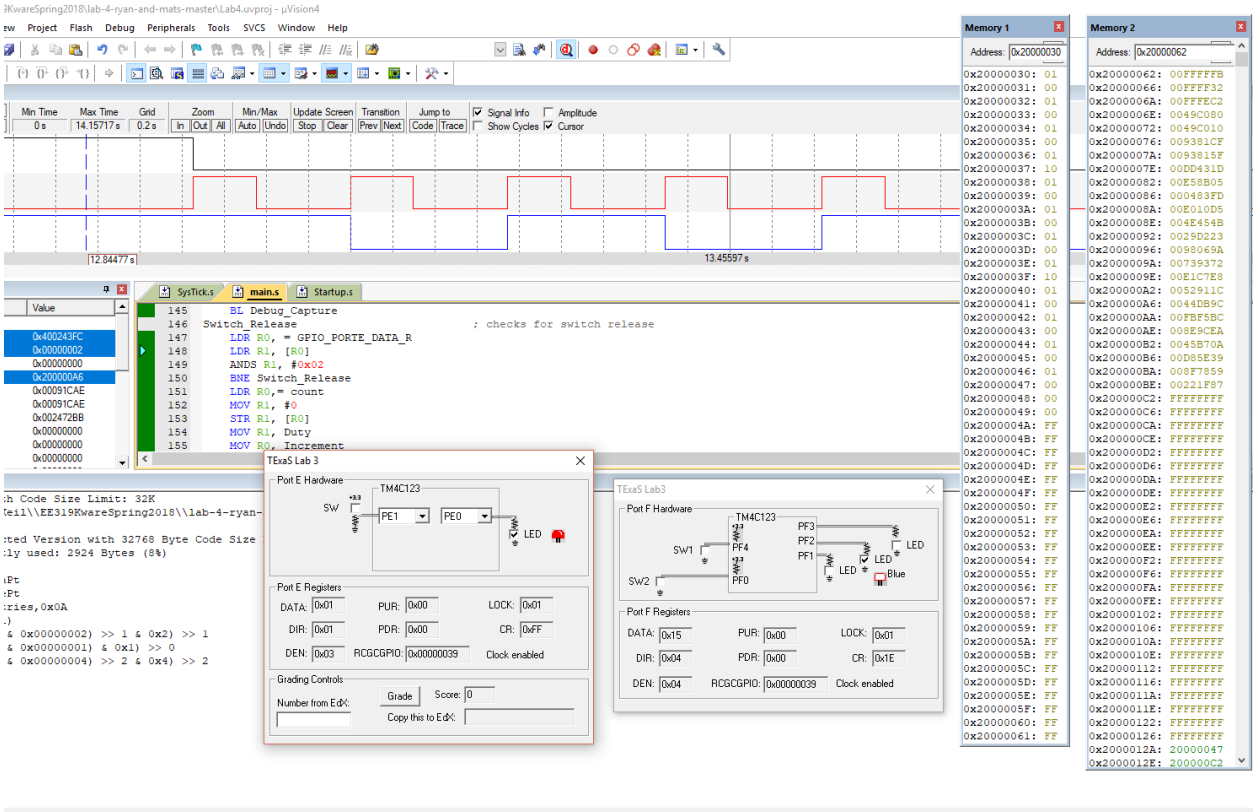
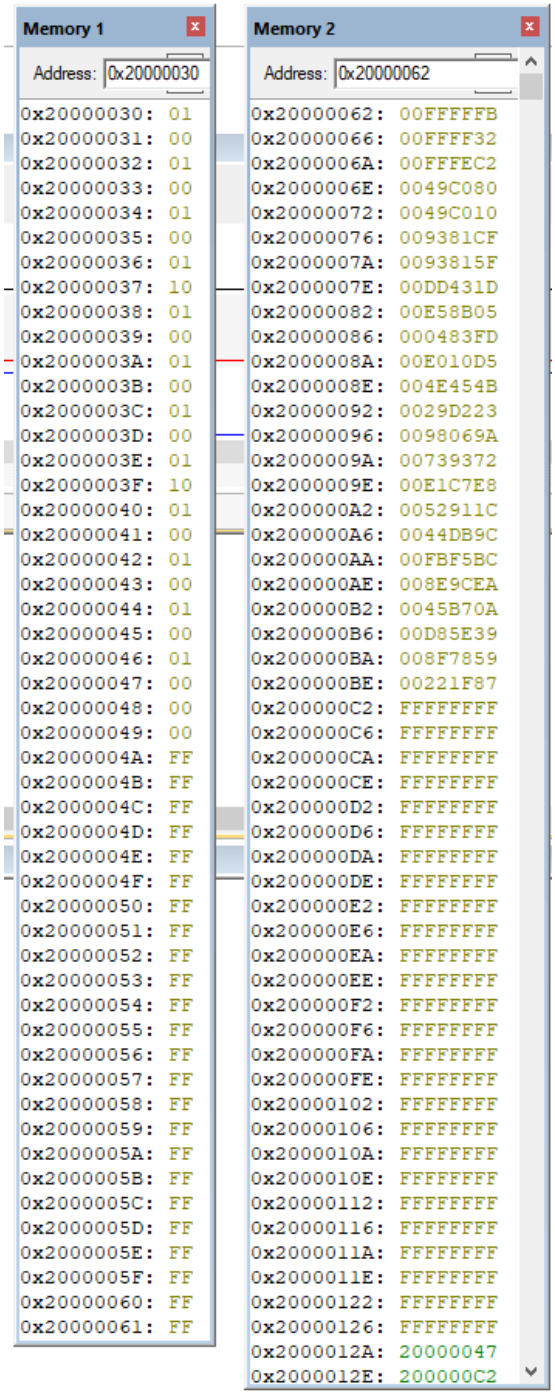


1 kDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE DEFINE DIR Display Enter EVALuate EXIT FUNC Go INCLUDE IRLOG ITMLOG KILL LogicAnalyze LOAD LOG MAP MODE Ostep Perf



:LL LogicAnalyze LOAD LOG MAP MODE Ostep Perf

```
;****************** main.s ***************
; Program written by: Mats Oosterlaken & Ryan Taylor
; Date Created: 2/14/2017
; Last Modified: 2/26/2018
; Brief description of the program
;   The LED toggles at 8 Hz and a varying duty-cycle
;   Repeat the functionality from Lab3 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
; You will only verify the variable duty-cycle feature of Lab 3 and not the "breathing" feature.
; 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
   EIGTH_SECOND EQU 2985833
   NVIC_ST_CURRENT_R EQU 0xE000E018
; 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            ; initializing array of 50, 32 bit ints
TimeBuffer      SPACE 50*4          ; initializing array of 50, 32 bit ints
DataPt          SPACE 4             ; pointer to DataBuffer
TimePt          SPACE 4             ; pointer to TimeBuffer
NEntries        SPACE 4             ; Entry counter
count           SPACE 4             ; time difference counter

; ROM Area
    IMPORT TExaS_Init        IMPORT SysTick_Init
;-UUU-Import routine(s) from other assembly files (like SysTick.s) here
    AREA    |.text|, CODE, READONLY, ALIGN=2         THUMB
    EXPORT Start              EXPORT TimePt
Start
        ; TExaS_Init sets bus clock at 80 MHz
        BL TExaS_Init ; voltmeter, scope on PD3
        ;place your initializations here
          BL Debug_Init
        CPSIE I  ; TExaS voltmeter, scope runs on interrupts
        BL LED_Init          ; initializes clock, DEN, DIR

    Increment RN 4      ; A constant value of delay that will be used to increment the
    duty, its 20% duty and this 1/40th of a second
    Duty      RN 5      ; keeps track of the percent duty
    offDuty RN 6        ; keeps track of the opposite perccent duty.
    LDR Duty,= 0x00          LDR offDuty,= EIGTH_SECOND   MOV R0, #0x05          SDIV Increment, offDuty, R0
     LDR R0,= count          MOV R1, #0     STR R1, [R0]
loop                                                    ; the Main loop
```

```
92              BL Heart_Beat
 93             BL Change_Duty          ; checks if the switch PE1 has been pressed and increments
        accordingly
 94             BL Check_Duty           ; checks if the Duty is more than 100 or 0 and acts accordingly 95      MOV R1, #0x01
96              BL LED_Out
97                                                      ; Turn on LED
98                                                      ;CMP Duty, R1
99                                                      ;BEQ loop
100                                                     MOV R0, Duty
101                                                     BL Delay    ; activates a delay of the duty
102                                                     MOV R1, #0x00
103                                                     BL LED_Out          ; Turn off LED
104                                                     MOV R0, offDuty
105                                                     BL Delay    ; activates a delay of the off duty
106     B loop 107  108  Heart_Beat 109     LDR R0, =GPIO_PORTF_DATA_R 110      LDR R1, [R0] 111            EOR R1, #0x04 112
STR R1, [R0] 113            BX LR 114
115 LED_Out                                                     ; Subroutine: changes the value of the LED according to the
        value of R1 (Uses R1, Clears R0, R1, R2)
116     PUSH {R0,LR} 117        LDR R0, =GPIO_PORTE_DATA_R    ; toggles PE0 118   LDR R2, [R0] 119            BIC R2, #0x01 120
ORR R2, R1 121    STR R2, [R0] 122          LDR R1,= EIGTH_SECOND 123     LDR R0,= count 124        LDR R12, [R0] 125          CMP R12,
#7 126    BEQ capture_skip 127      BL Debug_Capture 128      ADD R12, #1 129       STR R12,[R0] 130  capture_skip 131   POP {R0,PC} 132
BX LR 133
134 Delay                                                       ; Subroutine: delay of the value loaded into R0 (Uses R0,
        Clears R0)
135     SUBS R0, R0, #0x01 136      BPL Delay 137     BX LR 138
139     Change_Duty         ; Subroutine: checks if the switch PE1 has been pressed and
        increments accordingly (Clears R0, R1, R2)
140     PUSH {R0, LR} 141       LDR R0, = GPIO_PORTE_DATA_R 142       LDR R1, [R0] 143          ANDS R1, #0x02 144        BEQ
        no_change 145     BL Debug_Capture
146 Switch_Release                                              ; checks for switch release
147     LDR R0, = GPIO_PORTE_DATA_R 148       LDR R1, [R0] 149          ANDS R1, #0x02 150        BNE Switch_Release 151    LDR R0,=
count 152         MOV R1, #0 153   STR R1, [R0] 154          MOV R1, Duty 155        MOV R0, Increment 156     MOV R2, offDuty
157             ADD R1, R0  ; Duty gets incremented by 20%
158             SUBS R2, R0 ; offDuty gets decremented by 20%
159     MOV Duty, R1 160        MOV offDuty, R2 161   no_change 162       POP {R0, PC} 163          BX LR 164
165     Check_Duty          ; Subroutine: checks if the Duty is more than 100 or 0 and acts
        accordingly (Clears R1)
166     ;CMP Duty, #0x00   ; if duty = 0 it leaves it off 167          ;BEQ loop
168     LDR R1,= EIGTH_SECOND 169     CMP R1, Duty
170     BGE notnegative    ; if duty is greater than 100% then reset duty and offduty 171       MOV Duty, #0x00
172     LDR offDuty, =EIGTH_SECOND 173   notnegative 174   BX LR 175
176     LED_Init  ; Subroutine: initializes clock, DEN, DIR (Clears R0, R1)
177     LDR R0, =SYSCTL_RCGCGPIO_R      ; initialize clock 178        LDR R1, [R0] 179          ORR R1, R1, #0x30 180     STR R1, [R0] 181
        NOP 182         NOP
183             LDR R0, =GPIO_PORTE_DIR_R ; allow Output direction for port PE0
184             LDR R1, [R0] 185      ORR R1, R1, #0x01 186     STR R1, [R0]
187             LDR R0, =GPIO_PORTE_DEN_R           ; Digital enable for PE0 and PE1
188             LDR R1, [R0] 189      ORR R1, R1, #0x03 190     STR R1, [R0] 191
192             LDR R0, =GPIO_PORTF_DEN_R           ; Digital enable for PF4
193             LDR R1, [R0] 194      ORR R1, R1, #0x04 195     STR R1, [R0]
196             LDR R0, =GPIO_PORTF_DIR_R ; allow Output direction for port PE0
197             LDR R1, [R0] 198      ORR R1, R1, #0x04 199     STR R1, [R0] 200          BX LR 201   202   Debug_Init 203     PUSH {R0-R4, LR}
        204         LDR R0,= DataBuffer 205   LDR R1,= TimeBuffer 206   MOV R2, #0xFF
207                     MOV R3, #0xFFFFFFFF
208                 MOV R4, #0
209                 loop1       STR R2,[R0]       ; setting DataBuffer[0] - DataBuffer[50] to 0xFF
210                 ADD R0, #1
211                 STR R3,[R1]         ; setting TimeBuffer[0] - TimeBuffer[50] to 0xFFFFFFFF
212                 ADD R1, #4 213      ADD R4, #1 214    CMP R4, #50 215         BNE loop1 216     LDR R0,= DataBuffer 217   LDR
                    R1,=DataPt
218                 STR R0,[R1]         ; initializing DataPt to beginning of DataBuffer
219                 LDR R0,= TimeBuffer 220       LDR R1,=TimePt
```

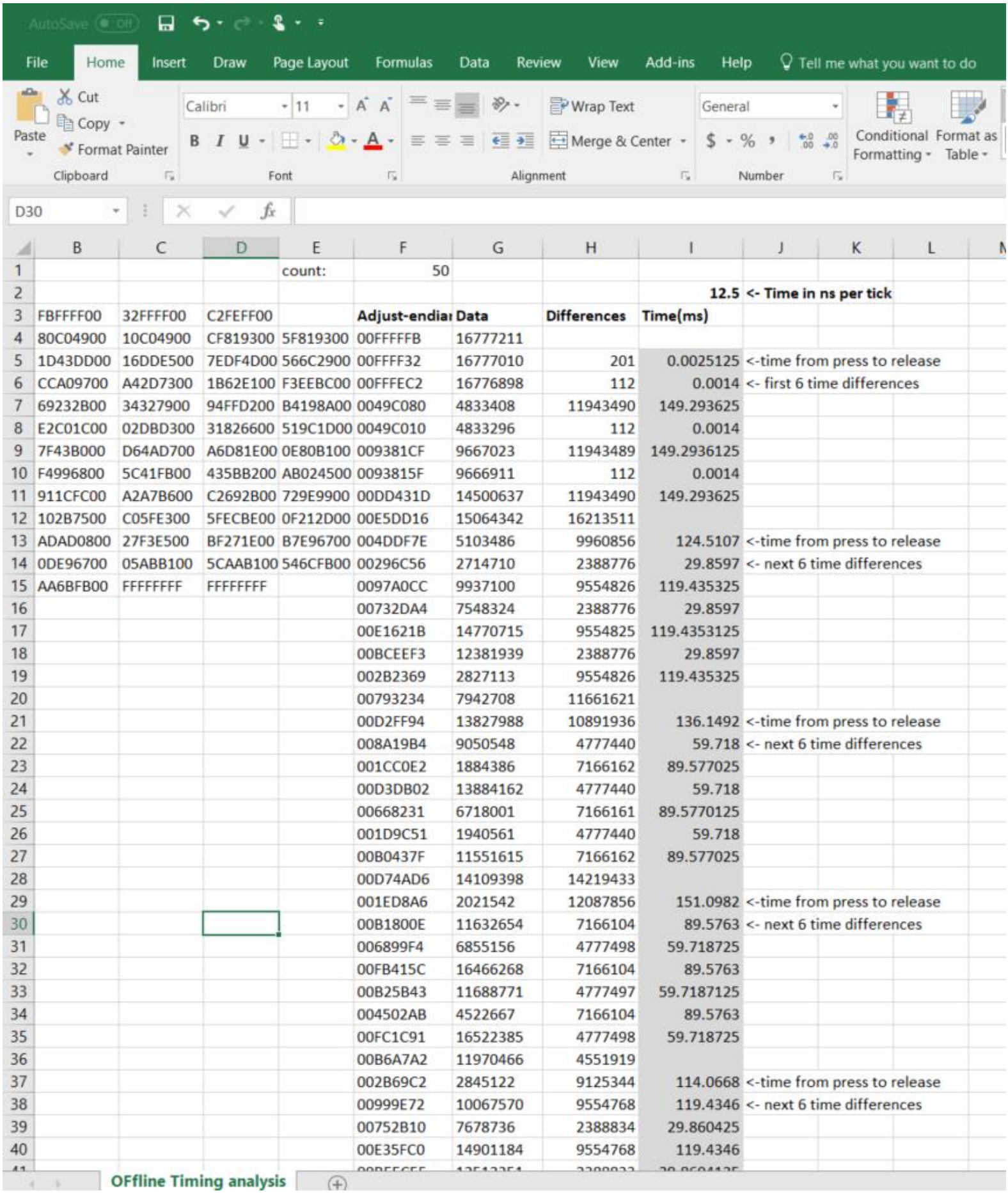
```
221                         STR R0,[R1 ]                                ; initializing TimePt to beginning of TimeBuffer
222       LDR R0,= NEntries 223       MOV R1 , #0
224                         STR R1,[R0]             ; initializing NEntries to 0
225                         BL SysTick_Init         ; activate SysTick Timer
226       POP {R0-R4, PC} 227         BX LR 228
229  Debug_Capture          ;30 instructions, 60 cycles; 750 nano seconds; 100*7.5E-7*8 = .0006% intrusiveness 230          PUSH {R0-R2, LR}
231       LDR R0,= NEntries 232       LDR R1,[R0] 233          CMP R1 , #50 234          BEQ done 235       LDR R0,= GPIO_PORTE_DATA_R 236
LDR R1,[R0] 237           AND R1,#0x03
238                         AND R2, R1,#0x01    ; R2 = PE0 input
239                         AND R3, R1,#0X02    ; R3 = PE1 output
240                         LSL R3, R3, #3
241                         ADD R1 , R2, R3     ; R1 = 000R2.000R3
242                           LDR R2,= DataPt
243                         LDR R3,[R2]         ; R3 = DataBuffer[i]
244                         STR R1,[R3]         ; DataBuffer[i] = 000R2.000R3
245                         ADD R3, #1 ; Incrementing DataPt
246                         STR R3,[R2]
247                         LDR R0,= NVIC_ST_CURRENT_R ;Dump time into TimeBuffer using the pointer TimePt
248                         LDR R0, [R0] 249    LDR R2,= TimePt 250       LDR R3, [R2] 251          STR R0, [R3]
252                         ADD R3, #4 ;increment time pointer
253                         STR R3, [R2] 254    LDR R0,= NEntries 255     LDR R1,[R0] 256           ADD R1 , #1
257                           STR R1 , [R0]                               ; Incrementing NEntries
258  done           POP {R0-R2, LR} 259       BX LR
260               ALIGN     ; make sure the end of this section is aligned
261               END       ; end of file
                  262       263
```

C:\Keil\EE319KwareSpring2018\lab-4-ryan-and-mats-master\SysTick.s

```
1  ; SysTick.s 2  ; Module written by: **-UUU-*Your Names**update this*** 3  ; Date Created: 2/14/2017 4  ; Last Modified: 2/12/2018 5  ; Brief
Description: Initializes SysTick 6
   7  NVIC_ST_CTRL_R              EQU
                                  0xE000E010
   8  NVIC_ST_RELOAD_R            EQU
                                  0xE000E014
   9  NVIC_ST_CURRENT_R           EQU
                                  0xE000E018
10
11   AREA    |.text|, CODE, READONLY, ALIGN=212       THUMB
13  ; -UUU- You add code here to export your routine(s) from SysTick.s to main.s
14   EXPORT SysTick_Init 15    IMPORT TimePt 16
17  ;------------SysTick_Init-----------18  ; ;-UUU-Complete this subroutine
19     ; Initialize SysTick running at bus clock.
20     ; Make it so NVIC_ST_CURRENT_R can be used as a 24-bit time 21  ; Input: none 22  ; Output: none
23  ; Modifies:
24  SysTick_Init
25  ; **-UUU-**Implement this function**** 26  PUSH {R0-R4, LR}    ;why ? this is uneccesary
27   LDR R0,= NVIC_ST_CTRL_R
28   MOV R1 , #0
29   STR R1,[R0]
30   LDR R0,= NVIC_ST_RELOAD_R
31  MOV R1 , #0x00FFFFFF
32   STR R1,[R0]
33   LDR R0,= NVIC_ST_CURRENT_R
34   MOV R1 , #0
35   STR R1,[R0]
36   LDR R0,= NVIC_ST_CTRL_R
37  MOV R1 , #0x00000005
```

```
38   STR R1,[R0]
39   LDR R0,= NVIC_ST_CURRENT_R    ;stores initial time
40   LDR R1, [R0]
41   LDR R2,=TimePt   ;external
42   LDR R3, [R2]
43   STR R1, [R3]
44   ADD R3, #4
45   STR R3, [R2]
46  POP {R0-R4, PC}
47              BX LR                                       ; return
48
49
50   ALIGN   ; make sure the end of this section is aligned 51    END      ; end of file
52
```

Page 1



	B	C	D	E	F	G	H	I	J	K	L
1				count:	50						
2								12.5	<- Time in ns per tick		
3	FBFFFF00	32FFFF00	C2FEFF00		Adjust-endia	Data	Differences	Time(ms)			
4	80C04900	10C04900	CF819300	5F819300	00FFFFFB	16777211					
5	1D43DD00	16DDE500	7EDF4D00	566C2900	00FFFF32	16777010	201	0.0025125	<-time from press to release		
6	CCA09700	A42D7300	1B62E100	F3EEBC00	00FFFEC2	16776898	112	0.0014	<- first 6 time differences		
7	69232B00	34327900	94FFD200	B4198A00	0049C080	4833408	11943490	149.293625			
8	E2C01C00	02DBD300	31826600	519C1D00	0049C010	4833296	112	0.0014			
9	7F43B000	D64AD700	A6D81E00	0E80B100	009381CF	9667023	11943489	149.2936125			
10	F4996800	5C41FB00	435BB200	AB024500	0093815F	9666911	112	0.0014			
11	911CFC00	A2A7B600	C2692B00	729E9900	00DD431D	14500637	11943490	149.293625			
12	102B7500	C05FE300	5FECBE00	0F212D00	00E5DD16	15064342	16213511				
13	ADAD0800	27F3E500	BF271E00	B7E96700	004DDF7E	5103486	9960856	124.5107	<-time from press to release		
14	0DE96700	05ABB100	5CAAB100	546CFB00	00296C56	2714710	2388776	29.8597	<- next 6 time differences		
15	AA6BFB00	FFFFFFFF	FFFFFFFF		0097A0CC	9937100	9554826	119.435325			
16					00732DA4	7548324	2388776	29.8597			
17					00E1621B	14770715	9554825	119.4353125			
18					00BCEEF3	12381939	2388776	29.8597			
19					002B2369	2827113	9554826	119.435325			
20					00793234	7942708	11661621				
21					00D2FF94	13827988	10891936	136.1492	<-time from press to release		
22					008A19B4	9050548	4777440	59.718	<- next 6 time differences		
23					001CC0E2	1884386	7166162	89.577025			
24					00D3DB02	13884162	4777440	59.718			
25					00668231	6718001	7166161	89.5770125			
26					001D9C51	1940561	4777440	59.718			
27					00B0437F	11551615	7166162	89.577025			
28					00D74AD6	14109398	14219433				
29					001ED8A6	2021542	12087856	151.0982	<-time from press to release		
30					00B1800E	11632654	7166104	89.5763	<- next 6 time differences		
31					006899F4	6855156	4777498	59.718725			
32					00FB415C	16466268	7166104	89.5763			
33					00B25B43	11688771	4777497	59.7187125			
34					004502AB	4522667	7166104	89.5763			
35					00FC1C91	16522385	4777498	59.718725			
36					00B6A7A2	11970466	4551919				
37					002B69C2	2845122	9125344	114.0668	<-time from press to release		
38					00999E72	10067570	9554768	119.4346	<- next 6 time differences		
39					00752B10	7678736	2388834	29.860425			
40					00E35FC0	14901184	9554768	119.4346			

41	00BEEC5F	12512351	2388833	29.8604125	
42	002D210F	2957583	9554768	119.4346	
43	0008ADAD	568749	2388834	29.860425	
44	00E5F327	15069991	2275974		
45	001E27BF	1976255	13093736	163.6717	<-time from press to release
46	0067E9B7	6810039	11943432	149.2929	<- next 6 time differences
47	0067E90D	6809869	170	0.002125	
48	00B1AB05	11643653	11943432	149.2929	
49	00B1AA5C	11643484	169	0.0021125	
50	00FB6C54	16477268	11943432	149.2929	
51	00FB6BAA	16477098	170	0.002125	
52	FFFFFFFF	4.295E+09	-4.262E+09		
53	FFFFFFFF	4.295E+09	0	0	
54					
55					