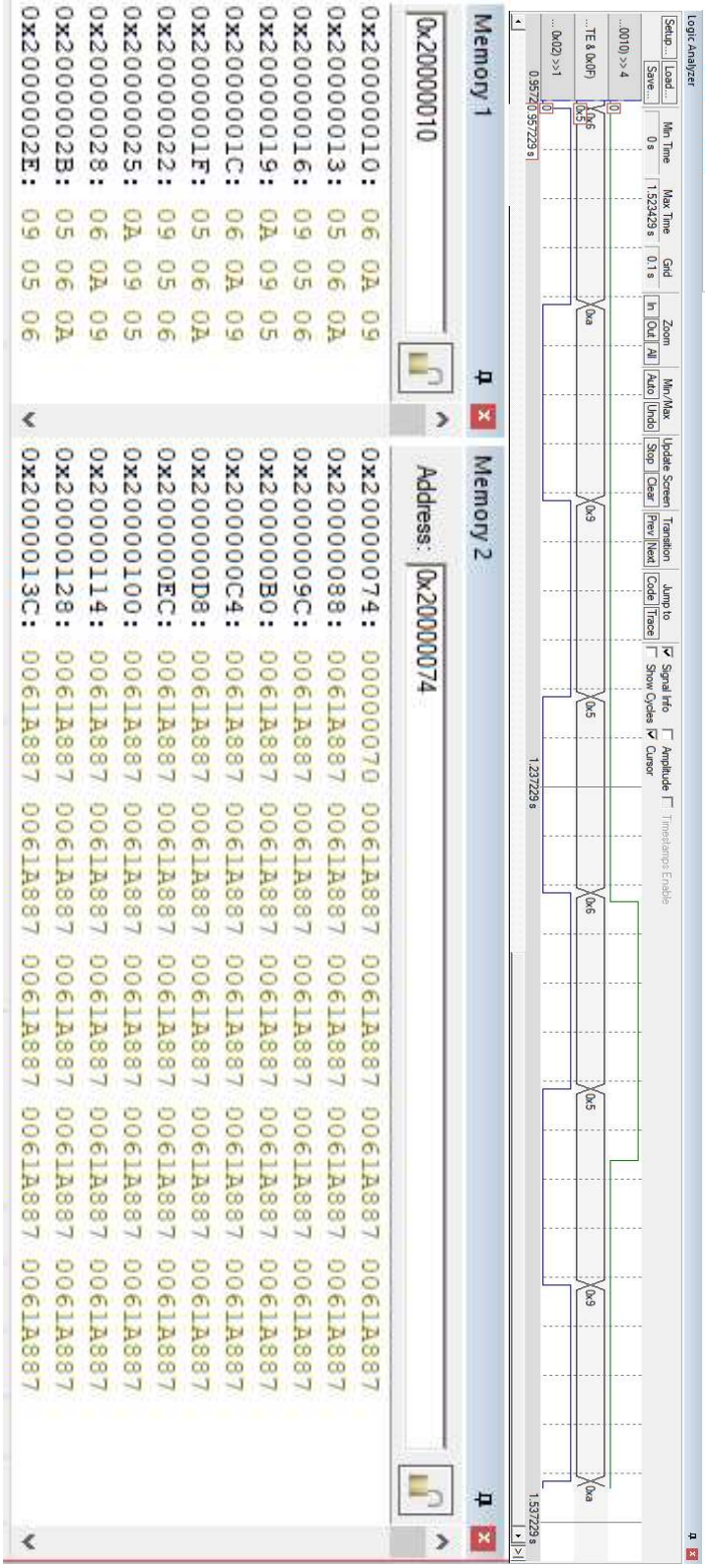
Lab 4 Deliverables

Eric Chen and Dennis Liu

2019-03-05

2. Screenshots



3. Source Code

```
22  SysTick_Init
23   ; **-UUU-**Implement this function****
24      LDR R0,=NVIC_ST_CTRL_R
25      MOV R1, #0
26      STR R1,[R0] ;disable Systick during setup
27
28      LDR R0,=NVIC_ST_RELOAD_R
29      LDR R1,=0x00FFFFFF
30      STR R1,[R0] ;max reload value
31
32      LDR R0,=NVIC_ST_CURRENT_R
33      MOV R1,#0
34      STR R1,[R0] ;any write to current clears
35
36      LDR R0,=NVIC_ST_CTRL_R
37      MOV R1,#0x05
38      STR R1,[R0]
39
40      BX  LR                          ; return
41
42
43      ALIGN                           ; make sure the end of this section is aligned
44      END                             ; end of file
```

```
77  loop
78          LDR  R1,=GPIO_PORTF_DATA_R
79          LDR  R4,[R1]
80          EOR  R4,R4,#0x02
81          STR  R4,[R1] ;toggle LED
82
```

```
Switch_Init
    ; activate clock for Port A
      LDR R1, =SYSCTL_RCGCGPIO_R
      LDR R0, [R1]
      ORR R0, R0, #0x21  ; Clock for A and F
      STR R0, [R1]
      NOP
      NOP                ; allow time to finish activating
    ; set direction register
      LDR R1, =GPIO_PORTA_DIR_R
      LDR R0, [R1]
      BIC R0, R0, #0x10    ; Input on PA4
      STR R0, [R1]
    ; 5) enable digital port
      LDR R1, =GPIO_PORTA_DEN_R
      LDR R0, [R1]
      ORR R0, R0, #0x10    ; enable PA4
      STR R0, [R1]

      LDR R1, =GPIO_PORTF_DIR_R
      LDR R0, [R1]
      ORR R0, R0, #0x02
      STR R0, [R1]

      LDR R1, =GPIO_PORTF_DEN_R
      LDR R0, [R1]
      ORR R0, R0, #0x02
      STR R0, [R1]
      BX  LR
```

```
185  Debug_Init
186        PUSH {R0-R4,LR}
187
188        LDR   R0,=DataBuffer
189        MOV   R1,#0xFF
190        MOV   R2,#100
191  loo   CMP   R2,#0
192        BEQ   done
193        STR   R1, [R0]
194        ADD   R0,#1
195        SUB   R2,#1
196        B     loo ;fill databuffer with 0xFF
197
198  done  LDR   R0,=TimeBuffer
199        LDR   R1,=0xFFFFFFFF
200        MOV   R2,#100
201  lo    CMP   R2,#0
202        BEQ   fin
203        STR   R1, [R0]
204        ADD   R0,#4
205        SUB   R2,#1
206        B     lo ;fill timebuffer with 0xFFFFFFFF
207
208  fin   LDR   R0,=DataPt
209        LDR   R1,=DataBuffer
210        STR   R1,[R0] ;set the datapt
211        LDR   R0,=TimePt
212        LDR   R1,=TimeBuffer
213        STR   R1,[R0] ;set the timept
214        LDR   R0,=minTime
215        LDR   R1,[R0]
```

```
216         ORR   R1,#0x0FFFFFFF
217         STR   R1,[R0] ;makes minTime a very large positive number
218
219         BL SysTick_Init
220         POP {R0-R4,PC}
221         BX LR
```

```
226   Debug_Capture
227         PUSH {R0-R10,LR}
228         LDR R7,=DataPt
229         LDR R4,[R7] ;R4 is contents of DataPt
230         LDR R8,=TimePt
231         LDR R5,[R8] ;R5 is contents of TimePt
232         LDR R9, =prevtime
233         LDR R10,[R9]
234
235         ;begin check to see if databuffer is full
236         LDR R6,=DataBuffer
237         ADD R6,R6,#100
238         CMP R4,R6
239         BEQ full ;check if databuffer full
240
241         LDR R6,=GPIO_PORTA_DATA_R
242         LDRB R0,[R6]
243         AND R0,#0x10 ;get only bit 4 of port a data
244
245         LDR R6,=GPIO_PORTE_DATA_R
246         LDRB R1,[R6]
247         AND R1,#0x0F ;only bits 0-3 of port e
248         ORR R0,R0,R1 ;combine them
249         STRB R0,[R4] ;store to datapointer address
250         ADD R4,#1 ;incr datapointer
251         STR R4,[R7] ;store datapointer new value
252
253         LDR R6,=NVIC_ST_CURRENT_R
254         LDR R2,[R6] ;R2 has systick current value
255
256         LDR R3,=0x00FFFFFF
```

```
257         SUB R1,R10,R2 ;find elapse time
258         AND R1,R1,R3
259
260         LDR R3,=maxTime
261         LDR R11,[R3]      ;R11 HAS MAXIMUM TIME DIFF
262         CMP R1,R11
263         BLE min           ;IF CURRENT VALUE<=MAX; SKIP STORING IT
264         STR R1,[R3]
265 min     LDR R3,=minTime
266         LDR R11,[R3]      ;R11 HAS MINIMUM TIME DIFF
267         CMP R1,R11
268         BGE skip          ;IF CURRENT VALUE>=MIN; SKIP STORING IT
269         MOV R11,#0xFF
270         CMP R1,R11
271         BLE skip          ;THIS CMP CHECKS TO SEE IF THE DIFF IS LESS THAN THE FIRST FAULTY VALUE-SKIPS IF THIS IS THE CASE
272         STR R1,[R3]
273
274 skip    STR R1,[R5] ;store elapse to timebuffer
275         ADD R5,R5,#4 ;advance timepointer to next
276         STR R5,[R8] ;store back timepointer
277         STR R2,[R9] ;current time to previous time
278
279 full    POP  {R0-R10,PC}
280         BX LR
```

4. Running Estimates

The capture function has 44 lines, requiring around 88 instruction cycles, requiring approx. 1100ns. The program has approx. 3.2e6 lines between captures, approx. 6.4e6 instruction cycles and requiring approx. 80ms to execute. Therefore, the overhead with an 80MHz clock is about 0.001375%, which is negligible.

5. Results

See above screenshots for memory data.