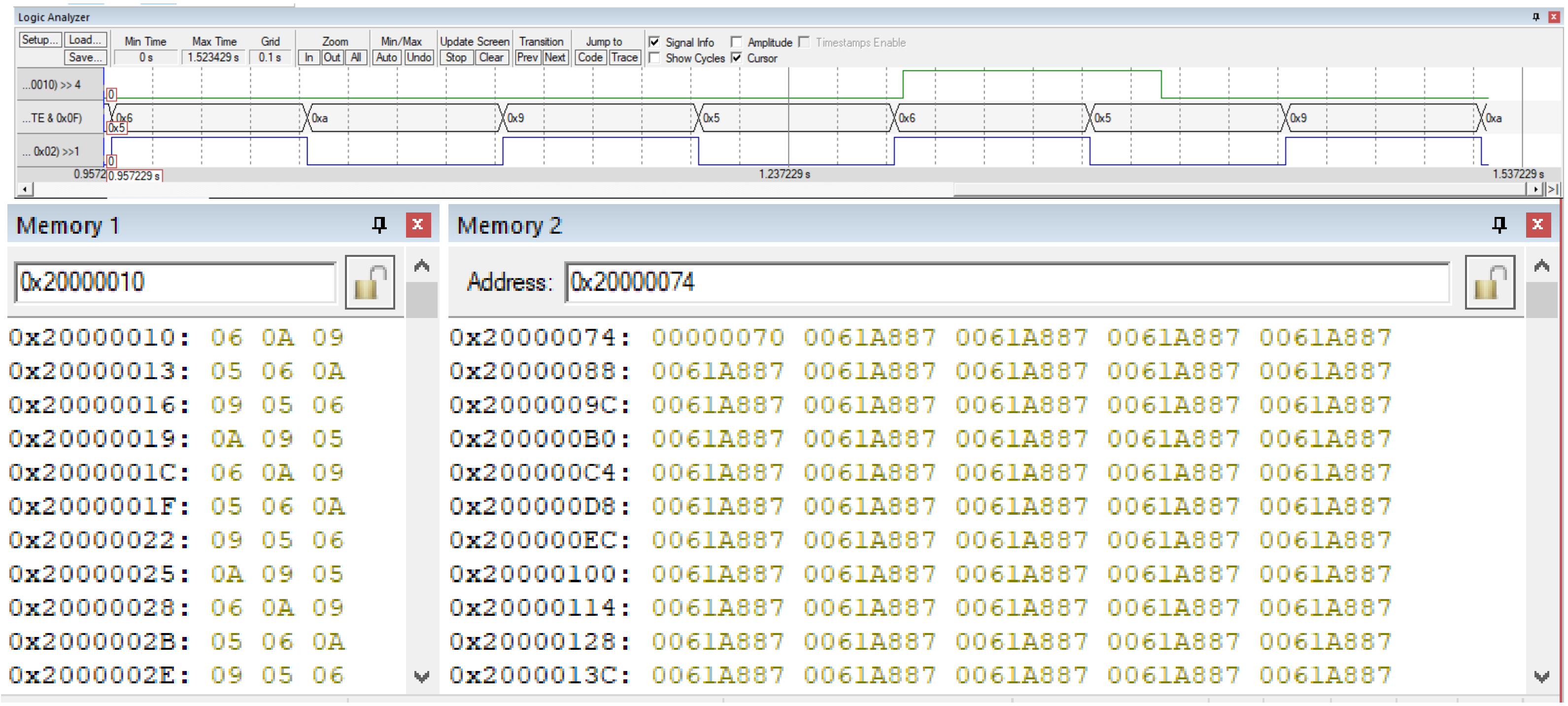
Lab 4 Deliverables

Eric Chen and Dennis Liu

2019-03-04

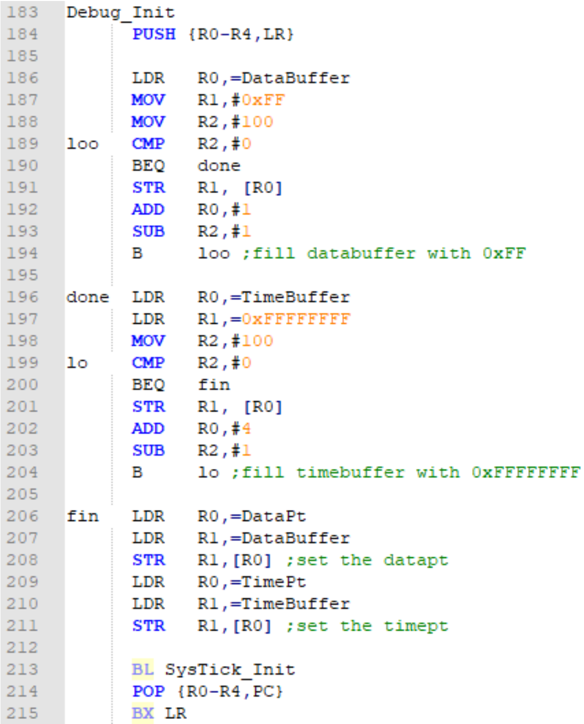
2. Screenshots



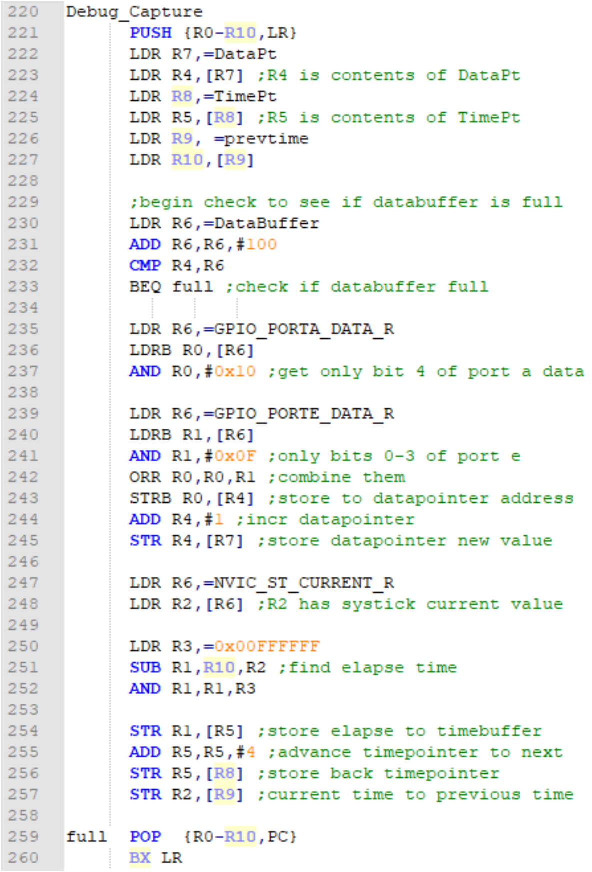
```
0x20000010

0x20000010: 06 0A 09
0x20000013: 05 06 0A
0x20000016: 09 05 06
0x20000019: 0A 09 05
0x2000001C: 06 0A 09
0x2000001F: 05 06 0A
0x20000022: 09 05 06
0x20000025: 0A 09 05
0x20000028: 06 0A 09
0x2000002B: 05 06 0A
0x2000002E: 09 05 06
```

```
Address: 0x20000074

0x20000074: 00000070 0061A887 0061A887 0061A887 0061A887
0x20000088: 0061A887 0061A887 0061A887 0061A887 0061A887
0x2000009C: 0061A887 0061A887 0061A887 0061A887 0061A887
0x200000B0: 0061A887 0061A887 0061A887 0061A887 0061A887
0x200000C4: 0061A887 0061A887 0061A887 0061A887 0061A887
0x200000D8: 0061A887 0061A887 0061A887 0061A887 0061A887
0x200000EC: 0061A887 0061A887 0061A887 0061A887 0061A887
0x20000100: 0061A887 0061A887 0061A887 0061A887 0061A887
0x20000114: 0061A887 0061A887 0061A887 0061A887 0061A887
0x20000128: 0061A887 0061A887 0061A887 0061A887 0061A887
0x2000013C: 0061A887 0061A887 0061A887 0061A887 0061A887
```

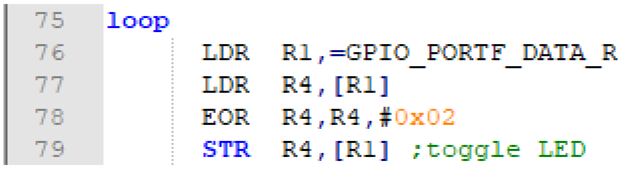
3. Source Code



```
Debug_Init
        PUSH {R0-R4,LR}

        LDR   R0,=DataBuffer
        MOV   R1,#0xFF
        MOV   R2,#100
loo     CMP   R2,#0
        BEQ   done
        STR   R1, [R0]
        ADD   R0,#1
        SUB   R2,#1
        B     loo ;fill databuffer with 0xFF

done    LDR   R0,=TimeBuffer
        LDR   R1,=0xFFFFFFFF
        MOV   R2,#100
lo      CMP   R2,#0
        BEQ   fin
        STR   R1, [R0]
        ADD   R0,#4
        SUB   R2,#1
        B     lo ;fill timebuffer with 0xFFFFFFFF

fin     LDR   R0,=DataPt
        LDR   R1,=DataBuffer
        STR   R1,[R0] ;set the datapt
        LDR   R0,=TimePt
        LDR   R1,=TimeBuffer
        STR   R1,[R0] ;set the timept

        BL SysTick_Init
        POP {R0-R4,PC}
        BX LR
```



```
Debug_Capture
        PUSH {R0-R10,LR}
        LDR R7,=DataPt
        LDR R4,[R7] ;R4 is contents of DataPt
        LDR R8,=TimePt
        LDR R5,[R8] ;R5 is contents of TimePt
        LDR R9, =prevtime
        LDR R10,[R9]

        ;begin check to see if databuffer is full
        LDR R6,=DataBuffer
        ADD R6,R6,#100
        CMP R4,R6
        BEQ full ;check if databuffer full

        LDR R6,=GPIO_PORTA_DATA_R
        LDRB R0,[R6]
        AND R0,#0x10 ;get only bit 4 of port a data

        LDR R6,=GPIO_PORTE_DATA_R
        LDRB R1,[R6]
        AND R1,#0x0F ;only bits 0-3 of port e
        ORR R0,R0,R1 ;combine them
        STRB R0,[R4] ;store to datapointer address
        ADD R4,#1 ;incr datapointer
        STR R4,[R7] ;store datapointer new value

        LDR R6,=NVIC_ST_CURRENT_R
        LDR R2,[R6] ;R2 has systick current value

        LDR R3,=0x00FFFFFF
        SUB R1,R10,R2 ;find elapse time
        AND R1,R1,R3

        STR R1,[R5] ;store elapse to timebuffer
        ADD R5,R5,#4 ;advance timepointer to next
        STR R5,[R8] ;store back timepointer
        STR R2,[R9] ;current time to previous time

full    POP  {R0-R10,PC}
        BX LR
```



```
loop
        LDR   R1,=GPIO_PORTF_DATA_R
        LDR   R4,[R1]
        EOR   R4,R4,#0x02
        STR   R4,[R1] ;toggle LED
```

4. Running Estimates

The capture function has 31 lines and approximately 62 instruction cycles, requiring approx. 775ns. The program has approx. 3.2e6 lines between captures and approx. 6.4e6 instruction cycles requiring approx. 80ms to execute. Thus, the debug capture overhead with an 80MHz clock is approx. 0.00971%, which is negligible.

5. results

See Screenshots for memory data.