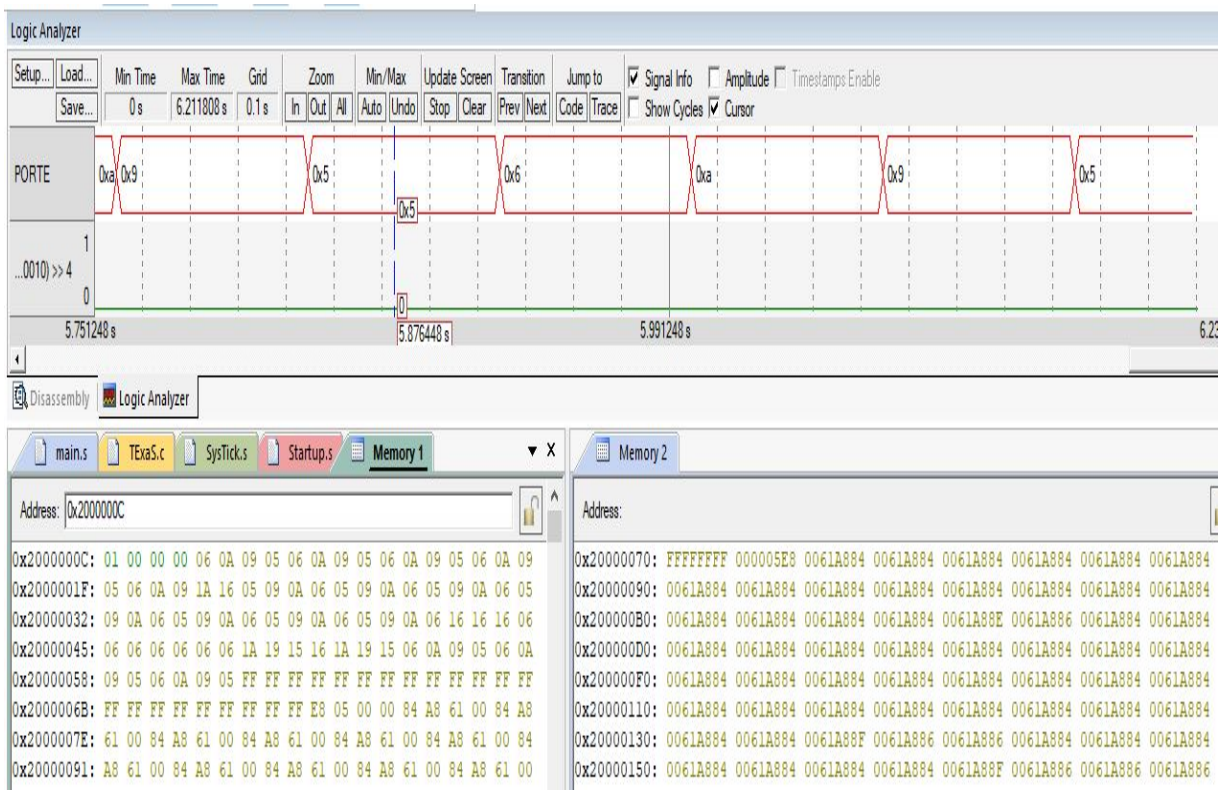
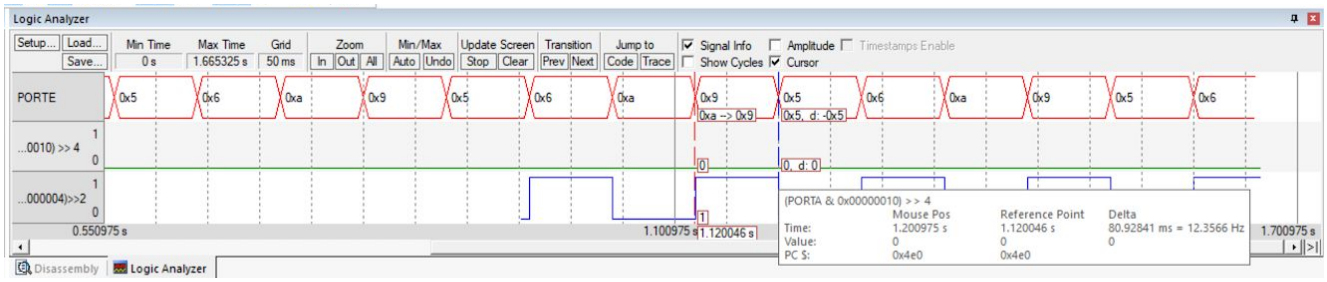
Lab 4. Minimally Intrusive Debugging Methods (Sp 2019)

Deliverables Emma Harper, Christopher Bowling





```
60A0905E805000083A8610083A86100
3A8610083A8610083A8610083A86100
3A8610083A8610083A8610083A86100
3A8610083A8610083A8610083A86100
3A8610083A8610083A8610083A86100
3A8610083A8610083A8610083A86100
3A8610083A8610083A8610083A86100
3A8610083A8610083A8610083A86100
3A8610083A8610083A8610083A86100
3A8610083A8610083A8610083A86100
3A8610083A8610083A8610083A86100
3A8610083A8610083A8610083A86100
3A8610083A8610083A892
```

Time Estimation:

Total clock cycles for program to run: about 6,400,085 clock cycles (6.4 million due to the delay function) while our Debug Capture takes about 68 cycles to fully run through. Therefore our intrusiveness is calculated to be about **0.00106%** which is considered to be minimally intrusive.

Timing Estimation for Heartbeat:

50% duty cycle since it is on for half of the time that it is off. This execution only takes 10 clock cycles which is approximately .000125ms to exexute out of the 6,400,085 clock cycles (about 80ms) that the loop runs through, therefore the intrusiveness was calculated to be about **0.000156%**.

Debug_Init

```
        PUSH {R0-R4,LR}
        LDR R0, =DataPt
        MOV R1, #0x00000000
        STR R1, [R0]
        LDR R0, =TimePt
        STR R1, [R0]
        MOV R1, #0x00FFFFFF
        LDR R0, =PTime
        STR R1, [R0]
        BL SysTick_Init
        LDR R0, =DataBuffer
        MOV R2, #0
        MOV R3, #0xFF
StoreFF
```

```
        STRB R3, [R0, R2]
        ADD R2, R2, #1
        CMP R2, #100
        BNE StoreFF
        LDR R0, =TimeBuffer
        MOV R2, #0
        MOV R3, #0xFFFFFFFF
TimeFF
        STR R3, [R0, R2, LSL #2]
        ADD R2, R2, #1
        CMP R2, #100
        BNE TimeFF


    POP {R0-R4,PC}
```

Debug_Capture

```
    PUSH {R0-R6,LR}          ;8 clock cycles
        LDR R0, =DataBuffer          ;2 cycles
        LDR R1, =DataPt     ; 2 cycles
        LDR R2, [R1]
        CMP R2, #100         ;R2 has the index for databuffer                ;1 cycle
        BEQ Branch ; if taken, 3, else 1
        LDR R4, =GPIO_PORTE_DATA_R
        LDRB R3, [R4]
        LDR R6, =GPIO_PORTA_DATA_R
        LDRB R5, [R6]
        ORR R3, R5 ;one cycle
        STRB R3, [R0,R2]    ;two cycles
        ADD R2, #1  ;one
        STR R2, [R1]
        LDR R0, =TimeBuffer
        LDR R1, =TimePt
        LDR R2, [R1]                                ; value of time pointer
        LDR R4, =NVIC_ST_CURRENT_R
        LDR R3, [R4]                                ; data of timer
        LDR R4, =PTime                              ; address of previous time
        LDR R5, [R4]
        CMP R5, R3 ;one
        SUB R5, R5, R3      ;one
        AND R5, R5, #0x00FFFFFF; only need first 24 bits- one cycle
        STR R3, [R4]
        STR R5, [R0, R2]
```

```
        ADD R2, #4   ;one cycle
        STR R2, [R1]
Branch
        POP  {R0-R6,PC}       ;4 for branch, 8 for registers
        ; 68 clock cycles through this subroutine: .85 microsecond each time its exexuted
```

Heartbeat_Init

```
        LDR R0, =SYSCTL_RCGCGPIO_R
        LDR R1, [R0]
        ORR R1, #0x20
        STR R1, [R0]
        NOP
        NOP
        NOP
        NOP
        LDR R0, =GPIO_PORTF_DIR_R
        LDR R1, [R0]
        MOV R1, #0x04 ;// bit 3 (PF2) in port F is output LED
        STR R1, [R0]
        LDR R0, = GPIO_PORTF_DEN_R
        LDR R1, [R0]
        MOV R1, #0x04
        STR R1, [R0]

        BX LR
```

Heartbeat

```
        LDR R0, =GPIO_PORTF_DATA_R
        LDR R1, [R0]
        EOR R1, R1, #0x04             ;toggle LED for heartbeat
        STR R1, [R0]
        BX LR
```