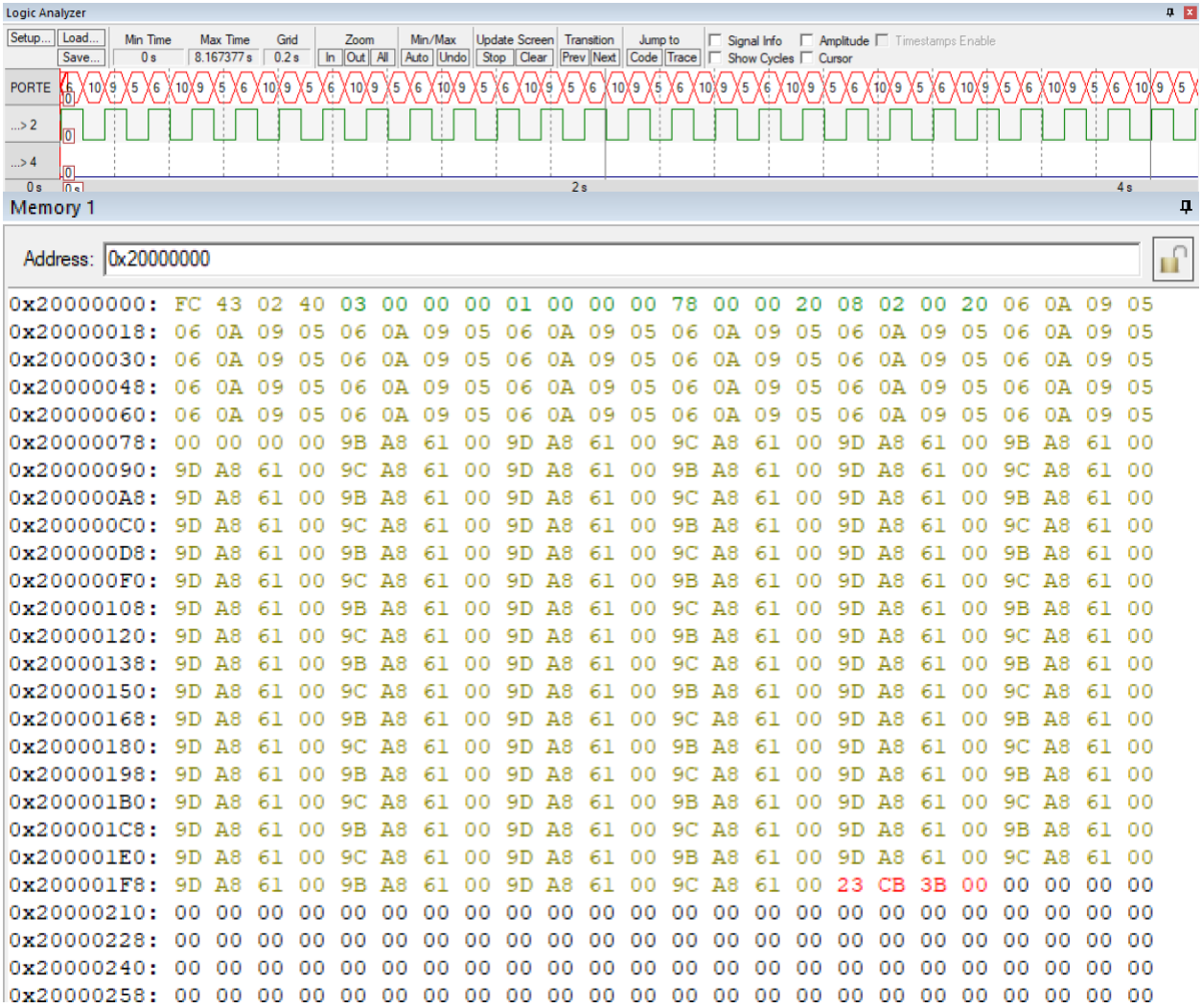
Screenshots



Main Code

```
Debug_Init
        ; save values of registers that will be used. use an even number of registers
    PUSH {R0-R4,LR}

        ;
        LDR R0, =DataBuffer
        LDR R1, =DataPt
        STR R0, [R1]            ; store data buffer starting address into the data pointer

        ; set all entries of the data buffer to 0xFF
        MOV R2, #0xFF
        MOV R4, #0              ; R4 will serve as a counter to check for the end of the array
loop1 STR R2, [R0]              ; store 0xFF in element of data buffer
```

```
        ADD R0, #1          ; increment location in array
        ADD R4, #1          ; increment counter
        CMP R4, #100            ; check if end of array is reached and exit loop if so
        BLO loop1


        LDR R0, =TimeBuffer
        LDR R1, =TimePt
        STR R0, [R1]

        ; set all entries of the time buffer to 0xFFFFFFFF
        MOV R2, #0xFFFFFFFF
        MOV R4, #0          ; reset the counter
loop2 STR R2, [R0]          ; store 0xFFFFFFFF in element of time buffer
        ADD R0, #4          ; increment to next location (since 32 bit, add 4, not 1)
        ADD R4, #1          ; increment counter
        CMP R4, #100            ; check if end of array is reached and exit loop if so
        BLO loop2

        ; call SysTick initialization
        BL SysTick_Init
    POP {R0-R4,PC}

;Debug capture
Debug_Capture
    PUSH {R0-R8,LR}
        ; implement heartbeat
        LDR R1, =SYSCTL_RCGCGPIO_R
        LDR R0, [R1]
        ORR R0, R0, #0x20                   ; set bit 5 (for Port F) of clock
        STR R0, [R1]
        NOP
        NOP

        LDR R1, =GPIO_PORTF_DIR_R
        LDR R0, [R1]
        ORR R0, R0, #0x04                   ; set bit 2 as output on Port F
        STR R0, [R1]

        LDR R1, =GPIO_PORTF_DEN_R
        LDR R0, [R1]
        ORR R0, R0, #0x04                   ; enable bit 2 on Port F
        STR R0, [R1]
```

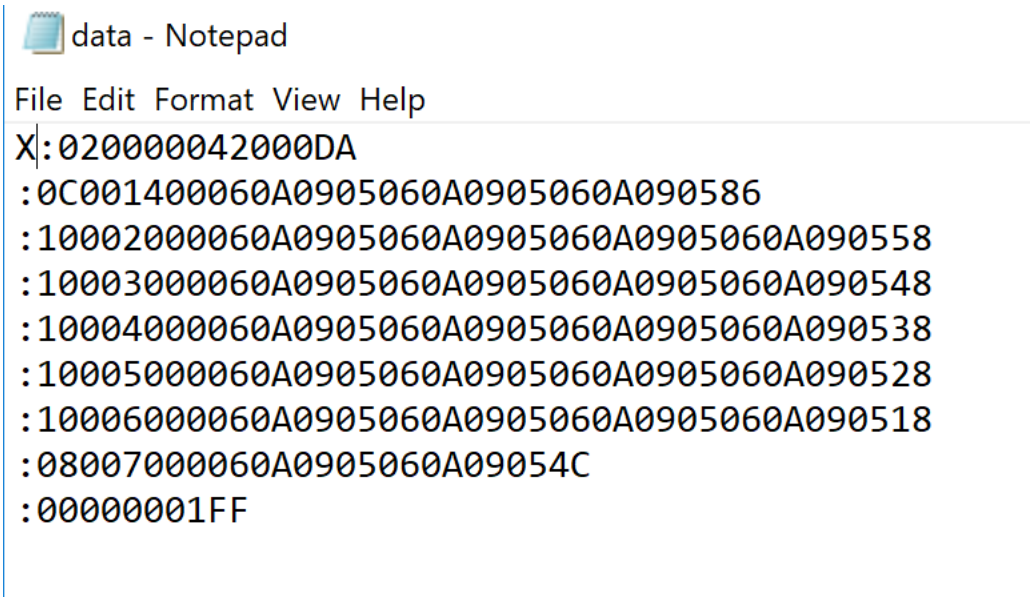
```
        LDR R1, =GPIO_PORTF_DATA_R      ; read data from Port F
        LDR R0, [R1]
        EOR R0, R0, #0x04                       ; toggle bit 2
        STR R0, [R1]                            ; store toggled value back in data

        ; return immediately if buffers are full
        LDR R0, =DataPt
        LDR R1, [R0]
        AND R2, R1, #0xFF       ; isolate 8 bits that vary across addresses in DataBuffer
        CMP R2, #0x78                   ; check if pointer has reached end of array
        BLO cont                        ; if end of array, buffer is full. return
        POP {R0-R8,LR}
        BX LR

        ; read Port E and current/old SysTick times
cont  LDR R0, =GPIO_PORTE_DATA_R
        LDR R1, [R0]                            ; R1 holds Port E data

        LDR R8, =NVIC_ST_CURRENT_R      ; R8 holds address of new time
        LDR R7, [R8]                            ; R7 holds the new time
        LDR R6, =Old                            ; R6 holds the address of Old
        LDR R5, [R6]                            ; R5 holds the old time

        ; mask capturing just bits 4, 3, 2, 1, and 0 of Port E
        AND R1, R1, #0x1F

        ; dump this port info into DataBuffer using the pointer DataPt
        LDR R2, =DataPt
        LDR R3, [R2]                            ; R3 holds the address of current element in
DataBuffer
        STR R1, [R3]                            ; put input/output information into the
element pointed to by DataPt

        ; increment DataPt to next address
        ADD R3, R3, #1                          ; data pointer points to next address
        LDR R2, =DataPt
        STR R3, [R2]

        ; calculate the 24 bit elapsed time
        SUB R4, R5, R7                          ; find time difference
        AND R4, R4, #0x00FFFFFF

        ; update the old variable with the new time (now old for next iteration)
```

```
        STR R7, [R6]                            ; old time is updated. R6 is still address of
Old

        ; dump elapsed time into TimeBuffer using pointer TimePt
        LDR R2, =TimePt
        LDR R3, [R2]                            ; R3 holds the address of current element in
TimeBuffer
        STR R4, [R3]                            ; dump elapsed time into TimeBuffer

        ; increment TimePt to next address
        ADD R3, R3, #4                          ; TimePt points to next address in
TimeBuffer
        LDR R2, =TimePt
        STR R3, [R2]                            ; update in variable

    POP  {R0-R8,PC}

        ; 29 instructions in Debug_Capture if array not full; else, there will only be 8
instructions and it will return
        ; 29 * 2 = 58 cycles
        ; 58 cycles * 12.5 ns = 725 ns
        ; use logic analyser to determine time between calls to Debug_Capture (80 ms)
        ; (725 * 10 ^ -9) / (80 * 10 ^ -3) * 100% = 0.00090625 %

    ALIGN     ; make sure the end of this section is aligned
    END       ; end of file
```

Estimation without heartbeat

```
  ; 29 instructions in Debug_Capture if array not full; else, there will only be 8 instructions and it
will return
        ; 29 * 2 = 58 cycles
        ; 58 cycles * 12.5 ns = 725 ns
        ; use logic analyser to determine time between calls to Debug_Capture (80 ms)
        ; (725 * 10 ^ -9) / (80 * 10 ^ -3) * 100% = 0.00090625 %
```

Data for DataBuffer



Data for TimeBuffer

data - Notepad

File Edit Format View Help

```
:020000042000DA
:0800780000000009BA86100DC
:100080009DA861009CA861009DA861009BA86100DB
:100090009DA861009CA861009DA861009BA86100CB
:1000A0009DA861009CA861009DA861009BA86100BB
:1000B0009DA861009CA861009DA861009BA86100AB
:1000C0009DA861009CA861009DA861009BA861009B
:1000D0009DA861009CA861009DA861009BA861008B
:1000E0009DA861009CA861009DA861009BA861007B
:1000F0009DA861009CA861009DA861009BA861006B
:100100009DA861009CA861009DA861009BA861005A
:100110009DA861009CA861009DA861009BA861004A
:100120009DA861009CA861009DA861009BA861003A
:100130009DA861009CA861009DA861009BA861002A
:100140009DA861009CA861009DA861009BA861001A
:100150009DA861009CA861009DA861009BA861000A
:100160009DA861009CA861009DA861009BA86100FA
:100170009DA861009CA861009DA861009BA86100EA
:100180009DA861009CA861009DA861009BA86100DA
:100190009DA861009CA861009DA861009BA86100CA
:1001A0009DA861009CA861009DA861009BA86100BA
:1001B0009DA861009CA861009DA861009BA86100AA
:1001C0009DA861009CA861009DA861009BA861009A
:1001D0009DA861009CA861009DA861009BA861008A
:1001E0009DA861009CA861009DA861009BA861007A
:1001F0009DA861009CA861009DA861009BA861006A
:080200009DA861009CA86100AB
:00000001FF
```