Lab Report

EECS 3100:046

Embedded Systems Lab

Lab 6:

Minimally Intrusive Debugging Methods

Landon Jackson & Logan Crawfis

Professor Brent Nowlin

July 2nd, 2021

Lab Objective

The purpose of this lab is to study, design and implement minimally intrusive debugging instruments. A debugging instrument can be considered minimally intrusive if the time it takes to collect and store information is short compared to the time when the information is collected.

Introduction

Analyzing software in real-time is important to confirm that the software is running as specified. This lab is a continuation of lab 5. The specifications of this lab were to implement two debugging instruments: a “dump” and a “heartbeat”. The Dump and heartbeat instruments will be used to visually confirm that our software is running correctly.

Project Description

The heartbeat and dump instruments are two debugging instruments that can be used to visualize software running in real-time on a microcontroller. The heartbeat is a square pulse wave used to indicate that the software is still running, when working with embedded devices it is important that the program can run nonstop, if the process is broken for any reason the heartbeat will provide that information to the end-user. The dump instrument allows the user to capture information of the software running that can be analyzed later. Information regarding the embedded devices runtime can allow for quick debugging, using the dump as a debugging tool allowed us to recognize issues in our code that needed to be fixed.

Design Procedure

The specifications of the lab instructed us to design and initialize two functions, Debug_Init and Debug_Capture, two buffers to hold our information DataBuffer and TimeBuffer and lastly to pointers DataPt and TimePt. The first step before implementing our functions was to allocate space in RAM for both the buffers and pointer. The first step of the Debug_Init function was to initialize both buffers, for this we created a loop to set all the entries of TimeBuffer and DataBuffer to 0xFFFFFFFF. The next step was to set the DataPt at the first address of the DataBuffer and the TimePt to the first address of the TimeBuffer. The last step was to activate the Systick Timer, this function was retrieved from the text. To implement the Debug_Capture function, first the registers were pushed and then the data from Port E and the timer were loaded into R2 and R3 respectively. Then the data from Port E was modified to fit the criteria given in the lab handout by using an and instruction on R2 with 0x02 and storing it in R4. Followed by shifting the data 3 bits to the left and then anding the original data with 0x01 and then using the orr instruction on that new data and the shifted data to get bit one moved to bit 4 and then recombine the values into one byte. Then the value of the modified data and the timer were stored into their respective buffer arrays and then indexed four bytes until the address for the data hit 0x200000F8 and then the index was reset to keep the data written into the allotted addresses.

Discussion

The first change that was made to the program from lab 5 was to add a Debug_Init routine to initialize the buffers, and counter for capturing data. Following this the capture portion of the debugging was added, and lastly the heartbeat. All of these new routines caused quite a few issues on their own. Issues were encountered with the debug capture where the indexing was being lost after each loop. After that was fixed the data was being written continuously even after the buffers were full. To fix this problem a compare was added to reset the index of the buffers back to the initial location after they were filled. The data and time buffers can be seen below in memory, where Fig 1. is the initial values of the two buffers and Fig 2. is the final values of buffers where the time buffer starts at 0x200000F8.

0x20000030:	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF
0x20000044:	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF
0x20000058:	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF
0x2000006C:	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF
0x20000080:	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF
0x20000094:	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF
0x200000A8:	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF
0x200000BC:	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF
0x200000D0:	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF
0x200000E4:	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF
0x200000F8:	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF
0x2000010C:	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF
0x20000120:	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF
0x20000134:	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF
0x20000148:	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF
0x2000015C:	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF
0x20000170:	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF
0x20000184:	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF
0x20000198:	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF
0x200001AC:	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF

Fig 1. Init Time/Data Buffer

Address: 0x20000030

0x20000030:	00000010	00000010	00000010	00000010	00000010
0x20000044:	00000010	00000010	00000010	00000010	00000010
0x20000058:	00000010	00000010	00000010	00000010	00000010
0x2000006C:	00000010	00000010	00000010	00000010	00000010
0x20000080:	00000010	00000010	00000010	00000010	00000010
0x20000094:	00000001	00000001	00000001	00000001	00000001
0x200000A8:	00000001	00000001	00000001	00000001	00000001
0x200000BC:	00000010	00000011	00000010	00000011	00000010
0x200000D0:	00000011	00000010	00000011	00000010	00000011
0x200000E4:	00000010	00000011	00000010	00000011	00000010
0x200000F8:	003F7491	00CCCC3D	005A23E9	00E77B95	0074D341
0x2000010C:	00022AED	008F8299	001CDA45	00AA31F1	0037899D
0x20000120:	00C4E149	005238F5	00DF90A1	006CE84D	00FA3FF9
0x20000134:	008797A5	0014EF51	00A246FD	002F9EA9	00BCF655
0x20000148:	004A4E01	00D7A5AD	0064FD59	00F25505	007FACB1
0x2000015C:	007DA5EC	00569D7C	002F950C	00088C9C	00E1842C
0x20000170:	00BA7BBC	0093734C	006C6ADC	0045626C	001E59FC
0x20000184:	00F75189	0084A935	001200E1	009F588D	002CB039
0x20000198:	00BA07E5	00475F91	00D4B73D	00620EE9	00EF6695
0x200001AC:	007CBE41	000A15ED	00976D99	0024C545	00B21CF1

Fig 2. Time/Data Buffer

The heartbeat caused quite a few problems. Initially it was not showing the delay, then after fixing that it was only working while the button was pressed. Ultimately the delay for the heartbeat was inconsistent, and depended on the amount of cycles that the program was going through until running the heartbeat routine again. This probably could have been fixed with a different implementation of the heartbeat using the timer, instead of the same style of delay that was used for the blinking LED on PE0. This inconsistency in the delay can be seen below in Fig 3. where the heartbeat is the bottom signal, the button is in the middle and the LED is the top signal.

To assure that our debugging instruments could be considered non-intrusive we had to calculate the time taken to execute our Debug_Capture subroutine. The subroutine contained 12 instructions, assuming 2 cycles per instruction we had 24 cycles in total, Once again making an assumption; 12.5 ns bus cycle time, we calculate the time to be 300ns . The Next step was to calculate the overhead {(300ns / 62ms) *100 = .0005%}. With this value our debugging instruments are considered minimally intrusive.

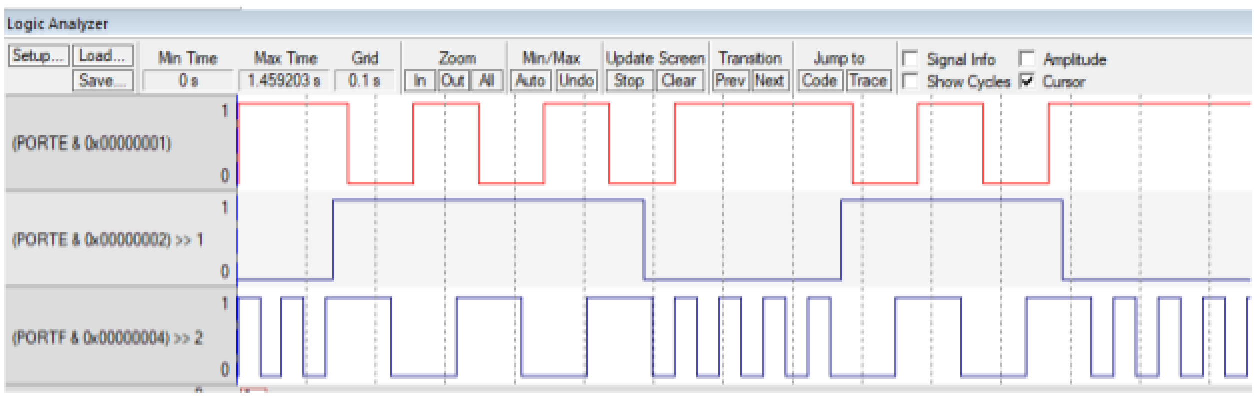

Fig 3. Logic analyzer output from simulation

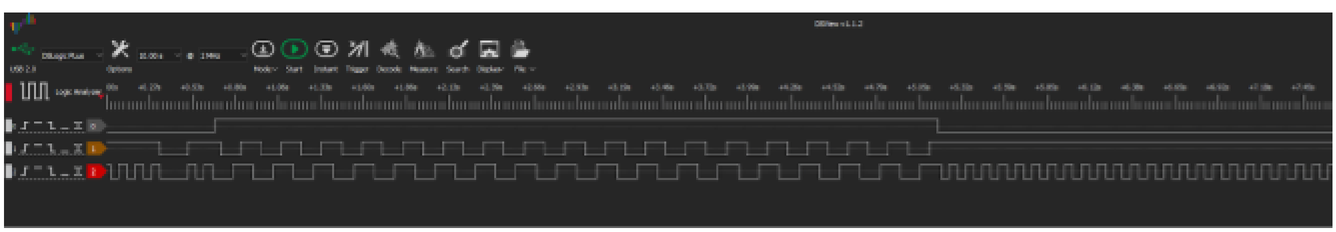

Figure 4. DSView output window where input 0 is PE1, input 1 is PE0 and input 2 is PF2.

Conclusion

This experiment was considerably more difficult than the previous lab experiments as it introduced a lot of new concepts all at once. The idea of the heartbeat, the timer, and the debug capture required a lot of trial and error compared to the previous weeks. Overall this experiment was extremely helpful in understanding those concepts, but was very difficult to get to that point.

The work for this lab was evenly split between partners. Each of us worked on new routines for our program and then worked through the bugs together. Then each of us wrote portions of the lab report.

```
:020000042000DA
:1000000000350C00480900007009000000000000E5
:10001000000000000000000000000000000000000E0
:100020000000000000000000010000000000000000CF
:10003000100000001100000010000000010000008E
:1000400001000000010000000100000001000000AC
:10005000010000000100000001000000100000008D
:10006000110000001000000011000000100000004E
:10007000110000000100000001000000010000006C
:10008000010000000100000001000000010000006C
:10009000010000000100000001000000100000004D
:1000A000110000001000000011000000100000000E
:1000B000110000000100000001000000010000002C
:1000C000010000000100000001000000010000002C
:1000D000010000000100000001000000010000001C
:1000E000010000000100000001000000FFFFFFFF11
:1000F000FFFFFFFFFFFFFFFF89F7D800354F6600C6
:10010000E1A6F3003CF65900CCED32005CE50B00B3
:10011000ECDCE4007CD4BD000CCC96009CC36F00EA
:100120002CBB4800B9B22100650AAF0011623C0047
:10013000BDB9C900691157001569E400C4C0710058
:1001400054B84A00E4AF230074A7FC00049FD50014
:100150009496AE00248E8700B4856000447D3900FB
:10016000D4741200616CEB000DC47800B91B06005A
:100170006573930011CB2000BD22AE006C7A3B006A
:10018000FC7114008C69ED001C61C600AC589F0026
:010190003C32
:00000001FF
```

```
;****************************************************************
; main.s
; Author: Logan Crawfis & Landon Jackson
; Date Created: 06/31/2021
; Last Modified: 07/02/2021
; Section Number: 046
; Instructor: Brent Nowlin
; Lab number: 6
; Brief description of the program
;   If the switch is presses, the LED toggles at 8 Hz
; Hardware connections
;   PE1 is switch input  (1 means pressed, 0 means not pressed)
;   PE0 is LED output (1 activates external LED on protoboard)
; Overall functionality is similar to Lab 5, with three changes:
;   1) Initialize SysTick with RELOAD 0x00FFFFFF
;   2) Add a heartbeat to PF2 that toggles every time through loop
;   3) Add debugging dump of input, output, and time
; Operation
;      1) Make PE0 an output and make PE1 an input.
;      2) The system starts with the LED on (make PE0 =1).
;   3) Wait about 62 ms
;   4) If the switch is pressed (PE1 is 1), then toggle the LED
;     once, else turn the LED on.
;   5) Steps 3 and 4 are repeated over and over
;****************************************************************
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_PORTE_AMSEL_R      EQU   0x40024528
GPIO_PORTE_PCTL_R       EQU   0x4002452C
GPIO_PORTF_DATA_R       EQU   0x400253FC
GPIO_PORTF_DIR_R        EQU   0x40025400
GPIO_PORTF_AFSEL_R      EQU   0x40025420
GPIO_PORTF_DEN_R        EQU   0x4002551C
GPIO_PORTF_LOCK_R               EQU         0x40025520
GPIO_PORTF_CR_R                 EQU   0x40025524
GPIO_PORTF_AMSEL_R      EQU   0x40025528
GPIO_PORTF_PCTL_R       EQU   0x4002552C
SYSCTL_RCGCGPIO_R       EQU   0x400FE608
NVIC_ST_CTRL_R          EQU   0xE000E010
NVIC_ST_RELOAD_R        EQU   0xE000E014
NVIC_ST_CURRENT_R       EQU   0xE000E018
NVIC_ST_CTRL_COUNT            EQU   0x00010000  ; Count flag
NVIC_ST_CTRL_CLK_SRC          EQU   0x00000004  ; Clock Source
NVIC_ST_CTRL_INTEN      EQU   0x00000002  ; Interrupt enable
NVIC_ST_CTRL_ENABLE           EQU   0x00000001  ; Counter mode
NVIC_ST_RELOAD_M        EQU   0x00FFFFFF  ; Counter load value
;SIZE                   EQU   50
PE0             EQU   0x40024004
PE1             EQU   0x40024008
PF2                     EQU         0x40025010
```

```
        IMPORT  TExaS_Init

        THUMB
        AREA    DATA, ALIGN=4
SIZE                            EQU   50
DataBuffer                      SPACE SIZE*4
TimeBuffer                      SPACE SIZE*4
DataPt                          SPACE 4
TimePt                          SPACE 4
DumpTruck                       SPACE 4

        ALIGN
        AREA    |.text|, CODE, READONLY, ALIGN=2
      PRESERVE8
         THUMB
         EXPORT  Start

Start

;====================TExaS_Init function========================
      BL  TExaS_Init ; voltmeter, scope on PD3
;==============================================================================
;====================PORT F INITIALIZE=========================================
;==============================================================================
;====================Clock Enable====================================
    LDR           R0, =SYSCTL_RCGCGPIO_R      ; set clock for PortF
        MOV             R1, #0x20
    STR           R1, [R0]
;====================Delay=========================================
    NOP
    NOP
;====================Lock===========================================
        LDR             R0, =GPIO_PORTF_LOCK_R
        LDR             R1, =0x4C4F434B
        STR       R1, [R0]
;====================CR=============================================
        LDR             R0, =GPIO_PORTF_CR_R
        MOV             R1, #0x04
        STR       R1, [R0]
;====================AMSEL==========================================
    LDR     R0, =GPIO_PORTF_AMSEL_R      ; clear amsel for normal operation
    MOV     R1, #0x00
    STR     R1, [R0]
;====================Direction======================================
    LDR     R0, =GPIO_PORTF_DIR_R       ; set PF2 as output
    MOV     R1, #0x04
    STR     R1, [R0]
;====================Analog=========================================
    LDR     R0, =GPIO_PORTF_AFSEL_R      ; disable analog functionality
    MOV     R1, #0x00
    STR     R1, [R0]
;====================Digital-Enable================================
    LDR     R0, =GPIO_PORTF_DEN_R       ; enable digital I/O on PF2
```

```
    MOV     R1, #0x04
    STR     R1, [R0]
;===================PCTL=========================================
    LDR     R0, =GPIO_PORTF_PCTL_R       ; clear PCTL for PF
    MOV     R1, #0x00
    STR     R1, [R0]
;==================================================================================
;===================PORT E INITIALIZE==============================================
;==================================================================================
;===================Clock Enable===================================
    LDR     R0, =SYSCTL_RCGCGPIO_R       ; set clock for PortE
    LDR    R2, [R0]
        MOV             R1, #0x10
        ORR             R1, R2
    STR     R1, [R0]
;===================Delay========================================
    NOP
    NOP
;===================AMSEL========================================
    LDR     R0, =GPIO_PORTE_AMSEL_R      ; clear amsel for normal operation
    MOV     R1, #0x00
    STR     R1, [R0]
;===================Direction====================================
    LDR     R0, =GPIO_PORTE_DIR_R        ; set PE1 as input and PE0 as output
    MOV     R1, #0x01
    STR     R1, [R0]
;===================Analog=======================================
    LDR     R0, =GPIO_PORTE_AFSEL_R      ; disable analog functionality
    MOV     R1, #0x00
    STR     R1, [R0]
;===================Digital-Enable===============================
    LDR     R0, =GPIO_PORTE_DEN_R        ; enable digital I/O on PE0 and 1
    MOV     R1, #0x03
    STR     R1, [R0]
;===================PCTL=========================================
    LDR     R0, =GPIO_PORTE_PCTL_R       ; clear PCTL for PE
    MOV     R1, #0x00
    STR     R1, [R0]

        BL  Debug_Init
;==================================================================================
;===================MAIN PROGRAM===================================================
;==================================================================================
    CPSIE  I    ; TExaS voltmeter, scope runs on interrupts
    LDR     R8, =PE0
        LDR        R9,=DataBuffer
        LDR        R10,=TimeBuffer
led_on
    LDR     R4, [R8]                  ; load PE3 data into R4
    ORR     R4,  R4, #0x01             ; clear PE3
    STR     R4, [R8]                  ; store cleared data onto PE3
    B       loop
led_toggle
    LDR        R4, [R8]                  ; load PE0 data into R4
```

```
    EOR        R4, R4, #0x01                ; toggle PE0
    STR        R4,    [R8]                  ; store toggled data into PE0
    MOV        R5, #3875                    ; start the delay process
        BL          Debug_Capture
    B        Delay
Delay
    MOVS     R2, #0                         ; reset count for timer
sub_loop
    ADD     R2, #1                          ; add 1 to R2 until it hits 255
    CMP     R2, #255                        ; after hitting 255 subtract 1 from R5
    BNE       sub_loop
    SUB     R5, #1                          ; sub 1 from R5 until it hits zero
    CMP     R5, #0
    BNE     Delay                           ; go back until first portion of sub loop has been run 1447 times
    B        loop
check                                   ; check that the led is already off
    LDR        R4, [R8]
    CMP        R4, #0x00
    BEQ        led_on
        BL          Debug_Capture
loop
    BL              heartbeat
        LDR     R0, =PE1
    LDR     R7, [R0]
    CMP     R7, #0x02
    BNE     check                           ; turns off led if PF4 is not pressed
    B        led_toggle; you input output delay
    B       loop
;==========================================================================================
;===================================HeartBeat==========================================
;==========================================================================================
heartbeat
        NOP
        NOP
        LDR              R0, =PF2
        LDR              R1,  [R0]
        EOR              R1, R1, #0x04
        STR        R1, [R0]
heartbeat_on
        MOV        R5, #2000                ; start the delay process
heartbeat_delay
    MOVS     R2, #0                     ;reset count for timer
heartbeat_sub_loop
    ADD     R2, #1                      ; add 1 to R2 until it hits 255
    CMP     R2, #255                    ; after hitting 255 subtract 1 from R5
    BNE     heartbeat_sub_loop
    SUB     R5, #1                      ; sub 1 from R5 until it hits zero
    CMP     R5, #0
    BNE     heartbeat_delay             ; go back until first portion of sub loop has been run 1447 times
    BX      LR
;==========================================================================================
;===================================Debug Init=========================================
;==========================================================================================
Debug_Init
```

```
    PUSH {R0-R3}
Data_Init
    MOV     R1, #0xFFFFFFFF ;value          ;initialize data buffer by filling all values with 0xFFFFFFFF
    MOV     R2, #0x00 ;count
    MOV     R3, #0x00
    LDR     R0,=DataBuffer
DataBuff_loop                               ;loop through all addresses to fill with 0xFFFFFFFF
    CMP     R2, #SIZE
    BEQ     Time_Init
    STR R1, [R0]
    ADD     R2, #0x01
    ADD     R0, #0x04
    B DataBuff_loop
Time_Init                                   ;initialize time buffer
    MOV     R2, #0x00 ;count
    MOV     R3, #0x00
    LDR     R0,=TimeBuffer
TimeBuff_loop                                   ;loop through time buffer to fill with 0xFFFFFFFF
    CMP     R2, #SIZE
    BEQ     Point_Init
    STR R1, [R0]
    ADD     R2, #0x01
    ADD     R0, #0x04
    B TimeBuff_loop
Point_Init
    LDR R0, =DataBuffer    ; Load address of data buffer into R0
    LDR R1, =DataPt        ; Load address of data pointer into R1
    STR R0, [R1] ; Point DataPt to the address of the data buffer

    LDR R0, =TimeBuffer    ; Load address of time buffer into R0
    LDR R1, =TimePt        ; Load address of time pointer into R1
    STR R0, [R1] ; Point TimePt to the address of the time buffer
SysTick_Init
    ; disable SysTick during setup
  LDR R1, =NVIC_ST_CTRL_R         ; R1 = &NVIC_ST_CTRL_R
  MOV R0, #0                      ; R0 = 0
  STR R0, [R1]                    ; [R1] = R0 = 0
  ; maximum reload value
  LDR R1, =NVIC_ST_RELOAD_R       ; R1 = &NVIC_ST_RELOAD_R
  LDR R0, =NVIC_ST_RELOAD_M;      ; R0 = NVIC_ST_RELOAD_M
  STR R0, [R1]                    ; [R1] = R0 = NVIC_ST_RELOAD_M
  ; any write to current clears it
  LDR R1, =NVIC_ST_CURRENT_R      ; R1 = &NVIC_ST_CURRENT_R
  MOV R0, #0                      ; R0 = 0
  STR R0, [R1]                    ; [R1] = R0 = 0
  ; enable SysTick with core clock
  LDR R1, =NVIC_ST_CTRL_R         ; R1 = &NVIC_ST_CTRL_R
                                  ; R0 = ENABLE and CLK_SRC bits set
  MOV R0, #(NVIC_ST_CTRL_ENABLE+NVIC_ST_CTRL_CLK_SRC)
  STR R0, [R1]                    ; [R1] = R0 = (NVIC_ST_CTRL_ENABLE|NVIC_ST_CTRL_CLK_SRC)
    POP {R0-R3}
    BX  LR
;===========================================================================================
;==============================Debug Capture=====================================
```

```
;==============================================================================
Debug_Capture
   PUSH {R4-R8}                              ;push registers
   LDR        R0, =GPIO_PORTE_DATA_R            ;load in data and timer addresses
   LDR    R1, =NVIC_ST_CURRENT_R
   LDR        R2, [R0]
   LDR    R3, [R1]
   AND    R4,R2,#0x02                        ;single out bit 1 of data
   LSL    R4,#3                              ;shift bit 1 to bit 4
   AND    R2,#0x01                      ;single out bit 0 of data
   ORR    R2,R4                              ;recombine the shifted bit 1 and the singled out bit 0
   STR    R2,[R9],#4                         ;store values into buffers
   STR    R3,[R10],#4
      LDR           R8, =0x200000F8
      CMP           R8, R9                            ;reset indexes of buffers when they are full
      BEQ           Reset
   POP    {R4-R8}
      BX  LR
;==============================================================================
;==================================Reset===========================================
;==============================================================================
Reset
      PUSH {R4-R8}
      LDR R9, =DataBuffer          ; Load address of data buffer into R0
      LDR R10, =TimeBuffer ; Load address of time buffer into R0
      POP    {R4-R8}
      BX  LR

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