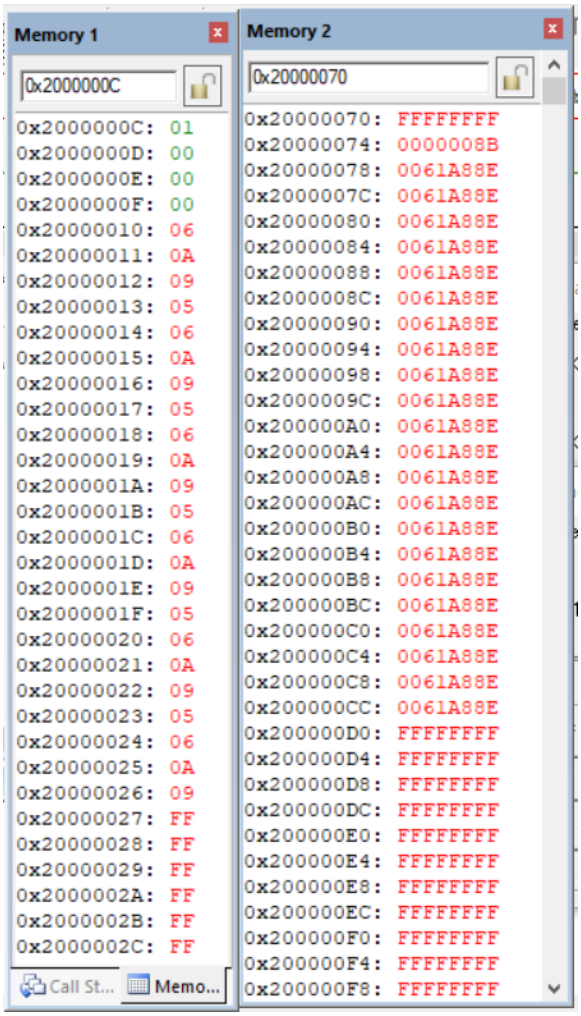
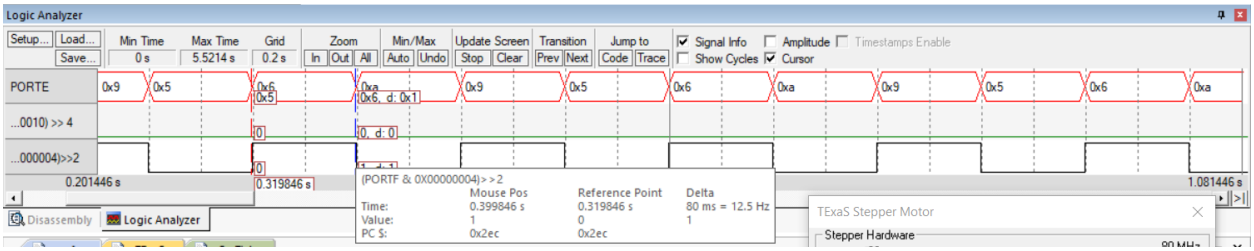
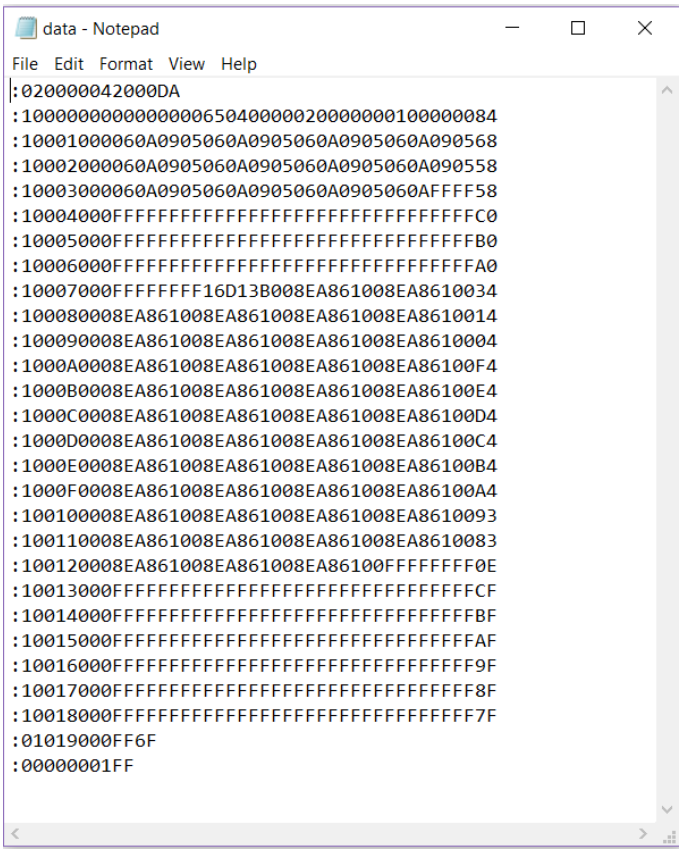





```
;62 cycles* 12.5ns = 775ns
;Percentage Overhead = ((775*10^-9)/.08)100 = 0.00097%

;Minimum elapsed time: 61cycles*12.5ns = 762.5ns
;Maximum elapsed time: 63cycles*12.5ns = 787.5ns
```



;****************** main.s ***************
; Program written by: **-UUU-*Your Names**update this***
; Date Created: 2/14/2017
; Last Modified: 1/18/2019
; You are given a simple stepper motor software system with one input and
; four outputs. This program runs, but you are asked to add minimally intrusive
; debugging instruments to verify it is running properly.
; If the input PA4 is low, the stepper motor outputs cycle 10,6,5,9,...
; If the input PA4 is high, the stepper motor outputs cycle 5,6,10,9,...
; Insert debugging instruments which gather data (state and timing)
; to verify that the system is functioning as expected.
; Hardware connections (External: One button and four outputs to stepper motor)
; PA4 is Button input (1 means pressed, 0 means not pressed)
; PE3-0 are stepper motor outputs
; PF2 is Blue LED on Launchpad used as a heartbeat
; Instrumentation data to be gathered is as follows:
; After every output to Port E, collect one state and time entry.
; The state information is the 5 bits on Port A bit 4 and Port E PE3-0
; place one 8-bit entry in your Data Buffer
; The time information is the 24-bit time difference between this output and the previous (in 12.5ns units)
; place one 32-bit entry in the Time Buffer
;   24-bit value of the SysTick's Current register (NVIC_ST_CURRENT_R)

```
;    you must handle the roll over as Current goes 3,2,1,0,0x00FFFFFF,0xFFFFFE,
; Note: The size of both buffers is 100 entries. Once you fill these
;       entries you should stop collecting data
; The heartbeat is an indicator of the running of the program.
; On each iteration of the main loop of your program toggle the
; LED to indicate that your code(system) is live (not stuck or dead).

SYSCTL_RCGCGPIO_R  EQU 0x400FE608
NVIC_ST_CURRENT_R  EQU 0xE000E018
GPIO_PORTA_DATA_R  EQU 0x400043FC
GPIO_PORTA_DIR_R   EQU 0x40004400
GPIO_PORTA_DEN_R   EQU 0x4000451C
GPIO_PORTE_DATA_R  EQU 0x400243FC
GPIO_PORTE_DIR_R   EQU 0x40024400
GPIO_PORTE_DEN_R   EQU 0x4002451C
GPIO_PORTF_DATA_R  EQU 0x400253FC
GPIO_PORTF_DIR_R   EQU 0x40025400
GPIO_PORTF_DEN_R   EQU 0x4002551C
; RAM Area
        AREA    DATA, ALIGN=2
Index     SPACE 4 ; index into Stepper table 0,1,2,3
Direction SPACE 4 ; -1 for CCW, 0 for stop 1 for CW

;place your debug variables in RAM here
DataBuffer SPACE 100
TimeBuffer SPACE 400
DataPt SPACE 4
TimePt SPACE 4
lastTime SPACE 4
; ROM Area
      IMPORT TExaS_Init
      IMPORT SysTick_Init
;-UUU-Import routine(s) from other assembly files (like SysTick.s) here
      AREA    |.text|, CODE, READONLY, ALIGN=2
      THUMB
Stepper DCB 5,6,10,9
      EXPORT  Start

Start
 ; TExaS_Init sets bus clock at 80 MHz
; PA4, PE3-PE0 out logic analyzer to TExasDisplay
    LDR  R0,=SendDataToLogicAnalyzer
    ORR  R0,R0,#1
    BL   TExaS_Init ; logic analyzer, 80 MHz
 ;place your initializations here
    BL   Stepper_Init ; initializestepper motor
    BL   Switch_Init  ; initialize switch input
;**********************
```

```
    BL   Debug_Init ;(you write this)
         BL      LED_Init
;*********************

    CPSIE  I                                ; TExaS logic analyzer runs on interrupts
    MOV  R5,#0                                      ; last PA4


loop
                LDR R1,=GPIO_PORTF_DATA_R
                LDR R0,[R1]
                EOR R0,#0X04
                STR R0,[R1]
    LDR  R1,=GPIO_PORTA_DATA_R
    LDR  R4,[R1]                            ;current value of switch
    AND  R4,R4,#0x10                        ; select just bit 4
    CMP  R4,#0
    BEQ  no                                 ; skip if not pushed
    CMP  R5,#0
    BNE  no                                 ; skip if pushed last time
    ; this time yes, last time no
    LDR  R1,=Direction
    LDR  R0,[R1]                            ; current direction
    ADD  R0,R0,#1                                   ;-1,0,1 to 0,1,2
    CMP  R0,#2
    BNE  ok
    MOV  R0,#-1                                     ; cycles through values -1,0,1
ok   STR  R0,[R1]                                   ; Direction=0 (CW)
no   MOV  R5,R4                                     ; setup for next time
    BL   Stepper_Step
    LDR  R0,=1600000
    BL   Wait                               ; time delay fixed but not accurate
    B    loop
;Initialize stepper motor interface
Stepper_Init
    MOV R0,#1
    LDR R1,=Direction
    STR R0,[R1]                             ; Direction=0 (CW)
    MOV R0,#0
    LDR R1,=Index
    STR R0,[R1]                             ; Index=0
  ; activate clock for Port E
    LDR R1, =SYSCTL_RCGCGPIO_R
    LDR R0, [R1]
    ORR R0, R0, #0x10                   ; Clock for E
    STR R0, [R1]
    NOP
    NOP                                 ; allow time to finish activating
```

```
   ; set direction register
    LDR R1, =GPIO_PORTE_DIR_R
    LDR R0, [R1]
    ORR R0, R0, #0x0F                ; Output on PE0-PE3
     STR R0, [R1]
   ; enable digital port
    LDR R1, =GPIO_PORTE_DEN_R
    LDR R0, [R1]
    ORR R0, R0, #0x0F                ; enable PE3-0
    STR R0, [R1]
    BX  LR
;Initialize switch interface
Switch_Init
   ; activate clock for Port A
    LDR R1, =SYSCTL_RCGCGPIO_R
    LDR R0, [R1]
    ORR R0, R0, #0x01                ; Clock for A
    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]
    BX  LR
LED_Init
        LDR R1,=SYSCTL_RCGCGPIO_R        ;turning on the clock
        LDRB R0,[R1]
        ORR R0,#0X20
        STRB R0,[R1]
        NOP                                                ;wait for stabilization bc 4 clock cycles
to completely stabilize
        NOP
        LDR R1, =GPIO_PORTF_DIR_R
        LDRB R0,[R1]
        ORR R0,#0X04
        STRB R0,[R1]
        LDR R1, =GPIO_PORTF_DEN_R        ;set digital enable (we use 4 pins)
        LDRB R0,[R1]
        ORR R0, #0X04
        STRB R0,[R1]
        BX LR
```

```
; Step the motor clockwise
; Direction determines the rotational direction
; Input: None
; Output: None
Stepper_Step
    PUSH {R4,LR}
    LDR  R1,=Index
    LDR  R2,[R1]                        ; old Index
    LDR  R3,=Direction
    LDR  R0,[R3]                        ; -1 for CCW, 0 for stop 1 for CW
    ADD  R2,R2,R0
    AND  R2,R2,#3                       ; 0,1,2,3,0,1,2,...
    STR  R2,[R1]                        ; new Index
    LDR  R3,=Stepper                    ; table
    LDRB R0,[R2,R3]                     ; next output: 5,6,10,9,5,6,10,...
    LDR  R1,=GPIO_PORTE_DATA_R      ; change PE3-PE0
    STR  R0,[R1]
    BL   Debug_Capture
    POP {R4,PC}
; inaccurate and inefficient time delay
Wait
    SUBS R0,R0,#1                       ; outer loop
    BNE  Wait
    BX   LR

Debug_Init
    PUSH {R0-R4,LR}
; you write this
                LDR R1,=DataBuffer
                MOV R2,#100                         ;set up count for the array
loop1   LDR R0,[R1]
                CMP R2,#0
                BEQ done                            ;check to see if array is at end
                MOV R3,#0XFF
                STR R3,[R1]                         ;make every entry 0xFF
                ADD R1,R1,#1                    ;move address
                SUB R2,R2,#1                    ;decrement counter
                B loop1
done    LDR R1,=TimeBuffer              ;first element in array
                MOV R2,#100                         ; set up count
loop2   LDR R0,[R1]
                CMP R2,#0
                BEQ done2                           ;check to see if array is at end
                MOV R3,#0XFFFFFFFF              ; make every enter 0xffffffff
                STR R3,[R1]
                ADD R1,R1,#4                    ;increment address by 4 because 32 bits
                SUB R2,R2,#1                    ;decrement counter
                B loop2
```

```
done2   LDR R1,=DataPt
        LDR R2,=TimePt
        LDR R3,=TimeBuffer
        LDR R4,=DataBuffer
        STR R3,[R2]
        STR R4,[R1]                                 ;store pointer to top of arrays

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

;Debug capture
Debug_Capture
    PUSH {R0-R12,LR}
; you write this
        LDR R1, =DataPt
        LDR R0, [R1]
        LDR R2, =DataBuffer
        ADD R2, R2,#100
        CMP R2, R0
        BNE continue                                ;check to see if at end if so continue
        POP  {R0-R12,PC}
continue

        ; Mask PA5, PE3-PE0
        LDR R1, =GPIO_PORTE_DATA_R
        LDR R2, [R1]                                ;  PortE Data
        LDR R1, =GPIO_PORTA_DATA_R
        LDR R3, [R1]                                ;  PortA Data
        ADD R4, R3, R2                              ;  Add portE data and port A data
        LDR R5, =DataPt
        LDR R6, [R5]                                ; R6 contains current address of dataBuf
        STRB R4, [R6]
        ADD R6, R6, #1
        STR R6, [R5]                                ; storing next data address
        LDR R1, =NVIC_ST_CURRENT_R
        LDR R2, [R1]                                ; R2 contains the current count value
        LDR R4, =lastTime               ; Calculate the 24-bit elapsed time
        LDR R5, [R4]
        SUB R5, R5, R2                              ; Previous - Current
        AND R5, #0x00FFFFFF
        LDR R8, =TimePt
        LDR R9, [R8]
        STR R5, [R9]                                ; Store elapsed time into timeBuf
        STR R2, [R4]                                ;store current time into lastTime
        ADD R9, R9, #4                              ; 32 bit element array
        STR R9, [R8]
```

```
                ;62 cycles* 12.5ns = 775ns
                ;Percentage Overhead = ((775*10^-9)/.08)100 = 0.00097%

                ;Minimum elapsed time: 61cycles*12.5ns = 762.5ns
                ;Maximum elapsed time: 63cycles*12.5ns = 787.5ns
    POP  {R0-R12,PC}
; edit the following only if you need to move pins from PA4, PE3-0
; logic analyzer on the real board
PA4  equ  0x40004040
PE30 equ  0x4002403C
UART0_DR_R equ 0x4000C000
SendDataToLogicAnalyzer
    LDR  R1,=PA4
    LDR  R1,[R1]                        ; read PA4
    LDR  R0,=PE30                       ; read PE3-PE0
    LDR  R0,[R0]
    ORR  R0,R0,R1                       ;combine into one 5-bit value
    ORR  R0,R0,#0x80
    LDR  R1,=UART0_DR_R
    STR  R0,[R1]                    ; send data at 10 kHz
    BX   LR


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