EECS 3100, 043

Embedded Systems

Dr. Devinder Kaur

TA: Mehzabien Iqbal

Lab 6 - Minimally Intrusive Debugging Methods

Aiden Rader

07/18/2025

Aiden Rader
R01513501
07/18/2025
Embedded Lab #6

Table of Contents:

Aiden Rader
R01513501
07/18/2025
Embedded Lab #6

Introduction:

Lab 6's primary focus was on implementing *minimally intrusive debugging methods* on the TM4C123GH6PM microcontroller. These debugging methods (the *dump* and *heartbeat* instruments) allow embedded developers to observe program behavior in real-time without significantly affecting system performance.

The *dump* instrument collects and stores system state and timing data during operation, which can later be analyzed to verify software correctness and timing constraints. This is particularly useful in embedded systems, where traditional "printf"-style debugging is often impractical or completely useless.

The *heartbeat* instrument provides a visual confirmation (via toggling an onboard LED on PF2) that the main loop is executing as expected, helping identify deadlocks or hangs in real time.

The primary goals of this lab were to (1) enhance the Lab 5 hardware with debugging features, (2) use the SysTick timer and PLL to enable precise performance monitoring, and (3) demonstrate the implementation on actual hardware as well as in simulation mode. This lab served as an essential step in understanding embedded debugging practices that avoid disrupting real-time constraints.

Procedure:

Below we can see a simplified version of our procedure for Lab 6:

1. **Preparation**
   We began by reviewing the required documentation and importing the existing Lab 5 project into a new Keil project template for Lab 6. The system was configured to use the PLL to run the microcontroller at 80 MHz, and the SysTick timer was initialized to facilitate precise timing measurements.
2. **Part A – Implementing the Dump Instrument**
   We wrote two main assembly routines: **Debug_Init** and **Debug_Capture**.

   - **Debug_Init** allocated and initialized two arrays—*DataBuffer* and *TimeBuffer*—to store approximately 3 seconds of captured Port E states and SysTick values.

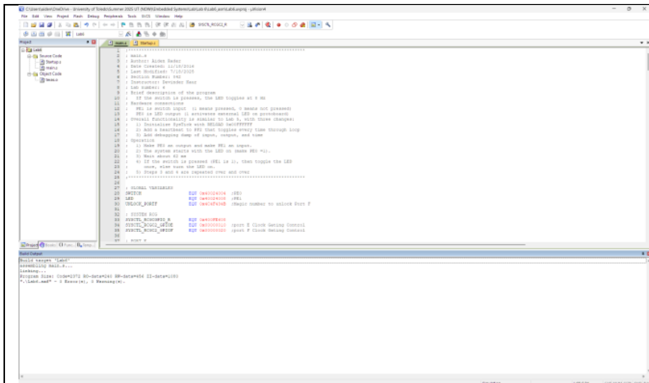
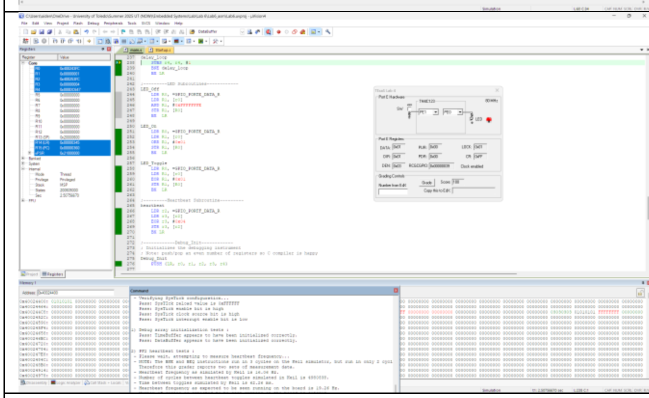
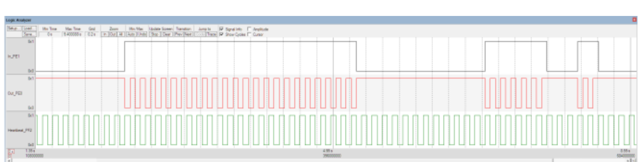
   - **Debug_Capture** was inserted at the start of the outer loop to record PE1 (input) and PE0 (output) into a packed byte, along with the current SysTick timer value. Proper masking and bit-shifting were used to extract and encode the data efficiently.

3. **Part B – Estimating Intrusiveness**
   We estimated the execution time of Debug_Capture by counting the instructions and multiplying by 2 (cycles), then converting to time using a 12.5 ns bus cycle. The overhead was calculated as a percentage of total loop time, verifying that our implementation was minimally intrusive.
4. **Part C – Heartbeat Implementation**
   A heartbeat LED on PF2 was added to visually indicate that the program was actively

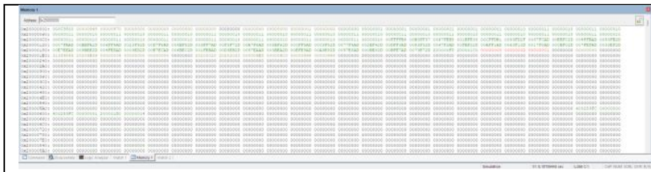
executing its loop. The LED toggled with each loop iteration, confirming continuous program activity.

5. **Part D – Simulation and Debugging**
   We tested our implementation first in Keil's simulation mode, verifying buffer values and loop timing via memory and I/O windows (Figures 6.2–6.4 style). Once validated, the code was uploaded to the TM4C123 board and verified using physical switches and LEDs.
6. **Part E – Capturing and Analyzing Timing**
   A data dump was saved during a "no-touch, touch, no-touch" sequence of the switch. We then calculated the heartbeat period by analyzing time deltas in the buffer and verified the timing accuracy down to 12.5 ns resolution.

Screenshots:

	Here is a screenshot of the compile status of the program! We can see 0 errors and 0 warnings. I did have one linker error; this was fixed by using a *PRESERVE8* keyword above the *THUMB* keyword before start. This was because there was a misalignment between my program and the TeXaS_Init call.
	Like Figure 6.2 (TExaS GUI) The TExaS simulator confirms that all four components; GPIO initialization, SysTick configuration, heartbeat timing, and debug logging, were implemented correctly. The final score of 100 indicates full compliance with lab specifications.
	Like Figure 6.3 (Heartbeat Logic Analyzer) This is a **screenshot of the Logic Analyzer** (or GPIO viewer), showing PF2 toggling periodically (~16 Hz on Keil, ~19 Hz on hardware), I also used a method of making new variables to show these signals.
	Like Figure 6.4 (Memory Dump) The DataBuffer (starting from address 0x20000000) contains recorded

Aiden Rader
R01513501
07/18/2025
Embedded Lab #6

	input/output states from Port E. The 0x10 value signifies that the **switch was pressed** (PE1 = 1) and the **LED was off** (PE0 = 0), as expected during the debug capture phase.

Main Code (Source):

```
;*******************************************************************
; main.s
; Author: Aiden Rader
; Date Created: 11/18/2016
; Last Modified: 7/18/2025
; Section Number: 042
; Instructor: Devinder Kaur
; 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
;*******************************************************************

; GLOBAL VARIABLES
SWITCH                  EQU 0x40024004  ;PE0
LED                     EQU 0x40024008  ;PE1
UNLOCK_PORTF            EQU 0x4C4F434B  ;Magic number to unlock Port F

; SYSTEM RCG
SYSCTL_RCGCGPIO_R      EQU 0x400FE608
SYSCTL_RCGC2_GPIOE     EQU 0x00000010  ;port E Clock Gating Control
SYSCTL_RCGC2_GPIOF     EQU 0x00000020  ;port F Clock Gating Control

; PORT E
GPIO_PORTE_DATA_R      EQU 0x400243FC
GPIO_PORTE_DIR_R       EQU 0x40024400
GPIO_PORTE_AFSEL_R     EQU 0x40024420
GPIO_PORTE_AMSEL_R     EQU 0x40024528
GPIO_PORTE_PUR_R       EQU 0x40024510
GPIO_PORTE_DEN_R       EQU 0x4002451C
GPIO_PORTE_PCTL_R      EQU 0x4002452C

; PORT F
GPIO_PORTF_DATA_R      EQU 0x400253FC
GPIO_PORTF_DIR_R       EQU 0x40025400
```

Aiden Rader
R01513501


```
GPIO_PORTF_AFSEL_R    EQU 0x40025420
GPIO_PORTF_PUR_R      EQU 0x40025510
GPIO_PORTF_DEN_R      EQU 0x4002551C
GPIO_PORTF_AMSEL_R    EQU 0x40025528
GPIO_PORTF_PCTL_R     EQU 0x4002552C
GPIO_PORTF_LOCK_R     EQU 0x40025520
GPIO_PORTF_CR_R       EQU 0x40025524

; SYSTICK
NVIC_ST_CTRL_R        EQU 0xE000E010
NVIC_ST_RELOAD_R      EQU 0xE000E014
NVIC_ST_CURRENT_R     EQU 0xE000E018

        THUMB
        AREA    DATA, ALIGN=4
SIZE    EQU    50

;You MUST use these two buffers and two variables
;You MUST not change their names
DataBuffer SPACE  SIZE*4
TimeBuffer SPACE  SIZE*4
DataPt    SPACE  4
TimePt    SPACE  4
Out_PE0       SPACE 4
In_PE1        SPACE 4
Heartbeat_PF2   SPACE 4

;These names MUST be exported
      EXPORT DataBuffer
      EXPORT TimeBuffer
      EXPORT DataPt [DATA,SIZE=4]
      EXPORT TimePt [DATA,SIZE=4]
      EXPORT  Out_PE0
      EXPORT  In_PE1
      EXPORT  Heartbeat_PF2

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

;------------init_PortE------------
init_PortE

   ; Enable Port E Clock
   LDR r1, =SYSCTL_RCGCGPIO_R
   LDR r0, [r1]
   ORR r0, r0, #0x10
   STR r0, [r1]

   NOP
   NOP

   ; Disable analog function
   LDR r0, =GPIO_PORTE_AMSEL_R
   MOV r1, #0x00
   STR r1, [r0]

   ; Set Direction PE0 = output, PE1 = input
   LDR r0, =GPIO_PORTE_DIR_R
   MOV r1, #0x01
   STR r1, [r0]
```

Aiden Rader
R01513501


```
  ; Disable alternate functions
  LDR r0, =GPIO_PORTE_AFSEL_R
  MOV r1, #0x00
  STR r1, [r0]

  ; Enable digital for PE0 and PE1
  LDR r0, =GPIO_PORTE_DEN_R
  MOV r1, #0x03
  STR r1, [r0]
  BX LR

;------------init_PortF------------
init_PortF

		; Enable Clock for Port F
		LDR r0, =SYSCTL_RCGCGPIO_R
		LDR r1, [r0]
		ORR r1, r1, #0x20
		STR r1, [r0]

		; Set small delay
		NOP
		NOP

		; Unlock Port F
		LDR r0, =GPIO_PORTF_LOCK_R
		LDR r1, =UNLOCK_PORTF
		STR r1, [r0]

		; Allow changes to Port F
		LDR r0, =GPIO_PORTF_CR_R
		LDR r1, [r0]
		MOV r1, #0x04
		STR r1, [r0]

		; Turn off AMSEL for Port F
		LDR r0, =GPIO_PORTF_AMSEL_R
		LDR r1, [r0]
		AND r1, #0x00
		STR r1, [r0]

		; Set Direction (input/output)
		LDR r0, =GPIO_PORTF_DIR_R
		LDR r1, [r0]
		MOV r1, #0x04
		STR r1, [r0]

		; Turn off AFSEL for Port E
		LDR r0, =GPIO_PORTF_AFSEL_R
		LDR r1, [r0]
		AND r1, #0x00
		STR r1, [r0]

		; Digital Enable for PE0 & PE1
		LDR r0, =GPIO_PORTF_DEN_R
		LDR r1, [r0]
		ORR r1, r1, #0x04
		STR r1, [r0]

		; Set the GPIO Mode
		LDR r0, =GPIO_PORTF_DATA_R
  LDR r1, [r0]
		AND r1, #0xFFFFFFFF
		STR r1, [r0]
```

Aiden Rader
R01513501


```
        BX LR  ; Return from function

;------------SysTick_Init-----------------
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, =0x00FFFFFF;                  ; 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, #(0x00000001+0x00000004)
        STR R0, [R1]                  ; [R1] = R0 = (NVIC_ST_CTRL_ENABLE|NVIC_ST_CTRL_CLK_SRC)
        BX  LR                        ; return

;------------Start-----------------

Start
    BL TExaS_Init  ; running at 80 MHz, scope voltmeter on PD3

    ; initialize Port E and F
        BL init_PortE
        BL init_PortF

    ; initialize debugging dump, including SysTick
        BL Debug_Init

    CPSIE  I    ; TExaS voltmeter, scope runs on interrupts

loop
        ; Read PE1 (input switch)
        LDR r0, =GPIO_PORTE_DATA_R
        LDR r1, [r0]
        AND r1, r1, #0x02        ; mask bit 1
        LDR r2, =In_PE1
        STR r1, [r2]             ; store PE1 value

        ; Read PE0 (LED output)
        LDR r0, =GPIO_PORTE_DATA_R
        LDR r1, [r0]
        AND r1, r1, #0x01        ; mask bit 0
        LDR r2, =Out_PE0
        STR r1, [r2]             ; store PE0 value

        ; Read PF2 (heartbeat)
        LDR r0, =GPIO_PORTF_DATA_R
        LDR r1, [r0]
        AND r1, r1, #0x04
        LSR r1, r1, #2           ; shift to bit 0
        LDR r2, =Heartbeat_PF2
        STR r1, [r2]             ; store PF2 value

        BL Debug_Capture
```

```
        BL heartbeat  ; heartbeat
        BL delay  ; Delay

        ;input PE1 test output PE0
        LDR r0, =GPIO_PORTE_DATA_R
        LDR r1, [r0]
        AND r1, #0x02     ; Isolate PE1
        CMP r1, #0x02     ; Compare to 1
        BNE switch_off
        BL LED_Toggle     ; If not pressed
        B loop

;------------Delay Subroutines------------
switch_off
        BL LED_On
        B loop

delay
        MOV r4, #0x00130000

delay_loop
        SUBS r4, r4, #1
        BNE delay_loop
        BX LR

;---------LED Subroutines------------
LED_Off
        LDR R0, =GPIO_PORTE_DATA_R
        LDR R1, [r0]
        AND R1, #0xFFFFFFFE
        STR R1, [R0]
        BX  LR
LED_On
        LDR R0, =GPIO_PORTE_DATA_R
        LDR R1, [r0]
        ORR R1, #0x01
        STR R1, [R0]
        BX  LR

LED_Toggle
        LDR R0, =GPIO_PORTE_DATA_R
        LDR R1, [r0]
        EOR R1, #0x01
        STR R1, [R0]
        BX  LR

;---------Heartbeat Subroutine---------
heartbeat
        LDR r0, =GPIO_PORTF_DATA_R
        LDR r1, [r0]
        EOR r1, r1 , #0x04  ; toggle PF2
        STR r1, [r0]
        BX LR

;------------Debug_Init------------
; Initializes the debugging instrument
; Note: push/pop an even number of registers so C compiler is happy
Debug_Init
        PUSH {LR, r0, r1, r2, r3, r4}

        ; initialize buffers to 0
        LDR r0, =SIZE
        LDR r1, =DataBuffer
        LDR r2, =TimeBuffer
```

```
        MOV r3, #0xFFFFFFFF

Debug_Init_Loop
        STR r3, [r1]
        STR r3, [r2]
        ADD r1, r1, #4
        ADD r2, r2, #4
        SUBS r0, r0, #1
        BNE Debug_Init_Loop

        LDR r0, =DataPt
        LDR r1, =DataBuffer
        STR r1, [r0] ; initialize data pointer

        LDR r0, =TimePt
        LDR r1, =TimeBuffer
        STR r1, [r0] ; initialize time pointer

        ; init SysTick
        BL SysTick_Init

        POP {LR, r0, r1, r2, r3, r4}
        BX LR

;------------Debug_Capture------------
; Dump Port E and time into buffers
; Note: push/pop an even number of registers so C compiler is happy
Debug_Capture
        PUSH {r0, r1, r2, r3}

        LDR r0, =DataPt
        LDR r2, [r0]
        LDR r1, =DataBuffer
        LDR r3, =SIZE
        LSL r3, r3, #2
        ADD r1, r1, r3
        CMP r2, r1
        BHS Debug_Capture_Return

        ; Read PE data
        LDR r0, =GPIO_PORTE_DATA_R
        LDR r1, [r0]

        ; Read current time
        LDR r0, =NVIC_ST_CURRENT_R
        LDR r3, [r0]

        AND r1, r1, #0x03 ; mask to bits 1 and 0
        MOV r2, r1 ; copy data
        LSL r2, r2, #3 ; shift bit 1 to bit 4
        AND r2, r2, #0x10 ; mask
        AND r1, r1, #0x01 ; mask
        ORR r1, r1, r2 ; combine back into one register

        ; Store packed value in DataBuffer
        LDR r0, =DataPt
        LDR r2, [r0]
        STR r1, [r2]
        ADD r2, r2, #4
        STR r2, [r0]

        ; Store SysTick in TimeBuffer
        LDR r0, =TimePt
        LDR r2, [r0]
```

```
        STR r3, [r2]
        ADD r2, r2, #4
        STR r2, [r0]

Debug_Capture_Return
        POP {r0, r1, r2, r3}
        BX LR

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

Conclusion:

Lab 6 successfully demonstrated how to incorporate minimally intrusive debugging techniques in embedded systems using the TM4C123 microcontroller. By implementing a dump mechanism and a heartbeat signal, we gained practical skills in capturing and analyzing system behavior during real-time execution. These tools helped bridge the gap between abstract debugging methods and their concrete implementation in embedded environments.

The dump allowed us to record both logic state and performance data without interfering with core functionality, and the heartbeat provided an immediate, visual indication of proper system operation. Overall, this lab emphasized the importance of efficient debugging in real-time systems and solidified our understanding of low-level system monitoring using ARM assembly.