ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 351E

MICROCOMPUTER LABORATORY EXPERIMENT REPORT

EXPERIMENT NO	:	5
EXPERIMENT DATE	:	11.12.2024
LAB SESSION	:	WEDNESDAY - 12.30
GROUP NO	:	G8

GROUP MEMBERS:

150200009	:	Vedat Akgöz
150200097	:	Mustafa Can Çalışkan
150200016	:	Yusuf Şahin

SPRING 2024

Contents

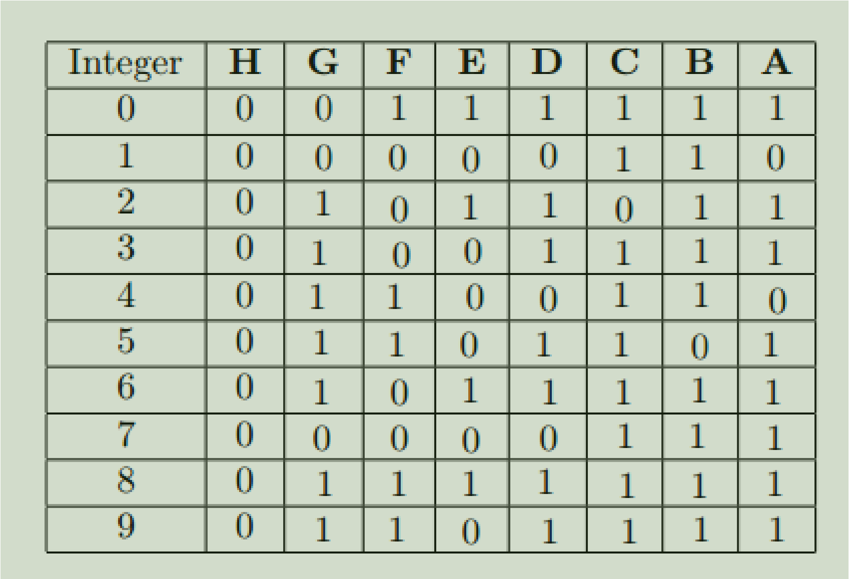
1 Introduction

This experiment aimed to enhance practical understanding of interfacing with a 7-segment display and utilizing interrupts on the MSP430 microcontroller. The objective was to develop programs to drive the 7-segment display and modify its behavior using interrupts. We gained experience in initializing GPIO ports, creating delay loops, and integrating interrupt service routines to manipulate counting modes.

2 Materials and Methods

2.1 Introduction Part

Below is the illustration of the 7-segment display used in this experiment:



Integer	H	G	F	E	D	C	B	A
0	0	0	1	1	1	1	1	1
1	0	0	0	0	0	1	1	0
2	0	1	0	1	1	0	1	1
3	0	1	0	0	1	1	1	1
4	0	1	1	0	0	1	1	0
5	0	1	1	0	1	1	0	1
6	0	1	0	1	1	1	1	1
7	0	0	0	0	0	1	1	1
8	0	1	1	1	1	1	1	1
9	0	1	1	0	1	1	1	1

Figure 1: 7-segment display and its GPIO Port connections.

The 7-segment display consists of 7+1 LEDs controlled by GPIO ports, as depicted in the figure. The table above outlines the inputs required to represent digits 0-9 on the display.

2.2 Part 1

The objective of Part 1 was to write an assembly program to display decimal numbers (0-9) on a 7-segment display. Below is the implemented assembly code:

```
SetupP1     bis.b   #0xFF,&P1DIR            ; Set Port 1 as output
            bis.b   #0x08,&P2DIR            ; Set P2.3 as output
            bis.b   #0x08,&P2OUT            ; Turn on P2.3
            mov.w   #0, R13                 ; Initialize R13 to 0 (counter/index)

Mainloop    mov.w   #array, R12             ; Reset R12 to array base address
            add.w   R13, R12                ; Calculate array offset
            mov.b   0(R12), &P1OUT          ; Send value to P1OUT (7-segment)
            call    #Delay                  ; 1-second delay
            inc.w   R13                     ; Increment counter
            cmp.w   #0x0A, R13              ; Check if counter == 10
            jne     Mainloop                ; If not, loop back
            mov.w   #0, R13                 ; Reset counter to 0
            jmp     Mainloop                ; Restart loop

Delay       mov.w   #0x0A, R14              ; Outer loop count (10)
L2          mov.w   #0x7A00, R15           ; Inner loop count (31250)
L1          dec.w   R15                    ; Decrement inner loop
            jnz     L1                      ; If not zero, loop
            dec.w   R14                    ; Decrement outer loop
            jnz     L2                      ; If not zero, loop
            ret                            ; Return from delay

exit        nop                            ; Program exit point
```

2.3 Part 2

In Part 2, an interrupt-based mechanism was implemented to toggle counting modes between even and odd numbers based on an external interrupt. The updated assembly code is shown below:

```
init_INT
    bis.b   #040h, &P2IE
    and.b   #0BFh, &P2SEL
    and.b   #0BFh, &P2SEL2
    bis.b   #040h, &P2IES
    clr     &P2IFG
    eint
```

```
SetupP1
    bis.b   #0FFh,&P1DIR
    bis.b   #008h,&P2DIR
    bis.b   #008h,&P2OUT
    mov.w   #00h, R5

SetupP2
    mov.w   R5, R13
    mov.w   #array, R12

Mainloop
    mov.w   #array, R12             ; Reset R12 to array base address
    add.w   R13, R12                ; Calculate array offset
    mov.b   0(R12), &P1OUT          ; Send value to P1OUT (7-segment)
    call    #Delay                  ; 1-second delay
    add.w   #2, R13                     ; Increment counter
    cmp.w   #0x0A, R13              ; Check if counter == 10
    jl      Mainloop              ; If not, loop back
    mov.w   #0, R13                 ; Reset counter to 0
    jmp     SetupP2                ; Restart loop


Delay
    mov.w   #0Ah, R14
L2
    mov.w   #07A00h, R15
L1
    dec.w   R15
    jnz     L1
    dec.w   R14
    jnz     L2
    ret

ISR
    dint
    xor.w   #01h, R5
    dec.w   R13
    clr     &P2IFG
    eint
    reti

exit
    nop
```

3 Discussions

In this experiment, we gained hands-on experience with GPIO configuration and delay function implementation on the MSP430 microcontroller. During **Part 1**, we utilized an array-based lookup table for 7-segment display data, which simplified the logic for converting digits into segment patterns. However, precise timing was crucial for ensuring accurate display updates. Calibrating the delay functions highlighted the delicate balance required between system responsiveness and computational overhead in embedded systems.

In **Part 2**, we integrated interrupts to dynamically toggle between counting modes, such as even and odd numbers. Configuring GPIO pins for interrupt-driven functionality was relatively straightforward.

4 Results

The outcomes of our implementation were as follows:

- **Part 1**: We successfully displayed digits 0 through 9 on the 7-segment display. The delay-based mechanism provided accurate and consistent timing for the display updates.
- **Part 2**: We implemented interrupt-driven functionality that allowed seamless switching between even and odd counting modes in response to external inputs. This demonstrated the practical advantages of using interrupts to handle dynamic external events.

5 Conclusion

This experiment enhanced our understanding of key concepts in embedded system development, such as GPIO configuration, delay function design, and interrupt handling.

It also emphasized the importance of modular and reusable code to ensure flexibility and scalability.

Key lessons learned from this experiment include:

1. Array-based lookup methods are effective for simplifying display logic.
2. Precise timing mechanisms are critical for ensuring system accuracy.
3. Interrupt-driven designs require careful handling of race conditions to maintain system stability.

REFERENCES