ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 351E
MICROCOMPUTER LABORATORY EXPERIMENT REPORT

EXPERIMENT NO	:	6
EXPERIMENT DATE	:	18.12.2024
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 focused on utilizing timers, one of the fundamental features of microcontrollers, specifically the MSP430 microcontroller, to explore two distinct applications: multi-digit display control and a chronometer.

2 Materials and Methods

2.1 Part 1

The objective of Part 1 was to implement an infinite loop in the main program to simultaneously control multiple digits of a 7-segment display. The desired behavior required a flashing effect, imperceptible to the human eye due to high-frequency switching, resulting in a constant light appearance.

The main loop code used in Part 1 is presented below:

```
MainLoop
    mov.b   3(R4), &P1OUT     ; Display digit 4 on P1OUT
    mov.b   #08h, &P2OUT      ; Select segment for digit 4
    call    #DELAY            ; Delay for visibility
    clr     &P1OUT
    clr     &P2OUT

    mov.b   2(R4), &P1OUT     ; Display digit 3 on P1OUT
    mov.b   #04h, &P2OUT      ; Select segment for digit 3
    call    #DELAY
    clr     &P1OUT
    clr     &P2OUT

    mov.b   1(R4), &P1OUT     ; Display digit 2 on P1OUT
    mov.b   #02h, &P2OUT      ; Select segment for digit 2
    call    #DELAY
    clr     &P1OUT
    clr     &P2OUT

    mov.b   0(R4), &P1OUT     ; Display digit 1 on P1OUT
    mov.b   #01h, &P2OUT      ; Select segment for digit 1
    call    #DELAY
```

```
    clr     &P1OUT
    clr     &P2OUT

    jmp     MainLoop          ; Repeat loop
```

2.2 Part 2

Part 2 involved designing a functional chronometer with the following features:

- **Reset**: Resets the time to 0.
- **Stop**: Stops the counter.
- **Start**: Starts the counter.
- **Save Best Time**: Saves the longest recorded duration when both *Start* and *Stop* buttons are pressed simultaneously.

The main code logic implemented in Part 2 is provided below:

```
MainProgram
            call #ConvertDisplayRoutine
            jmp MainProgram

ConvertDisplayRoutine
            call #NumericConversionRoutine
            mov.b @r4, &P1OUT
            mov.b #08h, &P2OUT
            nop
            nop
            clr &P1OUT
            clr &P2OUT
            mov.b @r5, &P1OUT
            mov.b #04h, &P2OUT
            nop
            nop
            clr &P1OUT
            clr &P2OUT
            mov.b @r6, &P1OUT
            mov.b #02h, &P2OUT
            nop
```

```
            nop
            clr &P1OUT
            clr &P2OUT
            mov.b @r7, &P1OUT
            mov.b #01h, &P2OUT
            nop
            nop
            clr &P1OUT
            clr &P2OUT
            ret

ButtonInterruptHandler
            push r15
            mov.b &P2IFG, r15
            bit.b #040h, r15
            jnz HandleResetRoutine
            bit.b #020h, r15
            jnz HandlePauseRoutine
            bit.b #080h, r15
            jnz HandleStartRoutine

ExitISR    clr &P2IFG
            pop r15
            reti

HandleResetRoutine
            mov.b #00h, &timer_sec
            mov.b #00h, &timer_cs
            bic.b #01h, &control_reg
            jmp ExitISR

HandlePauseRoutine
            bic.b #01h, &control_reg
            bit.b #0A0h, &P2IN
            jnz RecordBestTimeRoutine
            jz HandlePauseRoutine
            jmp ExitISR
```

```
HandleStartRoutine
            mov.b #01h, &control_reg
            jmp ExitISR

RecordBestTimeRoutine
            mov.b &timer_sec, r14
            cmp.b &record_sec, r14
            jl StoreRecordRoutine
            jmp ExitISR
            mov.b &timer_cs, r14
            cmp.b &record_cs, r14
            jhs ExitISR

StoreRecordRoutine
            mov.b &timer_sec, &record_sec
            mov.b &timer_cs, &record_cs
            jmp ExitISR

TimerInterruptHandler
            dint
            push r15
            cmp #00h, &control_reg
            jz EndTimerRoutine
            add.b #1b, &timer_cs
            mov.b &timer_cs, r15
            bic.b #0F0h , r15
            cmp #0Ah, r15
            jz IncrementSecondsRoutine
            jmp EndTimerRoutine

IncrementSecondsRoutine
            add.b #010h , &timer_cs
            bic.b #00Fh, &timer_cs
            mov.b &timer_cs, r15
            cmp #0A0h, r15
            jz SecondCycleRoutine
```

```
            jmp EndTimerRoutine

SecondCycleRoutine
            add.b #001h , &timer_sec
            bic.b #0FFh , &timer_cs
            mov.b &timer_sec, r15
            cmp #0Ah, r15
            jz RolloverCheckRoutine
            jmp EndTimerRoutine

RolloverCheckRoutine
            add.b #010h , &timer_sec
            bic.b #00Fh, &timer_sec
            mov.b &timer_sec, r15
            cmp #0A0h, r15
            jz ResetRoutine

EndTimerRoutine
            pop r15
            eint
            reti

NumericConversionRoutine
            push r14
            mov.b &timer_cs, r14
            bic.b #0F0h, r14
            mov.w #digit_map, r4
            add.w r14, r4
            mov.b &timer_cs, r14
            rra.b r14
            rra.b r14
            rra.b r14
            rra.b r14
            bic.b #0F0h, r14
            mov.w #digit_map, r5
            add.w r14, r5
            mov.b &timer_sec, r14
```

```
bic.b #0F0h, r14
mov.w #digit_map, r6
add.w r14, r6
mov.b &timer_sec, r14
rra.b r14
rra.b r14
rra.b r14
rra.b r14
bic.b #0F0h, r14
mov.w #digit_map, r7
add.w r14, r7
pop r14
ret
```

A sample initialization of timer variables:

```
data
seconds .byte 00h
centiseconds .byte 00h
```

Registers used for timer configuration included TA0CTL, TA0CCR0, and TA0CCTL0, adhering to the settings specified in the experiment guide.

3 Discussions

The experiment emphasized the importance of precise timer configuration and subroutine management in embedded systems. Part 1 highlighted the effectiveness of high-frequency multiplexing for display control, while Part 2 demonstrated the integration of timer interrupts and logical control for creating a multi-functional chronometer.

4 Results

4.1 Part 1

Successfully achieved stable multi-digit control on the 7-segment display, replicating the example output shown in Figure 1.

4.2 Part 2

Implemented and tested all chronometer features. The *Save Best Time* function was validated under various scenarios, ensuring correct operation.

5 Conclusion

Both parts of the experiment were completed successfully, demonstrating the practical applications of timers in embedded systems. The experiment reinforced the understanding of timer-based control mechanisms, interrupt handling, and real-time system design.

REFERENCES