

Section 11

Timers

Chapter 11.1 Timers

**Objectives of
this Chapter****Having studied this chapter you will be able to:**

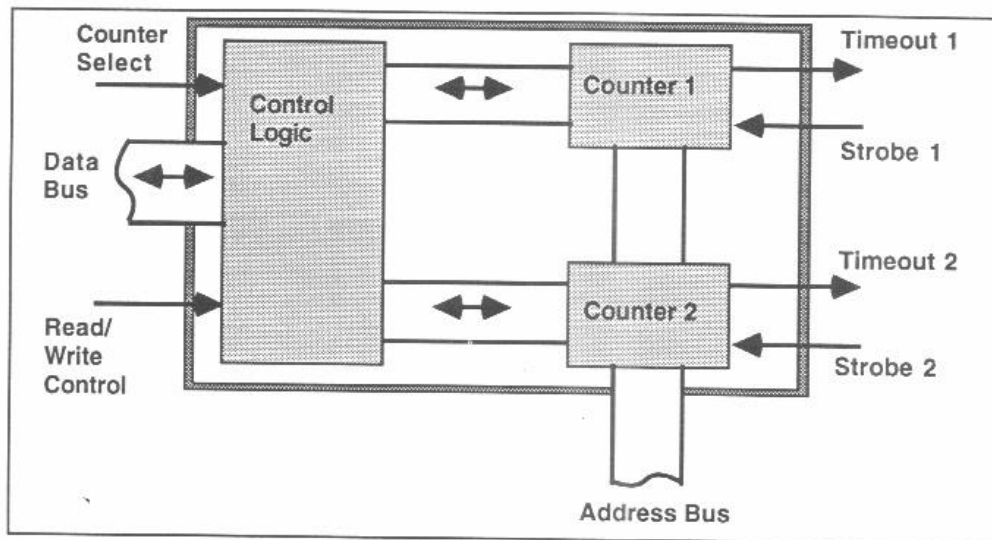
- Sketch a simplified block diagram of a typical Programmable Timer Device
- Explain how a Programmable Timer may be used as:
an Interval Timer
an Event Counter
- Describe the function of the 8256 MUART Timers
- Describe the following 8256 MUART Control Registers:
Command Register
Interrupt Enable and Reset Registers
Mode Register
- Write programs which use the 8256 MUART as a programmable timer.

Introduction

In previous chapters we saw how delays could be produced by using a time wasting routine. Clearly this is wasteful of CPU time. A Programmable Timer is a hardware device which can be used to interrupt the processor. So, for example, the timer could **interrupt** the CPU after a given time delay. This allows the processor to continue with some other program task in the meantime and is therefore far more efficient of CPU time.

Programmable Timers

A simplified block diagram for a typical Programmable Timer is shown below:



The counter registers appear as special data ports and so can be loaded or read using OUT and IN respectively. Each time a "strobe" signal is applied to a counter, the contents of that counter are **decremented** (ie: reduced by 1). When the contents of the counter reach zero, the **corresponding** timeout signal is asserted. This type of device can be used in two different ways:

- 1 as an Interval Timer
- 2 as an Event Counter

Applications of programmable timers include the generation of periodic interrupts and the measurement of frequency or elapsed time.

Interval Timer

The counter register is loaded with a value and the strobe input for that register connected to the system clock. This will cause a timeout signal to occur after a known delay.

For example:

Suppose Counter 1 is loaded with the value $FFFF_{16}$ (65536_{10}) and the strobe is connected to a clock of 2MHz.

$$\text{The clock period is } \frac{1}{2\text{MHz}} = 500\text{ns}$$

So the counter will be decremented every 500ns. This will produce a delay of $65536_{10} \times 500\text{ns} = 32.77\text{ms}$

Event Counter

The counter register is again loaded with a known value but the strobe input is now connected to the **event** signal (eg: a pulse caused by the breaking of an ultrasonic beam). When the number of events reaches the value originally loaded into the counter, a timeout signal will occur.

For example:

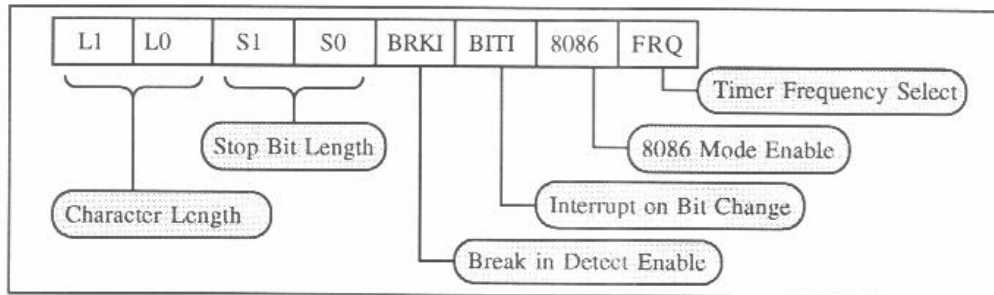
Suppose Counter 2 is loaded with the value $00C8_{16}$ (200_{10}) and the strobe is connected to the output of an ultrasonic detector. Once the ultrasonic beam has been broken 200_{10} times, the timeout signal will be generated.

The 8256 MUART Timers

You have already used the Parallel Interface section of this device for Input/Output programming. Now we shall examine how the Programmable Timer facilities of the 8256 can be used.

The 8256 MUART has five 8-bit timer registers. Certain of these can be cascaded to produce 16-bit timer registers. Chapter 12 of the PAT User Manual gives full details of all MUART Registers. For the present, we shall only consider some of the Control Registers.

Command Register 1



L1, L0, S1 and S0 are only concerned with RS232 serial communications.

BITI is used to select the source for level 1 interrupts:

BITI = 1: Interrupt generated by logic '1' to logic '0' transition on Port 1, bit 7.

BITI = 0: Interrupt generated by timeout of Counter/Timer 2.

BRKI is used to select the mode of operation of Port 1, bit 6:

BRKI = 1: Port 1, bit 6 indicates 'break in transmission' for RS232 serial communications.

BRKI = 0: Port 1, bit 6 operates as a normal general purpose Input/Output bit.

8086 determines the way in which the MUART uses the address lines. This should always be programmed at logic '1' for correct operation of the PAT microcomputer (ie '8086 mode').

FRQ selects the frequency at which timers will be clocked:

FRQ = 1: Timer input frequency is 1kHz.

FRQ = 0: Timer input frequency is 16kHz.

Notes:

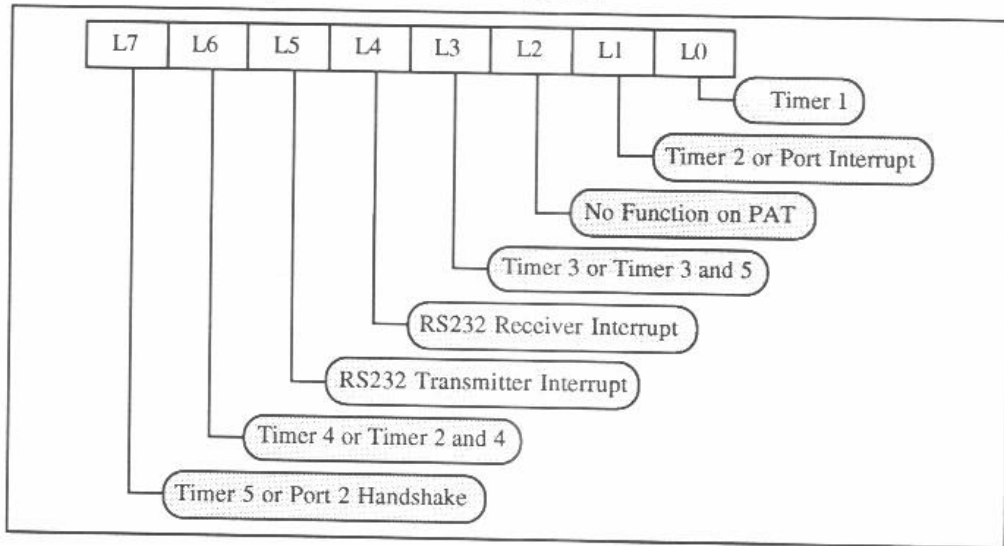
.....

.....

.....

.....

Interrupt Enable and Reset Registers



Writing a logic '1' to the appropriate bit will allow the corresponding register to produce an interrupt. Interrupts can be disabled by writing a logic '1' to the appropriate bit. Alternatively, reading the Interrupt Address register will clear all interrupts.

Notes:

.....

.....

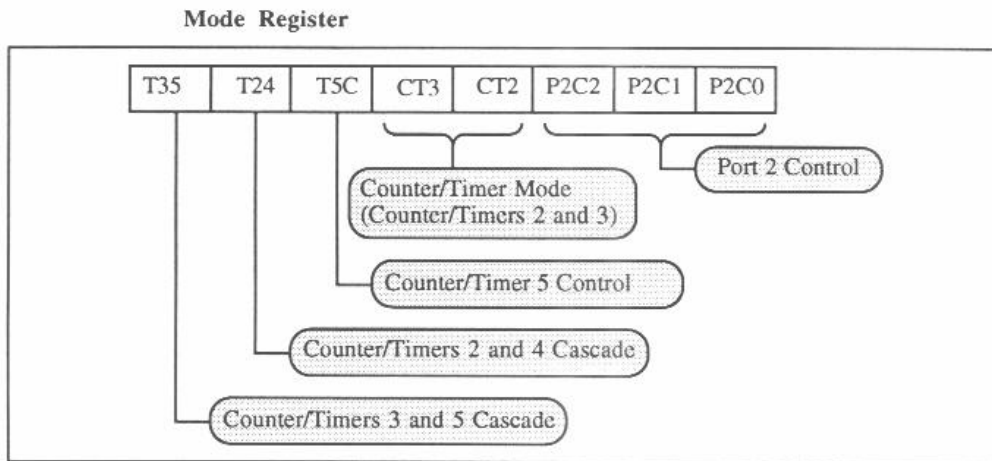
.....

.....

.....

.....

.....



We have already used P2C2, P2C1 and P2C0 to configure Port 2.

CT2 and CT3 determine whether Timer/Counters 2 and 3 are to be used as timers or counters:

- CT2 = 1: Timer/Counter 2 is a **counter**, counting logic '0' to logic '1' transitions on Port 1, bit 2.
- CT2 = 0: Timer/Counter 2 is a **timer**.
- CT3 = 1: Timer/Counter 3 is a **counter**, counting logic '0' to logic '1' transitions on Port 1, bit 2.
- CT3 = 0: Timer/Counter 3 is a **timer**.

Notes:

.....

.....

.....

.....

.....

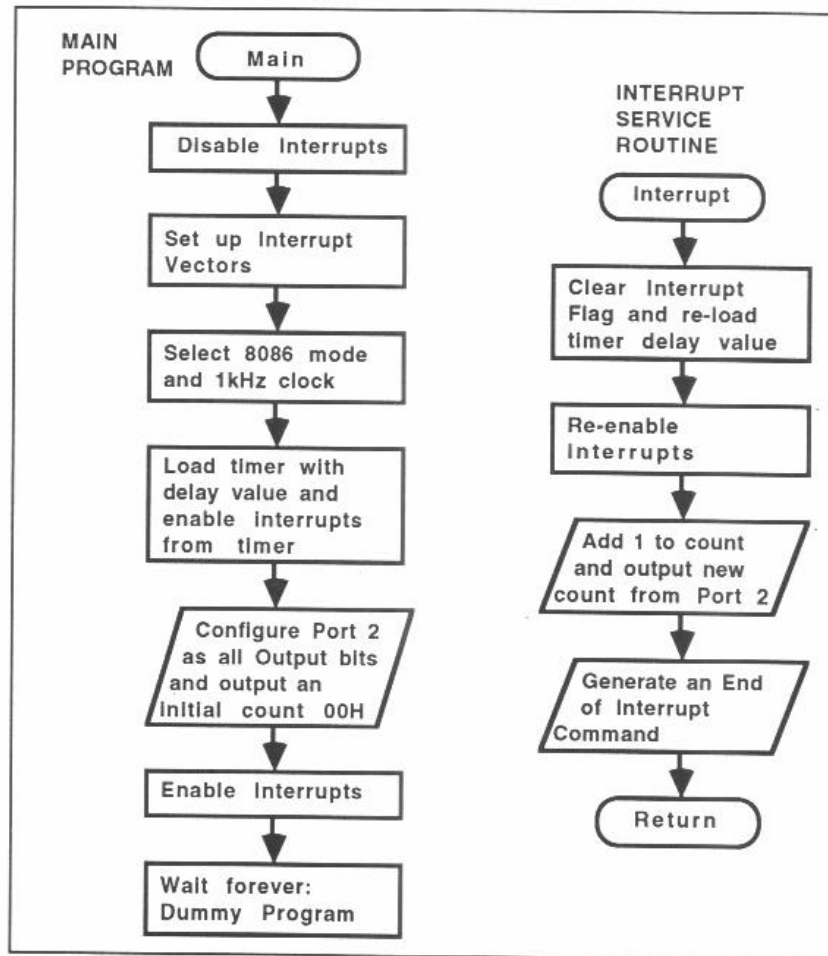
.....

Exercise 31

Write a program which uses a timer to produce an increasing binary count at Port 2. The rate of increase should be about four times per second.

Solution:

Flowchart:



The assembly language program will be:

```

;MAIN PROGRAM:
    ORG 0100H                ;Defines the start address for
                             ;object code as 0080:0100H
    INCLUDE PATCALLS.INC    ;Use the standard PAT system
                             ;labels
INT25V: EQU 0094H          ;Defines base address for
                             ;Interrupt 25H

;Disable interrupts:
    CLI                    ;Disable maskable interrupts

;Set up the Interrupt Vectors:
    MOV DX,DS              ;Saves the current Data Segment
    MOV AX,0000H          ;Data Segment points to
    MOV DS,AX             ;the bottom of memory
    MOV WORD PTR DS:INT25V,0300H ;Offset for Interrupt
                             ;Routine
    MOV WORD PTR DS:INT25V+2,0080H ;Segment for Interrupt
                             ;Routine
    MOV DS,DX             ;Restores the Data Segment

;Initialize the Timer:
    MOV AL,03H            ;Select 8086 mode and 1kHz
    OUT UCRREG1,AL        ;clock (1ms period)
    MOV AL,0FAH          ;Set 250ms delay
    OUT UTIMER1,AL
    MOV AL,01H            ;Enable interrupts from Timer 1
    OUT UIRQEN,AL

;Configure Port 2 as all outputs:
    MOV AL,03H            ;Make all of Port 2 outputs
    OUT UMODEREG,AL
    MOV BL,00H           ;Set initial count to 00H

;Enable interrupts:
    STI                    ;Enable maskable interrupts

;Dummy Main Program - Wait forever:
HERE: JMP HERE

```

Continued

Program continued:

```
; INTERRUPT SERVICE ROUTINE:
    ORG 0200H                ; Defines the start address for
                            ; object code as 0080:0200H

; Clear interrupt flag and re-load timer with delay value:
    IN  AL,UIRQADR           ; Clears interrupt flag
    MOV AL,0FAH             ; Re-load timer delay value
    OUT UTIMER1,AL
    MOV AL,01H              ; Re-enable interrupts from
    OUT UIRQEN,AL           ; Timer 1

; Increment the count and output new count:
    INC BL                  ; Adds 1 to the count
    MOV AL,BL
    OUT UPORT2,AL           ; Output new count

; Generate non-specific End of Interrupt command and return to main
; program:
    MOV AL,20H
    OUT 40H,AL
    IRET
```

Use this source program to generate an object code program and transfer to PAT for execution. Verify correct operation.

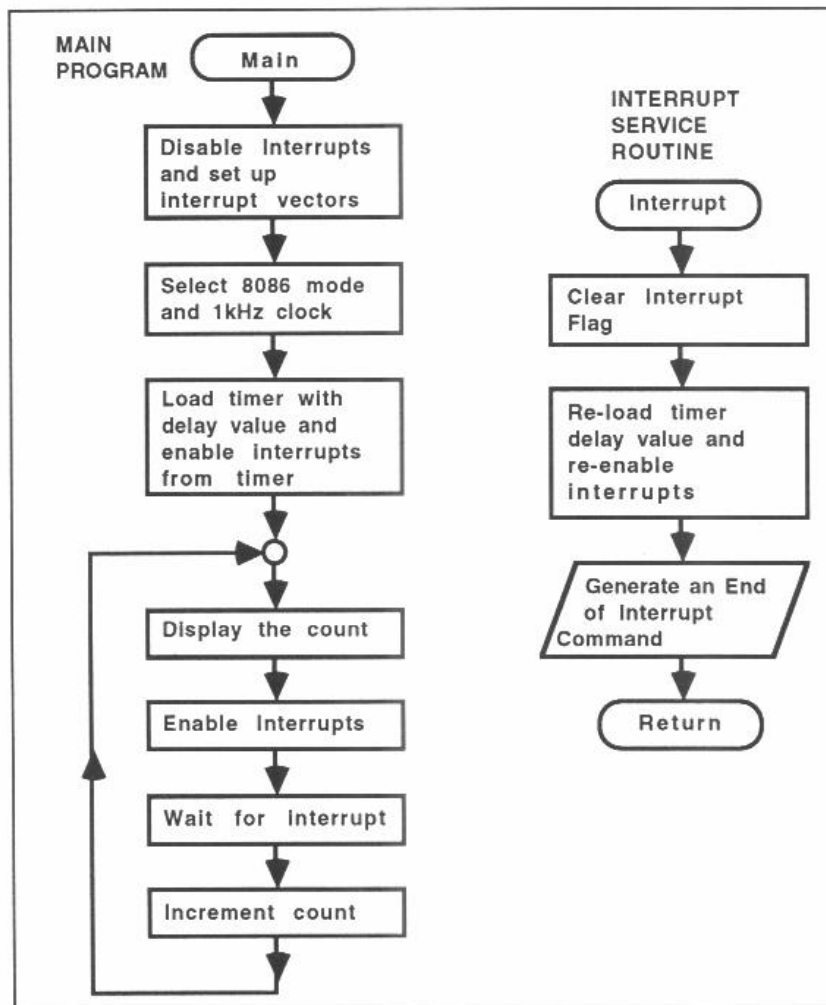
In this exercise the required delay of 250ms can be produced using a **single** Timer Register. Longer delays will require **two** Timer Registers to be **cascaded**. The next exercise shows how this may be achieved.

Exercise 32

Write a program which uses a timer to produce an increasing 2-digit hexadecimal count on the display. This count should increase once per second.

Solution:

Flowchart:



The source program will be:

```

;MAIN PROGRAM:
      ORG 0200H                ;Defines the start address for
                                ;object code as 0080:0200H
      INCLUDE PATCALLS.INC    ;Use the standard PAT system
                                ;labels
INT25V: EQU 0094H            ;Defines base address for
                                ;Interrupt 25H
;Disable interrupts:
      CLI                      ;Disable maskable interrupts
;Set up the Interrupt Vectors:
      MOV DX,DS                ;Saves the current Data Segment
      MOV AX,0000H            ;Data Segment points to
      MOV DS,AX                ;the bottom of memory
      MOV WORD PTR DS:INT25V,0300H ;Offset for Interrupt
                                ;Routine
      MOV WORD PTR DS:INT25V+2,0080H ;Segment for Interrupt
                                ;Routine
      MOV DS,DX                ;Restores the Data Segment
;Initialize the Timer:
      MOV AL,03H                ;Select 8086 mode and 1kHz
      OUT UCRREG1,AL           ;clock (1ms period)
      MOV AL,40H                ;Select Timers 2 and 4 cascaded
      OUT UMODEREG,AL
      CALL TIMINT              ;Load timer values for 1 second
                                ;delay and enable interrupts
                                ;from timer 24
;Display the current count:
      MOV AH,CLRSCR            ;Clear the display
      INT 28H
      MOV BL,00H                ;Set initial count to 00H
REPEAT:  PUSHA                 ;Save all General Purpose
                                ;Registers
      MOV AL,BL                ;Get current count
      MOV AH,WRBYTE            ;Display current count
      INT 28H

```

Continued

Program continued:

```

;Enable interrupts and wait for interrupt to occur:
    STI                                ;Enable maskable interrupts
    HLT                                ;Wait for an interrupt - delay
                                        ;of 1 second

;Add one to the count:
    MOV AH,CLRSCR                      ;Clear the display
    INT 28H
    POPA                               ;Restore General Purpose
                                        ;Registers
    INC BL                             ;Add 1 to count
    JMP REPEAT                         ;Jump back to display new count

; INTERRUPT SERVICE ROUTINE:
    ORG 0300H                          ;Defines the start address for
                                        ;object code as 0080:0300H

;Clear interrupt flag:
    IN AL,UIRQADR                      ;Clears interrupt flag

;Re-load timer with delay value and re-enable interrupts from timer 24:
    CALL TIMINT                        ;Load timer values for 1 second
                                        ;delay and enable interrupts
                                        ;from timer 24

;Generate non-specific End of Interrupt command and return to main
;program:
    MOV AL,20H
    OUT 40H,AL
    IRET

;SUBROUTINE TO LOAD TIMER VALUES AND ENABLE INTERRUPTS FROM TIMER 24:
TIMINT:  MOV AL,03H                    ;Loads Timer 4 with 03H - the
    OUT UTIMER4,AL                    ;High Byte of the count
    MOV AL,0E8H                       ;Loads Timer 2 with E8H - the
    OUT UTIMER2,AL                    ;Low Byte of the count
    MOV AL,40H                         ;Enables interrupts from Timer
    OUT UIRQEN,AL                     ;24
    RET                                ;Returns to main program

```

Use this source program to generate an object code program and transfer to PAT for execution. Verify correct operation.

A common requirement for a timer is to produce a 'real time clock'. This exercise produces such a clock.

Notice that here we have adopted a slightly different technique to the last exercise. The 'HLT' instruction has been used to effectively **halt** the processor until a valid interrupt occurs.

Practical Assignments

- 28 Write a program which uses a timer to sound the Piezo Sounder at 1kHz.
- 29 Write a program which allows motor speed to be controlled by the potentiometer position. A timer should be used to allow the motor speed in revolutions per second to be shown on the display.

Student Assessment 14

1. In an Interval Timer, a counter register is strobed by:.
 - a a Port
 - b an External Event
 - c the System Clock
 - d the Timer Control Register
2. If bit 4 of the 8256 MUART Mode Register is set then:
 - a Counter/Timer 2 will be a Counter
 - b Counter/Timer 2 will be a Timer
 - c Counter/Timer 3 will be a Counter
 - d Counter/Timer 3 will be a Timer
3. The 8256 MUART Timer frequency is programmed by:
 - a bit 0 of Command Register 1
 - b bit 1 of Command Register 1
 - c bit 0 of Command Register 2
 - d bit 1 of Command Register 2
4. The maximum delay which 8256 MUART Timer 1 can produce is:
 - a 25.5 ns
 - b 255 ns
 - c 25.5 ms
 - d 255 ms
5. The maximum delay which 8256 MUART Timer 2 and 4 cascaded can produce is:
 - a 65.5 s
 - b 655 s
 - c 6554 s
 - d 65535 s