

Section 8

Programming the Input/Output Ports

Chapter 8.1 Input and Output Programming

Objectives of this Chapter

Having studied this chapter you will be able to:

- Explain how data enters and leaves a microcomputer
- Program the 8256 MUART Parallel Data Ports
- Describe the 80286 Input and Output instructions
- Write programs which:
 - Configure the 8256 Parallel Data Ports as inputs, outputs or a mixture of both
 - Use the 80286 Input and Output instructions
 - Produce a delay of a given duration

Introduction

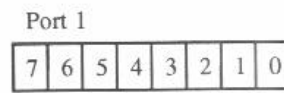
Data enters and leaves the microcomputer via **ports**. A port usually comprises 8 parallel connections to the external environment. These ports are often programmable so that the user can specify a combination of inputs or outputs. Ports may therefore be Output Ports, Input Ports or a mixture of both.

The PAT microcomputer uses a 8256 MUART (Multifunctional Universal Asynchronous Receiver Transmitter). This device has two 8-bit parallel ports called Port 1 and Port 2.

The 8256 device also includes counter/timer registers and serial communications.

The 8256 Ports can be configured under program control to provide a combination of inputs and outputs.

Every bit of Port 1 can be individually programmed as an input or an output bit:



Each bit can be programmed to be an Input or an Output

Unlike Port 1, Port 2 can only be programmed in terms of upper and lower **nibbles**. Thus the upper or lower 4 bits can be all input or all output but not a mixture of inputs and outputs:



Upper nibble can be programmed as all Inputs or all Outputs

Lower nibble can be programmed as all Inputs or all Outputs

Input and Output Instructions

The PAT microcomputer decodes Port 1 to Port Address 90H and Port 2 to Port Address 92H. The 80286 accesses these Ports by means of two special Input/Output instructions which address the Input/Output device directly:

IN

This instruction loads the Accumulator from an Input Port.

For example:

IN AL, 90H	;Loads the Accumulator from ;Input Port 1
------------	--

OUT

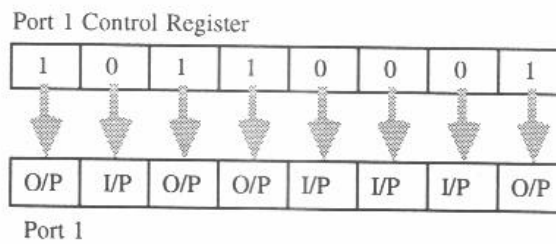
This instruction loads an Output Port from the Accumulator.

For example:

OUT 90H, AL	;Outputs the Accumulator from ;Output Port 1
-------------	---

Programming Input and Output Ports

Port 1 is programmed by means of the **Port 1 Control Register**. Each bit of this register which is at a logic "1" causes the **corresponding** bit of Port 1 to be an **output** bit:



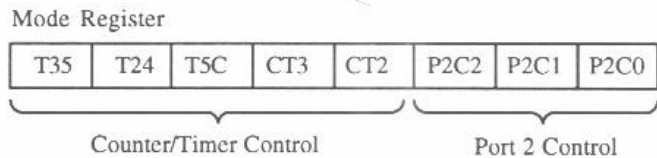
The Port 1 Control Register is decoded to Port Address 88H in the PAT microcomputer.

For example, to make bits 7,5,4 and 0 of Port 1 Outputs and bits 6, 3, 2 and 1 Inputs:

```

MOV  AL,B1H                ;Moves the mask B1H (101100012)
                                ;into the Accumulator
OUT  88H,AL                ;Outputs the mask to Port 1
                                ;Control Register
    
```

Individual bits cannot be programmed for Port 2. Only the upper and lower nibbles may be programmed as groups. This is achieved by writing a control byte to the 8256 Mode Register. This register controls both the Counter/Timers and Port 2:



When P2C2 is at logic "0", P2C1 and P2C0 program the direction of the upper and lower nibbles respectively thus:

P2C1	P2C0	Upper Nibble	Lower Nibble
0	0	Input Bit	Input Bit
0	1	Input Bit	Output Bit
1	0	Output Bit	Input Bit
1	1	Output Bit	Output Bit

For example, to make the upper nibble of Port 2 all outputs and the lower nibble all inputs:

```

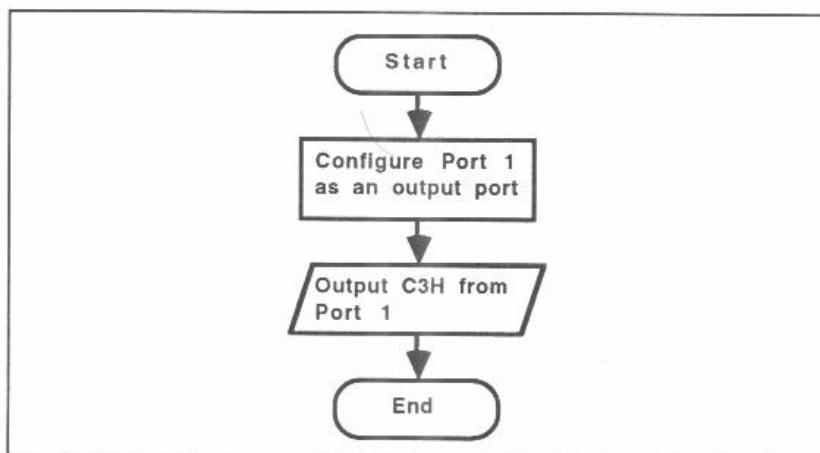
MOV  AL,02H                ;Moves the mask 02H (000000102)
                                ;into the Accumulator
OUT  86H,AL                ;Outputs the mask to Mode
                                ;Register
    
```

Exercise 17

Write a program which will output the value C3_H from Port 1.

Solution:

Flowchart:



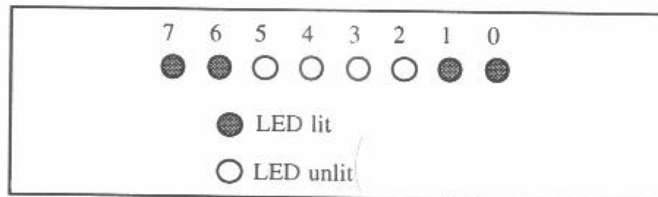
Source Program:

```

ORG 0100H                ;Defines the start address for
                        ;object code as 0080:0100H
UPORT1CTL EQU 088H      ;Defines address for Port 1
                        ;Control Register
UPORT1 EQU 090H         ;Defines address for Port 1
                        ;Data Register
MOV AL,0FFH
OUT UPORT1CTL,AL        ;Configures Port 1 as an output
                        ;port
MOV AL,0C3H
OUT UPORT1,AL           ;Outputs C3H from Port 1
MOV BX,0000H           ;Returns to PAT system
MOV AH,04H
INT 28H
  
```

Make sure that you have correctly connected the Applications Module to the PAT board and to the dc supply (see diagram on Page 2, Chapter 1.1).

Create the source program and generate the object program. Transfer the object program to PAT and execute. You should see the bit pattern for C3_H on the lower byte of the Applications Module Port Monitor thus:

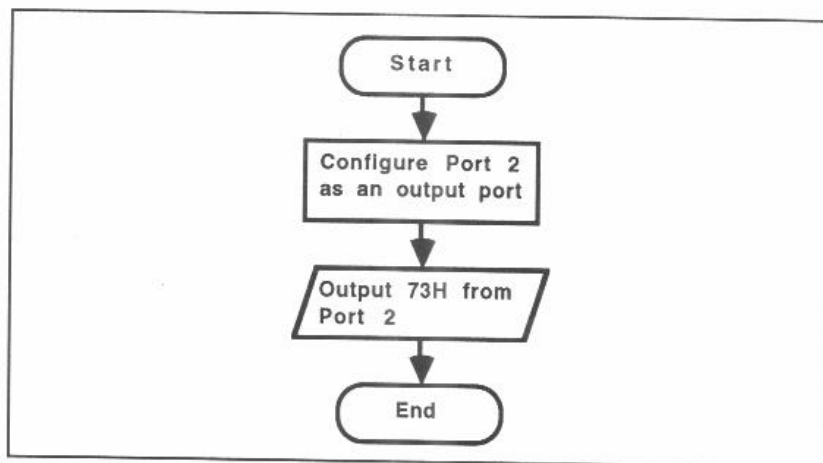


Exercise 18

Write a program which will output the value 73_H from Port 2.

Solution:

Flowchart:



Source Program:

```
ORG 0200H ;Defines the start address for
           ;object code as 0080:0200H
UMODEREG EQU 086H ;Defines address for Mode
           ;Register
UPOINT2 EQU 092H ;Defines address for Port 2
           ;Data Register
MOV AL,03H
OUT UMODEREG,AL ;Configures both nibbles of
                ;Port 2 as outputs
MOV AL,073H
OUT UPOINT2,AL ;Outputs 73H from Port 2
MOV BX,0000H ;Returns to PAT system
MOV AH,04H
INT 28H
```

Create this source program and generate the object program. Transfer the object program to PAT and execute. You should now see the bit pattern for 73H on the upper byte of the Applications Module Port Monitor.

Practical Assignment

- 14 Write a program which will add the bytes at locations 0080:1000H and 0080:1100H. The result should then be output from Port 1.

Exercise 19

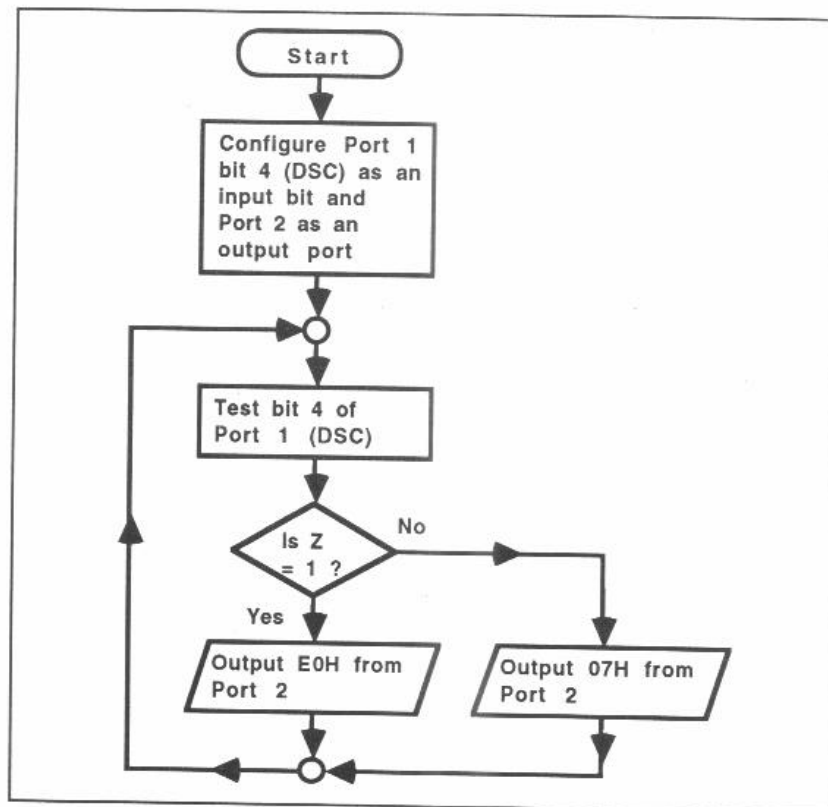
Write a program which will use the Applications Module motor disc detector as the input. If the input is a "1", output 07H from Port A. If the input is "0", output E0H from Port A.

Solution:

If you rotate the motor disc on the applications module, you will see the Port B monitor LED for bit 4 change. If the LED is lit, a "1" is present. If the LED is unlit, a "0" is present. This can be used as the input for this exercise.

This program should loop back to keep checking the input and change the output as required. This is the fundamental process of continuous microcomputer control.

Flowchart:



Notice that the program loops back. This will give a **continuous** loop. The output will change **whenever** the input changes.

Source Program:

```

                                ;Defines the start address for
                                ;object code as 0080:0200H
ORG 0400H

UMODEREG EQU 086H                ;Defines address for Mode
                                ;Register
UPORT1CTL EQU 088H                ;Defines address for Port 1
                                ;Control Register
UPORT1 EQU 090H                  ;Defines address for Port 1
                                ;Data Register
UPORT2 EQU 092H                  ;Defines address for Port 2
                                ;Data Register

MOV AL,00H
OUT UPORT1CTL,AL                 ;Configures all of Port 1 as
                                ;inputs

MOV AL,03H
OUT UMODEREG,AL                 ;Configures both nibbles of
                                ;Port 2 as outputs

CHECK: IN AL,UPORT1              ;Read Port 1 into Accumulator
TEST AL,10H                     ;Test bit 4 of Accumulator
JNZ OUTLOW                       ;If bit 4 is set, DSC is at
                                ;logic "1" so Jump to label
                                ;"OUTLOW" to output 07H

OUTHIGH: MOV AL,0E0H             ;Bit 4 is clear so output the
OUT UPORT2,AL                    ;marker value E0H from Port 2
JMP CHECK                       ;Jump back to test Bit 4 (DSC)
                                ;again.

OUTLOW: MOV AL,07H              ;Outputs the marker value 07H
OUT UPORT2,AL                    ;from Port 2
JMP CHECK                       ;Jump back to test Bit 4 (DSC)
                                ;again.

```

Use Merlin to assemble the object program from this source program. Transfer the object program to PAT and execute.

Rotate the motor disc and the input will switch between "1" and "0" (LED "On" and "Off"). Check that the output LEDs change between 07H and E0H as the input changes.

Time Delays

Often in Input/Output programs it will be necessary to provide a time delay. For example: to allow a peripheral device time to respond.

There are a number of ways of producing such delays. The simplest is to load a register with a value and then continually decrement the register until it reaches zero thus:

	MOV	CX,VALUE		;Loads the CX Register with a
				;value
WAIT:	LOOP	WAIT		;Decrements the CX Register and
				;Jumps back to decrement it
				;again until CX=0000H

The delay produced will depend upon the variable "value". It will also depend upon the time each instruction within the loop takes to execute. These times are given in the 80286 Programmer's Reference Manual as "clocks".

Refer to page B-70. This shows the number of clocks required for every type of MOVE instruction. Notice that to Move an Immediate word value onto a Register requires **2 clocks**. Now turn to page B-67. This shows the clock details for the LOOP instruction. You will find that a LOOP requires **8 clocks**.

Therefore each pass through the loop will take $2 + 8 = 10$ clocks.

Now, the clock period is related to the clock frequency thus:

$$\text{Cycle Time} = \frac{1}{\text{Clock Frequency}}$$

The PAT microcomputer has a 5MHz system clock, giving a time for each clock of 0.2µs or 200ns.

Recall that the simple delay loop program takes 10 clocks for each pass. This represents a time delay of $10 \times 200\text{ns} = 2000\text{ns}$ (or 2µs).

Now, the maximum size for "value" is FFFF_H, which is 65536₁₀. Therefore the maximum delay will be:

$$65536 \times 2\mu\text{s} = 131.07\text{ms} (= 0.13107\text{s})$$

Longer delays can be produced by adding a second loop which is **nested** with the first:

```

MOV  BX,07H           ;Loads the BX Register with the
                       ;value 07H
MOV  CX,FFFFH        ;Loads the CX Register with the
                       ;value FFFFH
WAIT: LOOP WAIT       ;Decrements the CX Register and
                       ;Jumps back to decrement it
                       ;again until CX=0000H
DEC  BX              ;Decrements the BX Register
JNZ  WAIT            ;Jumps back to decrement BX
                       ;Register again until BX=0000H

```

This will repeat a delay of 0.13107s **seven** times, giving a total delay of $7 \times 0.13107\text{s} = 0.91749\text{s}$. The action of this delay technique can be likened to a clock: The CX Register represents seconds and the BX Register minutes. The maximum delay which may be produced will be FFFF_H (65535₁₀) times 0.13107s. This would give a delay of 8589.7s (2 hours and 23 minutes).

Delays which are produced by these type of time-wasting program loops are called **software delays**. Other techniques for producing delays will be discussed in a subsequent chapter.

Exercise 20

Write a program section which gives a delay of 140μs

Solution:

$$\frac{140\mu\text{s}}{2\mu\text{s}} = 70_{10}$$

So 70₁₀ is the value to be loaded into the CX Register. The source program section will be:

```

MOV  CX,046H         ;Loads the CX Register with the
                       ;value 46H (7010)
WAIT: LOOP WAIT       ;Decrements the CX Register and
                       ;Jumps back to decrement it
                       ;again until CX=0000H

```

Software Delays Using the PAT

Any software delay loops which involve the accessing the MUART (eg Output a given byte for a specified time period) will actually give rather **longer** delays than those calculated. This is due to several factors involving the timing of data transfers within the PAT microcomputer system. However, as a general rule, actual software delays are about **1.5 times the calculated value**. Further details can be found in the PAT 80286 Microcomputer Technical Manual.

Practical Assignments

- 15 Write a program which will output an increasing binary count at Port 1.
[HINT: You will need to include a delay after each increment, otherwise the count will be too fast to see.]
- 16 Write a program which will output an increasing binary count at Port 2. The count should increase by one about every 1.0 second. The Applications Module motor disc detector is to be used as an input. If the input is a "0", the binary count may continue. If the input is "1", the binary count should be suspended.

Student Assessment 10

1. Data enters and leaves the microcomputer via a:
 - a Mode Register
 - b Port Control Register
 - c Data Direction Register
 - d Port Register
2. The 80286 assembly language mnemonics for "Read the byte input at Port 2 into the Accumulator" are:
 - a IN AL, UPORT2
 - b IN UPORT2, AL
 - c OUT AL, UPORT2
 - d OUT UPORT2, AL
3. The 80286 assembly language instructions required to make Port 1 an output port are:

<input type="checkbox"/> a MOV AL, 00H OUT UMODEREG, AL	<input type="checkbox"/> c MOV AL, 0FFH OUT UPORT1CTL, AL
<input type="checkbox"/> b MOV AL, 00H OUT UPORT1CTL, AL	<input type="checkbox"/> d MOV AL, 0FFH OUT UMODEREG, AL
4. The 80286 assembly language instructions required to make Port 2 an input port are:

<input type="checkbox"/> a MOV AL, 00H OUT UMODEREG, AL	<input type="checkbox"/> c MOV AL, 0FFH OUT UPORT1CTL, AL
<input type="checkbox"/> b MOV AL, 00H OUT UPORT1CTL, AL	<input type="checkbox"/> d MOV AL, 0FFH OUT UMODEREG, AL
5. A simple delay of 120ms using the LOOP instruction will require the CX Register to be :
 - a 0940H
 - b 0EA6H
 - c 940AH
 - d EA60H