

Section 10

Interrupts

Chapter 10.1 80286 Interrupts

Objectives of this Chapter

Having studied this chapter you will be able to:

- Describe the general response of a microprocessor to interrupt
- Understand the basic mechanisms of Polled Input/Output and Interrupt Input/Output
- Explain interrupt priority and describe the 80286 interrupt priorities
- Describe the two 80286 operating modes
- Describe interrupt masking and how 80286 interrupts can be masked
- Explain the need for a Return From Interrupt instruction
- Describe the following PAT Interrupt Responses:
 - Non-Maskable Interrupt
 - Maskable Interrupt
 - Cold Reset
 - Monitor Reset
 - Warm Reset

Introduction

An interrupt is a special **input** to a microprocessor which is examined as part of **every** instruction which the microprocessor executes. When an active transition occurs on this input, the current program is **suspended**.

The microprocessor will then start to execute an **interrupt service subroutine**. At the end of this routine the original program is usually resumed, from the point at which it was suspended.

The use of interrupts allows the microcomputer to respond quickly to external events.

Polling and Interrupts

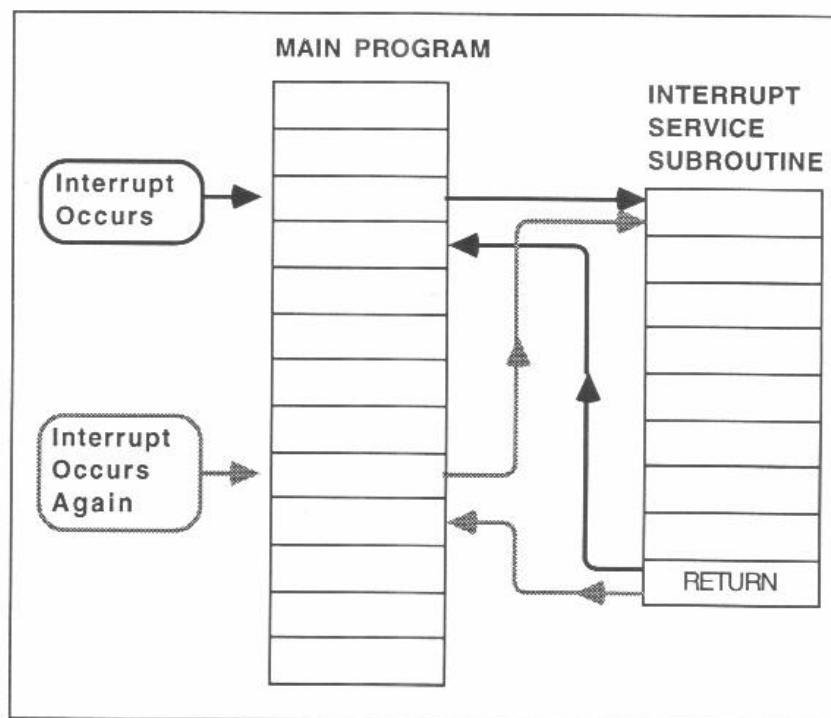
Many peripheral devices operate at a very much slower speed than the microprocessor. Consequently it will often be necessary for the microprocessor to wait while the peripheral responds. There are two basic techniques to achieve this synchronization:

Polled Input/Output

The microprocessor periodically checks the peripheral to see if it is ready for data transfer. This gives variable response times and also wastes microprocessor time in needless checking.

Interrupt Input/Output

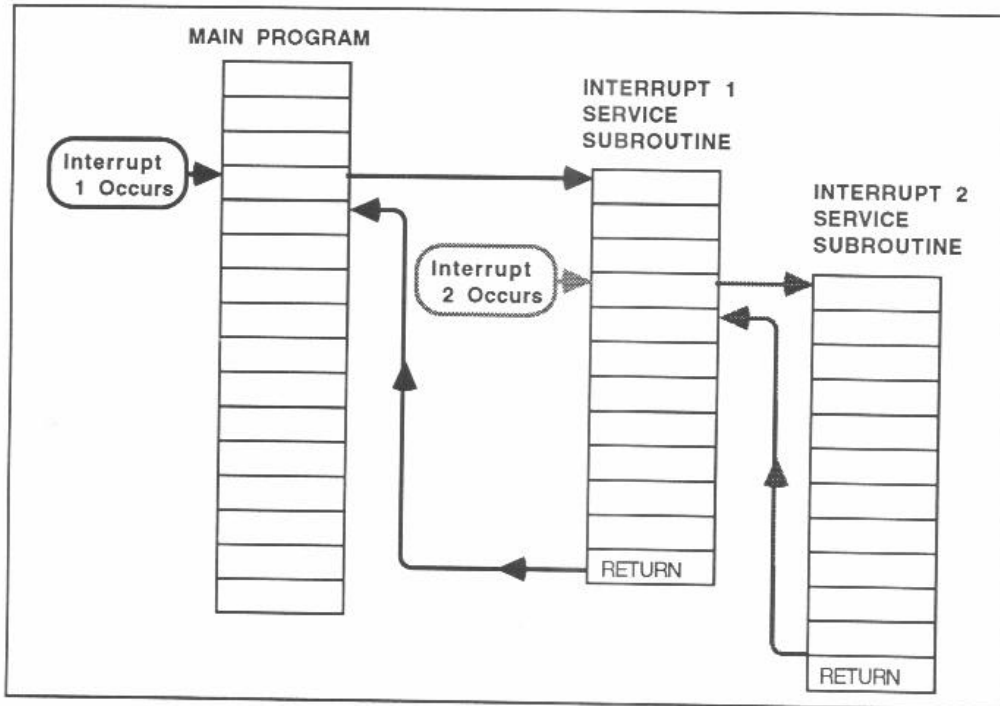
The peripheral signals it is ready for data transfer by **interrupting** the microprocessor. This has the advantage that the microprocessor does not waste time interrogating the peripheral over and over again. The microprocessor can actually be executing **another** program which is suspended when data transfer is required (sometimes called a background program).



Interrupts allow **external** events to cause a specific subroutine to be executed. So, an interrupt service subroutine can occur at **any** time during the execution of a program, unlike a normal subroutine which may only occur at a fixed position within a program sequence.

Since an interrupt may occur at **any** time within a program, the return address must be saved on the stack.

It is possible for microcomputer systems to have **multiple** interrupts, so **nesting** of interrupts may occur, where a second interrupt occurs while an interrupt service subroutine is already in progress:



The first return address is saved on the stack and then the second. Since the stack has a LIFO action, each address will be restored as it is required (ie second return address **then** first). The 80286 always saves the Instruction Pointer, Flag Register and CX Register automatically whenever an interrupt occurs.

An interrupt service subroutine may use registers which the main program uses so it is good practice for interrupt service subroutines to save any registers they use, generally by means of the PUSHA instruction.

The microprocessor will always complete the instruction in progress when an interrupt occurs before beginning the interrupt response sequence. When a particular interrupt occurs, the corresponding **interrupt vector** is loaded into the Instruction Pointer. This vector defines the start of the appropriate interrupt service routine. In the PAT, the first 1K of RAM is reserved for the storage of these vectors.

Interrupts may be generated by **external** hardware or by the execution of an INT instruction (**internal** interrupt). Some type of error condition (eg divide by zero, invalid opcode, etc) cause a special type of interrupt to occur. This is called an **Exception**.

Interrupt Masking

Certain interrupts can be "disabled" so that the microprocessor will **not** respond when they occur. Interrupts which can be ignored in this way are called **maskable interrupts**.

Maskable interrupts will be ignored if the Interrupt Enable Flag (bit 9 of the Flag Register) is **clear**. There are two 80286 instructions which are used to set or clear this flag respectively:

Set Interrupt Enable Flag (STI)

This instruction **allows** maskable interrupts to be acknowledged.

Clear Interrupt Enable Flag (CLI)

This instruction **prevents** maskable interrupts being acknowledged.

The 80286 has one maskable interrupt (INTR) and one non-maskable interrupt (NMI).

Return from Interrupt

Clearly, it will be necessary to terminate an interrupt service subroutine with a RETURN instruction, to restore the Instruction Pointer, Flags and CX Register from the stack. This allows the interrupted program to continue from the point at which it was interrupted.

The 80286 provides a **Return from Interrupt** instruction (IRET) which will return program control to the interrupted program.

Interrupt Priority

Some interrupts are more important than others. In the 80286 NMI has a higher priority than INTR. Consequently, NMI can interrupt INTR but **not** vice versa.

Non-Maskable Interrupt (NMI) Response

A positive edge (logic '0' to logic '1' transition) on the NMI pin will cause the sequence of events shown below:

- 1 The instruction in progress is completed
- 2 Other interrupts are disabled.
- 3 The Instruction Pointer, Flag Register and Code Segment Register are pushed on the stack.
- 4 Interrupt vector 2 is fetched (from locations 00008H to 0000BH.)
- 5 Program execution continues from the start of the interrupt service subroutine.
- 6 When an IRET instruction occurs, the Instruction Pointer, Flag Register and Code Segment Register are popped from the stack.
- 7 Execution of the Main Program continues from the point at which it was interrupted.

Notes:

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

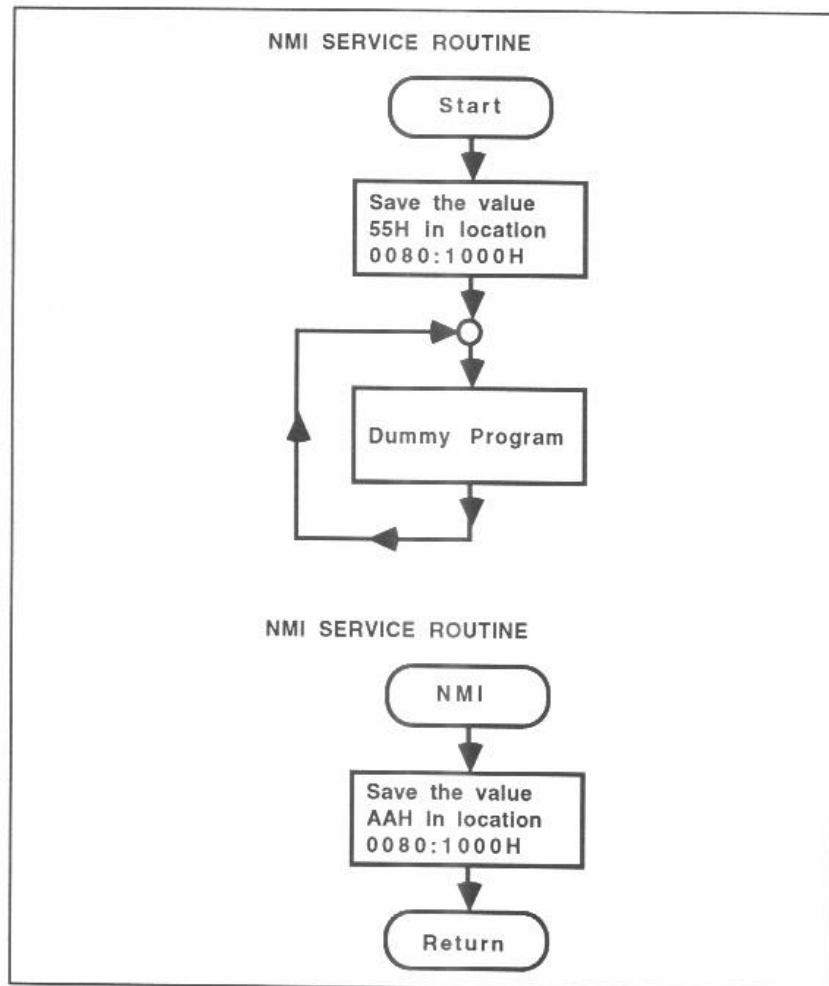
Exercise 29

Write a program which will place the value 55H in location 0080:1000H. If a Non-Maskable Interrupt occurs, this location should be changed to AAH.

Solution:

This will require two very simple programs:

Flowchart:



Source Program:

```

        ORG 0100H                ;Defines the start address for
;Main Program:                  ;Main Program as 0080:0100H
        MOV BYTE PTR DS:1000H,055H ;Moves the value 55H into
;                               ;location 0080:1000H
HERE:   JMP HERE                ;Wait forever - Dummy Main
;                               ;Program

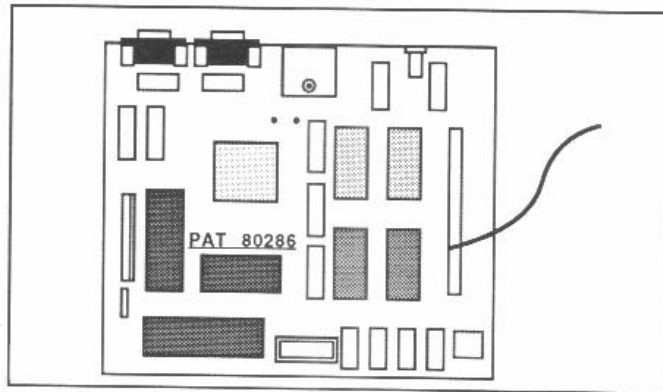
;NMI Service Routine:
        ORG 0200H                ;Defines the start address for
;                               ;NMI Interrupt Service Routine
;                               ;as 0080:0200H
        MOV BYTE PTR DS:1000H,0AAH ;Moves the value AAH into
;                               ;location 0080:1000H
        IRET                    ;Returns to Main Program
    
```

Use Merlin to produce source and object programs. Transfer the program to the PAT but do **not** run. Before you can run this program, it will be necessary to load the NMI Interrupt Vectors. Use the Terminal "C" command to change the contents of the NMI Vectors thus:

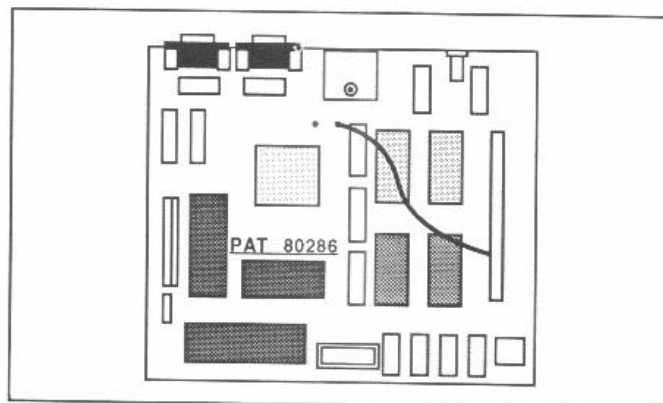
0000:0008	00	} Offset	} Start Address for NMI Interrupt Service Routine
0000:0009	02		
0000:000A	80	} Segment	
0000:000B	00		

Having entered the vectors, the program can be run. Press Reset and examine the contents of location 0080:1000H. You will find the value 55H since no NMI has occurred.

Now connect one of the flying leads provided with this curriculum manual to the NMI pin on the right hand side of the PAT board:



Run the program again but this time cause an NMI interrupt by briefly touching the free end of the flying lead on the 5V test point near the top of the PAT board thus:



Press Reset and examine the contents of location 0080:1000H. You will now find the value AA11. This shows that a NMI has occurred and the NMI Service Routine has been executed.

The program could be modified to load the NMI Vectors. This would remove the need for manual setting:

```

                ORG 0100H                ;Defines the start address for
                ;Main Program as 0080:0100H
NMIVCT EQU 0008H                ;Defines the start of the NMI
                ;Vectors

;Main Program:
    MOV BYTE PTR DS:1000H,055H        ;Moves the value 55H into
                ;location 0080:1000H
    MOV DX,DS                        ;Saves the Data Segment in the
                ;DX Register

;Set NMI Interrupt Vectors:
    MOV AX,0000H                    ;Sets the Data Segment to
                ;0000H
    MOV DS,AX
    MOV WORD PTR DS:NMIVCT,0200H      ;Stores the offset for
                ;the NMI Service
                ;Routine
    MOV WORD PTR DS:NMIVCT+2,00080H    ;Stores the
                ;segment for the
                ;NMI Service
                ;Routine

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

;NMI Service Routine:
    ORG 0200H                        ;Defines the start address for
                ;NMI Interrupt Service Routine
                ;as 0080:0200H
    MOV DS,DX                        ;Restores the Data Segment from
                ;the DX Register
    MOV BYTE PTR DS:1000H,0AAH        ;Moves the value AAH into
                ;location 0080:1000H
    IRET                              ;Returns to Main Program

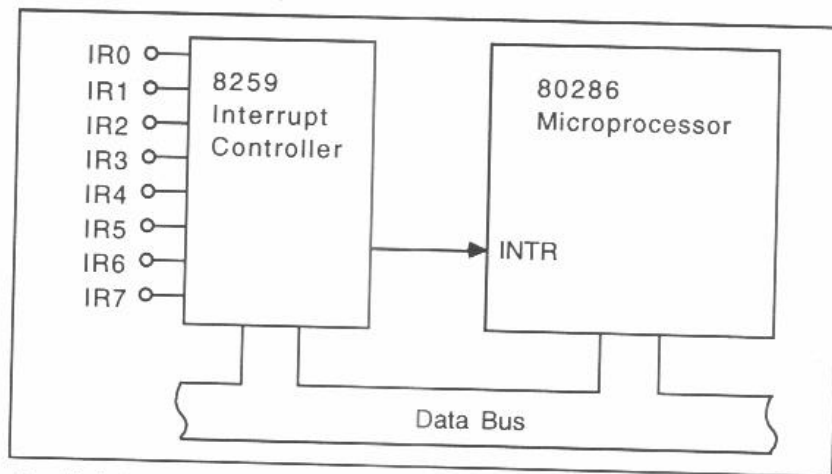
```

Modify your source program and create a new object program. Transfer to PAT and run. Verify that the marker value in location 0080:1000H changes between 55H and AAH (depending upon whether an interrupt has occurred).

Maskable Interrupt (INTR) Response

Recall that a Maskable Interrupt will only be acknowledged if the Interrupt Enable Flag (bit 9 of the Flag Register) is **set**.

In the PAT Microcomputer, a 8259 Interrupt Controller is connected to the INTR input of the 80286. This device extends the number of Maskable Interrupts available to eight.



Provided the Interrupt Enable Flag is set, a positive edge (logic '0' to logic '1' transition) on **any** of the 8259 inputs (IR0 to IR7) will cause the 80286 INTR input to be activated. The response will be as follows:

- 1 The instruction in progress is completed
- 2 Other interrupts are disabled.
- 3 The Instruction Pointer, Flag Register and Code Segment Register are pushed on the stack.
- 4 Interrupt Acknowledge sequence is generated, fetching the interrupt vector number from the Data Bus.
- 5 Interrupt vector is fetched.
- 6 Program execution continues from the start of the interrupt service subroutine. This should include instructions to clear the interrupt (since the 8259 will 'remember' that a particular interrupt is being processed).
- 7 When an IRET instruction occurs, the Instruction Pointer, Flag Register and Code Segment Register are popped from the stack.
- 8 Execution of the Main Program continues from the point at which it was interrupted.

The 8259 Interrupt Controller inputs are **prioritised**; IR0 being the highest and IR7 the lowest. Two of these inputs (IR1 and IR2) are inverted on the PAT board. This allows interrupts to be caused by both logic '1' and '0'. The inputs are assigned as follows:

8259 Input	Vector Number	Use
IR0	20H	Available to User, Active High
IR1	21H	Available to User, Active Low
IR2	22H	Available to User, Active Low
IR3	23H	Available to User, Active High
IR4	24H	Available to User, Active High
IR5	25H	Used by User MUART
IR6	26H	Used by System MUART
IR7	27H	Available to User, Active High

The Interrupt Vector can be found by simply multiplying the vector number by 4H:

8259 Input	Interrupt Vector	Priority
IR0	0080H	Highest
IR1	0084H	
IR2	0088H	
IR3	008CH	
IR4	0090H	
IR5	0094H	
IR6	0098H	
IR7	009CH	Lowest

The 80286 must issue an End of Interrupt command at the end of the interrupt service routine. This can be achieved by writing the value 20H to the first address in the Interrupt Controller (40H) thus:

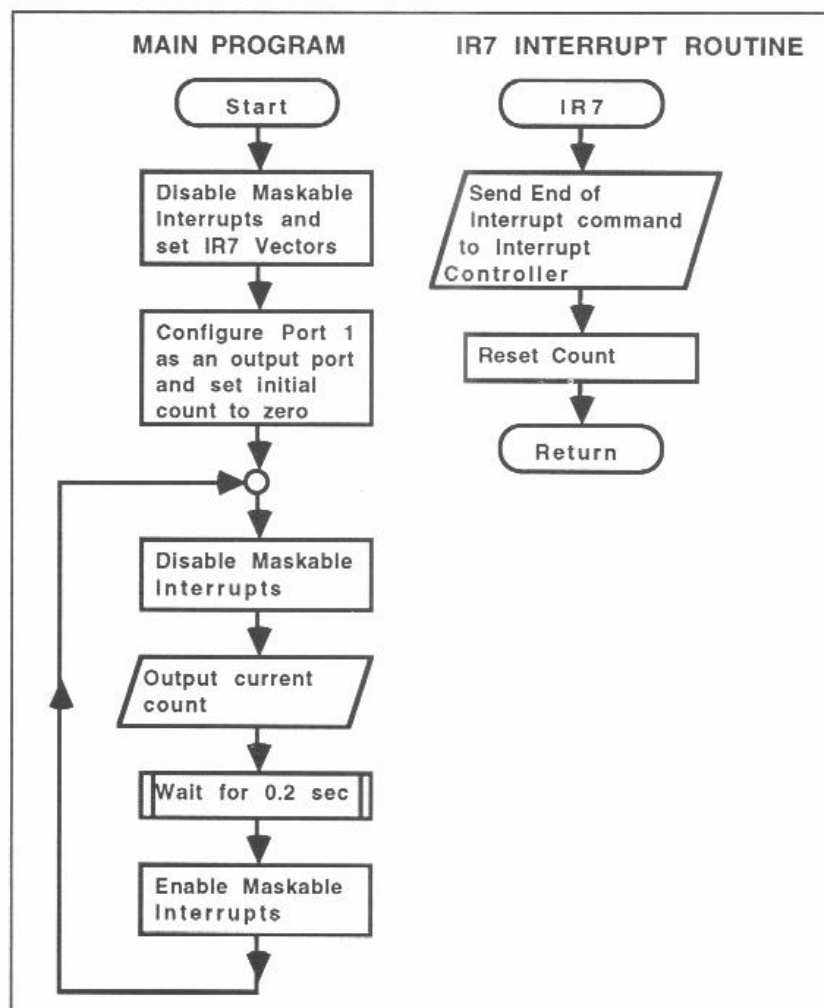
```
MOV AL, 20H  
OUT 40H, AL
```

Exercise 30

Write a program which will increment Port 1 about every 0.2 sec. If a IR7 occurs the count should be reset to zero.

Solution:

Flowcharts:



Source Program:

```

                INCLUDE PATCALLS.INC                ;Use standard Port Labels
I27VCT:        EQU 009CH                          ;Defines vector for IR7
;Main Program:
                ORG 0500H                          ;Defines the start address for
                ;the Main Program as 0080:0500H
                CLI                                ;Disable Maskable Interrupts
                MOV DX,DS                          ;Saves Data Segment in DX
                ;Register
;Set Interrupt Vectors:
                MOV AX,0000H                       ;Sets Data Segment to 0000H
                MOV DS,AX
                MOV WORD PTR DS:I27VCT,0600H      ;Stores the offset for
                ;the IR7 interrupt routine
                MOV WORD PTR DS:I27VCT+2,0080H    ;Stores the segment for
                ;the IR7 interrupt
                ;routine
;Output Increasing Count at Port 1:
                MOV DS,DX                          ;Restores Data Segment Register
                MOV AL,0FFH                        ;Configure Port 1 as all
                OUT UPORT1CTL,AL                  ;outputs
                MOV AL,00H                        ;Set initial count to zero
OUTPUT:        CLI                                ;Disable Maskable Interrupts
                OUT UPORT1,AL                     ;Output current count
                MOV CX,0FFFFH                     ;Delay of about 0.2 sec
DELY:          LOOP DELY
                INC AL                             ;Adds one to the count
                STI                                ;Enable Maskable Interrupts
                JMP OUTPUT                         ;Jump back to label "OUTPUT"
;IR7 Service Routine:
                ORG 0600H
                MOV AL,20H                        ;Output End of Interrupt to
                OUT 40H,AL                        ;Interrupt Controller
                MOV AL,00H                        ;Reset count to zero
                IRET                              ;Return to main program

```

Create this program and run. Verify that an incrementing count is produced at Port 1. Now briefly connect IR7 to +5V. You should find that the count resets to zero.

Reset

This interrupt input has priority over all others. The reset response is initiated by holding the Reset input at logic high (1) for at least 16 clock cycles and then applying a logic low (0).

In the Reset process, the 80286 will initialize various registers and then begin execution from location F000:FFF0H.

Disassemble the first instruction in the PAT monitor by using the "D" command. You should find the following:

```
PAT: D F000:FFF0;1
F000:FFF0 E9D00      JMP 0000
```

So the first instruction is a JUMP. This instruction causes the 80286 to Jump to location F000:0000H.

Disassemble from this location and you will be able to see the first few instructions of the PAT Monitor Program:

```
PAT: D F000:0000
F000:0000 FA          CLI
F000:0000 B4D5        MOV AH,D5
F000:0000 9E          SAHF
F000:0000 7340        JNB 0046
F000:0000 753E        JNE 0046
F000:0000 7B3C        JNP 0046
F000:0000 793A        JNS 0046
F000:0000 9F          LAHF
```

You can verify this by running the program from location F000:0000H thus:

```
G F000:0000
```

You will now see the "PAT" prompt, indicating that the monitor is running.

The PAT Monitor allows three types of reset:

Cold Reset

This takes place when power is applied to PAT board. This type of reset will set variables to their start-up values and set all RAM locations to 00_H.

Monitor Restart

This type of restart happens when the reset switch is pressed **twice** (with a delay of about 0.5 seconds). A Monitor Restart will set variables to their start up values but **not** clear the contents of RAM.

Warm Reset

Whenever the reset switch is pressed once, a Warm Restart occurs. This type of reset will not change any variables or programs.

A flowchart detailing the restart sequence is shown on the next page.

Notes:

.....

.....

.....

.....

.....

.....

.....

.....

.....

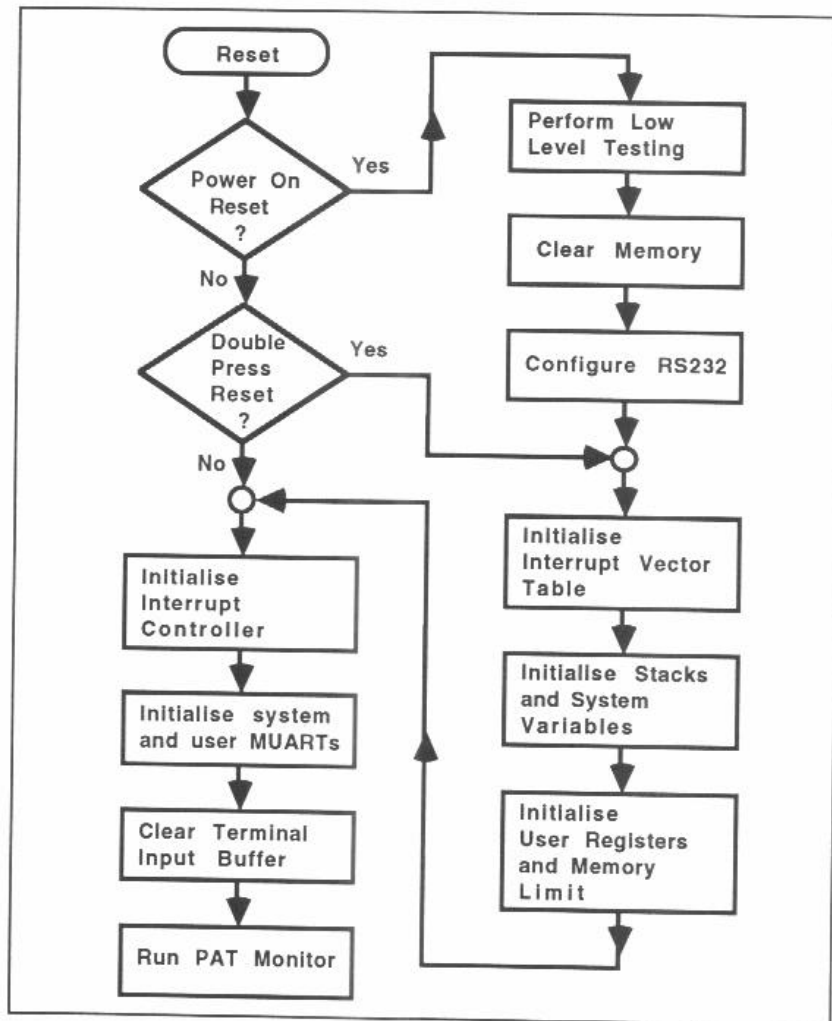
.....

.....

.....

.....

.....



The PAT user manual details the states of variables following each type of restart.

Student Assessment 13

1. Interrupt inputs which the processor may ignore are called:
 - a Maskable
 - b Non-maskable
 - c Low Priority
 - d Nested
2. The process of a microprocessor periodically checking a peripheral to see if it is ready for data transfer is called:
 - a Nested Input/Output
 - b Interrupt Input/Output
 - c Stack Input/Output
 - d Polled Input/Output
3. The 80286 instruction which allows maskable interrupts to be acknowledged is:
 - a CLI
 - b INT
 - c SEI
 - d STI
4. The vector for an IR5 interrupt is at:
 - a 0090H
 - b 0094H
 - c 0095H
 - d 009CH
5. The highest priority interrupt on the PAT microcomputer board is:
 - a IR0
 - b IR7
 - c NMI
 - d Reset