

**Faculty of Information Technology**

**Computer Systems Engineering Department**

**COMPUTER DESIGN LAB #ENCS411**

**EXP #5**

 Programmable Interrupt Controller Application

**Name : Mohammad Modallal**

**ID : 1120174**

**Section# : 3**

**Dr.** [**Khader Mohammad**](https://ritaj.birzeit.edu/student/view_office_hr?employee_id=5442)

**Date:30/3/2015**

1. Contents

[1 Abstract 3](#_Toc387987979)

[2 Introduction to Interrupt 4](#_Toc387987976)

 [2.1 Defnition of Interrupt 4](#_Toc387987981)

 [2.2 Type of Interrupt 4](#_Toc387987982)

 [2.3 Interupt Vector 4](#_Toc387987983)

 [2.4 The Operation of a Real Mode Interrupt 6](#_Toc387987984)

 [2.5 Initialization command words 7](#_Toc387987984)

 [2.6 operation command words 8](#_Toc387987984)

[3 Procedure 9](#_Toc387987989)

[3.1 User Defined Software Interrupts 9](#_Toc387987990)

[3.2 8259a Interrupt Control (Polling Technique) 10](#_Toc387987991)

[3.3 8259a Interrupt Control (Interruption Technique) 13](#_Toc387987991)

[3.4 8259a Interrupt Control (Interruption Technique) (C Code) 15](#_Toc387987991)

[4 Conclusion 16](#_Toc387987992)

[5 References 17](#_Toc387987992)

**1.Abstract**

 The aim of this experiment is to understand the 8086 Interrupt capabilities using Programmable Interrupt Controller [PIC 8259] that includes reviewing 8259 PIC control, its initialization[ICW] and operational modes[OCW].

**2.Introduction**

**2.1 Definition**

When a Process is executed by the [CPU](http://ecomputernotes.com/fundamental/introduction-to-computer/what-is-cpu) and when a user Request for another Process then this will create disturbance for the Running Process. This is also called as the **Interrupt**[[1]](http://en.wikipedia.org/wiki/Diode%22%20%5Cl%20%22cite_note-5).

**2.2 Types of Interrupts**

Generally there are three types o Interrupts :

 1)   Internal Interrupt

2)   Software Interrupt.

3)   External Interrupt.

**2.3 Interrupt Vectors**

The interrupt vectors and vector table are crucial to an understanding of

hardware and software interrupts. The **interrupt vector table** is located in the first 1024 bytes of memory at addresses 000000H–0003FFH. It contains 256 different four-byte interrupt vectors.An interrupt vector contains the address (segment and offset) of the interrupt service procedure[[2]](http://en.wikipedia.org/wiki/Diode%22%20%5Cl%20%22cite_note-5).

Figure 2–1 illustrates the interrupt vector table for the microprocessor. The first five interrupt vectors are identical in all Intel microprocessor family members, from the 8086 to the Pentium. Other interrupt vectors exist for the 80286 that are upward-compatible to the 80386, 80486, and Pentium–Core2, but not downward-compatible to the 8086 or 8088. Intel reserves the first 32 interrupt vectors for their use in various microprocessor family members. The last 224 vectors are available as user interrupt vectors. Each vector is four bytes long in the real mode and contains the starting address of the interrupt service procedure. The first two bytes of the vector contain the offset address IP and the last two bytes contain the segment address CS[[2]](http://en.wikipedia.org/wiki/Diode%22%20%5Cl%20%22cite_note-5).

Each interrupt has physical address 20 bits wide and is computed from the 16-bit contents of IP and CS. For finding an interrupt address vector, the IP and CS position must be found. For example, for the interrupt type n :

The table address for IP=4×n  **and** The table address for CS=4×n+2.



**2.4 The Operation of a Real Mode Interrupt**

When the microprocessor completes executing the current instruction, it determines whether an interrupt is active by checking :

(1) instruction executions

(2) single-step

(3) NMI

(4) coprocessor segment overrun

(5) INTR

(6) INT instructions

If one or more of these interrupt conditions are present, the following sequence of events occurs[[2]](http://en.wikipedia.org/wiki/Diode%22%20%5Cl%20%22cite_note-5):

**1.** The contents of the flag register are pushed onto the stack.

**2.** Both the interrupt (IF) and trap (TF) flags are cleared. This disables the INTR pin and the trap or single-step feature.

**3.** The contents of the code segment register (CS) are pushed onto the stack.

**4**. The contents of the instruction pointer (IP) are pushed onto the stack.

**5.** The interrupt vector contents are fetched, and then placed into both IP and CS so that the next instruction executes at the interrupt service procedure addressed by the vector.

**2.5 Initialization Command Words**

There are four initialization command words (ICWs) for the 8259A that are selected when the A0 pin is a logic 1. When the 8259A is first powered up, it must be sent ICW1, ICW2, and ICW4. If the 8259A is programmed in cascade mode by ICW1, then we also must program ICW3. So if a single 8259A is used in a system, ICW1, ICW2, and ICW4 must be programmed. If cascade mode is used in a system, then all four ICWs must be programmed[[2]](http://en.wikipedia.org/wiki/Diode%22%20%5Cl%20%22cite_note-5).

Figure 2–2 lists the binary bit patterns for all four initialization command words of the 8259A.



**2.6 Operation Command Words**

The operation command words (OCWs) are used to direct the operation of the 8259A once it is programmed with the ICW[[2]](http://en.wikipedia.org/wiki/Diode%22%20%5Cl%20%22cite_note-5). The OCWs are selected when the A0 pin is at a logic 0 level, except for OCW1, which is selected when A0 is a logic 1. Figure 2–3 lists the binary bit patterns for all three operation command words of the 8259A.



**3.Procedure**

**3.1 User Defined Software Interrupts**

**Setp1:** I write the following code and save it to an Assembly file(.asm).

The following code make **INT21h** work as addition operation wich add the content of BX to AX and store the result in AX.This done by store the address of **add** instruction in the Interrupt vector table(IP=OFFSET INT\_SER , CS=0) .

****

**Step2:** I Compile and build this ASM file and execute it on MDA-8086 kit.

The content of AX after executing the code is AX=AX+BX=1234+6789=79BD

**3.2 8259a Interrupt Control (Polling Technique)**

**Srep1:**I initialize command words using this information

1. ICW4 is needed 🡺[ICW1]

2. Edge triggered mode 🡺[ICW1]

3. An address interval of 8 🡺[ICW1]

4. Single mode 🡺[ICW1]

5. Interrupt vector of 40H 🡺[ICW2]

6. Normal end of interrupt 🡺[ICW4]

7. Non-buffered mode 🡺[ICW4]

8. Not SFNM🡺[ICW4]

**The I/O ports for the 8259 if direct addressing mode is used with only 8086 A4 being “1” and 8086 A1 being connected to A0 of 8259:**

Address of I/O ports are **10H(start address)** and **12H(end address)**.



By using figure 2\_2 the ICW`s are:







By using figure 2­­­\_3 the OCW`s are:

****

**Setp2:** I write the following code and save it to an Assembly file(.asm).I Compile and build this ASM file and execute it on MDA-8086 kit.

The following code will control the LEDs such that only one LED is lit(ON) every time i pressed the push button(send interrupt to the CPU by requesting a service via the IR2 input to the 8259 controller).



**🡺EOI(***End Of Interrupt***)**command: Reset interrupt in PIC after accepted by ISR.

🡺 **Polling**[[1]](http://en.wikipedia.org/wiki/Diode%22%20%5Cl%20%22cite_note-5)

* Polling is the process where the computer waits for an external device to check for it readiness.
* The computer does not do anything else than check the status of the device
* Polling is often used with low-level

**3.3 8259a Interrupt Control (Interruption Technique)**

**Setp1:** I write the following code and save it to an Assembly file(.asm).I Compile and build this ASM file and execute it on MDA-8086 kit.

* The following code control the seven-segment display .when i pressed the push button, the seven segment display will be incremented until it reached 9 it will go to zero again, and so 8259a used to control the lighting sequence, through software interrupts.
* The code in part 2 and 3 are checking the interrupt occurrence in different ways. In part 3 i use the interrupt to check that, since i write the ISR for INT 24H in my code. The Micro Processor remains idle (we use the infinite loop with ‘nop’ instruction) until it is interrupted (using IR2 of PIC). but in part 2,the interrupt occurrence is checking using polling command.
* How come results changes on the Seven Segment while the CPU enters an infinite loop?

The run of program stay in loop until interrupt occur.



**3.4 8259a Interrupt Control (Interruption Technique) (C Code)**

**Setp1:** I write the following code and save it to an Assembly file(.asm).I Compile and build this ASM file and execute it on MDA-8086 kit.

* The following code is a C code to implement the same previous function but, it will count from A-F.
* **asm** used to insert lines of assembly language code into a C program.
* **INTERRUPT\_IN** used to push all registers.



**4.Conclusion**

In this experiment i learned the interrupt and how it implement in the interrupt vectors table(two byte for IP&to byte for CS) . Also, I learned how to defined software interrupts and how to change it`s function .I learn how to program the PIC (ICWs and OCWs). I learn how to convert the assembly code into C code and how to write assemble instructions in C code .

**5.Reference**

[1] <http://ecomputernotes.com/fundamental/input-output-and-memory/what-is-interrupt-types-of-interrupts>

[2]The Intel Microprocessors by Barry.B.Bary (Eighth Edition)

[3] [http://dis-dpcs.wikispaces.com/6.5.3+Interrupts+and+Polling](http://dis-dpcs.wikispaces.com/6.5.3%2BInterrupts%2Band%2BPolling)