

EEM332 Lecture8: Input/Output instructions of 8086 Polling/Interrupt

CPU

Control Unit
Arithmetic Logic Unit
Registers

CONTROL BUS
DATA BUS
ADDRESS BUS

Memory System
Input/Output System

Basic Input/Output Ports

— 8086/8088 Microprocessors —

Basic I/O Techniques

RAM 0000 FFFF
I/O Space 0000 FFFF
Memory Mapped I/O
I/O Mapped I/O
IN OUT
LOD STO MOV

Memory mapped I/O

- [Devices and memory share an address space](#)
- I/O looks just like memory read/write
- No special commands for I/O
- Large selection of memory access commands available

Isolated I/O

- [Separate address spaces](#)
- Need I/O or memory select lines
- **Special commands for I/O**
 - Limited set

- Communicating with I/O devices using **IN and OUT instructions** is referred to **isolated I/O**.
- However, there are many microprocessors, such as the new **RISC processors**, that do not have **IN and OUT inst**
- In such cases these microprocessors use what is called **memory-mapped I/O**.
- In memory-mapped I/O, a memory location is assigned to be an input or output port.

8088/8086 INPUT/OUTPUT INSTRUCTIONS (Isolated)

- All x86 microprocessors, from the 8088 to the Pentium, can access external devices called *ports* using I/O instructions.
- The x86 CPU is one of the few processors that have I/O space in addition to memory space.
- Memory can contain both opcodes and data
- I/O ports contain data only. Two instructions: “OUT” and “IN”.
- These instructions can send data from the accumulator (AL or AX) to ports or bring data from ports into the accumulator.
- In accessing ports, we can use an 8-bit or 16-bit data port.

Format	Inputting data	Outputting data	Adress rang
(1)	IN dast,source IN AL, port #	OUT dest, source OUT port #, AL	00 – FF A0 –A7
(2)	MOV DX, port # IN AL, DX	MOV DX, port # OUT DX, AL	0000 – FFFF A0- A15

Example : In a given 8088-based system, **port address 22H** is an **input port** for monitoring the temperature. Write Assembly language instructions to monitor that port continuously for the temperature of **100 degrees**. **If it reaches 100, then BH should contain ‘Y’.**

Solution: (format₍₁₎..**256 port**)

```
BACK: IN  AL,22H ;get the temperature from port # 22H
      CMP AL,100 ;is temp = 100?
      JNZ BACK  ;if not, keep monitoring
      MOV BH,'Y' ;temp = 100, load 'Y' into BH
```

Format (2):

Inputting Data

MOV **DX**,port#

IN AL,DX

Outputting Data

MOV **DX**,port#

OUT DX,AL

- port# is also the address of the port, except that it can be **from 0000 to FFFFH**, allowing up to 65,536 input and 65,536 output ports.
- **The 16-bit port address is carried on address bus A0 – A15**
- The use of a register as a pointer for the port address has an advantage in that the port address can be changed very easily, especially in cases of dynamic compilations where the port address can be passed to DX.

Example:

- The following program sends **values 55H and AAH to I/O port address 300H (a 16-bit port address)**; the program **toggles the bits of port address 300H continuously**.

```
BACK:  MOV DX,300H      ;DX = port address 300H
        MOV AL,55H
        OUT DX,AL       ;toggle the bits
        MOV AL,0AAH
        OUT DX,AL       ;toggle the bits
        JMP BACK
```

- We can only use register DX for 16-bit I/O addresses. Also notice the use of register AL for 8-bit data.
- The following code transfers the contents of register BL to port address 378H.

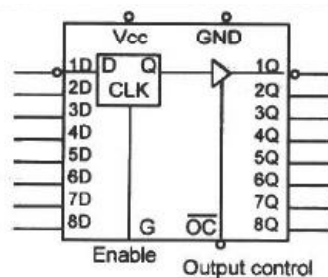
```
MOV DX,378H      ;DX=378 the port address
MOV AL,BL        ;load data into accumulator
OUT DX,AL       ;write contents of AL to port
                  ;whose address is in DX
```

- The following program gets data from port address 300H and sends it to port address 302H.

```
MOV      DX,300H    ; load port address
IN       AL,DX      ;bring in data
MOV      DX,302H
OUT     DX,AL      ;send it out
```

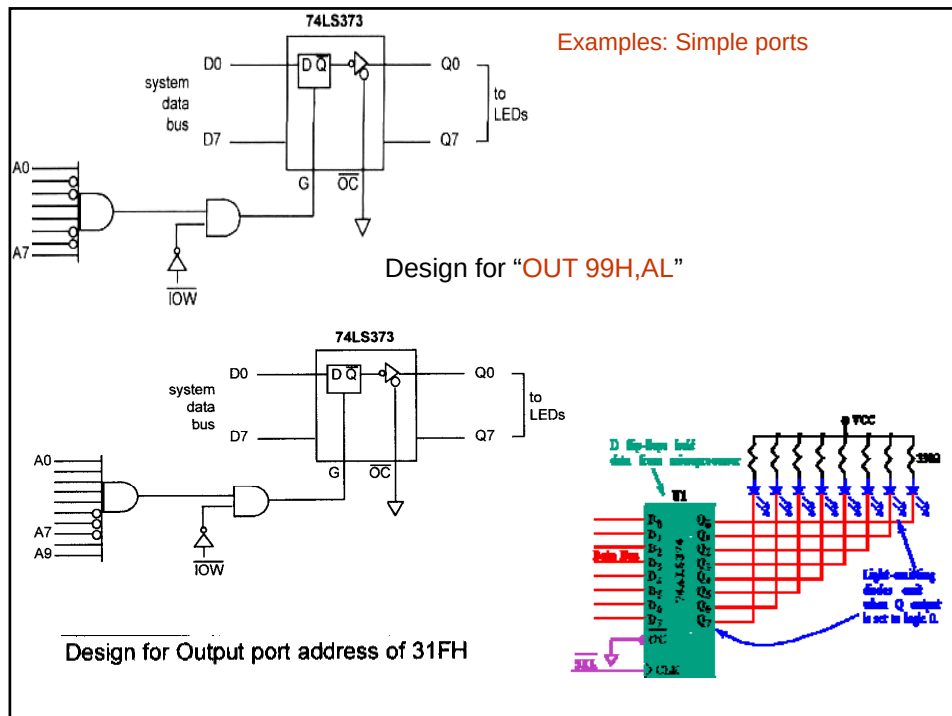
ADDRESS DECODING AND DESIGN

- In every computer, whenever data is sent out by the CPU via the data bus, **the data must be latched by the receiving device.**
- While **memories have an internal latch** to grab the data, a **latching system must be designed** for simple I/O ports.
- The **74LS373** can be used for this purpose.
- The concept of address bus decoding for I/O instructions is exactly the same as for memory. The following are the steps:
 - The control signals **IOR** and **IOW** are used along with the decoder.
 - For an **8-bit** port address, **A0 – A7** is decoded.
 - If the port address is **16-bit** (using DX), **A0 - A15** is decoded.



Function Table			
Output Control	Enable		Output
	G	D	
L	H	H	H
L	H	L	L
L	L	X	Q0
H	X	X	Z

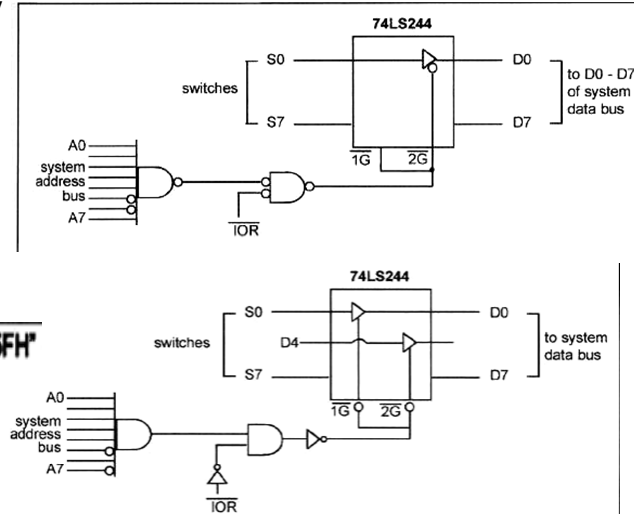
Figure 11-1. 74LS373 D Latch



EXAMPLES: Input port By using Buffer (74LS244)

The design of "IN AL,9FH" using the 74LS244 as a tri-state buffer is depicted below.

Input port design for "IN AL,5FH"



The 74LS244 not only plays the role of buffer, but also provides the incoming data with sufficient driving capability to travel all the way to the CPU. Indeed, the 74LS244 chip is widely used for buffering and providing high driving capability for unidirectional buses. The 74LS245 is used for bidirectional buses,

Input port design using the Octal buffer 74LS244

- When data is coming in by way of a data bus, it must come in through a **three-state (tri-state) buffer**.
- A tri-state buffer is internal and therefore invisible.
- For the simple input ports we use the 74LS244 chip.
- **1G and 2G each control only 4 bits of the 74LS244**, they both must be activated for the 8-bit input

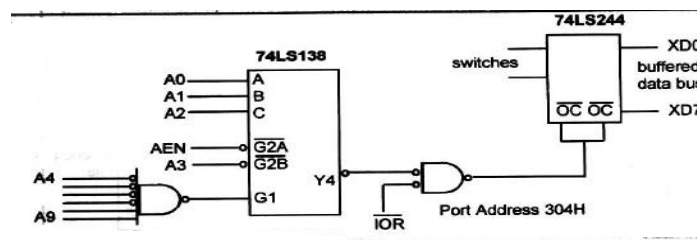
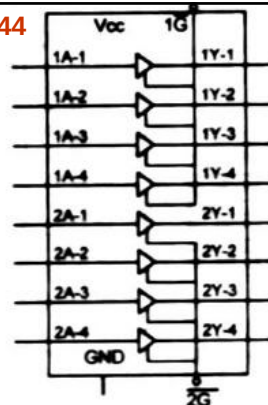


Figure 11-9. Using 74LS138 for I/O Address Decoder

8255 PPI CHIP ([Parallel Peripheral Interface](#))

- One of the most widely used I/O chips.
- It is a 40-pin DIP chip and has three separately accessible [ports, A, B, and C](#), which can be programmed, hence the name PPI (programmable peripheral interface).
- The individual ports can be programmed to be input or output.
- They can also be changed dynamically, in contrast to the 74LS244 and 74LS373, which are hard-wired.

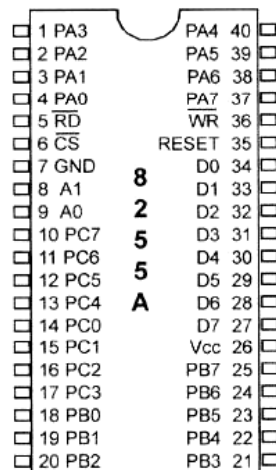


Figure 11-11. 8255 PPI Chip

Port A (PA0 – PA7) (8-bit port)

- Port A can be programmed all as I/O port.

Port B (PB0 – PB7) (8-bit port)

- Port B can be programmed all as I/O port

Port C (PC0 – PC7) (8-bit port)

- Port C can be programmed all as I/O. It can also be split into two parts, **CU** (upper bits PC4 - PC7) and **CL** (lower bits PC0 – PC3). Each can be used for I/O individually.

RD and WR

- If the 8255 is using peripheral I/O design, IOR and IOW of the system bus are connected to these two pins. If the port uses memory-mapped I/O, MEMR and MEMW activate them.

RESET

- This is an active-high signal input into the 8255 used to clear the control register. When RESET is activated, all ports are initialized as input ports. This pin must be connected to the RESET output of the system bus or ground, making it inactive. Like all IC input pins, it should not be left unconnected.

8255 PPI CHIP

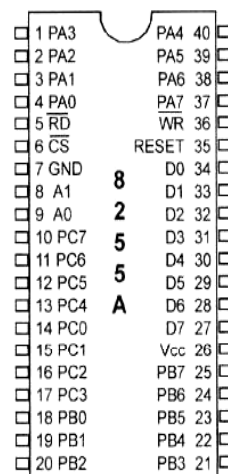


Figure 11-11. 8255 PPI Chip

8255 PPI CHIP

A0, A1, and CS

- While **CS (chip select)** selects the entire chip, address pins A0 and A1 select the specific port within the 8255.
- These three pins are used to access ports A, B, C, or the control register, as shown in Table

CS	A1	A0	Selects
0	0	0	Port A
0	0	1	Port B
0	1	0	Port C
0	1	1	Control Register
1	x	x	8255 is not selected

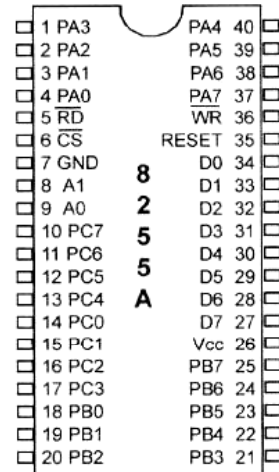


Figure 11-11. 8255 PPI Chip

Mode selection of the 8255A

While ports A, B, and C are used for I/O data, it is the control register that must be programmed to select the operation mode of the three ports A, B, and C. The ports of the 8255 can be programmed in any of the following modes.

- Mode 0, simple I/O mode.** In this mode, any of the ports A, B, CL, and CU can be programmed as input or output. In this mode, all bits are out or all are in. In other words, there is no control of individual bits. Mode 0 is the most widely used mode in current system I/O interfacing design.
- Mode 1.** In this mode, **ports A and B can be used as input or output ports with [handshaking capabilities](#)**. This mode is not used due to timing incompatibility with devices such as the printer.
- Mode 2.** In this mode, **port A can be used as a bidirectional I/O port with [handshaking capabilities](#)**. This mode is rarely used.
- BSR mode (Bit set/reset)**

- The 8255 chip is programmed in any of the modes (Mode 0, 1 or 2) by sending a byte (or control word) to the control register of the 8255. For example to make ports A, B, and C input or output ports, we make D7 = 1 according to Figure 11-12.

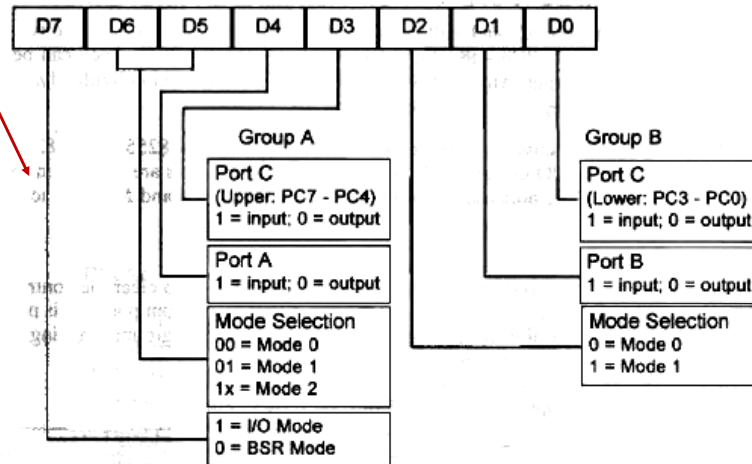


Figure 11-12. 8255 Control Word Format (I/O Mode)

- If D7 = 0, we have **BSR mode**. In **BSR (bit set/reset) mode**, the bits of port C are programmed individually. This mode is also rarely used.

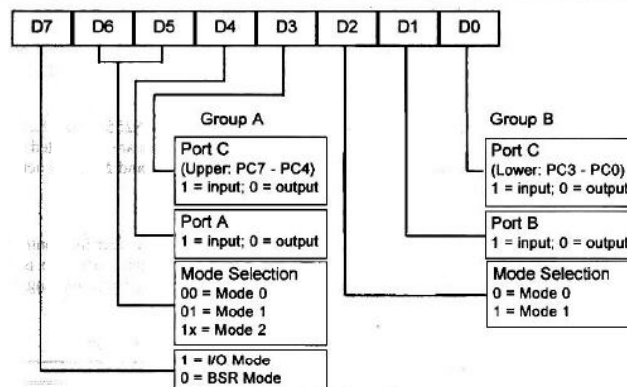


Figure 11-12. 8255 Control Word Format (I/O Mode)

- To select simple I/O mode of 0, we need **1000 0000 as the control word**.
- Similarly, to get PB as input, and PA and all of PC as output, we must have 1000 0010 or 82H for the control word.

Example 11-4: Find the control word if **PA = out**, **PB = in**, **PC0- PC3 = in**, and **PC4 - PC7 = out**. (Mode 0)

Solution:

From Figure 11-12 we get the control word of 1000 0011 in binary or 83H

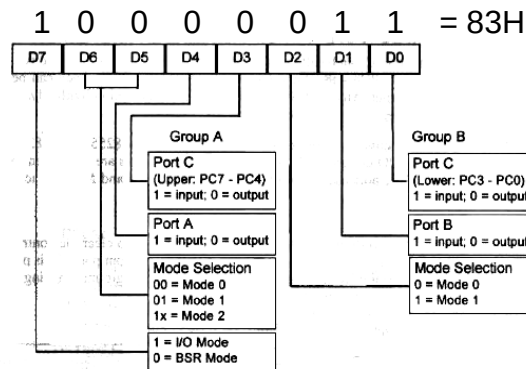


Figure 11-12. 8255 Control Word Format (I/O Mode)

(b) Find the control byte (word) for this configuration. **Port A as input**, **B as output**, and all the bits of port C as output.

Solution: (Mode 0)

(b) The control word is 90H, or 1001 0000.

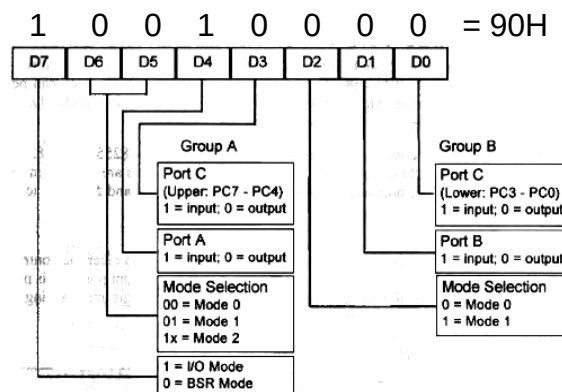


Figure 11-12. 8255 Control Word Format (I/O Mode)

Example : The 8255 shown in Figure 11-13 is configured as follows: port A as input, B as output, and all the bits of port C as output. (Mode 0)

- Find the port addresses assigned to A, B, C, and the control register.
- Find the control byte (word) for this configuration. (Mode 0)

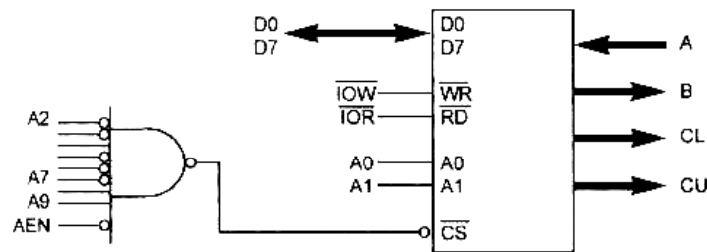


Figure 11-13. 8255 Configuration for Example 11-5

- Find the port addresses assigned to A, B, C, and the control register.

(a) The port addresses are as follows: (See Table 11-3)

<u>CS</u>	<u>A1</u>	<u>A0</u>	<u>Address</u>	<u>Port</u>	(Mode 0)
11000100	0	0	310H	Port A	
11000100	0	1	311H	Port B	
11000100	1	0	312H	Port C	
11000100	1	1	313H	Control register	

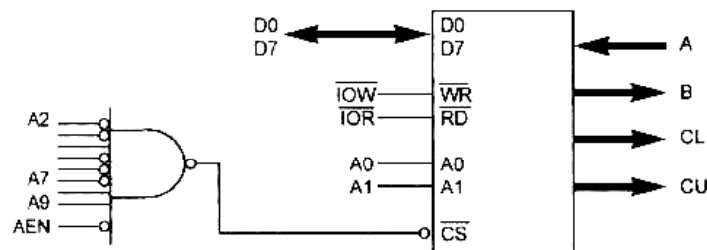


Figure 11-13. 8255 Configuration for Example 11-5

Example 4-5

(Mode 0)

The 8255 shown in Figure 4-6 is configured as follows: port A as input, B as output, and all the bits of port C as output.

- Find the port addresses assigned to A, B, C, and the control register.
- Find the control byte (word) for this configuration.
- Program the ports to input data from port A and send it to both ports B and C.

Solution:

- The port addresses are as follows:

CS*	A1	A0	Address	Port
0101 00	0	0	50H	Port A
0101 00	0	1	51H	Port B
0101 00	1	0	52H	Port C
0101 00	1	1	53H	Control register

- The control word is 90H, or 1001 0000.

- One version of the program is as follows:

```

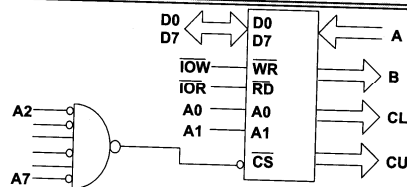
MOV AL,90H      ;control byte PA=in, PB=out, PC=out
OUT 53H,AL      ;send it to control register
IN  AL,50H      ;get the data from PA
OUT 51H,AL      ;send it to both PB
OUT 52H,AL      ; and PC
    
```

Using the EQU directive one can rewrite the above program as follows:

```

PORTA EQU 50H
PORTB EQU 51H
PORTC EQU 52H
CNTLREG EQU 53H

...
MOV AL,90H      ;control byte PA=in, PB=out, PC=out
OUT CNTLREG,AL  ;send it to control register
IN  AL,PORTA    ;get the data from PA
OUT PORTB,AL    ;send it to both PB
OUT PORTC,AL    ; and PC
    
```



Example 4-6

- Find the port address for Figure 4-7.
- Find the control word if PA=out, PB=in, PC0 - PC3 =in, and PC4 - PC7=out.
- Program the 8255 to get data from port A and send it to port B. In addition, data from PCL is send out to the PCU.

(Mode 0)

Solution:

- The port addresses are as follows:

CS*	A1	A0	Address	Port
0111 11	0	0	7CH	Port A
0111 11	0	1	7DH	Port B
0111 11	1	0	7EH	Port C
0111 11	1	1	7FH	Control register

- The control word is 83H, or 1000 0011.

- The code is as follows.

```

MOV AL,83H      ;control byte PA=out, PB=in, PCL=in, PCU=out
OUT 7FH,AL      ;send it to control register
IN  AL,7DH      ;get the data from PB
OUT 7CH,AL      ;send it to PA
IN  AL,7EH      ;get the bits from PCL
AND AL,0FH      ;mask the upper bits
ROL AL,1        ;shift the bits
ROL AL,1        ;to upper position
ROL AL,1
ROL AL,1
OUT 7EH,AL      ;send it to PCU
    
```

Alternately, the four instructions above of "ROL AL,1" could be replaced with the following two instructions:

```

MOV CL,4        ;count = 4
ROL AL,CL       ;rotate 4 times
    
```

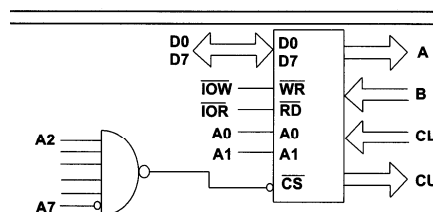
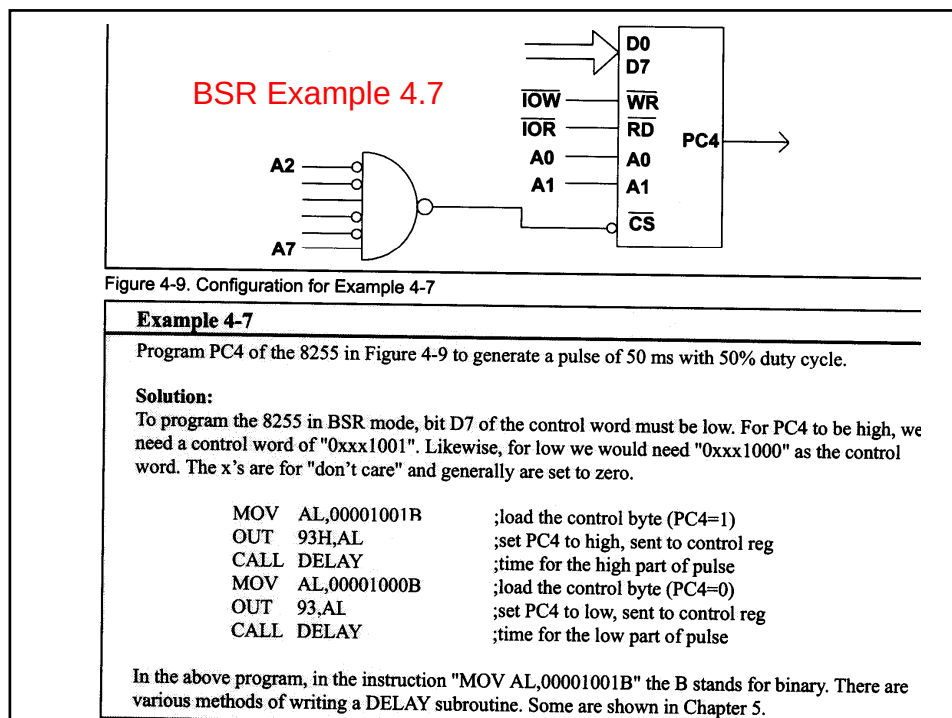
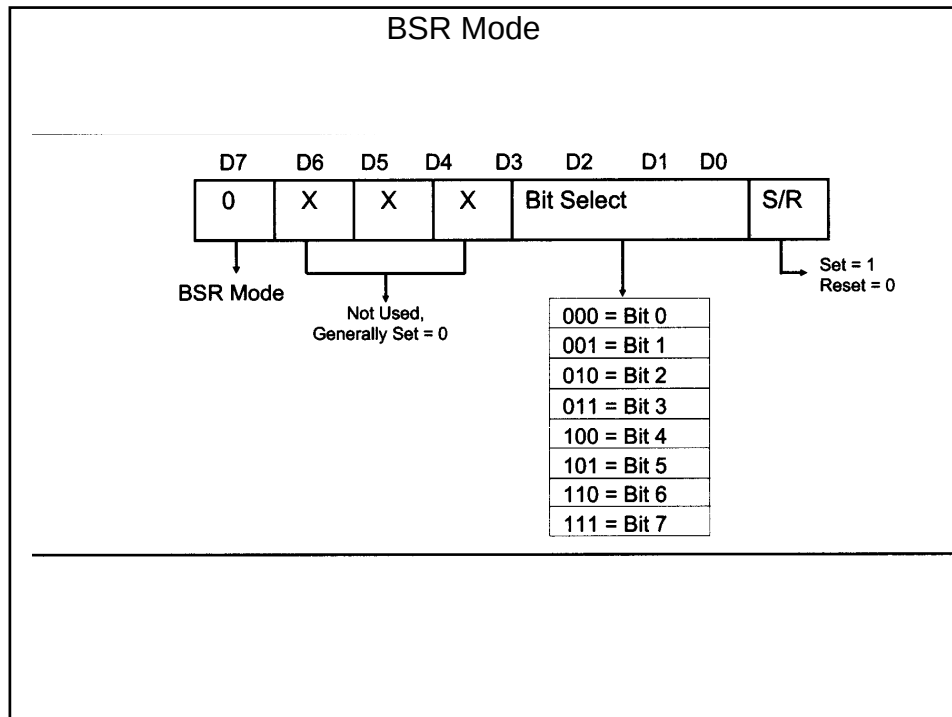
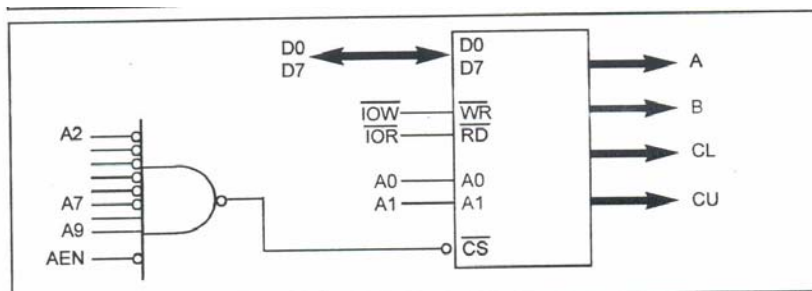
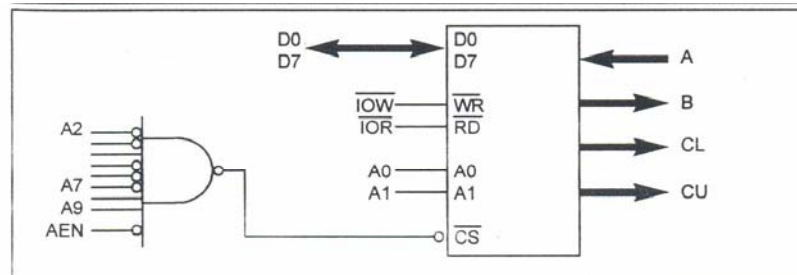


Figure 4-7 Configuration for Example 4-6



Homework::



Polling & Interrupts

- 2 techniques are available to service external devices:
 - Polling
 - Interrupts

Polling:

- CPU periodically polls ALL external devices & takes action if required

Interrupts:

- external device indicates request for attention by sending a signal via a control line
- it stops the current CPU activity and responds to the device that requested attention

Polling:

- if the frequency of polling is high – a significant overhead is incurred on a CPU
- If, however, the frequency of polling is low - events may be undetected - data lost due to latency etc.

