Lecture 22: 80x86 IO Interfacing

Seyed-Hosein Attarzadeh-Niaki

Based on the slides by Hongzi Zhu

Review

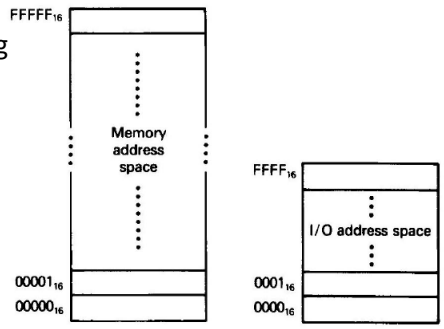
- Memory and memory interfacing
  - Memory chips
  - Address decoding
  - Data integrity
  - 8086 memory interfacing

Outline

- IO interfacing
  - Basic IO interfacing
  - 8255 programmable peripheral interface
  - 8253 programmable interval timer

I/O in x86 Family

- X86 microprocessors have an I/O space in addition to memory space
- Use special I/O instructions accessing I/O devices at ports (i.e., addresses for I/O)
- Memory can contain machine codes and data, I/O ports only contain data
- Also referred to as *peripheral I/O* or *isolated I/O*
- I/O ports are 8 bits in width.
  - a 16-bit port is actually two consecutive 8-bit ports being addressed



I/O Instructions – 8-Bit Instance

```
                Inputting Data          Outputting Data
Format:         IN      dest,source     OUT   dest,source
```

- Direct I/O instructions:
  - port# ranges from 00h to 0ffh, 256 ports in total

```
(1)             IN      AL,port#        OUT   port#,AL
```

- Indirect I/O instructions:
  - port# ranges from 0000h to 0ffffh, 65536 ports in all
  - use a 16-bit address that resides in the DX register

```
(2)             MOV     DX,port#        MOV   DX,port#
                IN      AL,DX           OUT   DX,AL
```

- Note: no segment concept for port addresses
- Instructions are also provided to transfer strings of data between memory and I/O.
  - INS and OUTS, found except the 8086/8088

I/O Example

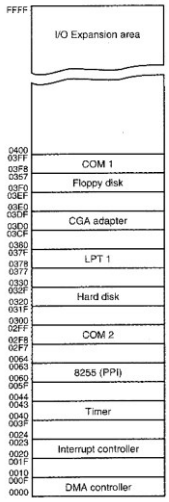
In a given 8088-based system, port address 22H is an input port for monitoring the temperature. Write Assembly language instructions to monitor that port continuously for the temperature of 100 degrees. If it reaches 100, then BH should contain 'Y'.

Solution:

```
BACK:   IN    AL,22H
        CMP   AL,100
        JNZ   BACK
        MOV   BH,'Y'
```

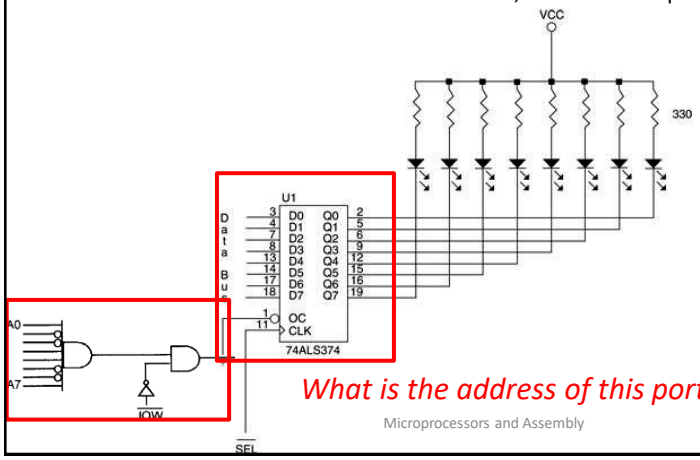
Personal Computer I/O Map

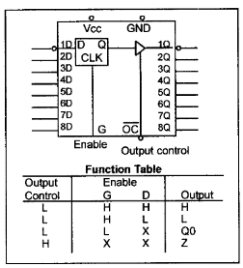
- the PC uses part of I/O map for dedicated functions, as shown here
- I/O space between ports 0000H and 03FFH is normally reserved for the system and ISA bus
- ports at 0400H–FFFFH are generally available for user applications, main-board functions, and the PCI bus
- 80287 coprocessor uses 00F8H–00FFH, so Intel reserves I/O ports 00F0H–00FFH



Output Port Design: Latches

- Latch the data coming from the CPU
- Address decoding
- TTL-compatible voltages: 0-> 0.0 V to 0.4 V , 1-> 2.4 V to 5.0 V
- currents: 0-> 0.0 to 2.0 mA, 1-> 0.0 to 400 µA



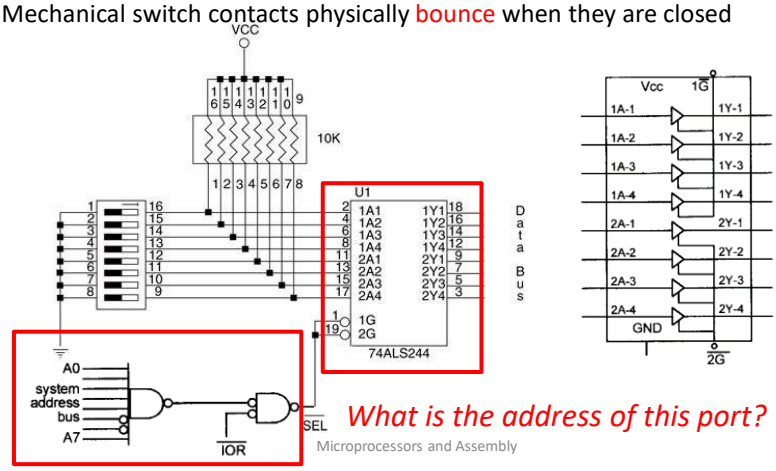


74LS373 D Latch

What is the address of this port?

Input Port Design: Three-State Buffers

- Use *tri-state buffer* to connect to system data bus
- Address decoding
- A pull-up resistor ensures when the switch is open, the output signal is a logic 1.
- Mechanical switch contacts physically bounce when they are closed



What is the address of this port?

I/O Instructions – 16-Bit Instance

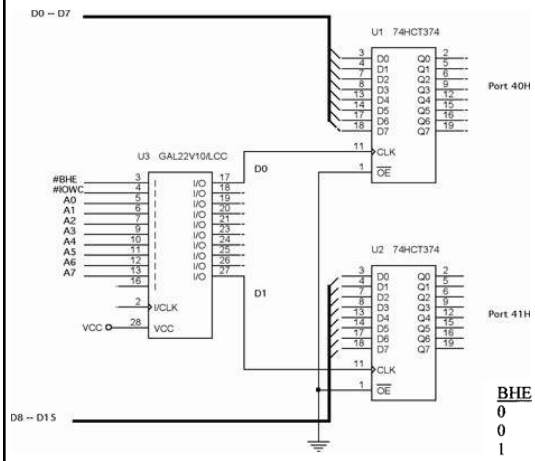
- For 16-bit I/O modules
- Direct I/O instructions:
  - port# ranges from 00h to 0ffh, 256 ports in total

```
IN AX, port#        OUT port#, AX
```

- Indirect I/O instructions:
  - port# ranges from 0000h to 0ffffh, 65536 ports in all
  - use a 16-bit address that resides in the DX register

```
MOV DX, port#       MOV DX, port#
IN AX, DX           OUT DX, AX
```

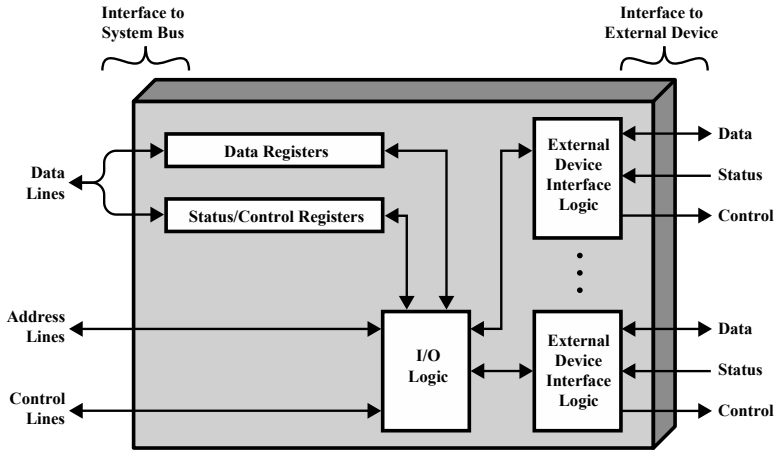
Interfacing 8-bit I/O Modules To A 16-Bit Data Bus



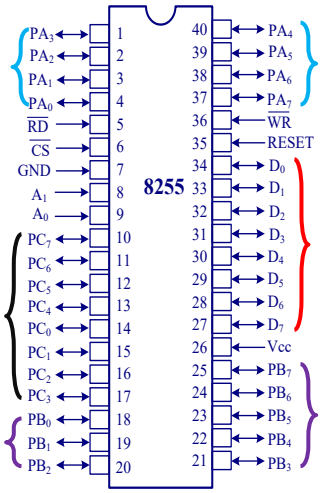
- data for even-address ports are carried on data bus D0-D7
- data for odd-address ports are carried on data bus D8-D15

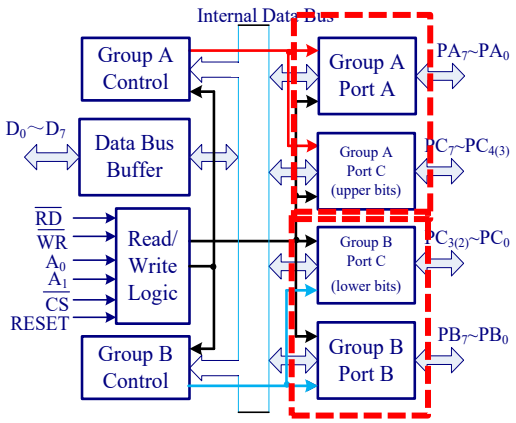
BHE	A0	
0	0	Even-addressed words (uses D0 - D15)
0	1	Odd-addressed byte (uses D8 - D15)
1	0	Even-addressed byte (uses D0 - D7)

Recall: I/O Module Structure



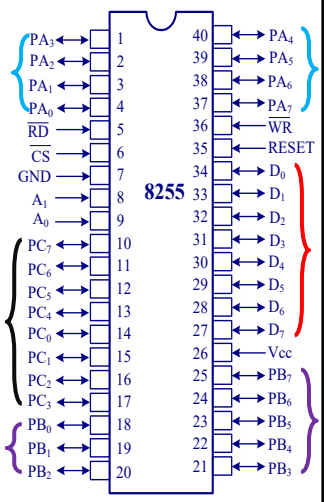
8255 Package & Internal Structure





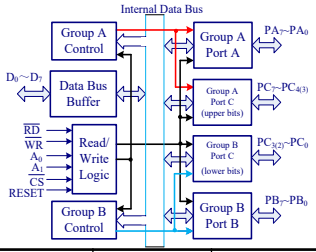
Internal Structure and Pins

- Three data ports: A, B, and C
  - Port A (PA_0~PA_7): can be programmed **all** as input/output
  - Port B (PB_0~PB_7): can be programmed **all** as input/output
  - Port C (PC_0~PC_7): can be split into two separate parts *PCU* and *PCL*; any bit can be programmed **individually**
- Control register (CR)
  - Internal register: used to setup the chip
- Group A, Group B and control logic
  - Group A (PA & PCU)
  - Group B (PB & PCL)



Internal Structure and Pins

- Data bus buffer
  - An interface between CPU and 8255
  - Bidirectional, tri-state, 8-bit
- Read/Write control logic
  - Internal and external control signals
  - **RESET**: high-active, clear the control register, all ports are set as input port
  - **~CS**, **~RD**, **~WR**
  - **A_1, A_0:** port selection signals



~CS	A_1	A_0	~RD	~WR	Function
0	0	0	0	1	PA->Data bus
0	0	1	0	1	PB->Data bus
0	1	0	0	1	PC->Data bus
0	0	0	1	0	Data bus->PA
0	0	1	1	0	Data bus->PB
0	1	0	1	0	Data bus->PC
0	1	1	1	0	Data bus->CR
1	×	×	1	1	D_0~D_7 in float

Operation Modes

- Input/output modes
  - Mode 0, simple I/O mode:
    - **PA**, **PB**, **PC**: PCU{PC_4~PC_7}, PCL{PC_0~PC_3}
    - No Handshaking: *negotiation between two entities before communication*
    - Each port can be programmed as input/output port
  - Mode 1:
    - **PA**, **PB** can be used as input/output ports with *handshaking*
    - PCU{PC_3~PC_7}, PCL{PC_0~PC_2} are used as handshake lines for PA and PB, respectively
  - Mode 2:
    - Only **PA** can be used for *bidirectional handshake* data transfer
    - PCU{PC_3~PC_7} are used as handshake lines for PA
- Bit set/reset (BSR) mode
  - Only **PC** can be used as output port
  - Each line of PC can be set/reset individually

Control Register & Op. Modes

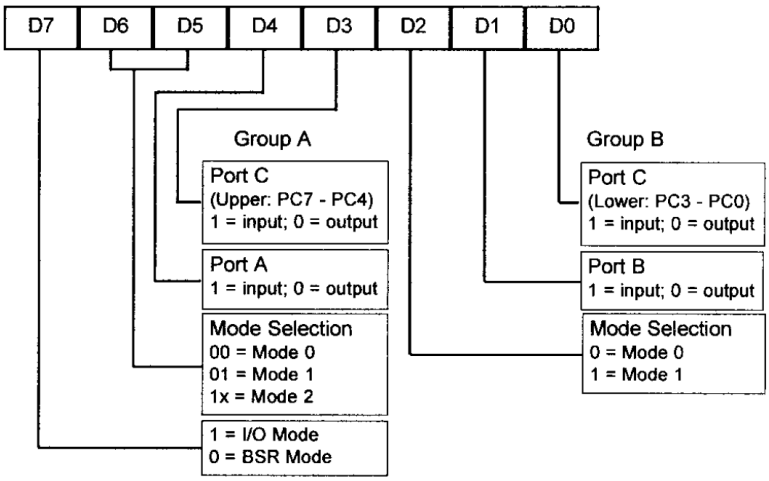
- An 8-bit internal register in 8255
- Selected when A_1=1, A_0=1
- Mode selection word
  - Input/output modes

1	D_6	D_5	D_4	D_3	D_2	D_1	D_0

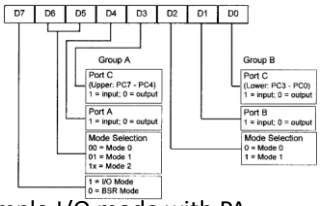
  - BSR mode

0	D_6	D_5	D_4	D_3	D_2	D_1	D_0

Mode Selection with Control Register



Select Input/output Mode Examples



1. Write ASM instructions for setting the 8255 in simple I/O mode with PA and PB being output port and PC being input port.

```
MOV    AL, 10001001B
MOV    DX, ControlPort
OUT    DX, AL
```

2. Assume that the address of the control register of the 8255 is 63H, give out the instructions that set up the 8255 in mode 0 where PA, PB and PCU are used as input ports and PCL is used as output port.

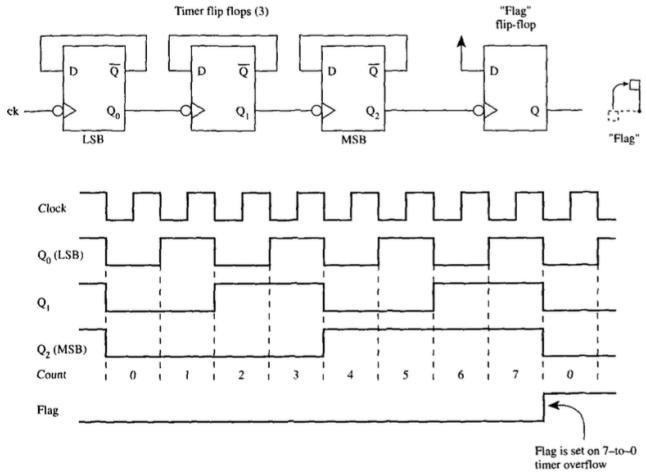
```
MOV    AL, 10011010B
OUT    63H, AL
```


Time in Embedded Real-time Systems

- Time is an inherent part of real-time systems
- **Problem**: Instruction-sets do not expose the precise timing of the underlying hardware to the programmer
- **Solutions?**

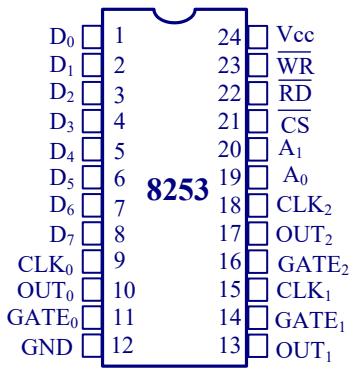
1. Number of executed instructions × cycle time
   - Problems: imprecise timing (especially with modern hardware), inflexible software
2. A dedicated hardware for timing

Simplified Structure of a Timer/Counter

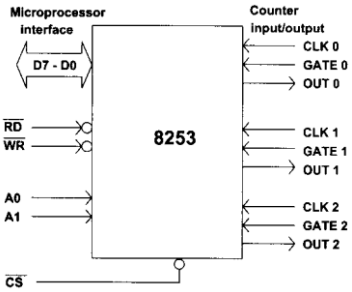


The 8253/54 PIT

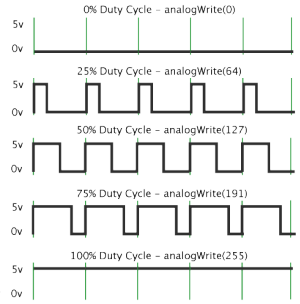
- The 8253/54 Programmable interval timer is used to generate a lower frequency for various uses e.g.,
  - Event counter
  - Accurate time delays
- 8253 used in the first PC decoded at ports 40H–43H
  - Replaced with 8254 later



Interface to the System

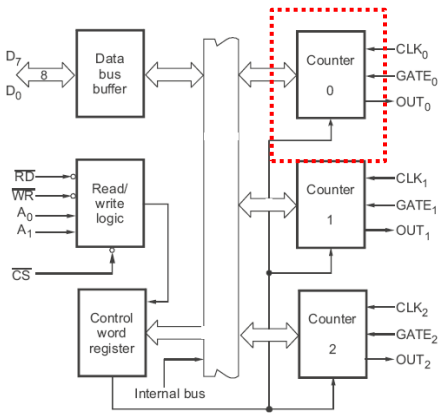


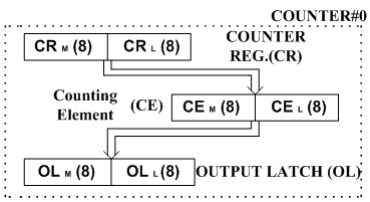
- There are three independent counters.
- The input frequency can be divided from 1 to 65536 (Binary), or from 1 to 10000 (BCD)
- Shape of the output frequency:
  - ❖Square-wave
  - ❖One-shot
  - ❖Square-wave with various duty cycles.



- Gate is used to enable (High) or disable (Low) the counter.
- Bidirectional bus D0-D7 is connected to D0-D7 of the system bus (even addresses).

Internal Structure





To operate a 16-bit down counter

- ❖ a 16-bit count is loaded in the counter
- ❖ begins to decrement the count until it reaches 0
- ❖ generates a pulse that can be used to interrupt the CPU

Summary of the Features

- Three independent 16-bit down counters
- 8254 can handle inputs from DC to 10 MHz (5MHz 8254-5 8MHz 8254 10MHz 8254-2) whereas 8253 can operate up to 2.6 MHz
- Three counters are identical and pre-settable, and can be programmed for either binary or BCD count
- Counter can be programmed in six different modes
- Compatible with all Intel and most other microprocessors
- 8254 has a powerful command called READ BACK command which allows the user to check the count value, programmed mode and current mode and current status of the counter

8253 Bus Access

Data bus buffer

- interface the 8253/4 to the system data bus
- Bi-directional, tri-state, 8-bit

Read/Write control logic

- ~CS
  - Tied to a decoded address
- ~RD, ~WR
  - In isolated I/O: ~IOR, ~IOW
  - Memory-mapped I/O: ~MEMR, ~MEMW
- A1, A0
  - Select the control word register and counters
  - usually connected to address lines A1, A0

A_1	A_0	Selection
0	0	Counter 0
0	1	Counter 1
1	0	Counter 2
1	1	Control word Register

How to R/W 16-bit counters from a single address?

/CS	/RD	/WR	A1A0	FUNCTION
0	1	0	00	Write counter0 (to CR0)
0	1	0	01	Write counter1 (to CR1)
0	1	0	10	Write counter2 (to CR2)
0	1	0	11	Write control port
0	0	1	00	Read counter0 (from OL0)
0	0	1	01	Read counter1 (from OL1)
0	0	1	10	Read counter2 (from OL2)
0	0	1	11	Read control port (for *8254*)
1	X	X	XX	Not available

Write/Read Operations

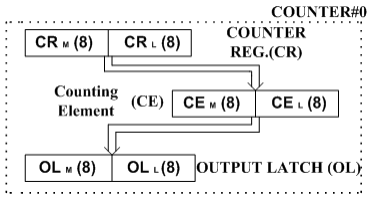
- WRITE:
  - Write a control word into control register
  - Load the low-order byte of a count in the counter register
  - Load the high-order byte of a count in the counter register
- READ:
  - **Simple Read**: two I/O read operations, first one for low-order byte and then one for the high order byte
  - **Counter Latch Command**: one I/O write operation used to write a control word to the control register to latch a count in the output latch, then two I/O read operations are used to read the latched count as in Simple Read.
  - ➢ **Read-Back Command**: for 8254 only

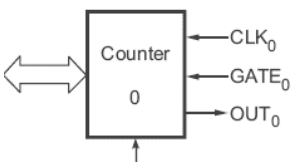
Operation of 8253

- 8253 takes one CLK pulse to convey the count from CR to CE
- CE will start to count only when GATE = 1
  - *When to check the GATE?*

  On every CLK pulse's rising (0-to-1) edge
  - *When to count down?*

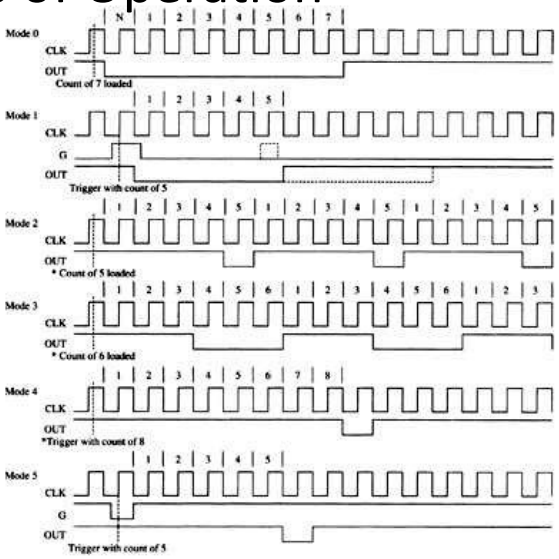
  On every CLK pulse's falling (1-to-0) edge





Modes of Operation

- Mode 0: Interrupt on Terminal Count
- Mode 1: Hardware Retriggerable One-shot
- Mode 2: Rate Generator
- Mode 3: Square Wave Rate Generator
- Mode 4: Software Triggered Strobe
- Mode 5: Hardware Triggered Strobe



8253/4 Gate Pin Operation

Modes 0 and 4

- Counting is suspended while GATE is low
- Resumed while GATE is high

Modes 1 and 5

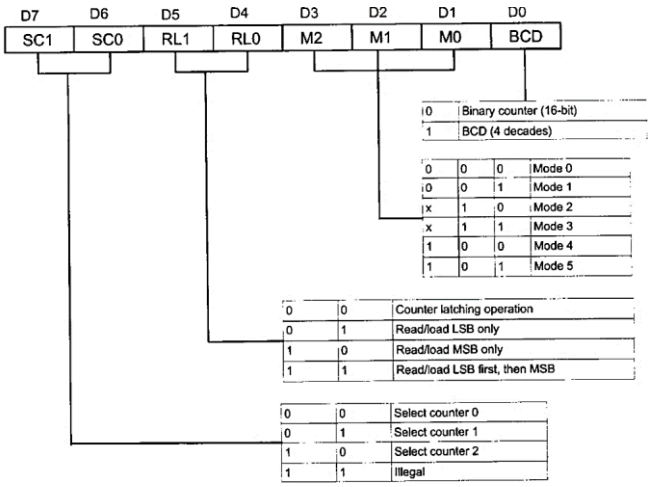
- Rising edge of GATE starts counting
- GATE may go low without affecting counting
- Another rising edge will restart the count from the beginning

Modes 2 and 3

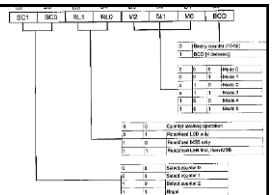
- GATE low forces OUT high immediately (without waiting for a clock pulse) and resets the counter (on the next clock falling edge)
- When GATE goes high again, counting restarts from the beginning

Signal Status Modes	Low Or Going Low	Rising	High
0	Disables Counting	— —	Enables Counting
1	— —	1) Initiates Counting 2) Resets Output after Next Clock	— —
2	1) Disables Counting 2) Sets Output Immediately High	Initiates Counting	Enables Counting
3	1) Disables Counting 2) Sets Output Immediately High	Initiates Counting	Enables Counting
4	Disables Counting	— —	Enables Counting
5	— —	Initiates Counting	— —

Control Word Format



Programming Example



Example 1: *Write a program to initialize counter 2 in mode 0 with a count of C030H. Assume address for control register = 0BH, counter 0 = 08H, counter 1 = 09H and counter 2 = 0AH.*

Sol. : Control word

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SC_1	SC_2	RW_1	RW_0	M_2	M_1	M_0	BCD
1	0	1	1	0	0	0	0

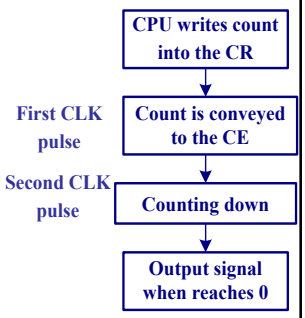
= B0H

Source Program

```
MOV  AL,B0H
OUT  0BH,AL       ; Loads control word (B0H) in the control
                  ; register.
MOV  AL,30H
OUT  0AH,AL       ; Loads lower byte of (30H)the count.
MOV  AL,0C0H
OUT  0AH,AL       ; Loads higher byte (C0H) of the count.
```

Mode 0: Interrupt on Terminal Count I

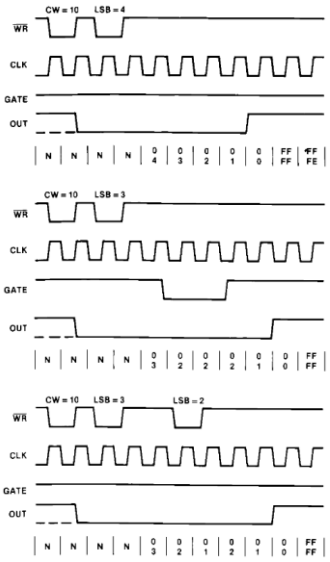
- Normal Operation:
  - The output will be initially **low** after the mode set operation;
  - After the count is loaded into the selected CR the output will remain **low**
  - When the terminal count is reached, the output will go **high** and remain high until the selected counter is reloaded
  - **Output: N clock pulses low and high afterwards after writing a count**



Mode 0: Interrupt on Terminal Count II

- Gate disable:
  - Gate = 1 enables counting
  - Gate = 0 disables counting
- New count:
  - If a new count is written to the counter, it will be loaded on the next CLK pulse and counting will continue from the new count
  - In case of two byte count:
    - Writing the first byte disables the current counting
    - Writing the second byte loads the new count on the next CLK pulse and counting will continue from the new count

Mode 0: Interrupt on Terminal Count III



When loading a new count *N*, the actual number of CLK pulses in OUT is *N*+1

Does not automatically repeat

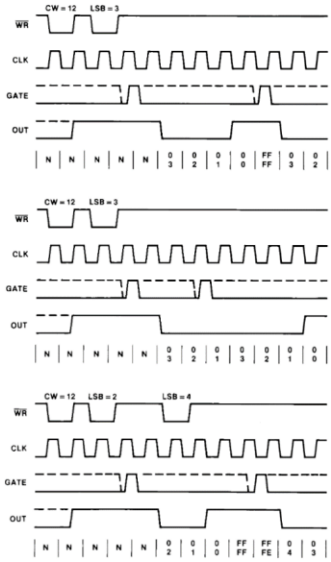
Mode 1: Hardware Retriggerable One-shot I

- Normal Operation:
  - The output will be initially **high** after the mode set operation;
  - The output will go **low** on the CLK pulse following the rising (0-to-1) edge of the gate input;
  - The output will go **high** on the terminal count and remain high until the next rising edge of the gate input.
  - **Output: one-shot of N clock pulses on every trigger**

Mode 1: Hardware Retriggerable One-shot II

- Retriggering:
  - retriggerable, hence the output will remain low for the full count after any rising edge of the gate input
- New count:
  - If the counter is loaded during one shot pulse, the current one shot is not affected unless the counter is retriggered
  - If retriggered, the counter is loaded with the new count and the one-shot pulse continues until the new count expires

Mode 1: Hardware Retriggerable One-shot III



When loading a new count *N*, the current counting will not be affected

Does not automatically repeat

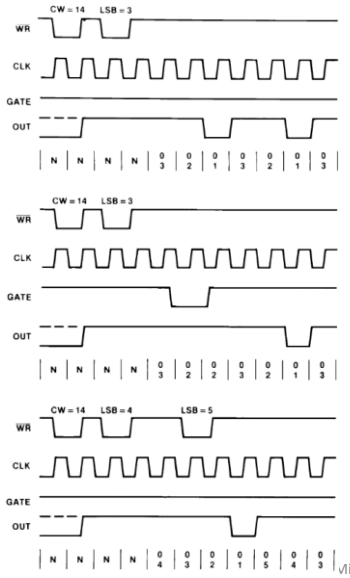
Mode 2: Rate Generator I

- Also called *divide-by-N counter*
- Normal Operation:
  - The output will be initially **high**;
  - The output will go **low** for one clock pulse before the terminal count;
  - The output then goes **high**, the counter reloads the initial count and the process is repeated
  - **Output: periodical signal with a period of N-1 clock pulses high and 1 clock pulse low**

Mode 2: Rate Generator II

- Gate disable:
  - If Gate=1 it enables a counting otherwise it disables counting (Gate=0)
  - If Gate goes low during a low output pulse, output is set immediately high
- New count:
  - The current counting sequence is not affected when the new count is written
  - If a trigger (a rising edge of GATE) is received after writing a new count but before the end of the current period, the new count will be loaded with the new count on the next CLK pulse and counting will continue from the new count
  - Otherwise, the new count will be loaded at the end of the current counting cycle
  - **Note : In mode 2, a count of 1 is illegal.** ***Why?***

Mode 2: Rate Generator III

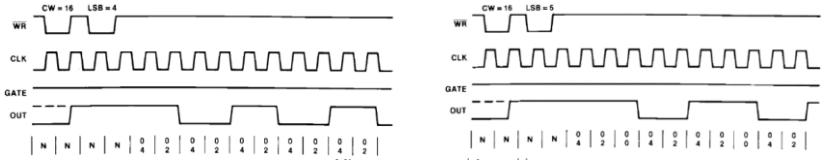


When loading a new count *N*, the current counting will not be affected

Automatically repeat on terminal count

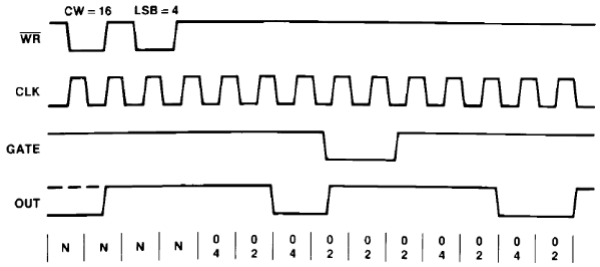
Mode 3: Square Wave Rate Generator I

- Normal Operation:
  - The output will be initially high;
  - For even count, counter is decremented by 2 on the falling edge of each clock pulse; when reaches terminal count, the state of the output is changed and the counter is reloaded with the full count and the whole process is repeated
  - For odd count, the first clock pulse decrements the count by 1. Subsequent clock pulses decrement the clock by 2. After timeout, the output goes low and the full count is reloaded. The first clock pulse (following the reload) decrements the count by 3 and subsequent clock pulse decrement the count by two. Then the whole process is repeated.
  - **Output: if the count is odd, the output will be high for (n+1)/2 counts and low for (n-1)/2 counts.**



Mode 3:
Square Wave Rate Generator II

- Gate disable:
  - If Gate is 1 counting is enabled otherwise it is disabled.
  - If Gate goes low while output is low, output is set high immediately. After this, When Gate goes high, the counter is loaded with the initial count on the next clock pulse and the sequence is repeated.



Mode 3:
Square Wave Rate Generator III

- New count:
  - The current counting sequence is not affected when the new count is written.
  - If a trigger is received after writing a new count but before the end of the current half-cycle of the square wave, the counter will be loaded with the new count on the next CLK pulse and counting will continue from the new count.
  - Otherwise, the new count will be loaded at end of the current half-cycle.

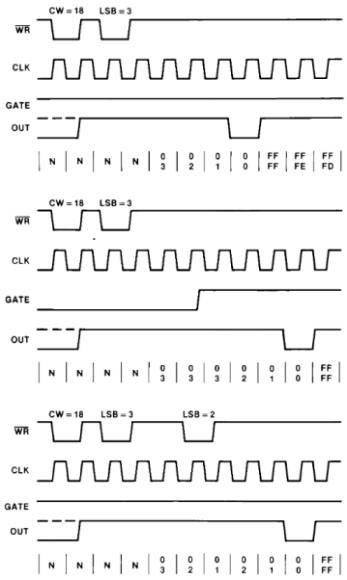
When loading a new count *N*, the current half will not be affected

Automatically repeat on terminal count

Mode 4: Software Triggered Strobe I

- Similar to mode 2 except that the counter is not auto-reloaded
- Normal Operation:
  - The output will be initially **high**;
  - The output will go **low** for one CLK pulse after the terminal count
- Gate disable:
  - If Gate is one, the counting is enabled; otherwise, it is disabled
- New count:
  - If a new count is written during counting, it will be loaded on the next CLK pulse and counting will continue from the new count. If the count is two byte then:
    - Writing the first byte has no effect on counting
    - Writing the second byte allows the new count to be loaded on the next CLK pulse

Mode 4: Software Triggered Strobe II



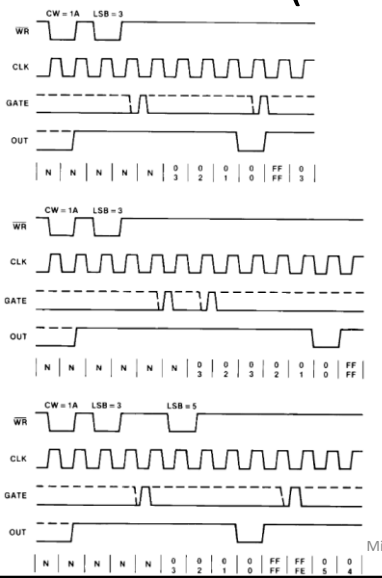
When loading a new count *N*, the actual number of CLK pulses in OUT is *N*+1

Automatically repeat

Mode 5: Hardware Triggered Strobe (Retriggerable) I

- Similar to mode 4 except that triggered by GATE.
- Normal Operation:
  - The output will be initially **high**;
  - The counting is triggered by the rising edge of the Gate
  - The output will go **low** for one CLK pulse after the terminal count
- Retriggering:
  - If the triggering occurs during the counting, the initial count is loaded on the next CLK pulse and the counting will be continued until the terminal count is reached
- New count:
  - the current counting sequence will not be affected. If the trigger occurs after the new count but before the terminal count, the counter will be loaded with the new count on the next CLK pulse and counting will continue from there

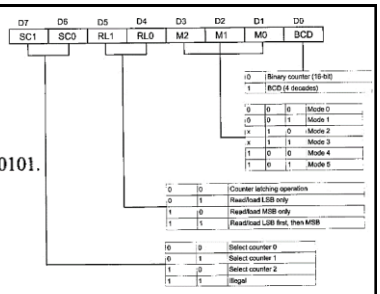
Mode 5: Hardware Triggered Strobe (Retriggerable) II



When loading a new count *N*, the current counting will not be affected

Automatically repeat on terminal count

Example:
Setting Up a Counter



Pin $\overline{CS}$ of a given 8253/54 is activated by binary address A7 - A2 =100101.

CS		A1A0	Port	Port address (hex)
1001	01	00	Counter 0	94
1001	01	01	Counter 1	95
1001	01	10	Counter 2	96
1001	01	11	Control register	97

(a) counter 0 for binary count of mode 3 (square wave) to divide CLK0 by number 4282 (BCD)
(b) counter 2 for binary count of mode 3 (square wave) to divide CLK2 by number C26A hex
(c) Find the frequency of OUT0 and OUT2 in (a) and (b) if CLK0 =1.2 MHz, CLK2 = 1.8 MHz.

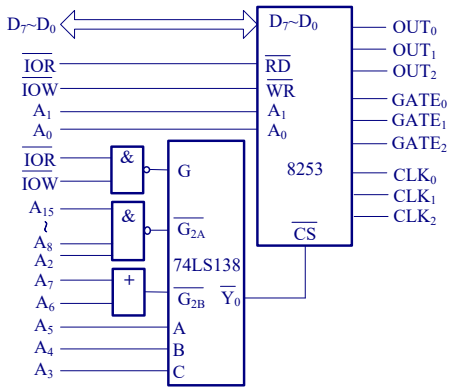
Solution:

(c) The output frequency for OUT0 is 1.2MHz divided by 4282, which is 280 Hz. Notice that the program in part (a) used instruction "MOV AX,4282H" since BCD and hex numbers are represented in the same way, up to 9999. For OUT2, CLK2 of 1.8 MHz is divided by 49770 since C26AH = 49770 in decimal. Therefore, OUT2 frequency is a square wave of 36 Hz.

```
OUT   96H,AL    ;send the low byte
MOV   AL,AH     ;to count 2
OUT   96H,AL    ;send the high byte to counter 2  )
```


Example

- The frequency of CLK is 2MHz, write initiation program to let counter 0 generate an interruption request after 100μs, let counter 1 generate 50% duty cycle square wave with a period of 10μs, and let counter 2 generate a negative pulse every 1ms.



```
MOV DX, 0FF07H
MOV AL, 00010000B       ;counter 0, write LSB only, mode 0, binary
OUT DX, AL
MOV AL, 01010110B       ;counter 1, write LSB only, mode 3, binary
OUT DX, AL

MOV DX, 0FF04H
MOV AL, 200             ; initial count for counter 0
OUT DX, AL
MOV DX, 0FF05H
MOV AL, 20              ;initial count for counter 1
OUT DX, AL

MOV DX, 0FF07H
MOV AL, 10110100B       ;counter 2, write LSB and MSB, mode 2
OUT DX, AL

MOV DX, 0FF06H
MOV AX, 2000            ; initial count for counter 2
OUT DX, AL
MOV AL, AH
OUT DX, AL
```


Next Lecture

- 8086 Interrupts