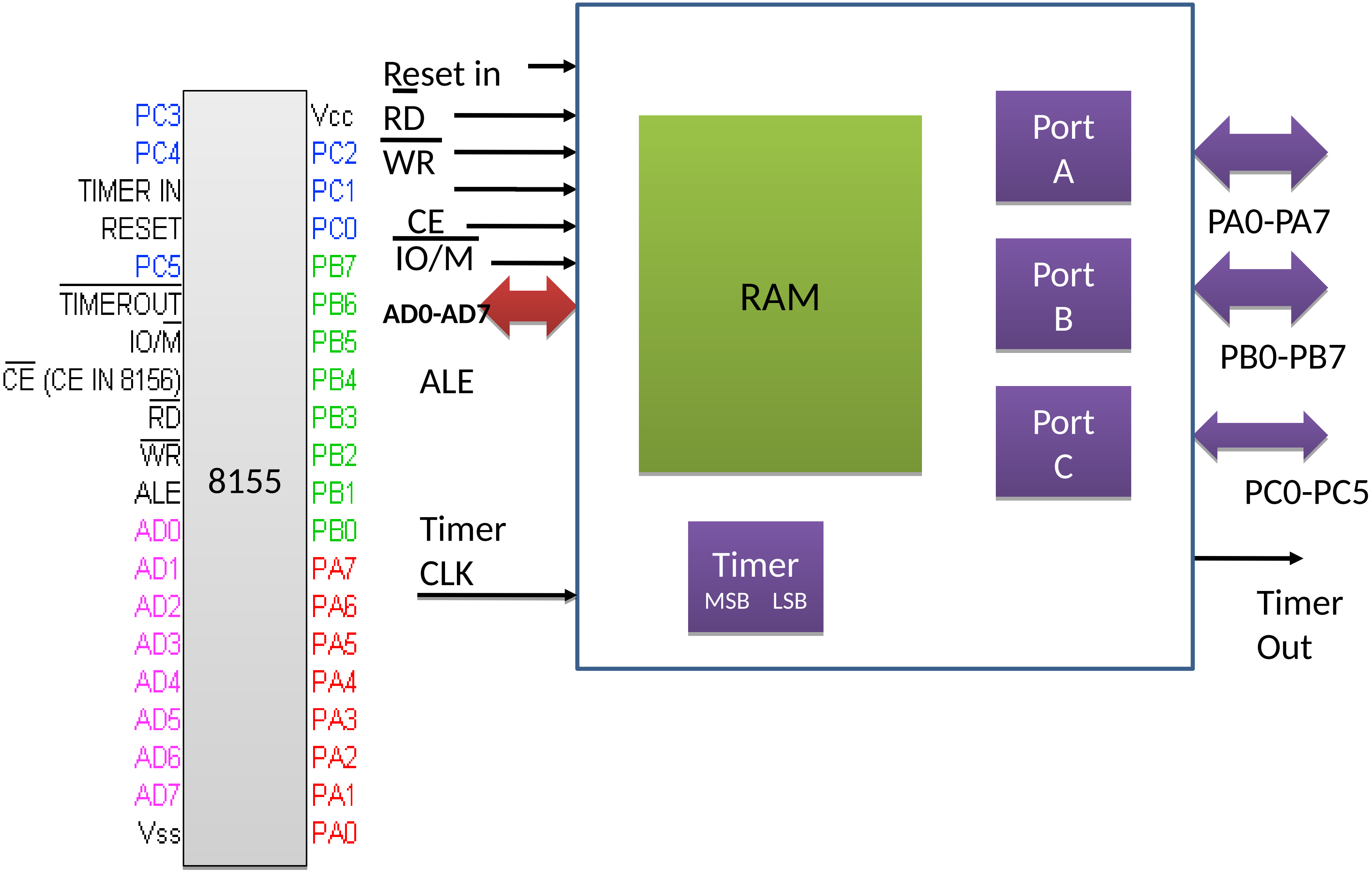
Peripheral Interface Device 8155 (I/O Interface & Timer)

Outline

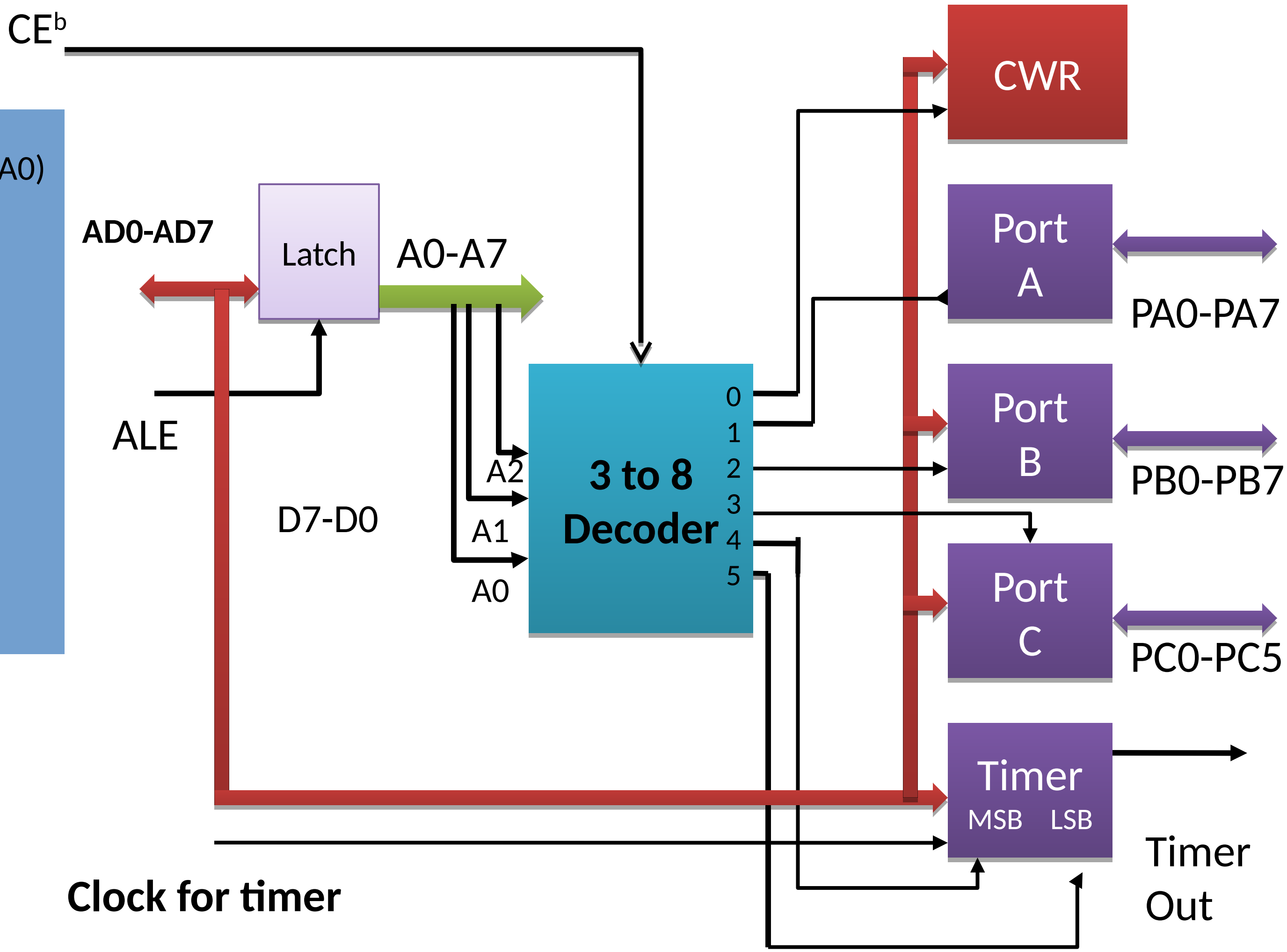
- Review
  - Programmable Interface device 8155
    - Block diagram, Address Calculation diagram
    - Interfacing LED using 8155
- 8155 Timer
  - Modes of timer
  - Square wave generation using 8155 interfaced timer
- 8155 Handshake & Interrupt mode
- Interfacing A/D Converter using Handshake mode using 8155

8155 Block Diagram

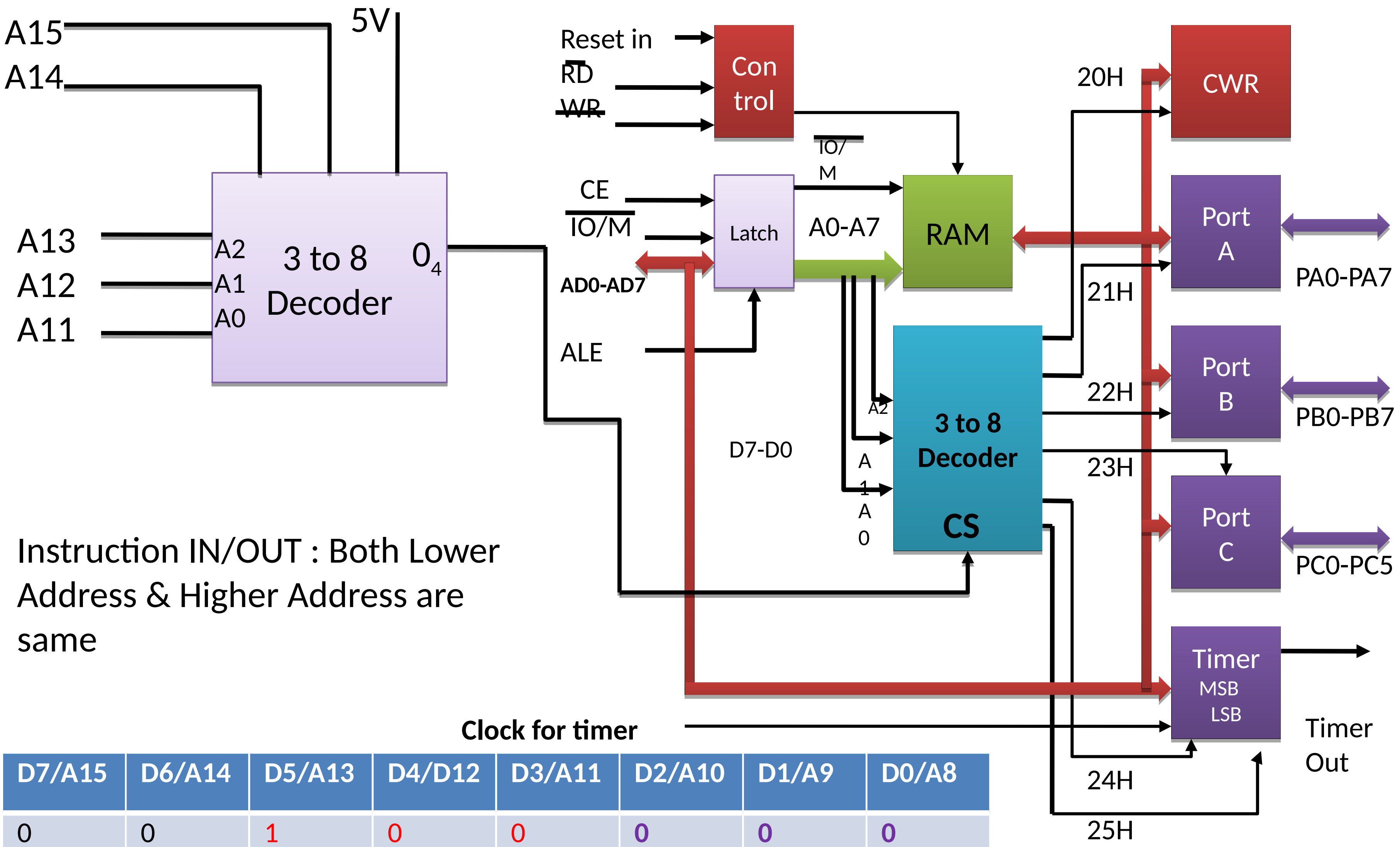


Expanded Block Diagram

A2	A1	A0	Port (ALE high, AD0=A0)
0	0	0	Command /Status Register
0	0	1	PA
0	1	0	PB
0	1	1	PC
1	0	0	Timer LSB
1	0	1	Timer MSB



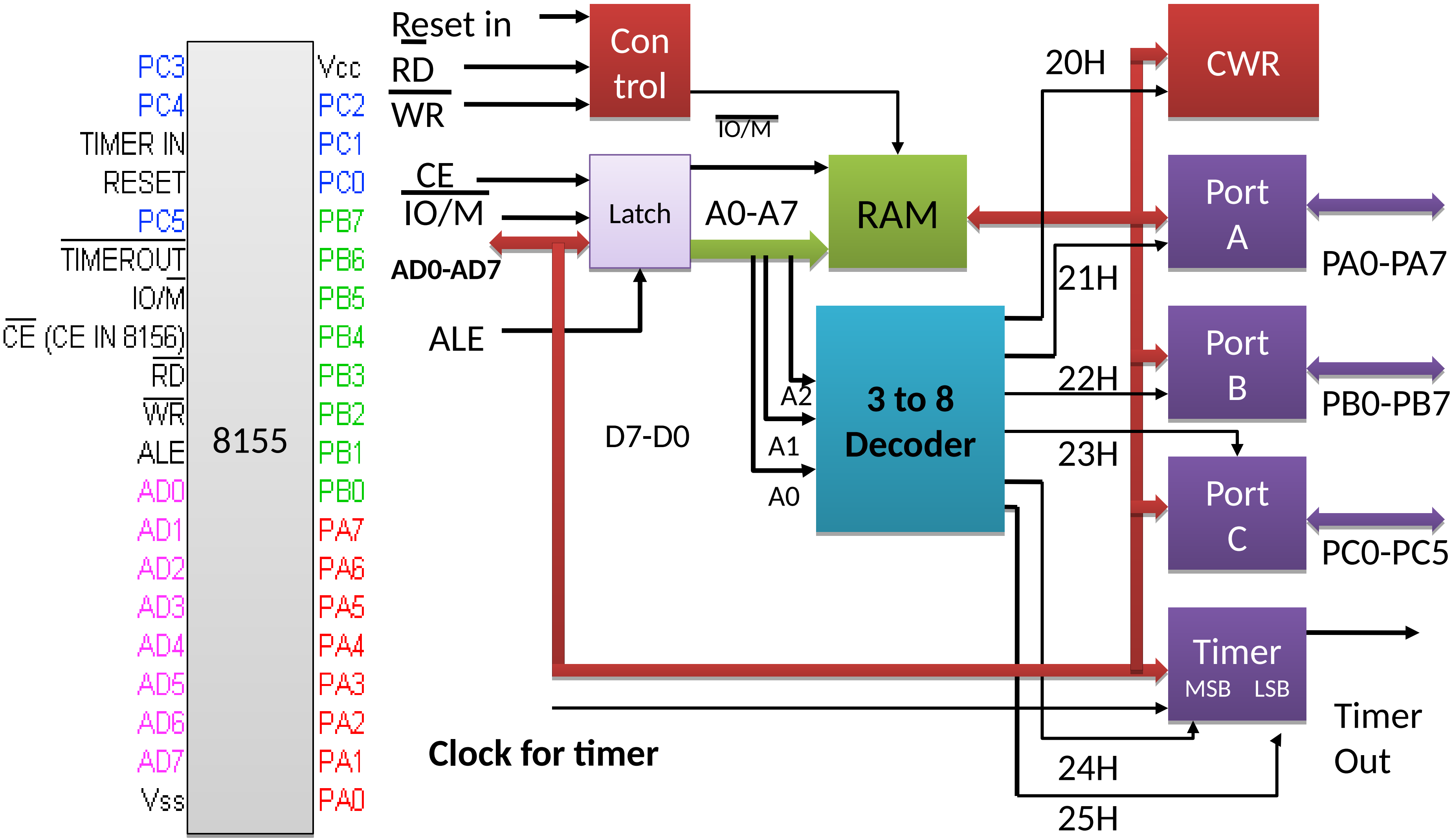
Calculate Address of Port of 8155



Instruction IN/OUT : Both Lower Address & Higher Address are same

D7/A15	D6/A14	D5/A13	D4/D12	D3/A11	D2/A10	D1/A9	D0/A8
0	0	1	0	0	0	0	0

8155 Block Diagram



Control word (command reg) format

D7	D6	D5	D4	D3	D2	D1	D0
Timer Command		IEB	IEA	PC		PB	PA

- D0, D1: mode for PA and PB, 0=IN, 1=OUT
- D2, D3: mode for PC
- D4, D5: interrupt EN for PA and PB, 0=disable 1=enable
- D6, D7: Timer command:
  - 00: No effect
  - 01: Stop if running else no effect
  - 10: Stop after terminal count (TC) if running, else no effect
  - 11: Start if not running, reload at TC if running.
- Port C bits
  (D2, D3)

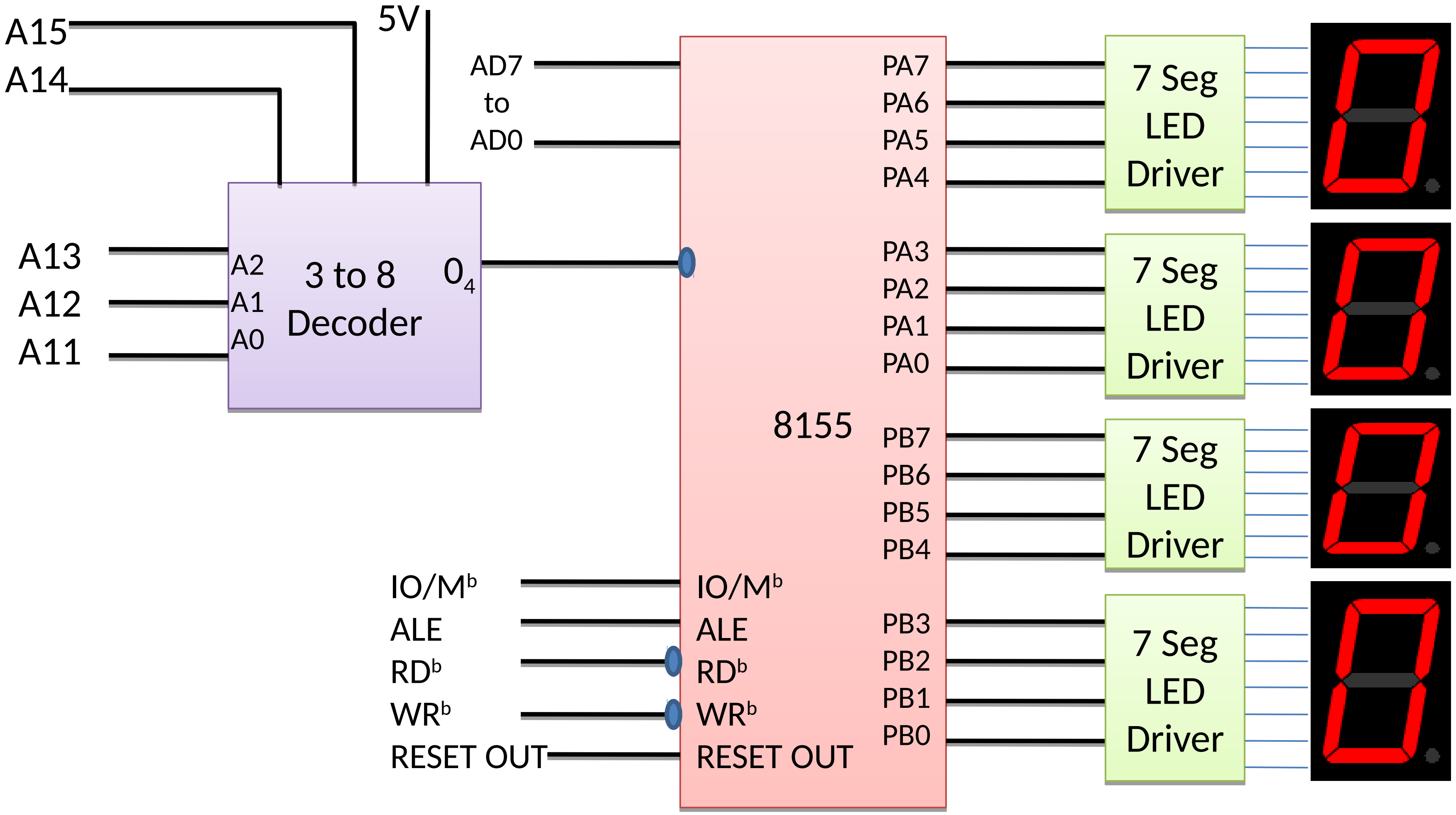
ALT	D3	D2	PC5	PC4	PC3	PC2	PC1	PC0
1	0	0	IN	IN	IN	IN	IN	IN
2	0	1	OUT	OUT	OUT	OUT	OUT	OUT
3	1	0	OUT	OUT	OUT	STB_A	BF_A	$INTR_A$
4	1	1	STB_B	BF_B	$INTR_B$	STB_A	BF_A	$INTR_A$

8155 Decode Registers

- Registers

A2	A1	A0	Port (ALE high, AD0=A0)
0	0	0	Command/Status Register
0	0	1	PA
0	1	0	PB
0	1	1	PC
1	0	0	Timer LSB
1	0	1	Timer MSB

Interfacing 7 Segment LEDs to output port using 8155



Interfacing LEDs Cntd..

- Port Address
  - Control Register=20H, Port A= 21H, Port B= 22H
- Control word:

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	1	1
Timer		Not Applicable		Use for Port C		Port B Output	Port A Output

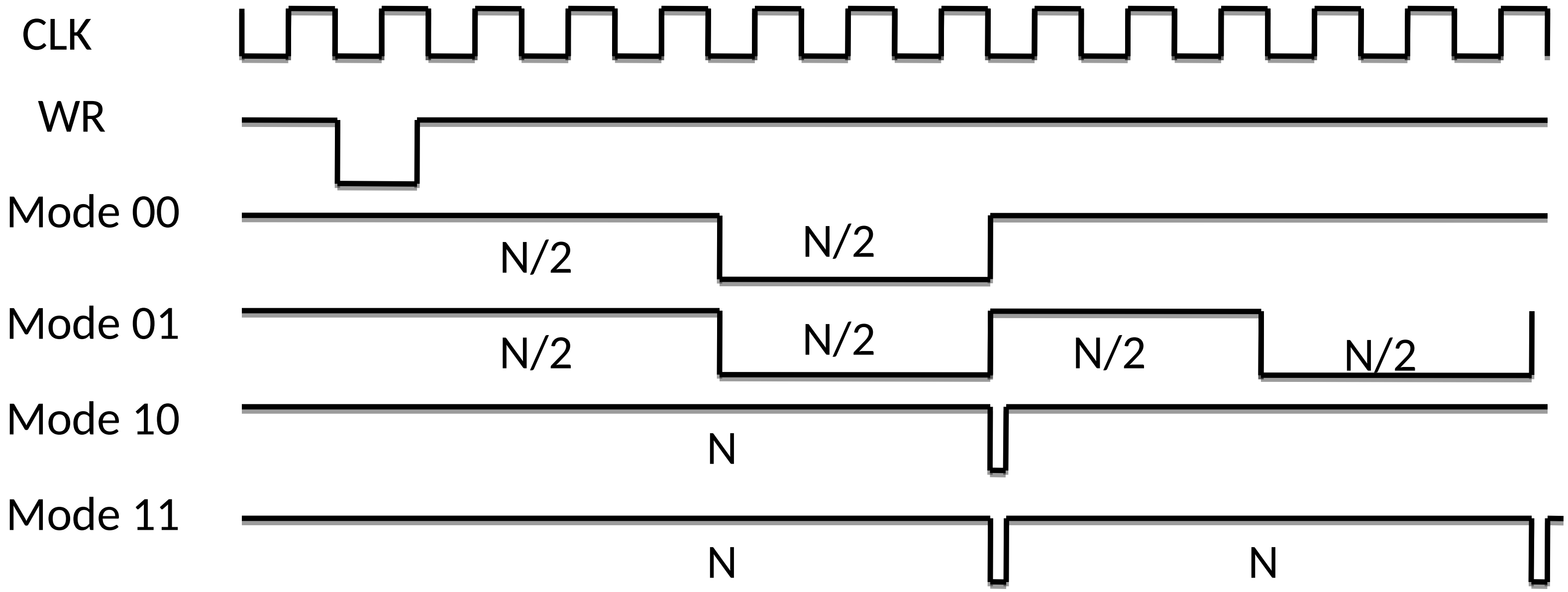
- Program
  - MVI A,03 ; initialize Port A &B for O/P
  - OUT 20H
  - MVI A, BYTE1 ; Display BYTE1 at port A
  - OUT 21H
  - MVI A, BYTE2 ; Display BYTE2 at port B
  - OUT 22H

8155: Timers

MSB							
M2	M1	T13	T12	T11	T10	T9	T8
LSB							
T7	T6	T5	T4	T3	T2	T1	T0

- M2, M1: mode bits:
  - 00: Single square wave of wavelength TC/2 (TC/2,TC/2 if TC even; [TC+1/2],[TC-1/2] if TC odd)
  - 01: Square waves of wavelength TC (TC/2,TC/2 if TC even; [TC+1/2],[TC-1/2] if TC odd)
  - 10: Single pulse *on* the TC'th clock pulse
  - 11: Single pulse *on* every TC'th clock pulse.

8155: Timer Modes Output



- 00: Single square wave of wavelength TC/2 (TC/2,TC/2 if TC even; [TC+1/2],[TC-1/2] if TC odd)
- 01: Square waves of wavelength TC (TC/2,TC/2 if TC even; [TC+1/2],[TC-1/2] if TC odd)
- 10: Single pulse *on* the TC'th clock pulse
- 11: Single pulse *on* every TC'th clock pulse.

Designing of Square Wave Generator Using 8155

- Design a square wave with pulse width 100µS
- Mode 1
- Clock Frequency 3 MHZ

Timer count: Pulse Period/Clock period

$= 200x10^{-6}/330x10^{-9} = 606$

= 25E H

= 02 (MSB), 5E (LSB)

Square Wave Generator Cntd..

- Timer port address: LSB 24H & MSB 25H
- Mode 1; M1=0, M2=1
  - M1 M2 T13 T12 T11 T10 T8 T7
  - 0 1 0 0 0 0 1 0 == (42H)
- Control word: (C0H)

D7	D6	D5	D4	D3	D2	D1	D0
Timer Command		IEB	IEA	PC		PB	PA
1	1	0	0	0	0	0	0

- Instructions to set counter & square wave generation

```
MVI       A , 5E      ;  LSB of count
OUT 24H ;  Load LSB  of  timer Register
MVI A, 42H      ;  MSB count with Mode 1
MVI 25H ;  Load  MSB  of timer Register
MVI A, C3H      ;  Load the control word for register
OUT 20H ;   Trigger the counter by loading to Ctrl word to ctrl Reg
```

Designing Interfacing Ckt Using 8155 to Read & Display from ADC to LEDs

- Set up Port A in the handshake mode to read data from A/D Converter
- Setup port B as output port to display data at seven segment LEDs
- Use line PC3 from port C to initiate a conversion
- Use the 8155 Timer to record conversation time

Control, Status & Timer

- Control word

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	1	1	0
Timer		NA (INTR not used)		Use for Port C		Port B OUT to LED	Port A IN from DAC

06H

CH=O/P
CL=Handshake
Mode

- Status word

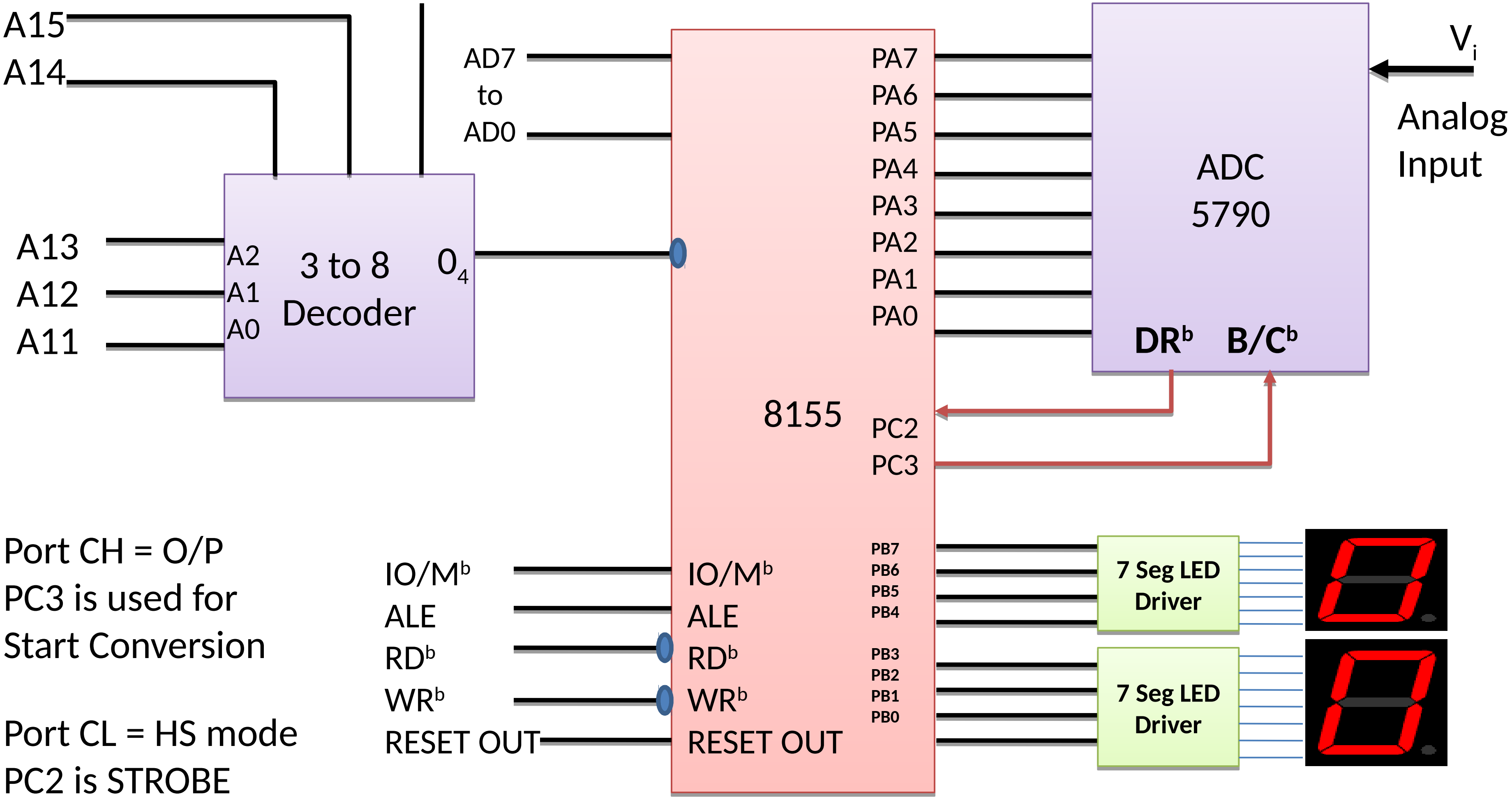
D7	D6	D5	D4	D3	D2	D1	D0
No	X	X	X	X	X	BFa	X
USe	Timer	INTEb	BFb	INTRb	INTAa	BFa	INTRa

Read the Data
Mask with 02H

- Timer (with ef ent)
  - Start timer
  - Stop timer

1	1	0	0	0	1	1	0	**C6H**
0	1	0	0	0	1	1	0	**46H**

Interface Diagram



Port CH = O/P
PC3 is used for
Start Conversion

Port CL = HS mode
PC2 is STROBE

Interfacing Program

```
  MVI    A,06H     ;Control world for I/O port
  OUT    20H  ; Set up port as specified
  MVI    A,00H     ; Load 0000H in Timer Reg
 OUT     24H
 OUT     25H

 MVI     A,08H     ;Byte to set PC3=1
 OUT     23H  ;Send start pulse
  MVI    A,C6H     ;Control word to start timer
  MVI    20H  ; Start timer
  MVI    A,00H     ;Byte to set PC3=0
  OUT    23H  ;Start conversion

ST:IN  20H      ; Read Status Register
   ANI     02H ;Check Status of DR$^{b}$
   JZ   ST        ; If BF$_{a}$=0 wait in
```

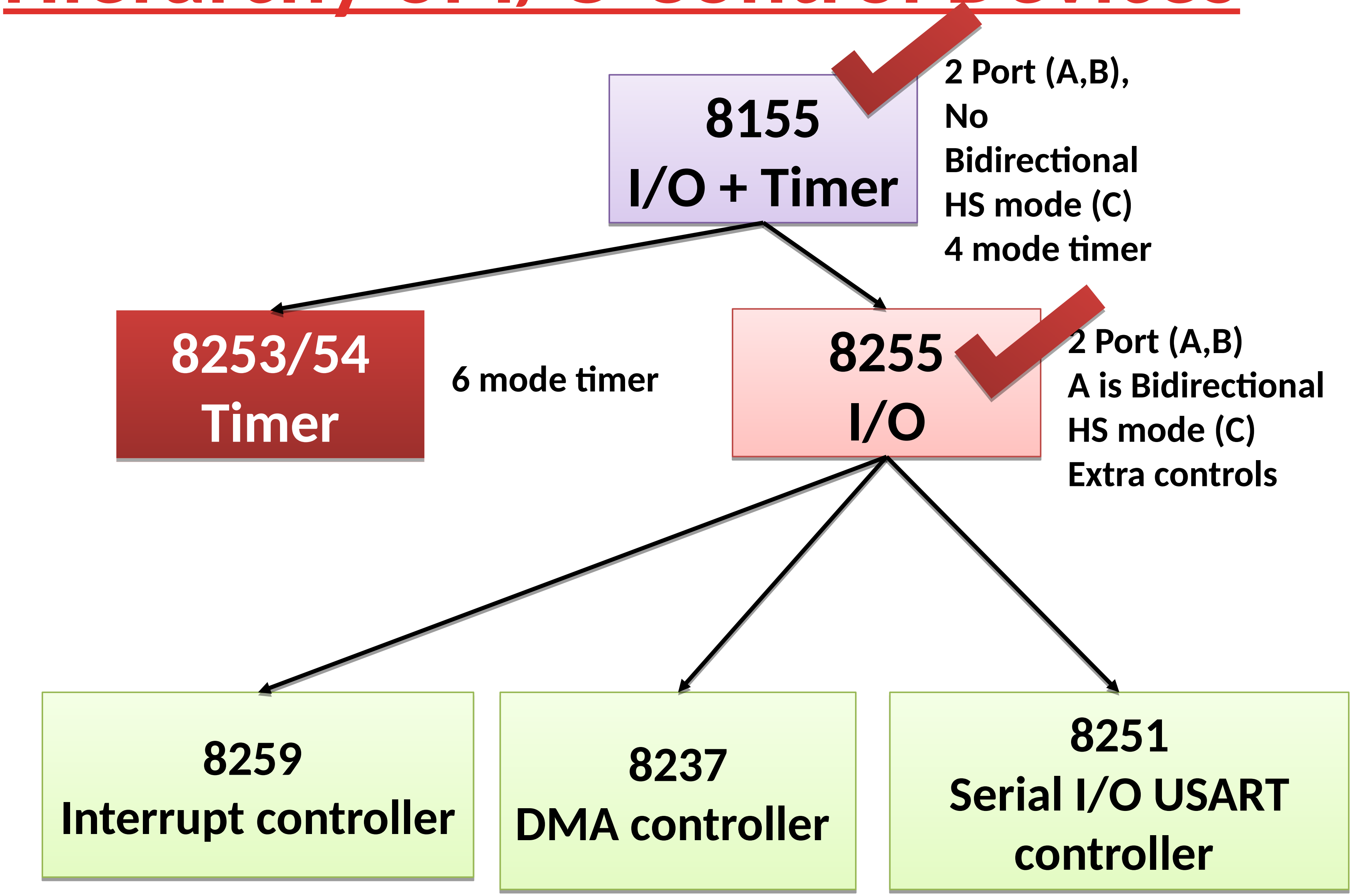
```
MVI  A,46H    ; Byte to stop Counter
OUT  20H      ;Stop Counter

IN     21H       ; Read A/D output
OUT  22H   ;Display data at port B (LEDs)


INT   24H    ; Read LSB of Timer
MOV  L,A
INT   25H     ; Read MSB of Timer
ANI   3FH     ;Mask the mode Bit D6,D7
MOV  H,A           ; Save MSB timer count in
H
LHLD  RWM  ; Store the count at
             ;Memory location RWM
HLT
```

8253/4 Programmable Interval Timer

Hierarchy of I/O Control Devices



8155
I/O + Timer

2 Port (A,B),
No
Bidirectional
HS mode (C)
4 mode timer

8253/54
Timer

6 mode timer

8255
I/O

2 Port (A,B)
A is Bidirectional
HS mode (C)
Extra controls

8259
Interrupt controller

8237
DMA controller

8251
Serial I/O USART controller

Outline

- Basic Difference of 8155 I/O timer Vs 8254
- 8254 Brief
- Architecture of 8254
- Control register
- Status register
- Modes of Counters with example
- Read-Back modes

8254 Description

- The 82C54 solves one of the most common problems in any microcomputer system, the generation of accurate time delays under software control.
- Instead of setting up timing loops in software, the programmer configures the 82C54 to match his requirements and programs one of the counters for the desired delay. After the desired delay, the 82C54 will interrupt the CPU. Software overhead is minimal and variable length delays can easily be accommodated

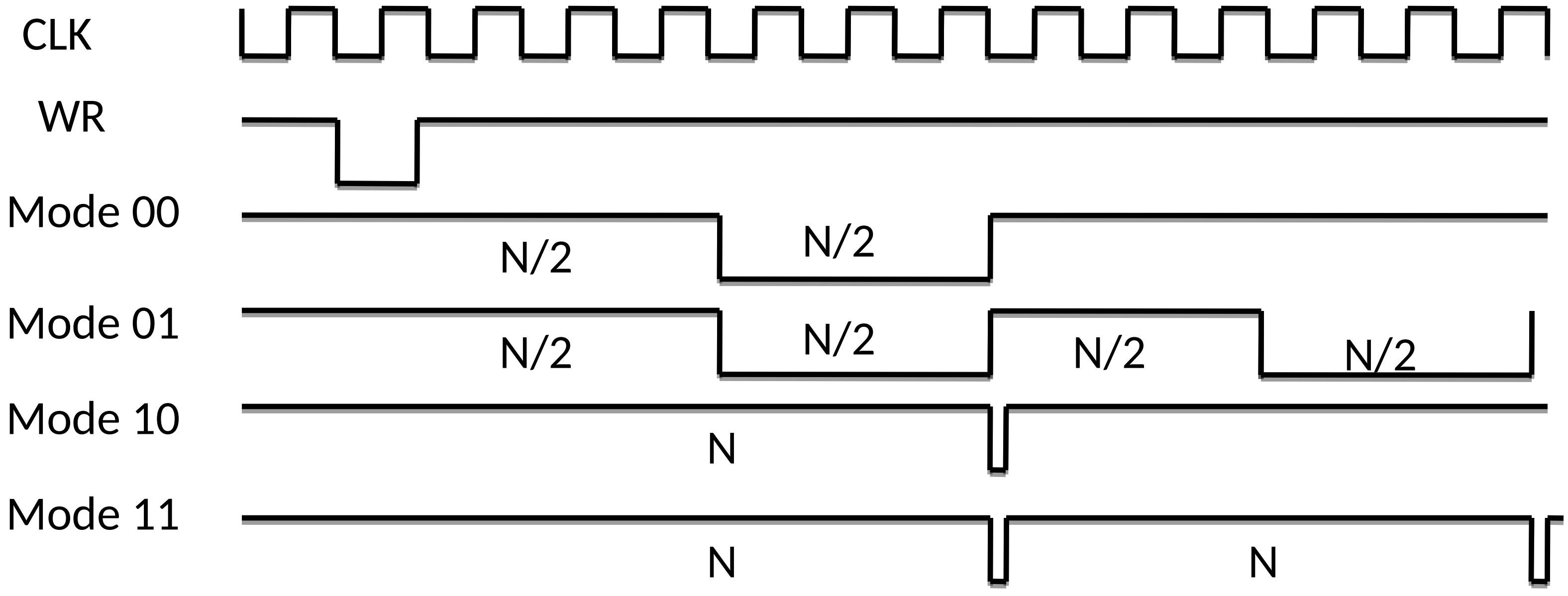
8254 Description

- Some of the other counter/timer functions common to microcomputers which can be implemented with the 82C54 are:
  - Real time clock
  - Event counter
  - Digital one-shot
  - Programmable rate generator
  - Square wave generator
  - Binary rate multiplier
  - Complex waveform generator
  - Complex motor controller

8254 Chip

- Main function:
  - Dividing clock frequency
- Three independent 16-bit counters
- Models
  - 8253: 2 MHz
  - 8254: 8 MHz
  - 8254-2: 10 MHz

8155: Timer Modes Output



- 00: Single square wave of wavelength TC/2 (TC/2,TC/2 if TC even; [TC+1/2],[TC-1/2] if TC odd)
- 01: Square waves of wavelength TC (TC/2,TC/2 if TC even; [TC+1/2],[TC-1/2] if TC odd)
- 10: Single pulse *on* the TC'th clock pulse
- 11: Single pulse *on* every TC'th clock pulse.

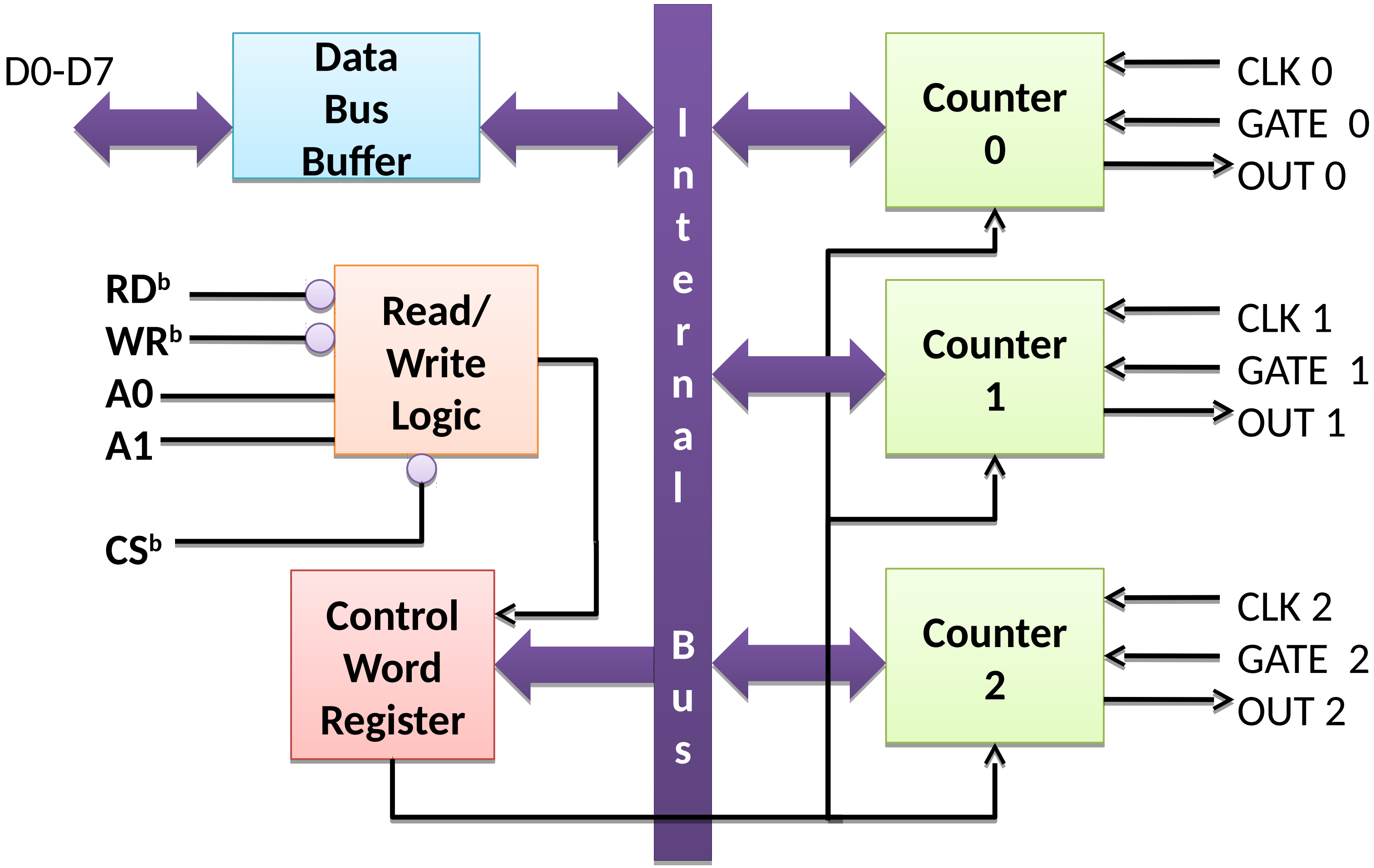
8254: Brief

- Three independent 16-bit programmable counters (timers).
- It generates accurate time delays and can be used for
  - Real time clock, an event Ctr, a digital one shot, a square wave gen, complex wave gen.
- Programmable and work DC to 8 MHz
- 5 different modes of operation

The 8254 PIT

- The 8254 Programmable Interval-timer is used by the PC system for (1) generating timer-tick interrupts (rate is 18.2 per sec), (2) performing dynamic memory-refresh (reads ram once every 15 microseconds), and (3) generates ‘beeps’ of PC speaker
- When the speaker-function isn’t needed, the 8254 is available for other purposes

8254 Block Diagram



Control Logic

- RD^b, WR^b, CS^b
- A0, A1: Selection of Counter and Control Register
- Suppose Address is (**80H,81H,82H,83H**) with interfacing Circuit

A1	A0	Selection
0	0	Counter 0
0	1	Counter 1
1	0	Counter 2
1	1	Control Register

Control Register

D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC2	RW1	RW0	M2	M1	M0	BCD
Select Counter 00: Counter 0 01: Counter 1 10: Counter 2 11: Read-Back Command		Read Write 00: Counter latch Command 01:RW LSByte only 10: RW MSByte only 11:RW LSByte first then Msbyte		000 : Mode 0 001 :Mode 1 X10 :Mode 2 X11 :Mode 3 100 :Mode 4 101 : Mode 5			0/1 = Binary / BCD Mode

Programming Counters

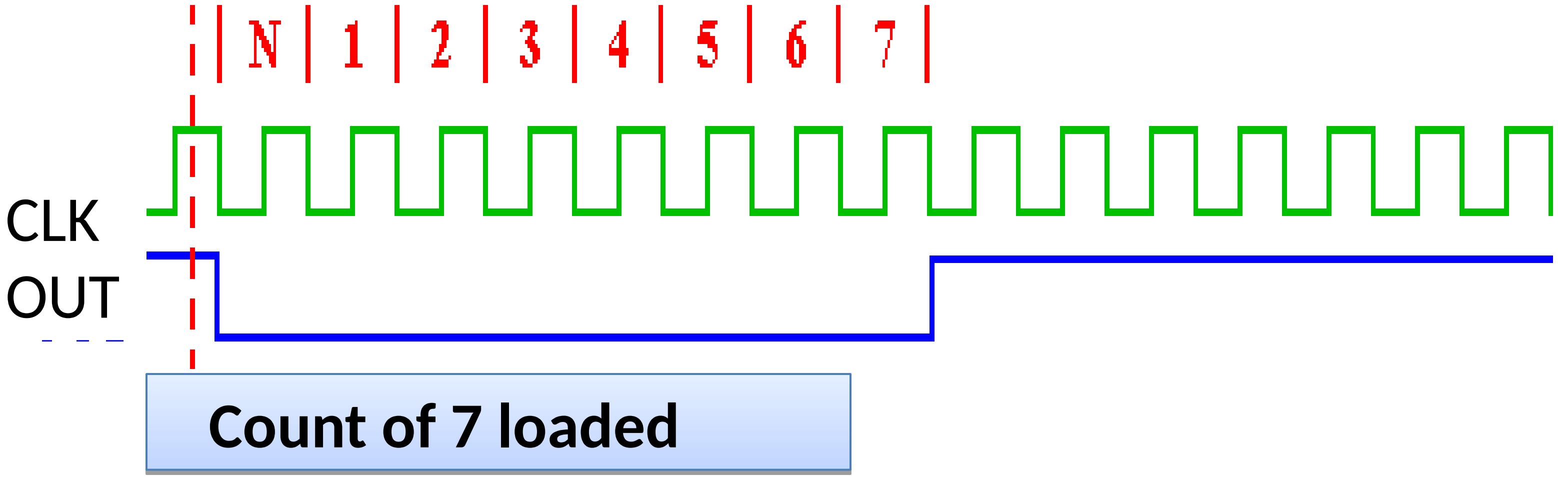
- Each counter may be programmed with a count of 1 to FFFFH.
  - Minimum count is 1 all modes except 2 and 3 with minimum count of 2.
- Each counter has a program control word used to select the way the counter operates.
  - If two bytes are programmed, then the first byte (LSB) stops the count, and the second byte (MSB) starts the counter with the new count.

Modes of 8254 Counter

- Mode 0 : Interrupt on Terminal count
- Mode 1 : Hardware Retriggerable One Shot
- Mode 2 : Rate Generator
- Mode 3 : Square wave generator
- Mode 4 : Software Triggered Strobe
- Mode 5 : Hardware Triggered Strobe

Mode 0: Interrupt on Terminal Count

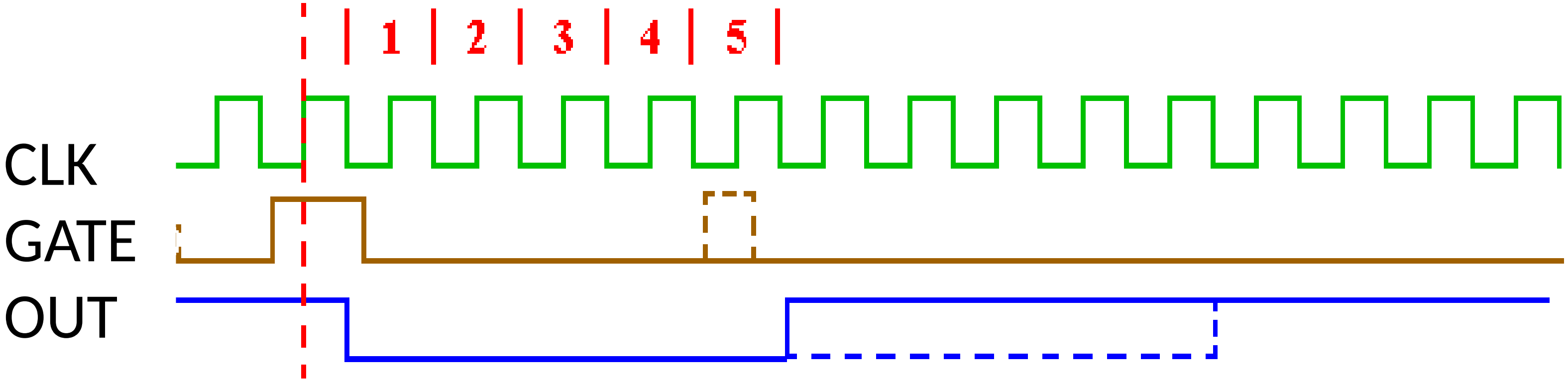
- The output becomes a logic 0 when the control word is written and remains there until N plus the number of programmed counts.



Count of 7 loaded

Mode 1: Hardware Retriggerable One Shot

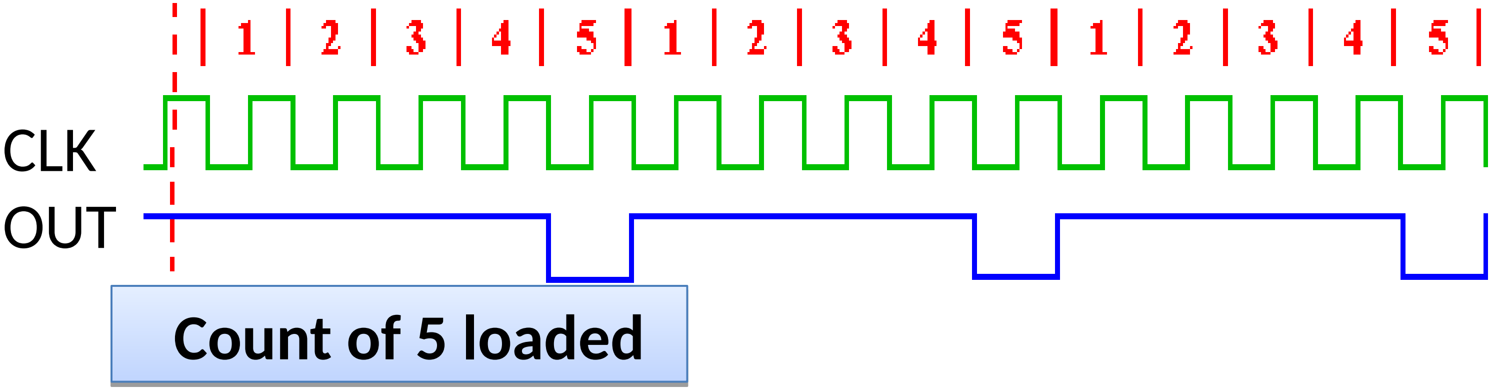
- The G input triggers the counter to output a 0 pulse for `count' clocks.
- Counter reloaded if G is pulsed again.



Triggered with count of 5

Mode 2: Rate Generator

- Counter generates a series of pulses 1 clock pulse wide.
- The separation between pulses is determined by the count.
- The cycle is repeated until reprogrammed or G pin set to 0.



Count of 5 loaded

Write instruction to generate pulse every 50mcroS from Ctr0

- Control word = **14H**
  - D7D6=00 Select ctr 0
  - D5D4=01 load 8 bit count
  - D3D2D1=010 mode 2
  - D0=0 Binary
- Count = $50x10^{-6}/0.5x10^{-6}$=64H

PULSE: MVI A 14H ; Control word

OUT CTRAdd 83H

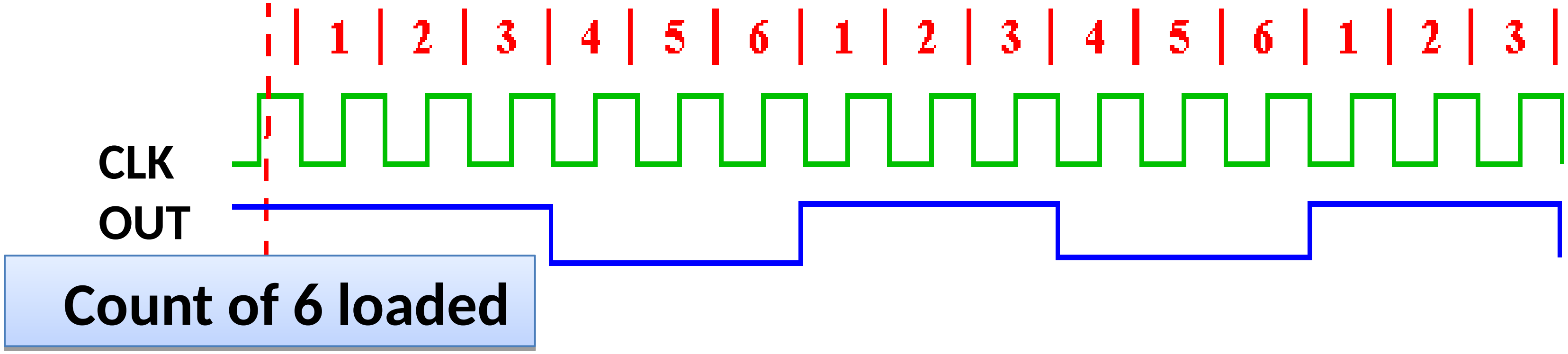
MVI A,64H ;Count value

OUT 80H ; load counter 0 with low order byte

HALT

Mode 3: Square wave generator

- Generates a continuous square-wave with G set to 1.
- If count is even, 50% duty cycle otherwise OUT is high 1 cycle longer



Count of 6 loaded

Write instruction for 1KhZ square wave at Ctr 1

- Control word = **76H**

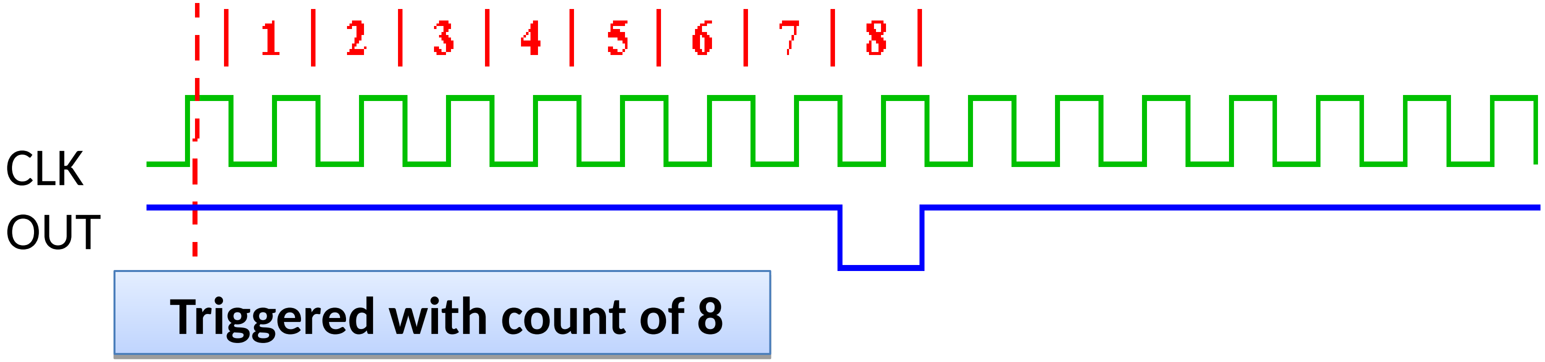
D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC2	RW1	RW0	M2	M1	M0	BCD
01		Load 16 bit (11)		011 (mode 3)			0

- Count= $1x10^{-3}/0.5x10^{-6}$=2000=07D0H
- Instructions

```
MVI   A,76H ; load Control word for Ctr 1 mode 3
OUT   83H    ; write to Ctrl reg
MVI   A, D0H; lower order byte cnt
OUT   81H
MVI   A,07H ; higher order byte
OUT   81H
HLT
```

Mode 4: Software Triggered Strobe

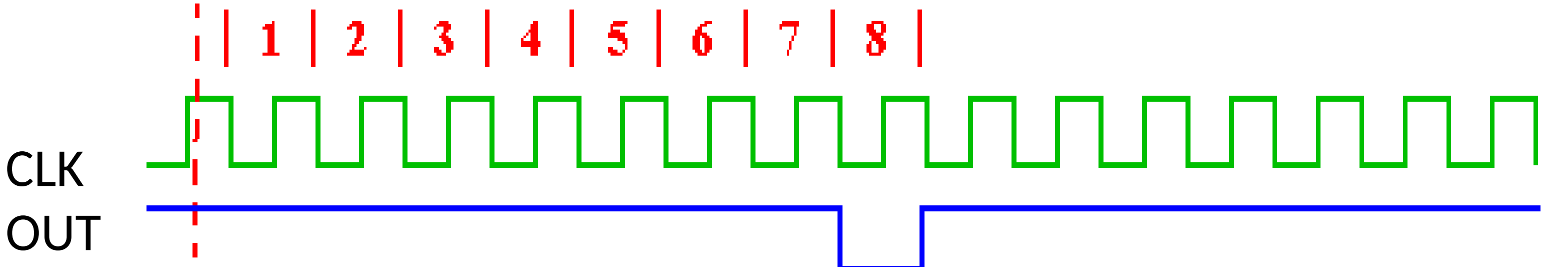
- Software triggered one-shot (**G must be 1**).
- OUT goes initially High, it goes low for one clock at the end of count
- The count must be reloaded for subsequent output



Triggered with count of 8

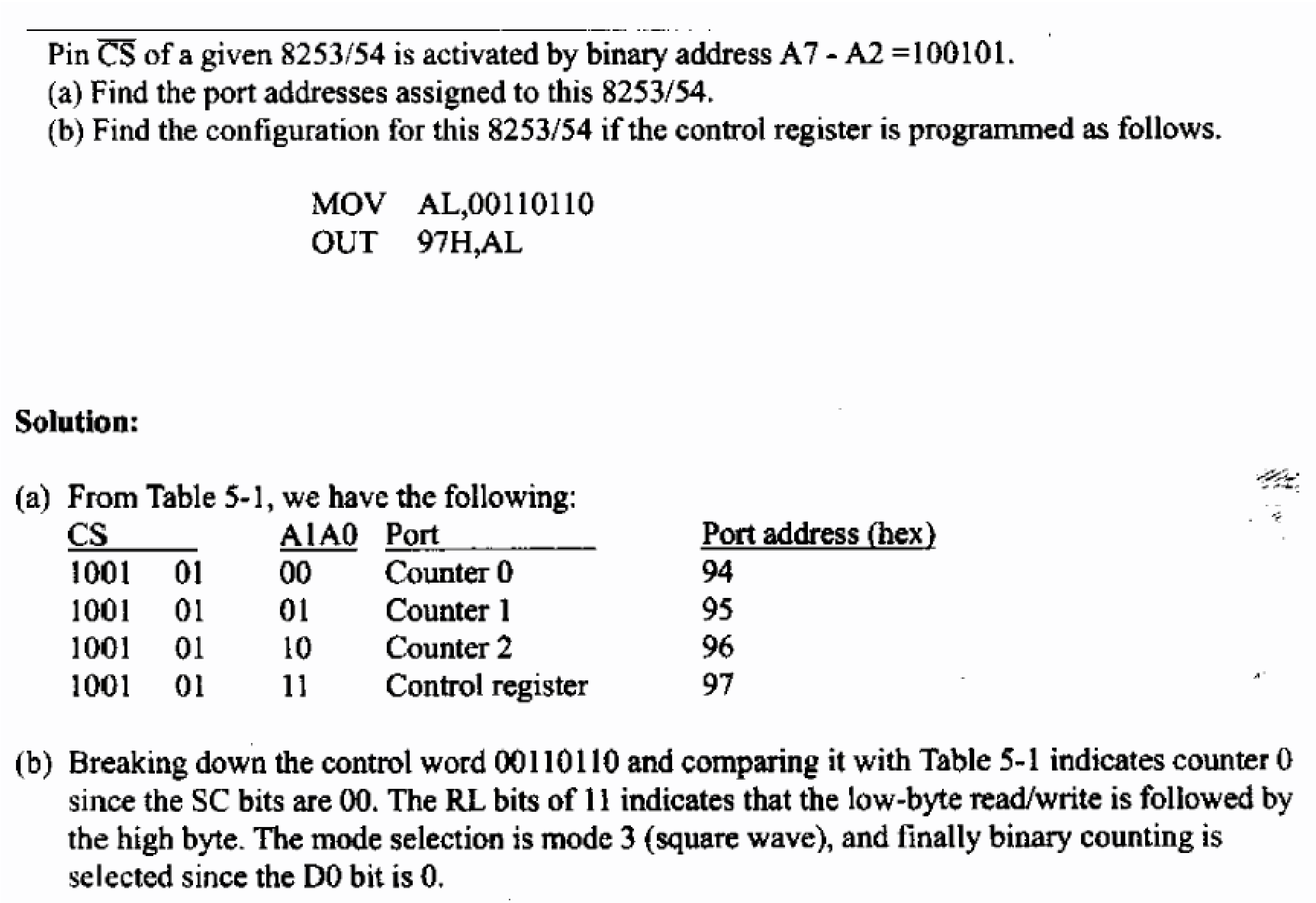
Mode 5: Hardware Triggered Strobe

- Hardware triggered is one-shot
- It is triggered by rising edge at the **Gate**
- Initially the OUT is low and Gate triggered from low to high the count begins
- OUT goes low for one clock period



H/W trigger with count of 8

Example



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

(a) Find the port addresses assigned to this 8253/54.

(b) Find the configuration for this 8253/54 if the control register is programmed as follows.

```
MOV   AL,00110110
OUT   97H,AL
```

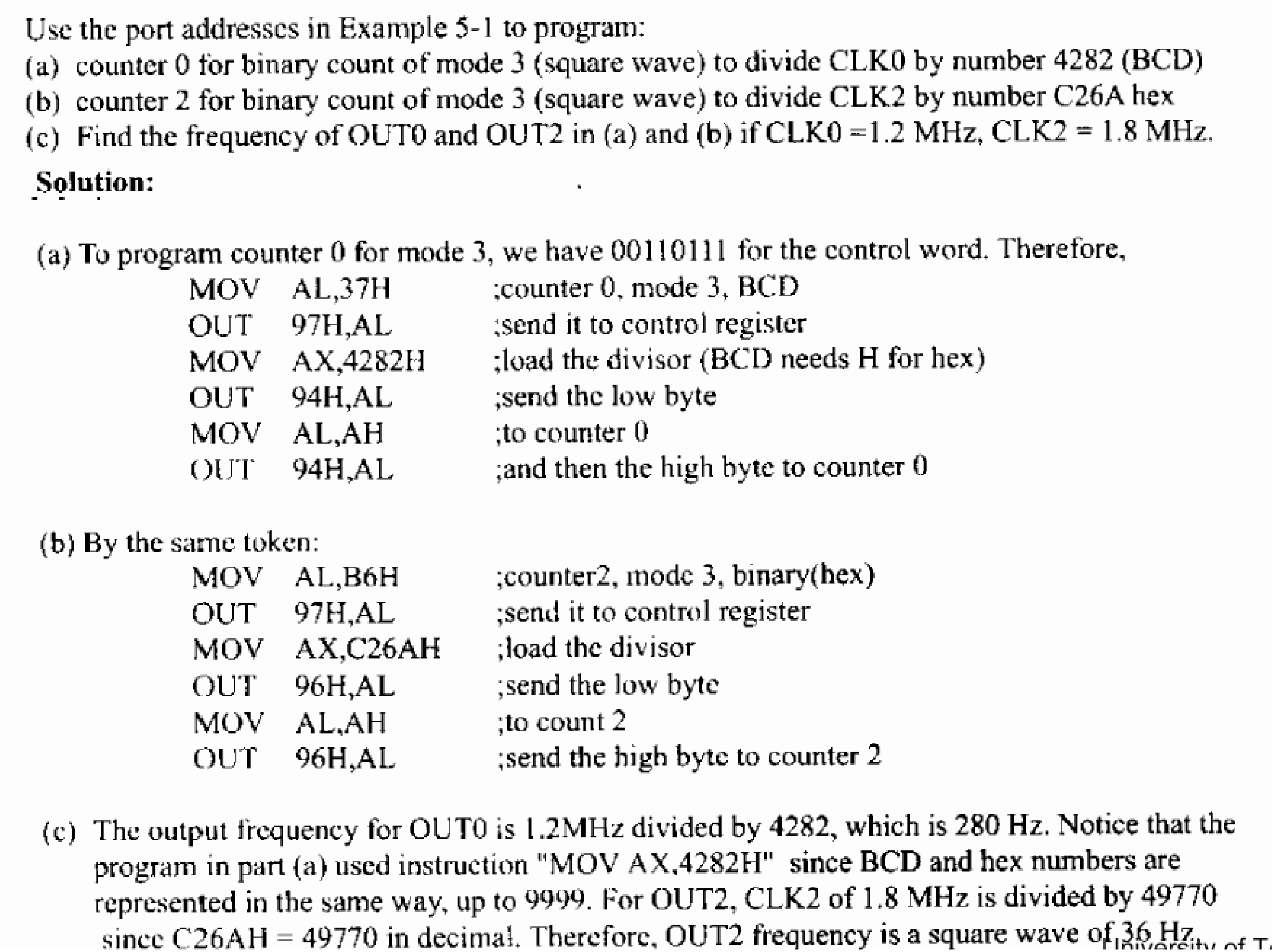
Solution:

(a) From Table 5-1, we have the following:

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

(b) Breaking down the control word 00110110 and comparing it with Table 5-1 indicates counter 0 since the SC bits are 00. The RL bits of 11 indicates that the low-byte read/write is followed by the high byte. The mode selection is mode 3 (square wave), and finally binary counting is selected since the D0 bit is 0.

Example 2



Use the port addresses in Example 5-1 to program:

(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:

(a) To program counter 0 for mode 3, we have 00110111 for the control word. Therefore,

```
MOV   AL,37H       ;counter 0, mode 3, BCD
OUT   97H,AL       ;send it to control register
MOV   AX,4282H     ;load the divisor (BCD needs H for hex)
OUT   94H,AL       ;send the low byte
MOV   AL,AH        ;to counter 0
OUT   94H,AL       ;and then the high byte to counter 0
```

(b) By the same token:

```
MOV   AL,B6H       ;counter2, mode 3, binary(hex)
OUT   97H,AL       ;send it to control register
MOV   AX,C26AH     ;load the divisor
OUT   96H,AL       ;send the low byte
MOV   AL,AH        ;to count 2
OUT   96H,AL       ;send the high byte to counter 2
```

(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.

8253 Decoding in PC

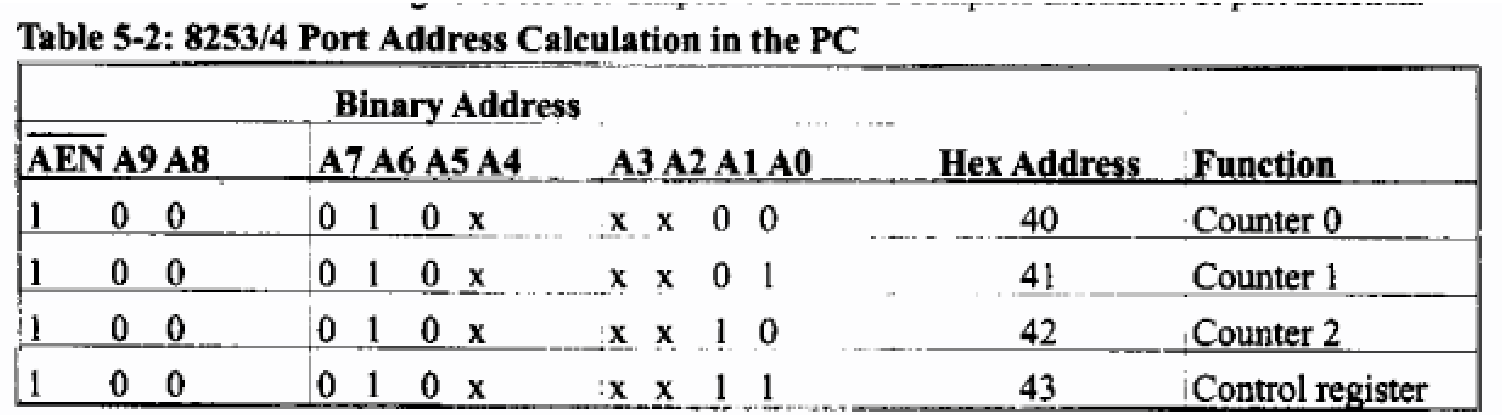


Table 5-2: 8253/4 Port Address Calculation in the PC

Binary Address				
$\overline{AEN}$ A9 A8	A7 A6 A5 A4	A3 A2 A1 A0	Hex Address	Function
1 0 0	0 1 0 x	x x 0 0	40	Counter 0
1 0 0	0 1 0 x	x x 0 1	41	Counter 1
1 0 0	0 1 0 x	x x 1 0	42	Counter 2
1 0 0	0 1 0 x	x x 1 1	43	Control register

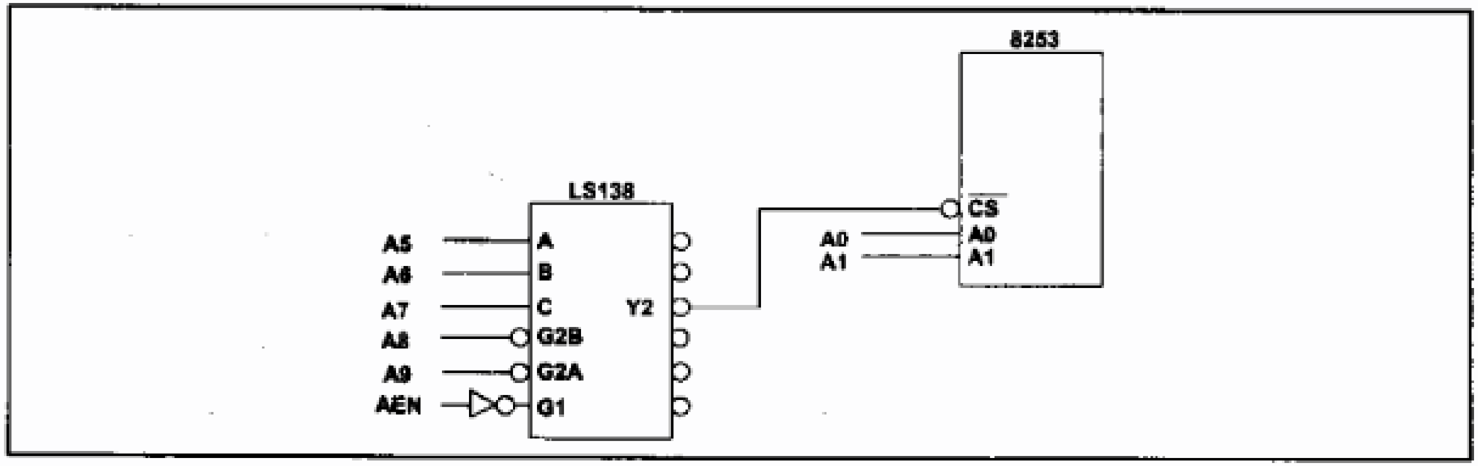


Figure 5-3. 8253 Port Selection in the PC/XT

PC Board

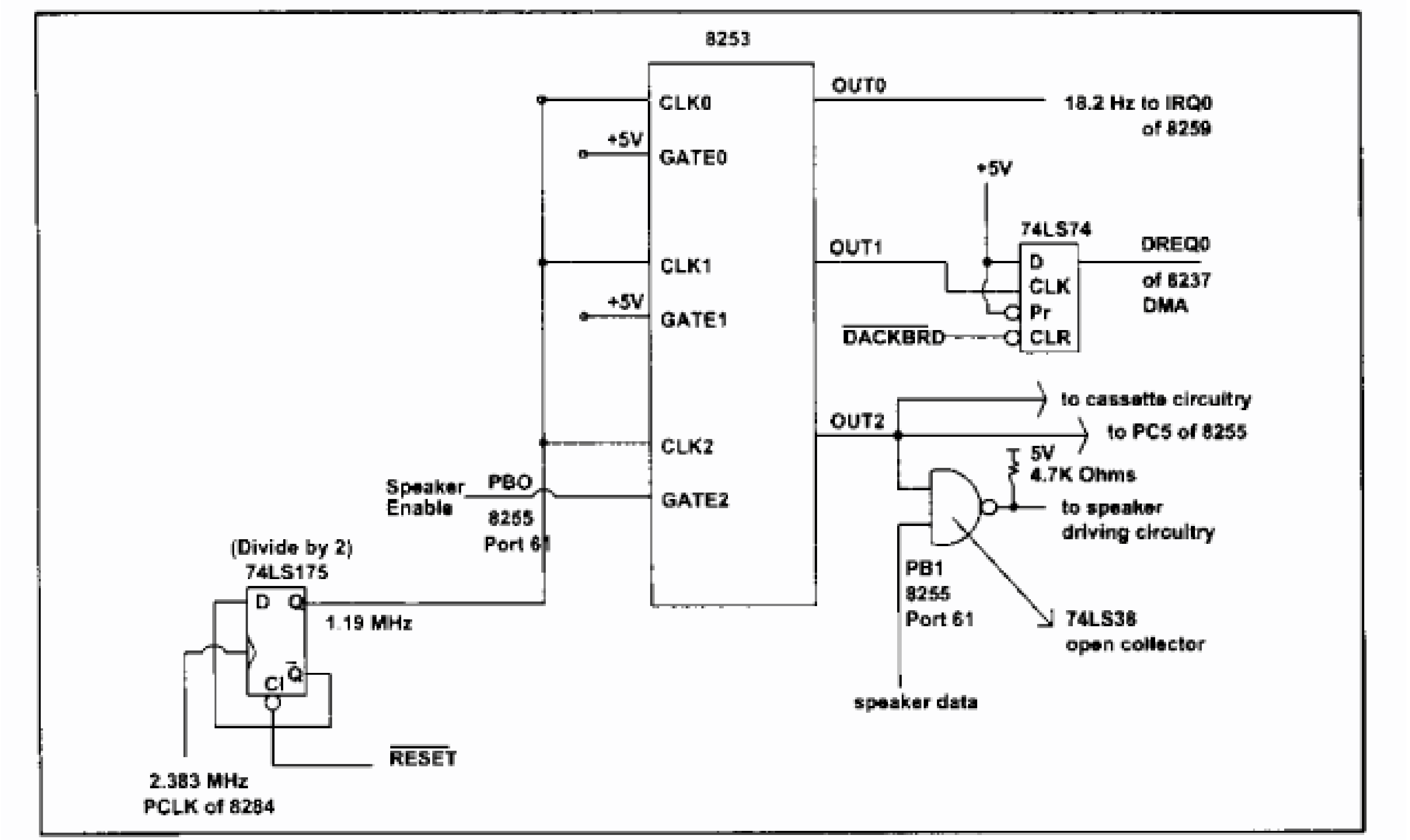


Figure 5-4. 8253 Chip Connections in the PC

Write a pgm to generate an interrupt every 1 Second

- Assume Clock Freq=2MhZ
- Count is too large
- Counter 1 load with 50,000 to generate 25ms
  - CNTLOAD=$50{,}000_{10}$=C350H
- Counter 2 load with 40 to generate 25msX40=-1S pulse (CNTLOAD=40_{10}=28H)
- Counter1 input is to counter 2
- Both Counter 1 & Counter 2 in Mode 2

Instruction to set up 1s interrupt

```
MVI A , 74H  ; Mode for 1st CTR
OUT 83H   ;Write in control register
MVI A,94H    ; Mode for 2nd CTR
OUT 83H   ; Write to control register
MVI A,50  ; low byte of CTR1=C350
OUT 81H   ; load to CTR1 low byte
MVI A,C3 ;  high byte of CTR1=C350
OUT 81H   ; load to CTR1 high byte
MVI A,28H    ; Count for Counter 2
OUT 82H   ; Load Counter 2
RET
```

Gate Setting of Counter

Modes	Low or Going Low	Rising	High
Mode 0	Disable Counting	-	Enable Counting
Mode 1	-----	1. Initiate Counting 2. Reset O/P after next Clock	---
	1. Disable counting 2. Set O/P immediately high	1. Reloads Counter 2. Initiate Counting	Enable Counting
	1. Disable counting 2. Set O/P immediately high	Initiates Counting	Enable Counting
	Disable Counting		Enable Counting
		Initiates Counting	

Read-Back Command

- This allow user to read the count and status of the counter
- Command Written in control register and count of the specified counter can be latched

D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC2	COUNT^{b}	STATUS^{b}	CNT2	CNT1	CNT0	0
11: Read-Back Command		If (D5=0) count is latecheded		D3=1 select counter 2 D2=1 select counter 1 D1=1 select counter 0			

word will latch the count of CNT0 & CNT1

Read-Back Command

- Status can be read if STATUS^{b} bit D4 =0
- D7=1 : Outpin is 1, 0 Outpin is 0
- D6=1: Null count, D6: 0= Count available for reading
- D5-D0:Counter Programmed mode

D7	D6	D5	D4	D3	D2	D1	D0
OUTPUT	NULL COUNT	RW1	RW0	M2	M1	M0	BCD

Control word

- Counter 1 (74H)

D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC2	RW1	RW0	M2	M1	M0	BCD
01		Load 16 bit (11)		010 (mode 2)			0

- Counter 2 (94H)

D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC2	RW1	RW0	M2	M1	M0	BCD
10		Load 8 bit (01)		010 (mode 2)			0

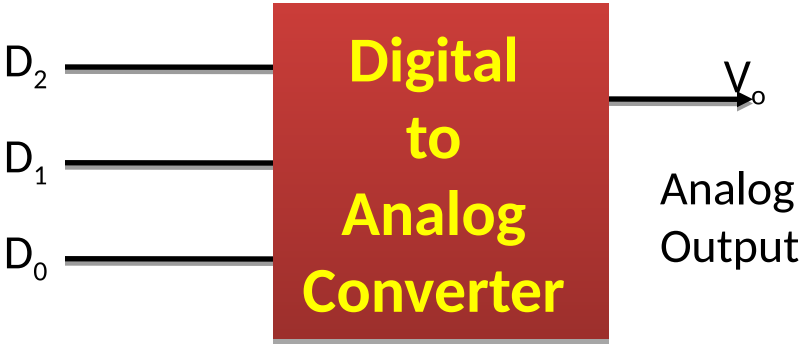
Outline

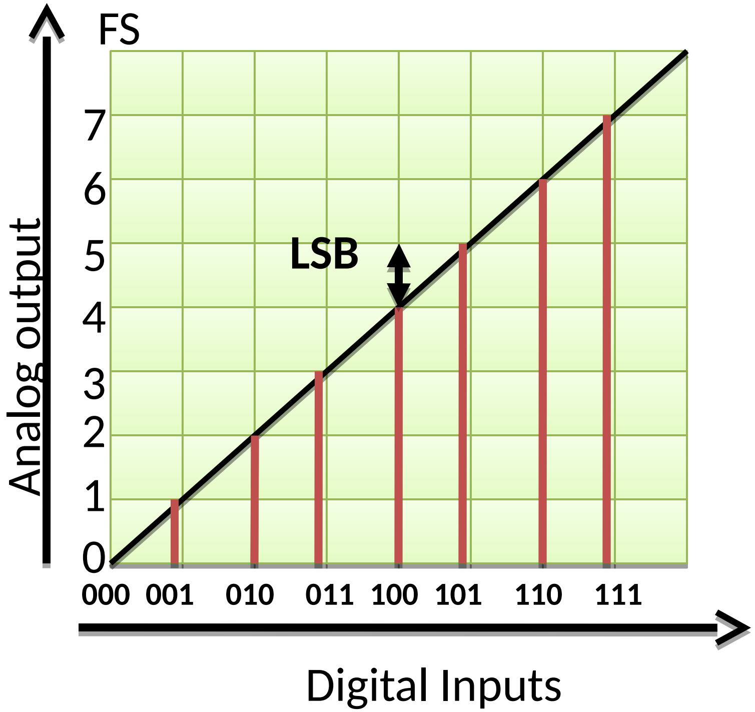
- Peripheral communications
- DAC
  - Properties
  - Generic Model
  - Interfacing
- ADC
  - Properties
  - Generic Model
  - Interfacing
- Display

Digital to Analog Converter

- Used for play sound in speaker
- Used by AC97 (Audio codec)
- MP3 Sound store digital format in HDD
- Slow as compared to processor/MPU
- Parameters
  - Resolution (8 bit/16 bit)
  - Settling time (1micro sec)

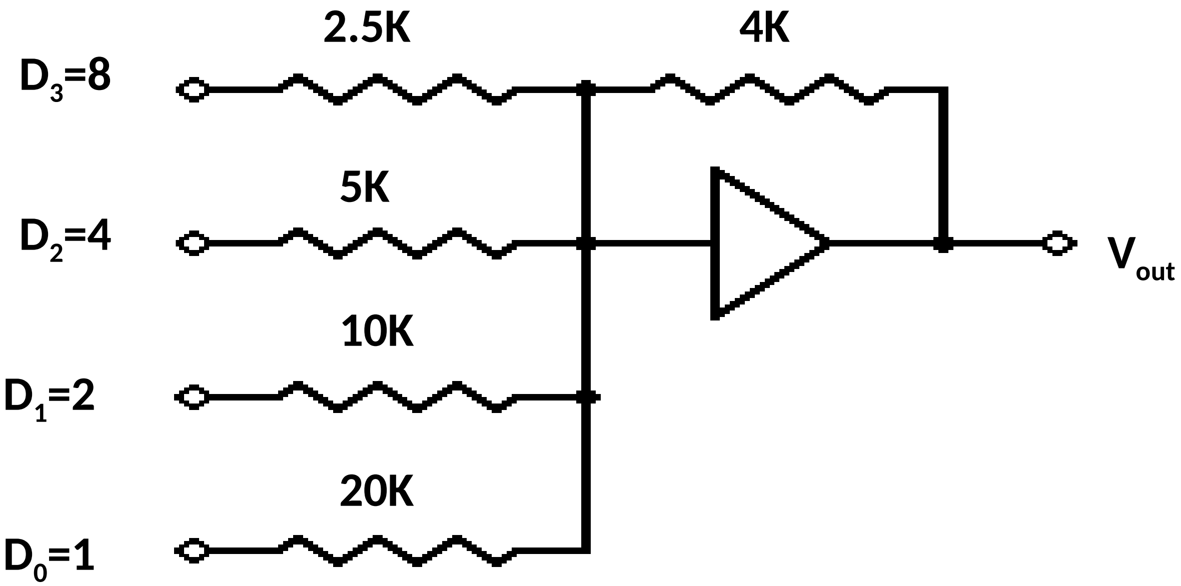
D/A converter





- FullScaleOutput=(FullScaleValue – 1LSBValue)
- 1MSB Value=1/2 * FSV

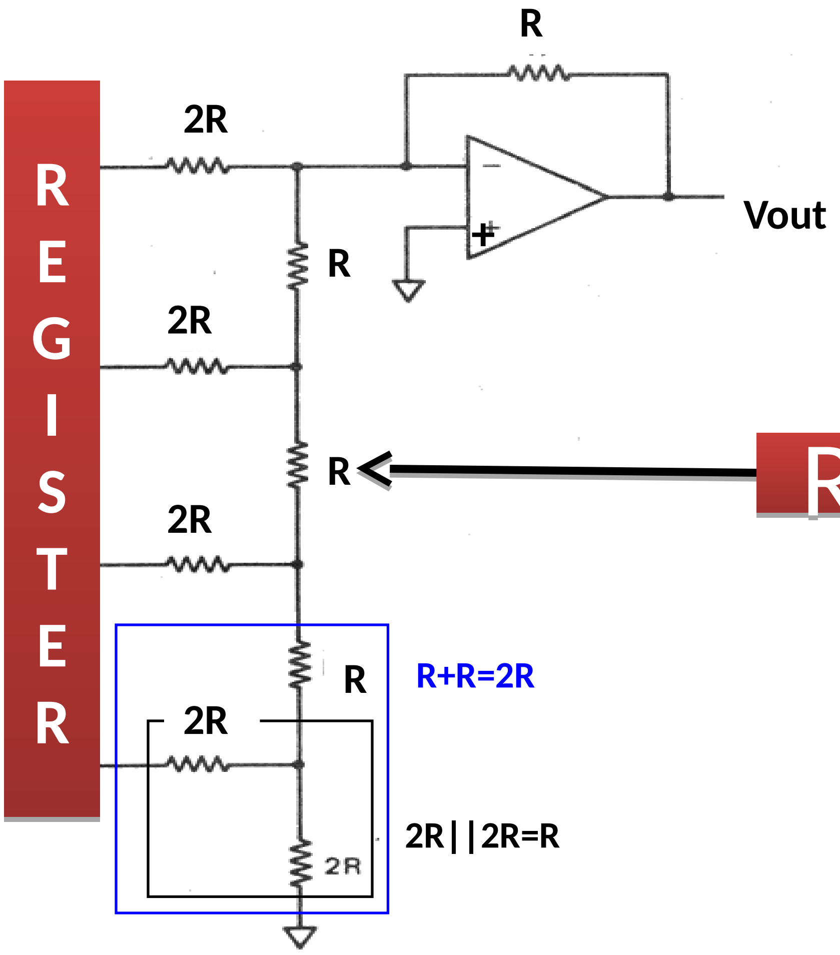
Circuit Realization



- Vo= Vref/R * ($A_1/2+ A_2/4+\ldots A_n/2^n$)
- Vo is proportional to values of Data Bits Value

Practical DAC : R-2R ladder network

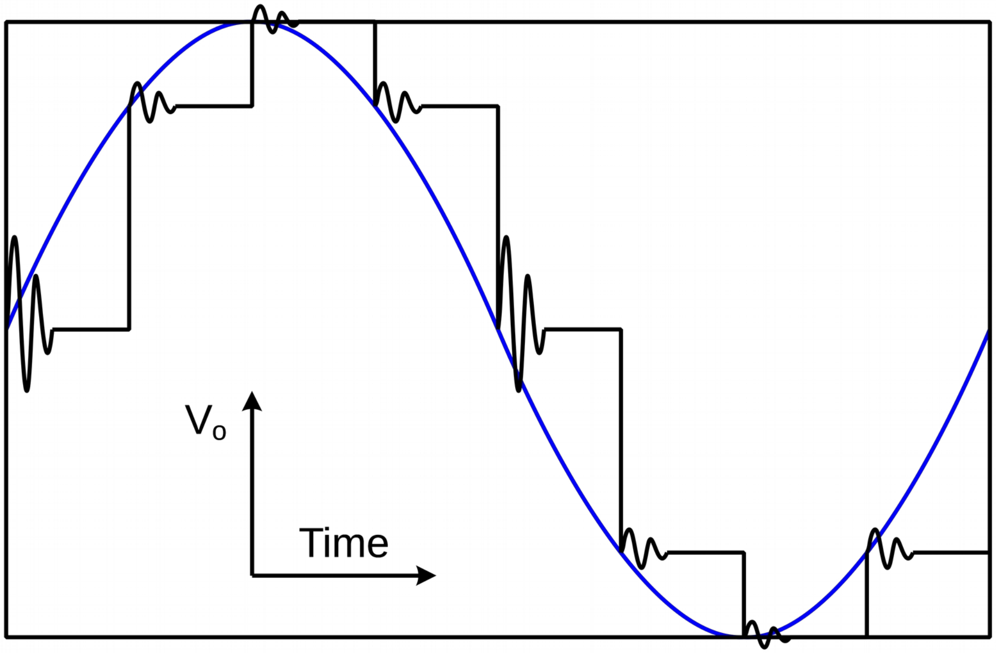
- Resistive Ladder Network



Resistor

- Require two type Resistor
- But small value
  - 5KΩ and 10KΩ

Output Glitches



- Cause: Signal and clock skew in circuits
- Especially severe at MSB transition where all bits are switching –

  0111…111 →

  1000…000

- Glitches cause waveform distortion, spurs and elevated noise floors
- High-speed DAC output is often followed by a de-glitching SHA (Hold Buffer)

Performance of DAC

- Resolution
- Reference Voltages
- Settling Time
- Linearity
- Speed
- Errors

Resolution

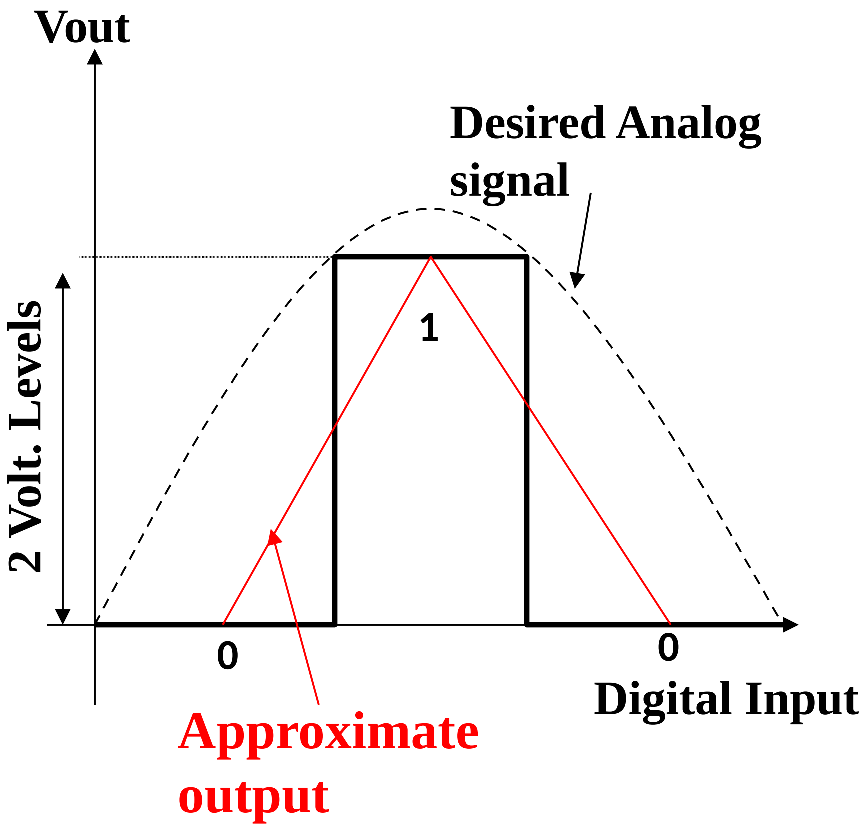
- Amount of variance in output voltage for every change of the LSB in the digital input.
- How closely can we approximate the desired output signal(Higher Res. = finer detail=smaller Voltage divisions)
- A common DAC has a 8 - 12 bit Resolution

$$\text{Resolution} = V_{LSB} = \frac{V_{\text{Ref}}}{2^{N}}$$

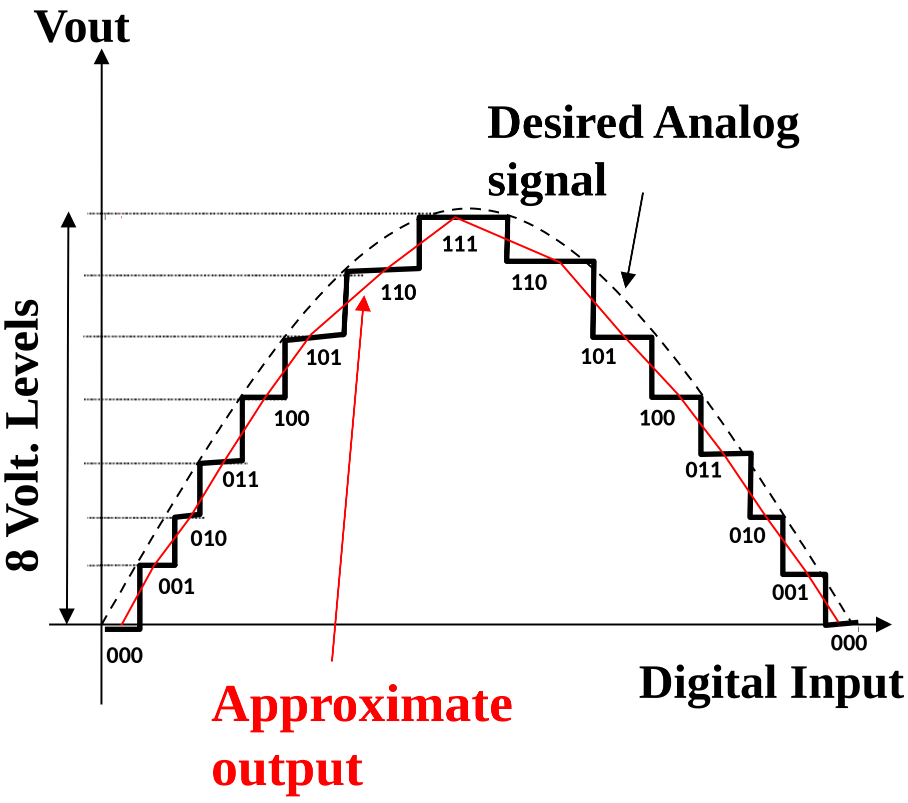
N = Number of bits

Resolution

Poor Resolution(1 bit)



Better Resolution(3 bit)



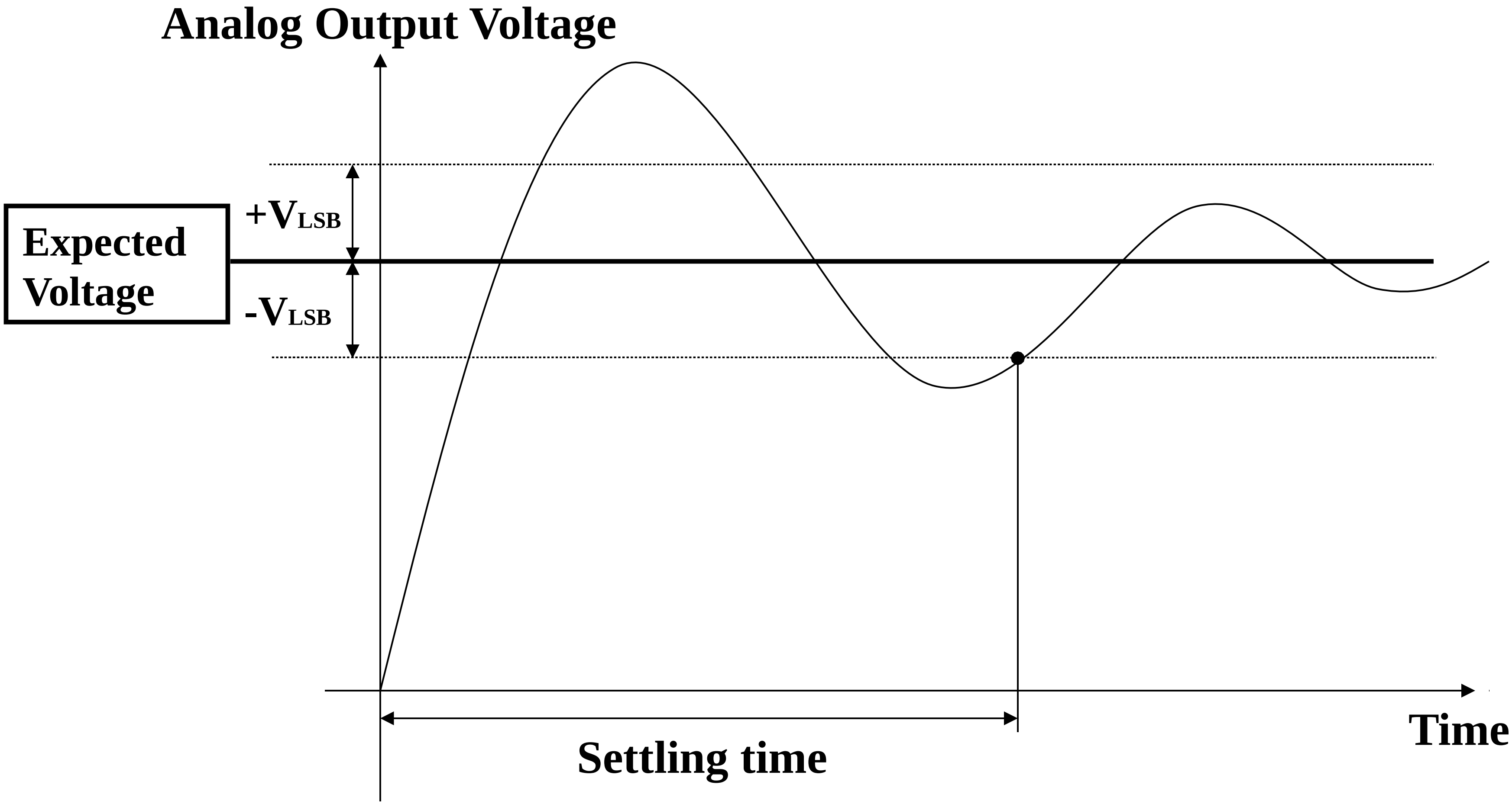
Reference Voltage

- A specified voltage used to determine how each digital input will be assigned to each voltage division.
- Types:
  - Non-multiplier: internal, fixed, and defined by manufacturer
  - Multiplier: external, variable, user specified

Settling Time

- Settling Time: The time required for the input signal voltage to settle to the expected output voltage(within +/- V_{LSB}).
- Any change in the input state will not be reflected in the output state immediately. There is a time lag, between the two events.

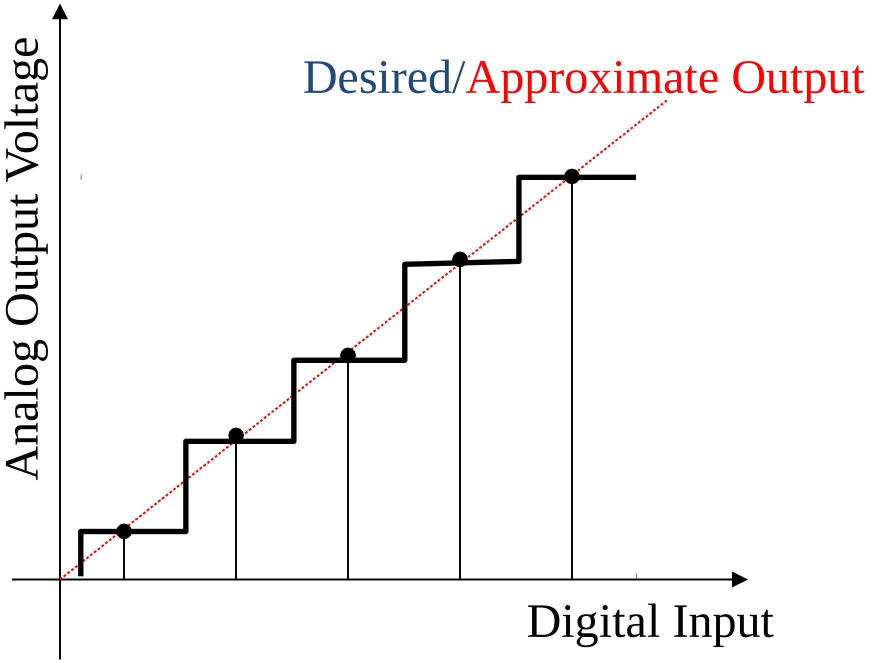
Settling Time



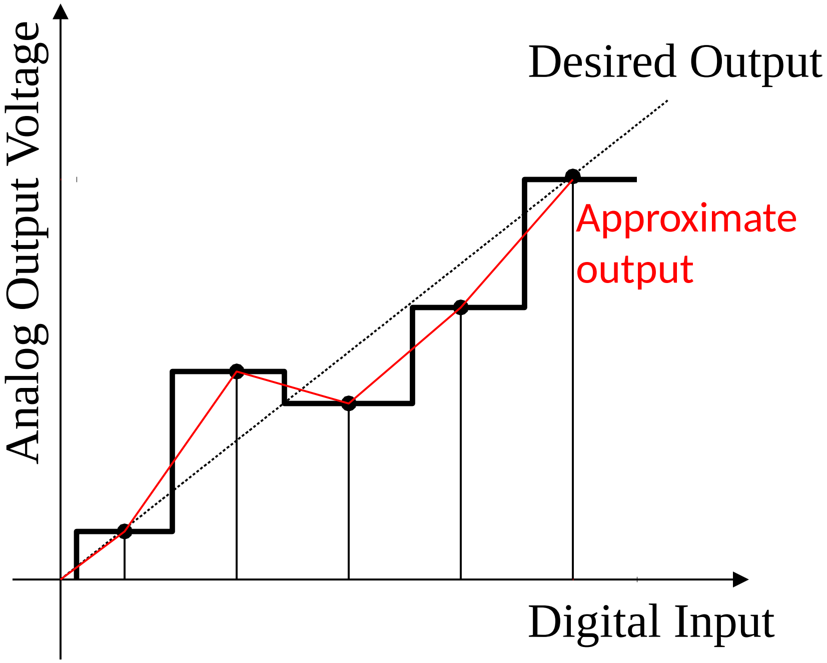
Linearity

Linearity(Ideal Case)

NON-Linearity(Real World)



Perfect Agreement



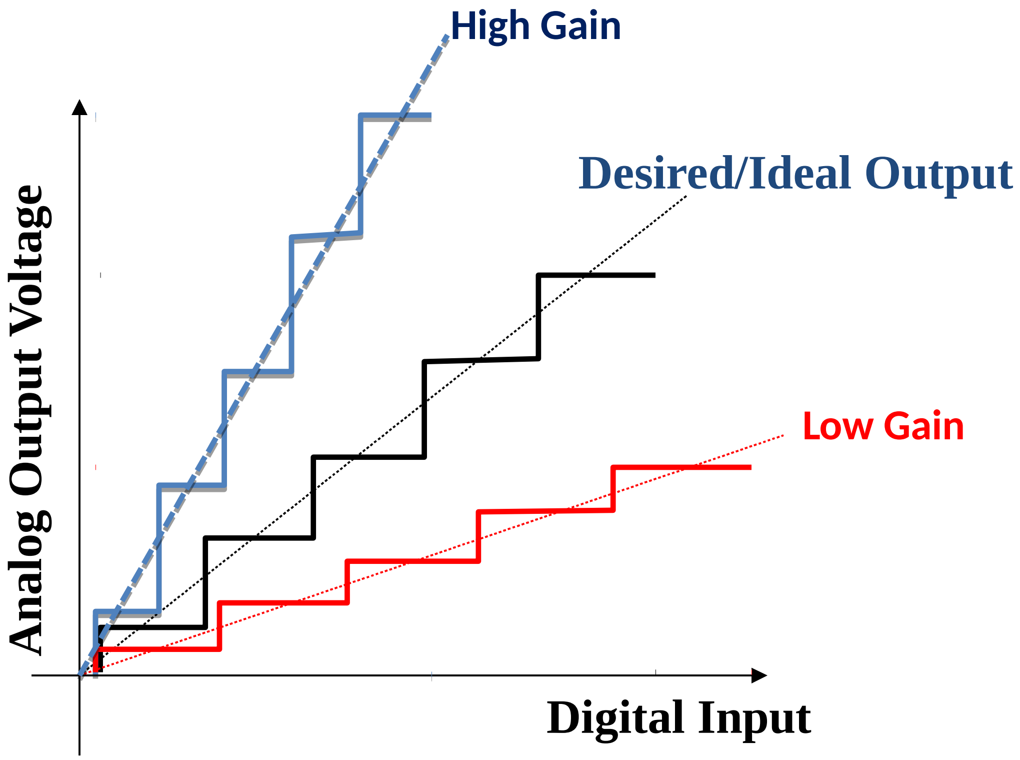
Miss-alignment

Errors Gain

- Gain Error: Difference in slope of the ideal curve and the actual DAC output

High Gain Error: Actual slope greater than ideal

Low Gain Error: Actual slope less than ideal

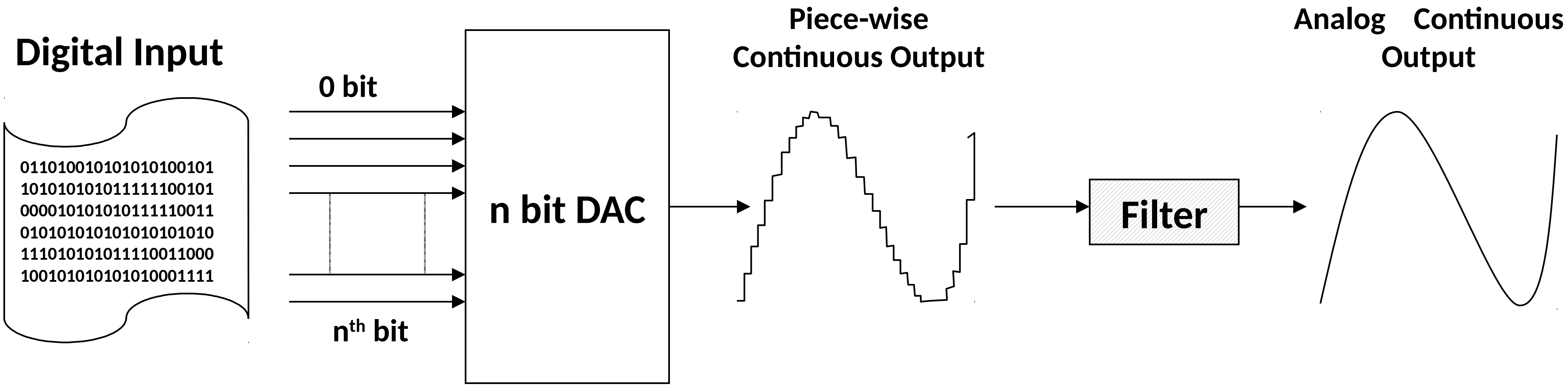


Speed

- Rate of conversion of a single digital input to its analog equivalent
- Conversion Rate
  - Depends on clock speed of input signal
  - Depends on settling time of converter

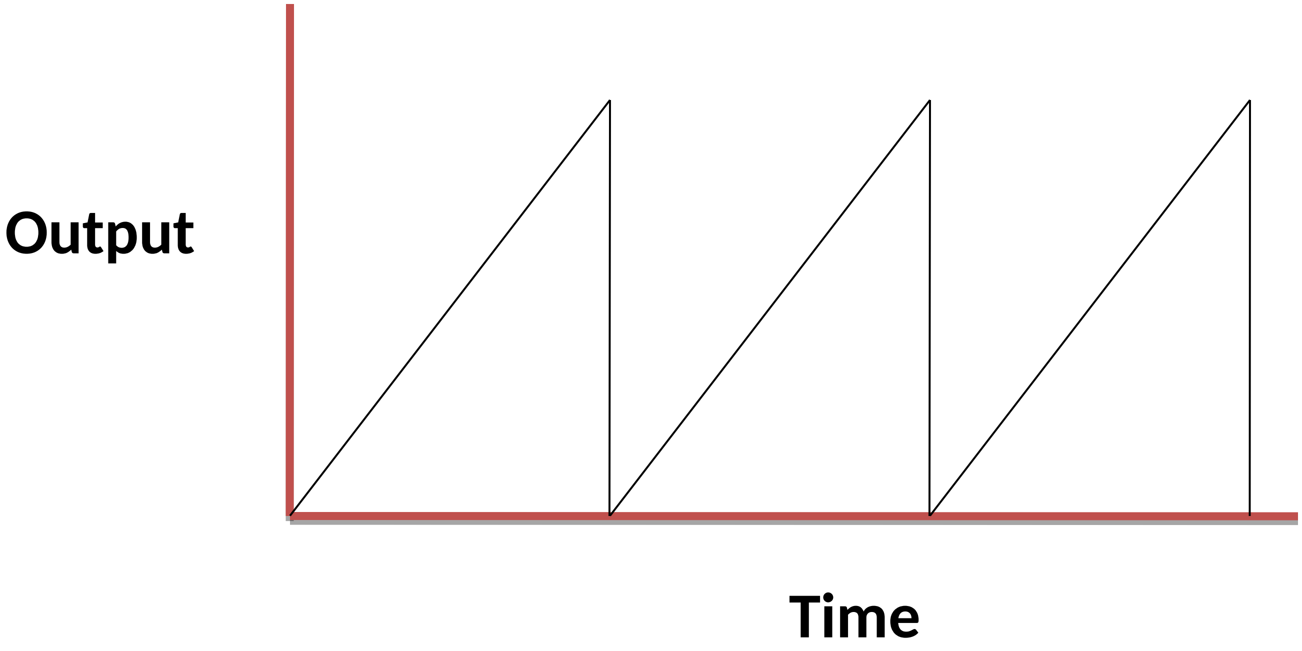
Generic Used: Audio player

- Used when a continuous analog signal is required.
- Signal from DAC can be smoothed by a Low pass filter

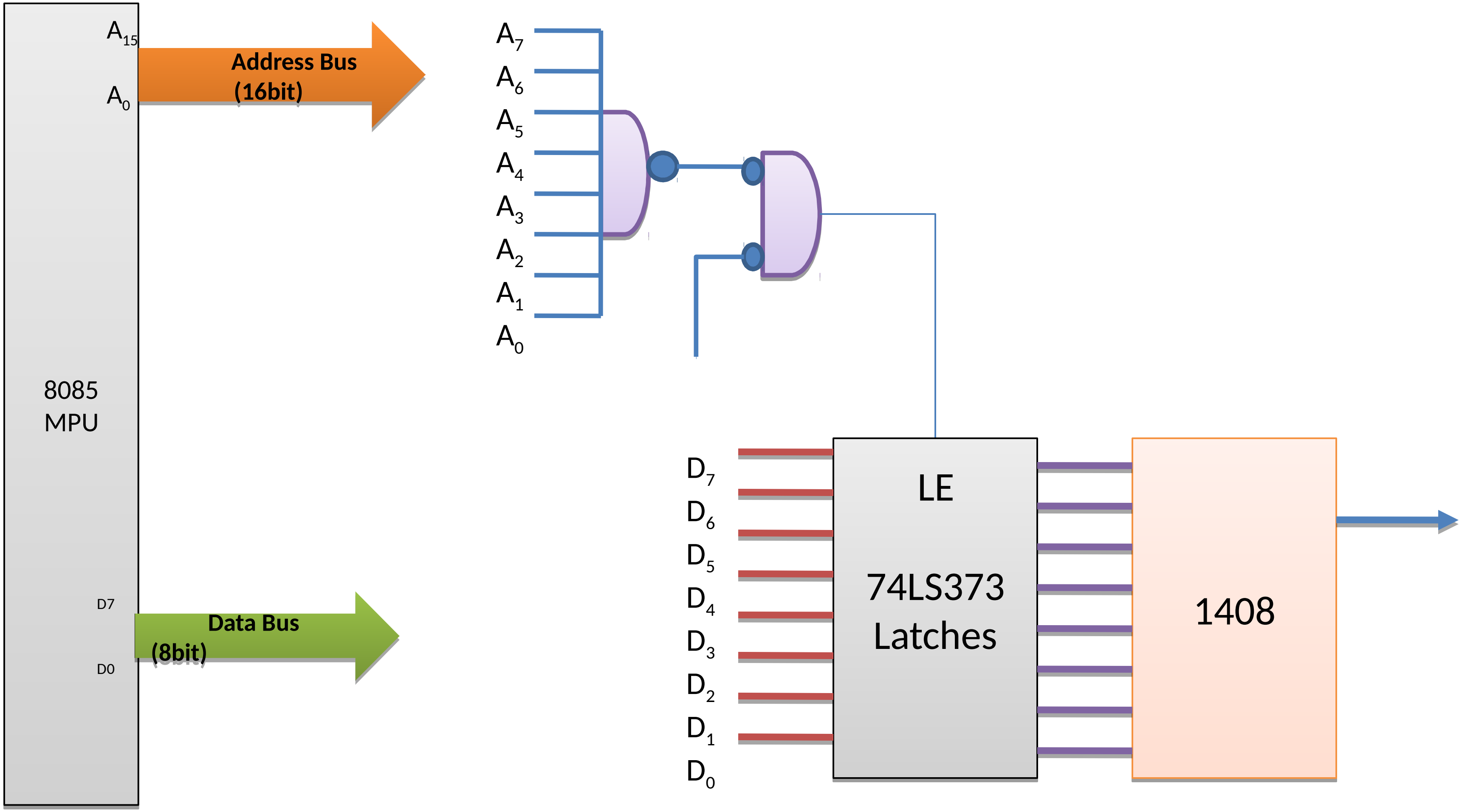


Interfacing 8-bitDAC with 8085

- Design an output port with Address FFH to interface 1408 DAC
- Write a program to generate a continuous RAMP waveform



Interfacing Diagram

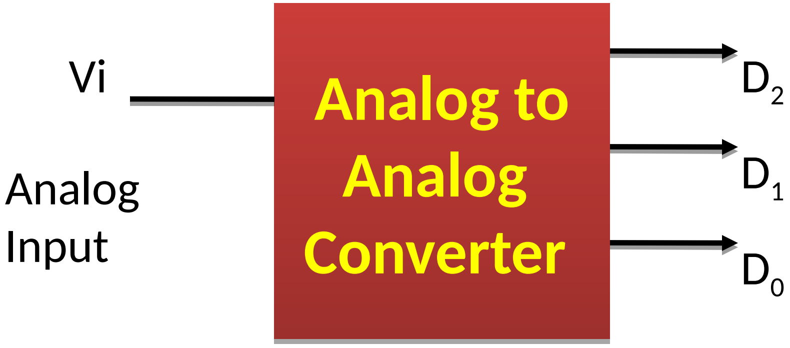


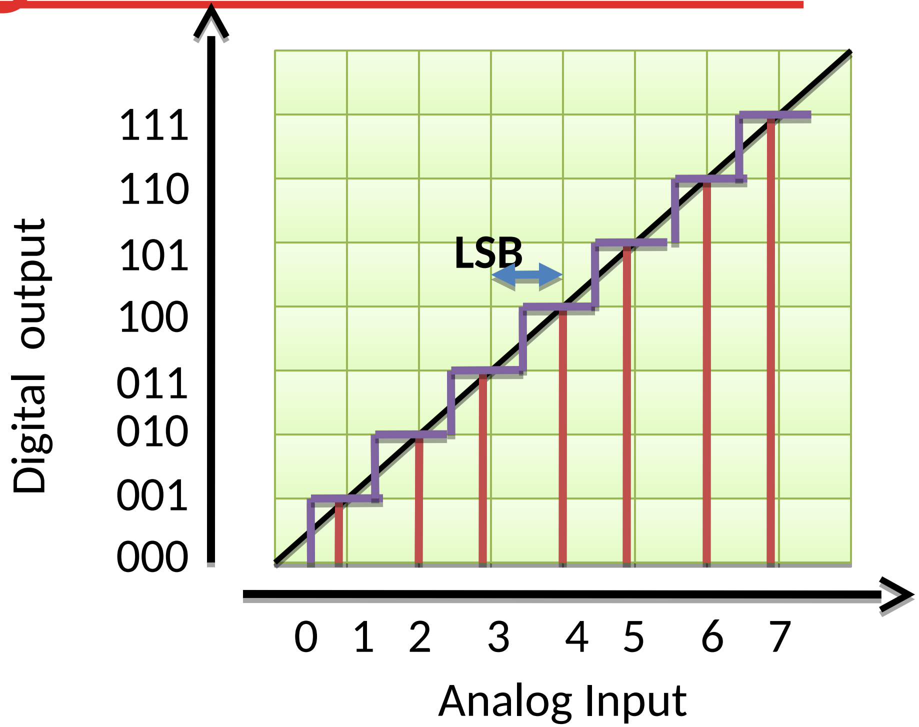
Program to generate continuous RAMP waveform

```
        MVI   A, 00H      ; Load Acc with first I/P
DTOA:   OUT   FFH         ; Output to DAC
        MVI   B, COUNT       ; Setup Reg. for Delay
        DCR   B
        JNZ   DELAY
        INR   A        ; Next Input
        JMP   DTOA        ; Go back to Output
```

Slope of RAMP can be varied by changing Delay

Analog to Digital Conversion

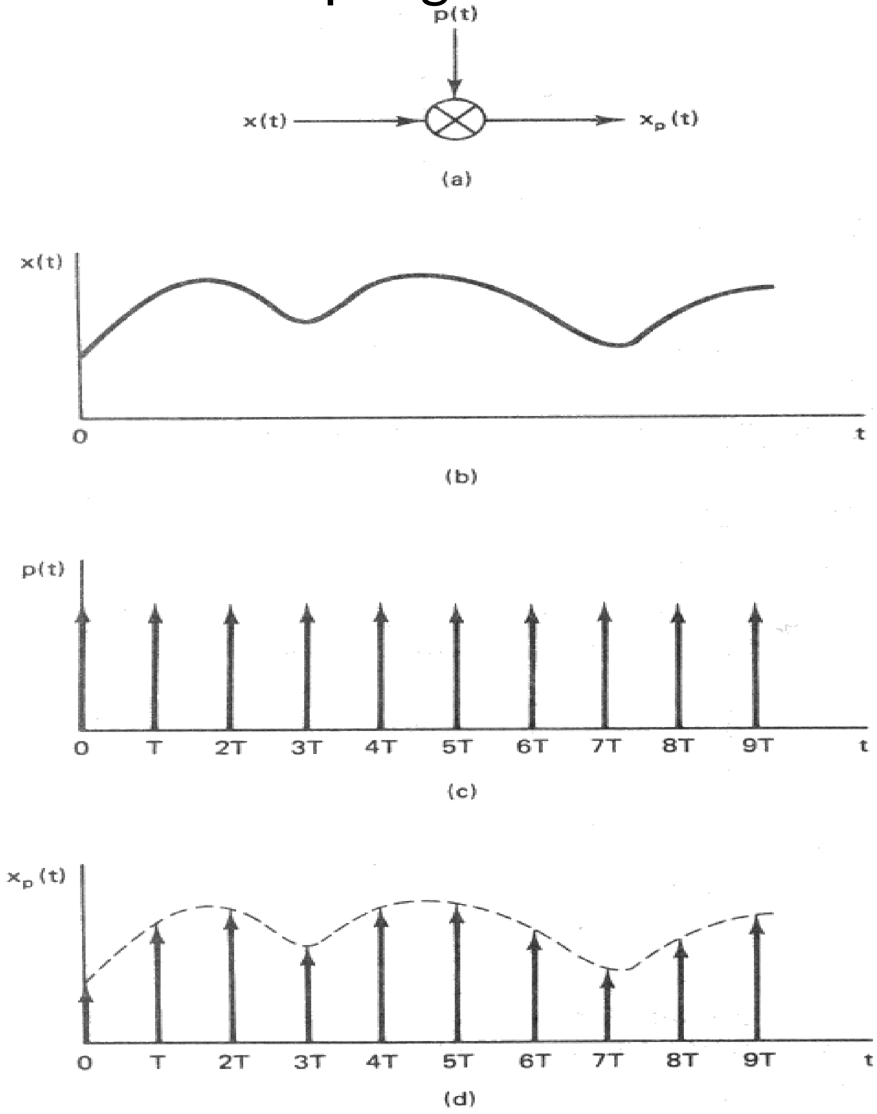




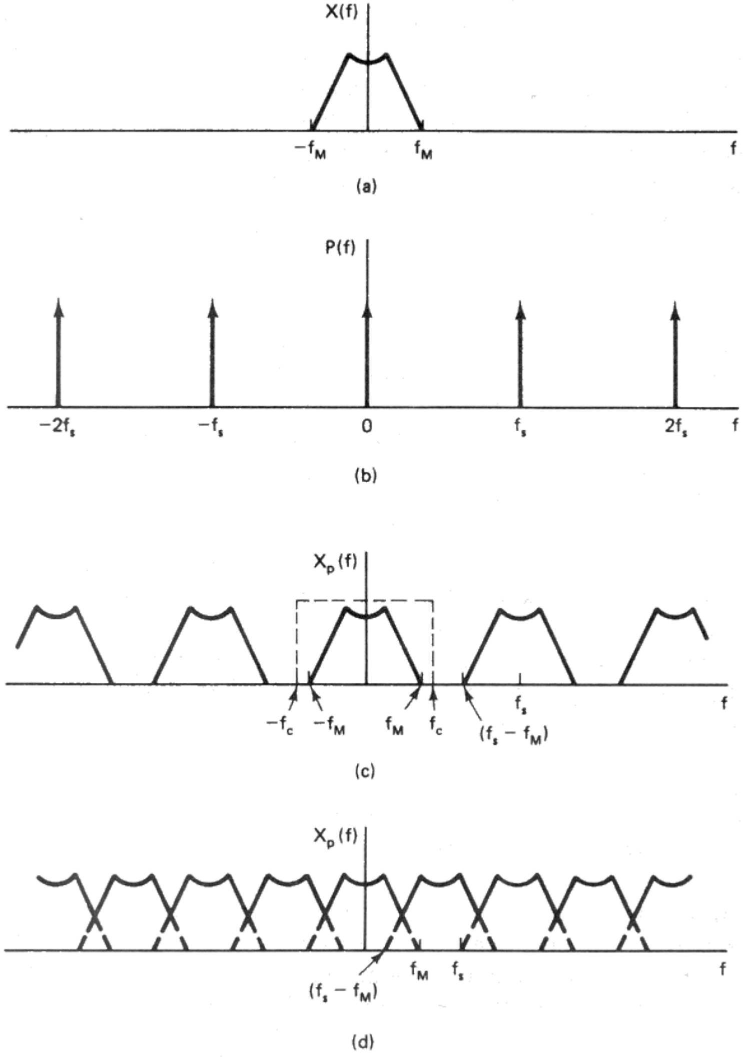
- ADC are slower then DAC
- Interfaced using Status Check

Sampling Concepts

- Sampling

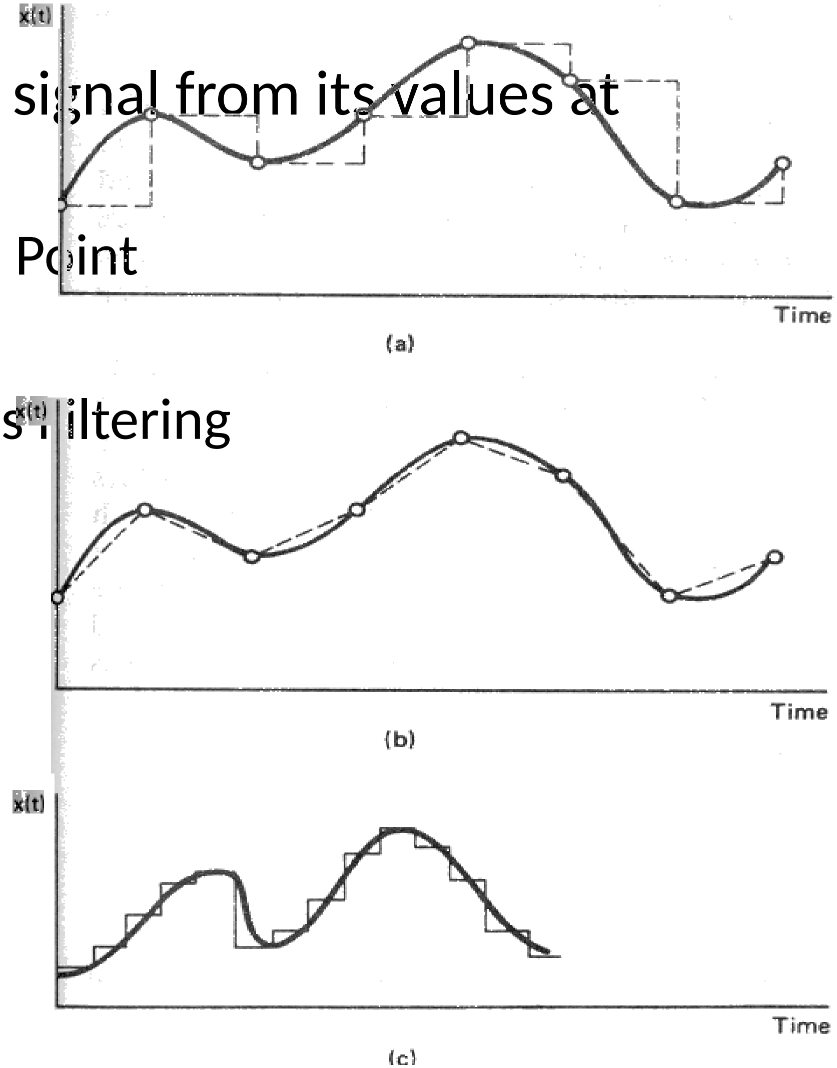


- Fourier Transform



Interpolation

- The process of reconstructing a signal from its values at discrete instants of time
  - Zero order hold or One Point
  - Linear or Two Point
  - Band limited or Low pass filtering

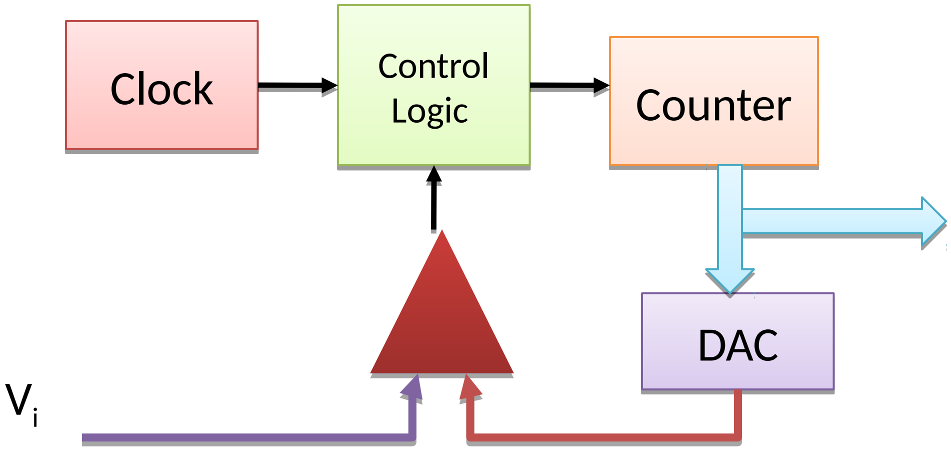


A/D Conversion Techniques

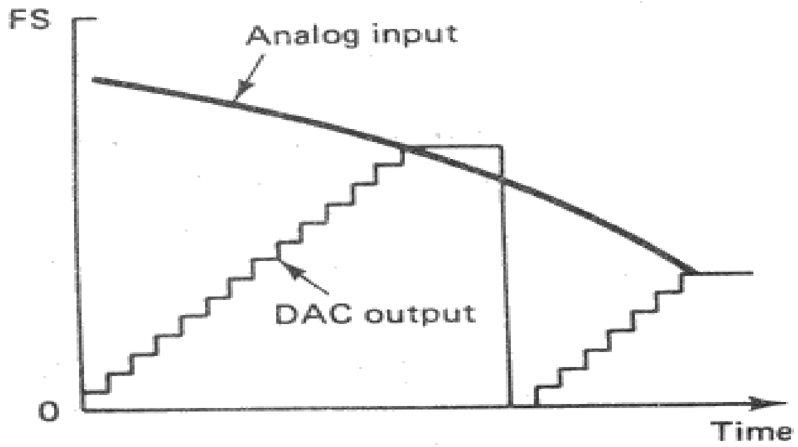
- Counter or Tracking ADC
- Successive Approximation ADC
  - Most Commonly Used
- Parallel or Flash ADC
  - Fast Conversion

Counter Type ADC

- Block diagram



- Waveform

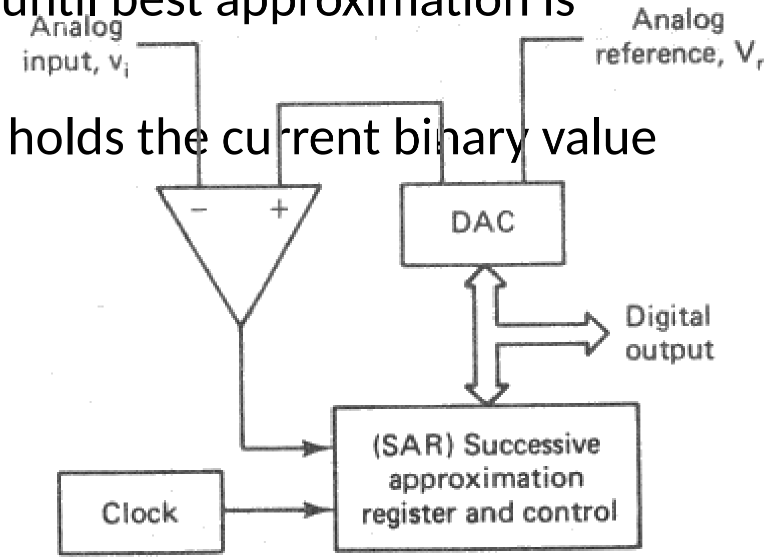


- Operation
  - Reset and Start Counter
  - DAC convert Digital output of Counter to Analog signal
  - Compare Analog input and Output of DAC
    - Vi < V_{DAC}
      - Continue counting
    - Vi = V_{DAC}
      - Stop counting
  - Digital Output = Output of Counter
- Disadvantage
  - Conversion time is varied
    - 2^n Clock Period for Full Scale input

Successive Approximation ADC

- Most Commonly used in medium to high speed Converters
- Based on approximating the input signal with binary code and then successively revising this approximation until best approximation is achieved
- SAR(Successive Approximation Register) holds the current binary value

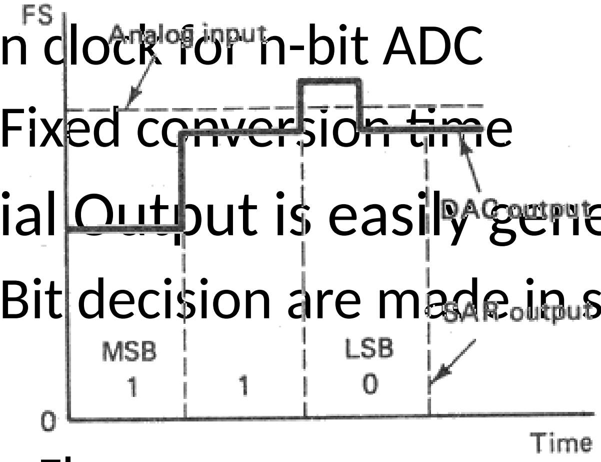
- Block Diagram



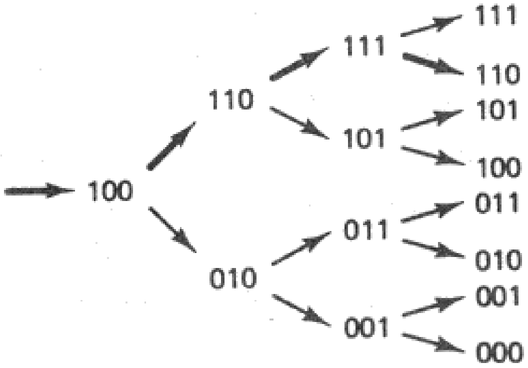
Successive Approximation ADC

- Conversion Time
  - n clock for n-bit ADC
  - Fixed conversion time
- Serial Output is easily generated
  - Bit decision are made in serial order

- Circuit waveform

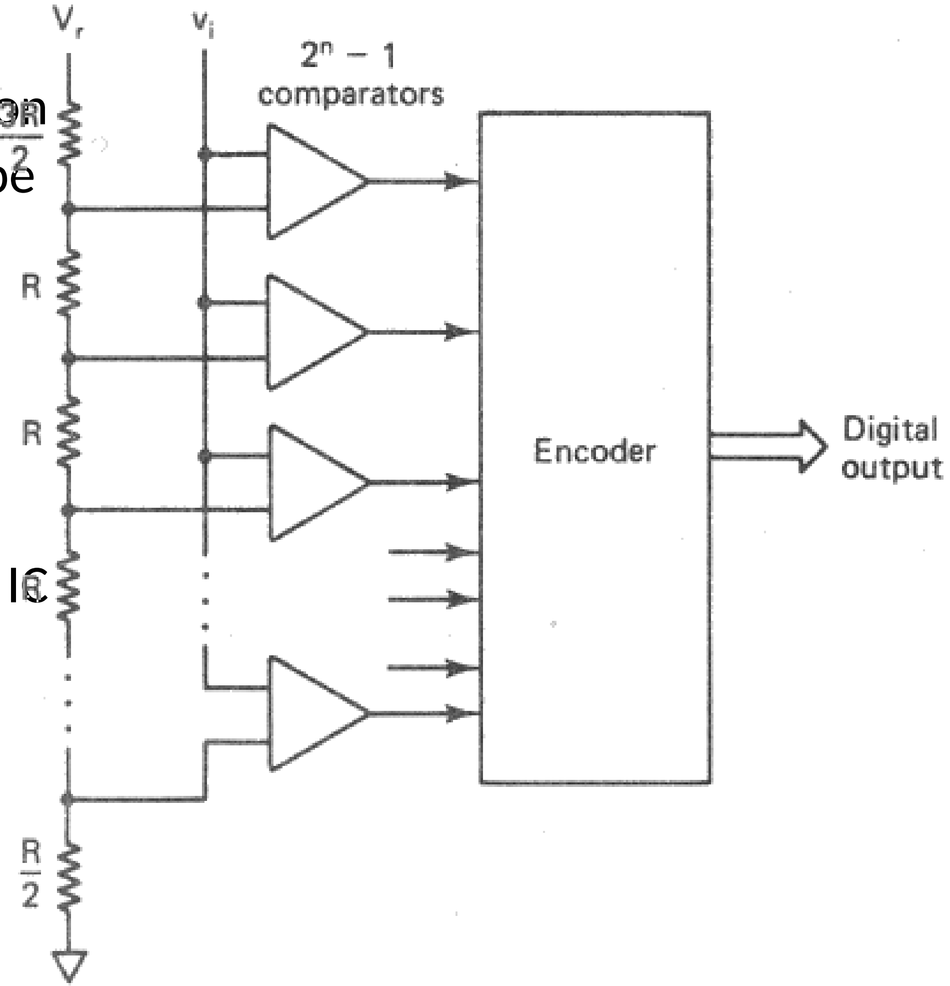


- Logic Flow

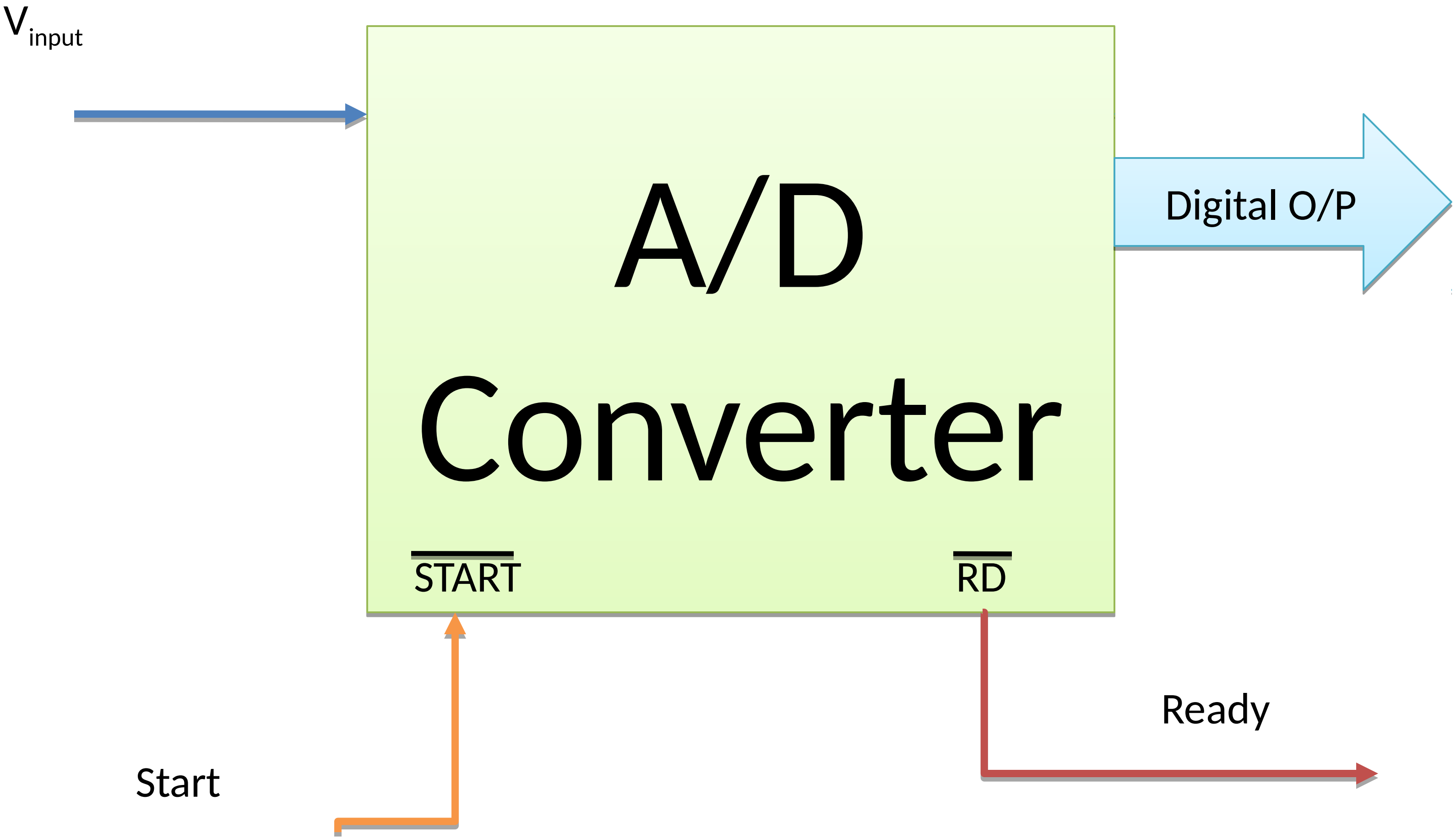


Parallel or Flash ADC

- Very High speed conversion
  - Up to 100MHz for 8 bit resolution
  - Video, Radar, Digital Oscilloscope
- Single Step Conversion
  - 2^n –1 comparator
  - Precision Resistive Network
  - Encoder
- Resolution is limited
  - Large number of comparator in IC



Generic ADC



V_{input}

A/D Converter

Digital O/P

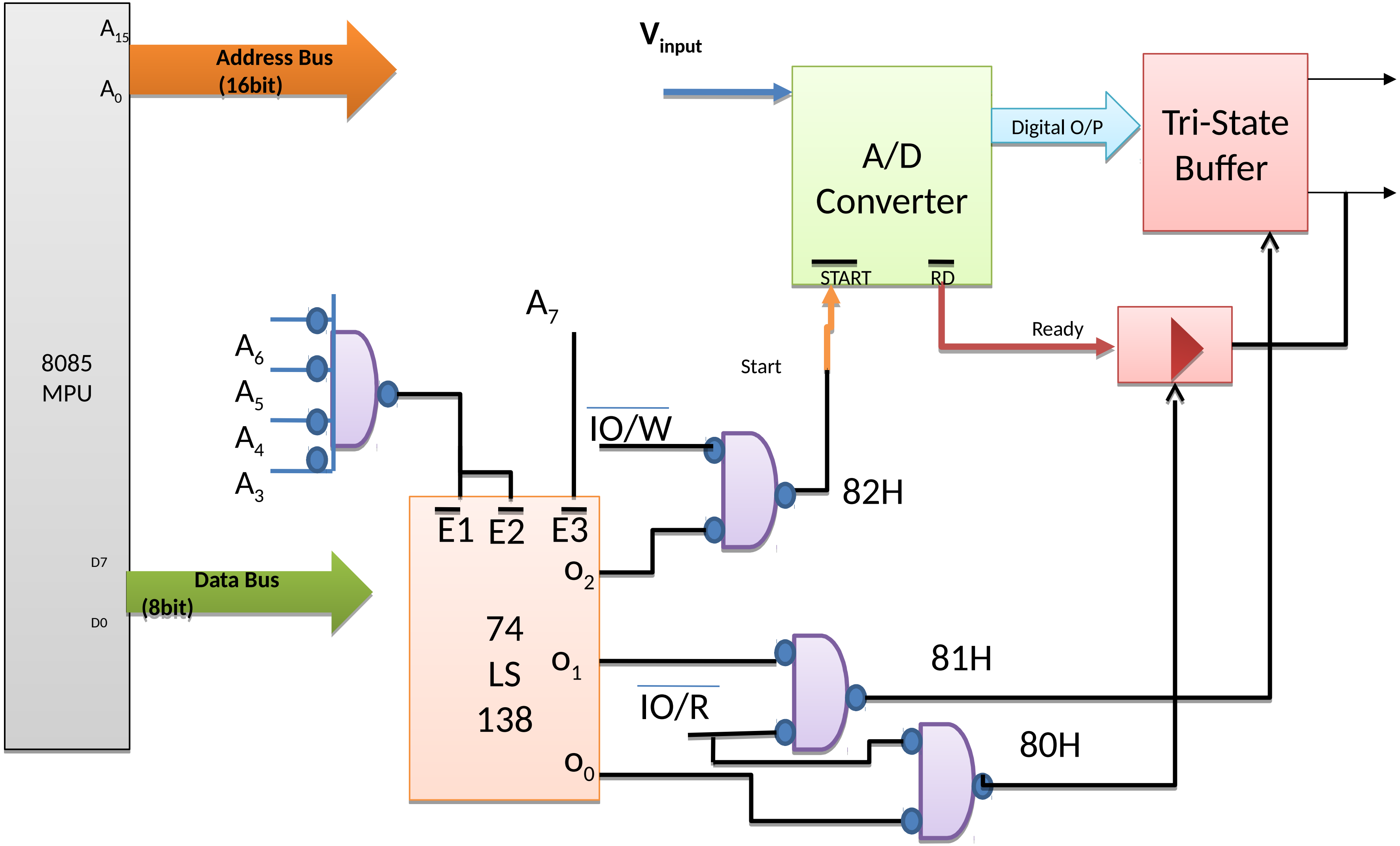
$\overline{\text{START}}$

$\overline{\text{RD}}$

Start

Ready

Interface ADC using Status Check

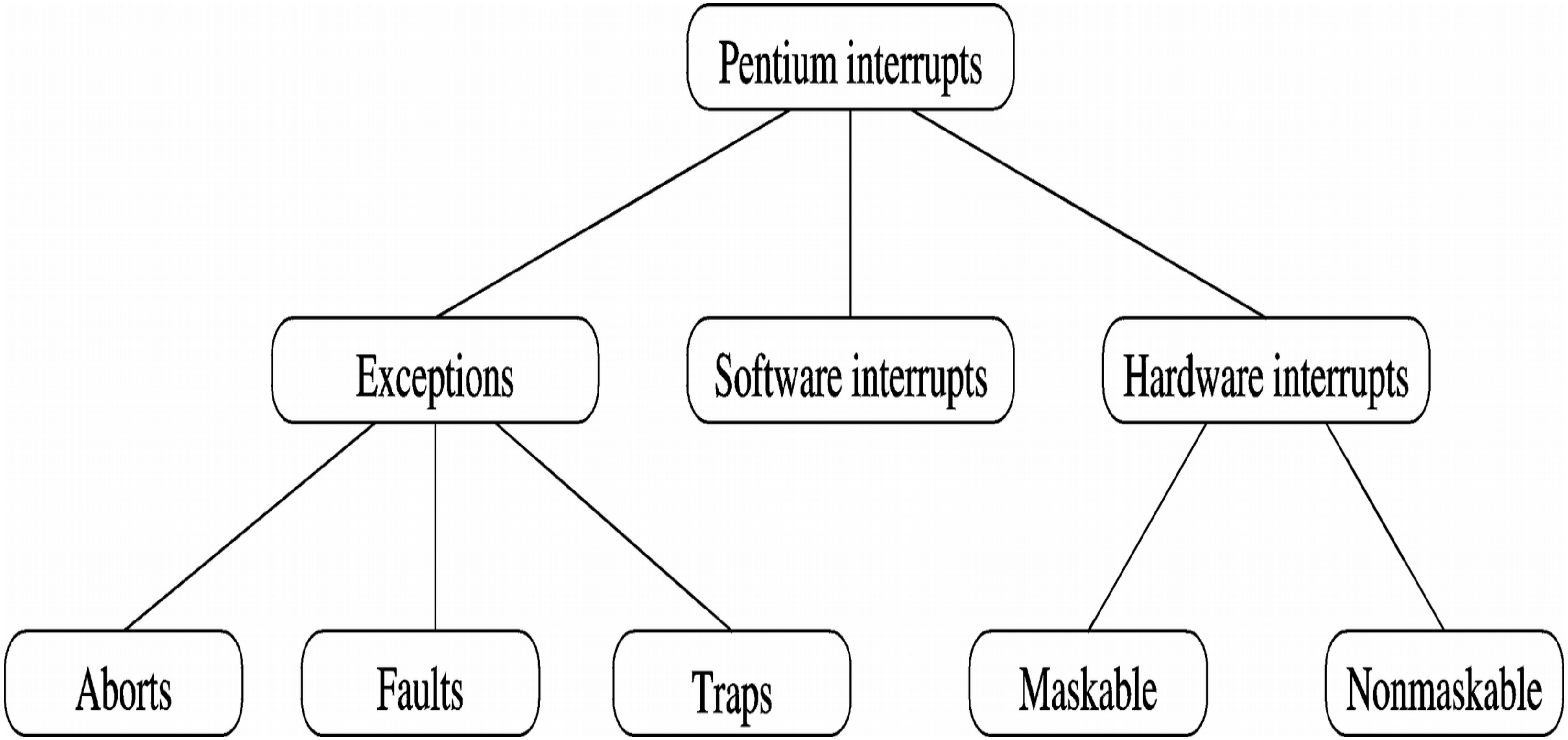


Program to Interface ADC

```
           OUT  82H   ; Start Conversion
TEST:          IN      80H  ;Read DR Status
           RAR        ; Rotate Do to carry
           JC TEST  ; if Do==1 conv. done
           IN 81H   ; Read the output
           RET              ; Return
```

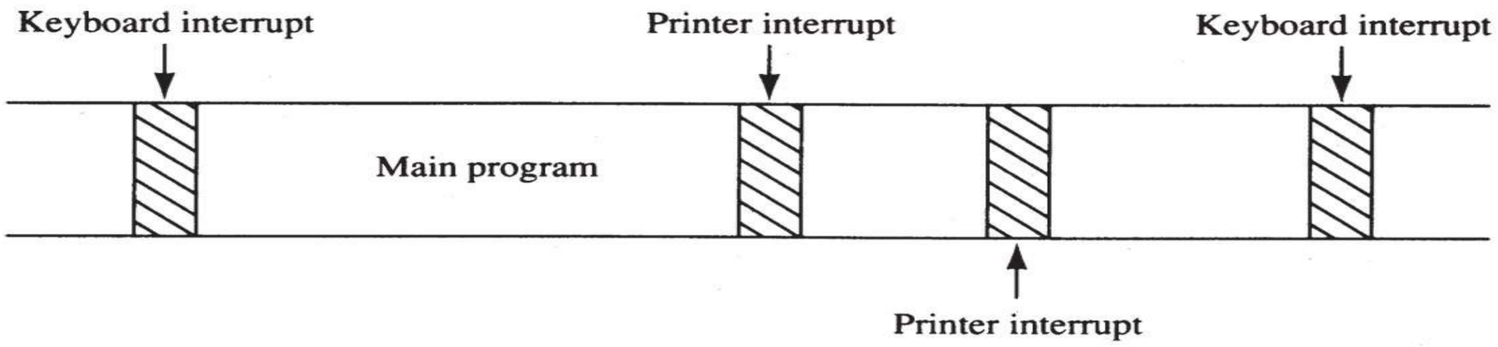
Interrupt Controller (8259)

A Taxonomy of Interrupts

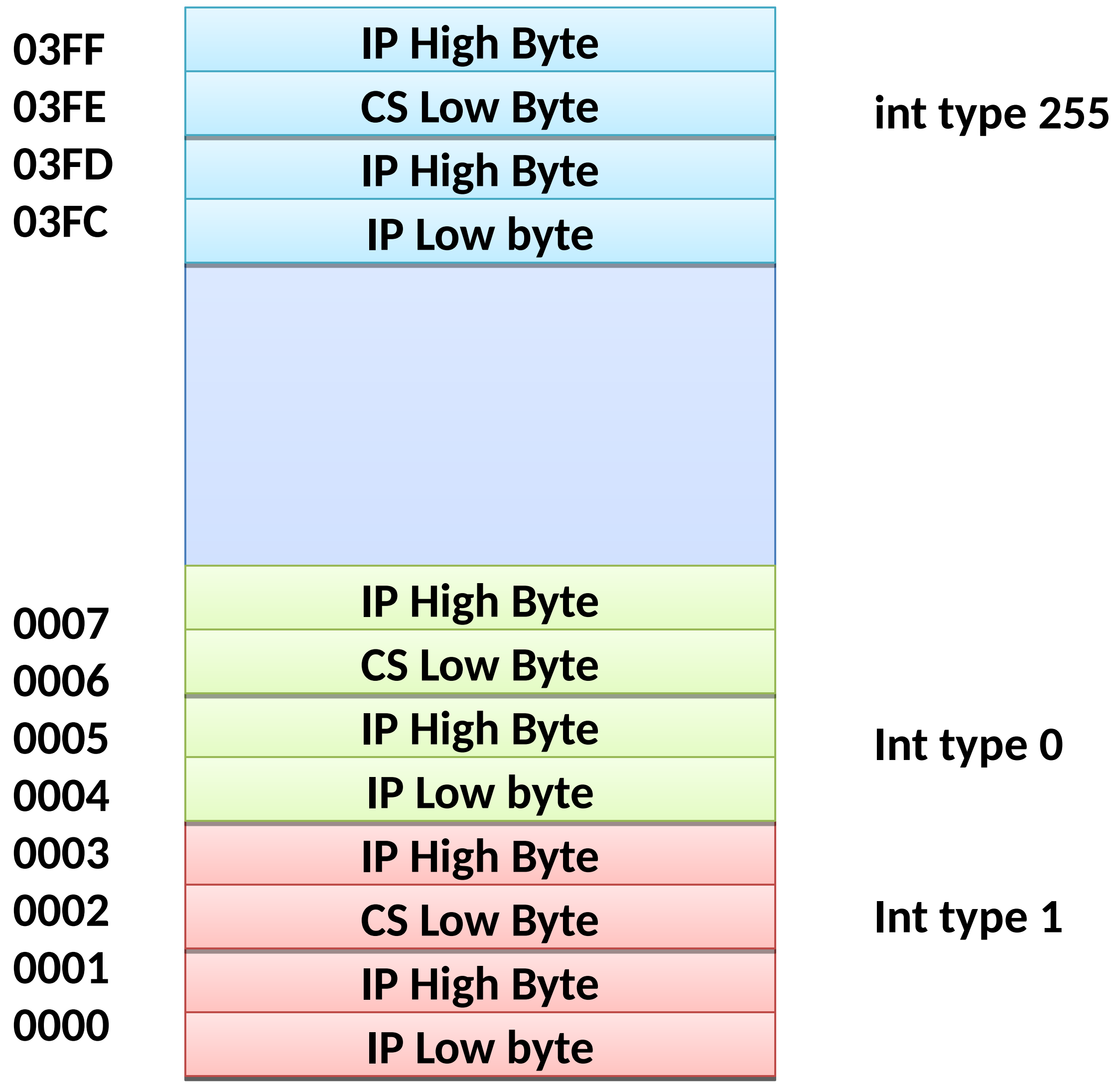


The Purpose of Interrupts

- Interrupts are useful when interfacing I/O devices at relatively low data transfer rates, such as keyboard inputs
- Interrupt processing allows the processor to execute other software while the keyboard operator is thinking about what to type next.



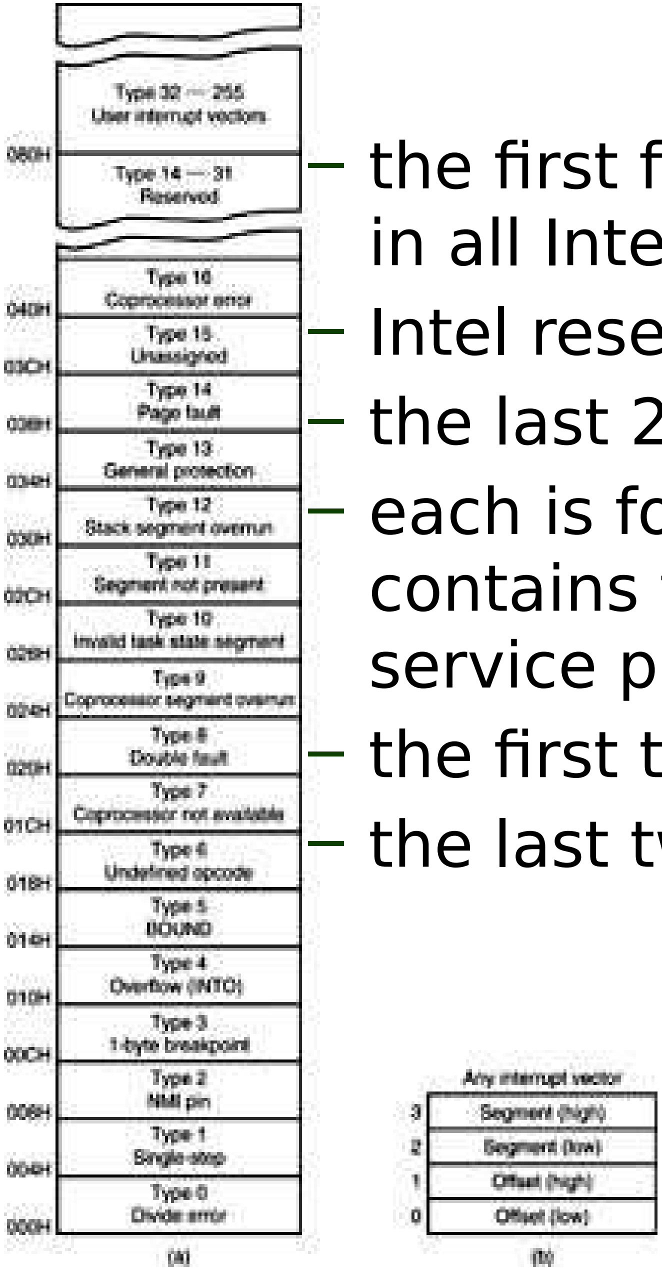
Interrupt Vector Table



03FF	IP High Byte	
03FE	CS Low Byte	int type 255
03FD	IP High Byte	
03FC	IP Low byte	
0007	IP High Byte	
0006	CS Low Byte	
0005	IP High Byte	Int type 0
0004	IP Low byte	
0003	IP High Byte	
0002	CS Low Byte	Int type 1
0001	IP High Byte	
0000	IP Low byte	

Memory in Hex

Figure 12–2 (a) The interrupt vector table for the microprocessor and (b) the contents of an interrupt vector.



- the first five interrupt vectors are identical in all Intel processors
- Intel reserves the first 32 interrupt vectors
- the last 224 vectors are user-available
- each is four bytes long in real mode and contains the starting address of the interrupt service procedure.
- the first two bytes contain the offset address
- the last two contain the segment address

Example

Question: A particular interrupt has a type number n=41H. If the corresponding ISR begins at address

FE00:0500H, determine the locations in the vector table to store this address.

Solution: The vector address is calculated by multiplying 41H by four.

00104:00; 00105:50; 00106:00; 00107:FE;

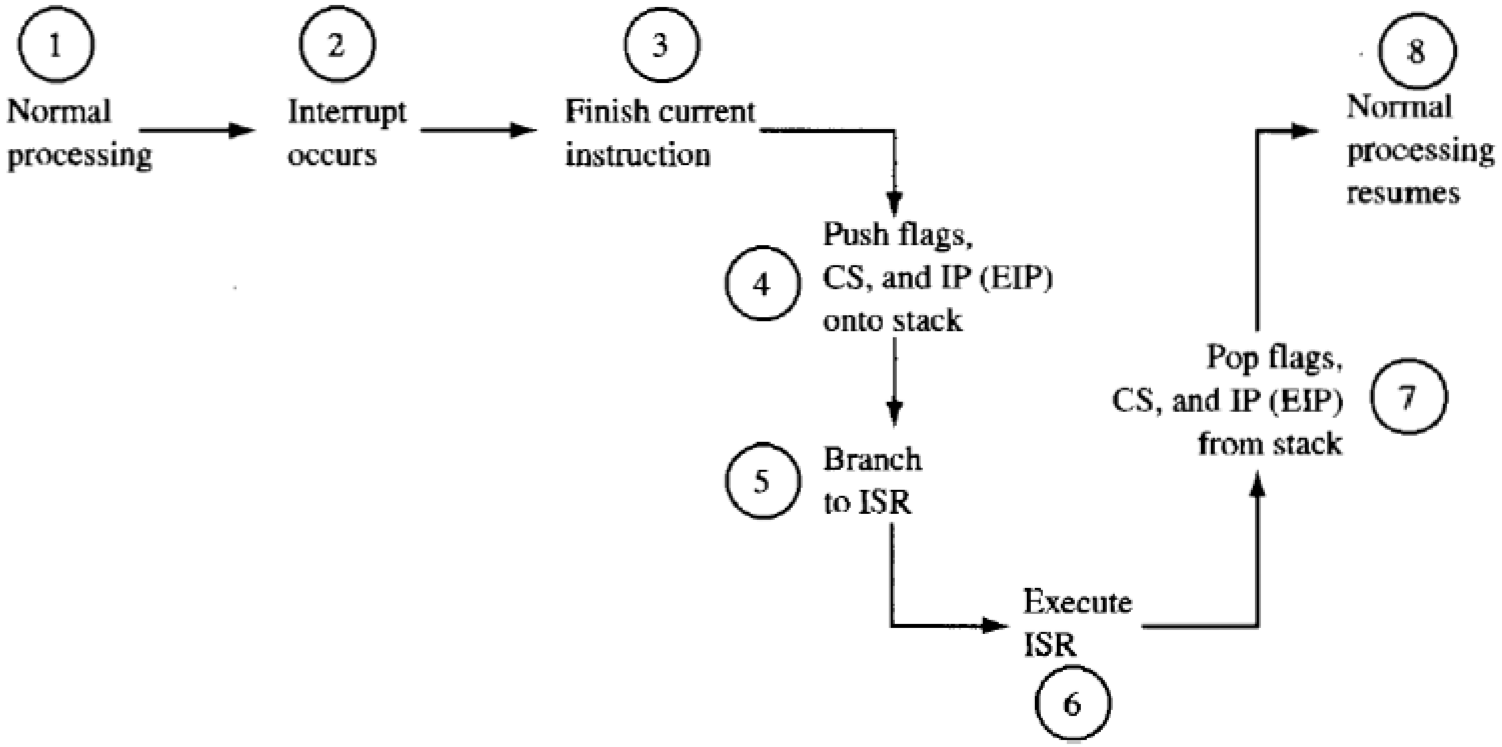
Interrupt Vector Table (x86)

- Using the Interrupt Vector Table shown below, determine the address of the ISR of a device with interrupt vector 42H.
- Answer: Address in table = 4 X 42H = 108H
- (Multiply by 4 since each entry is 4 bytes)
- Offset Low = [108] = 2A, Offset High = [109] = 33
- Segment Low = [10A] = 3C, Segment High = [10B] = 4A
- Address = 4A3C:332A = 4A3C0 + 332A = 4D6EAH

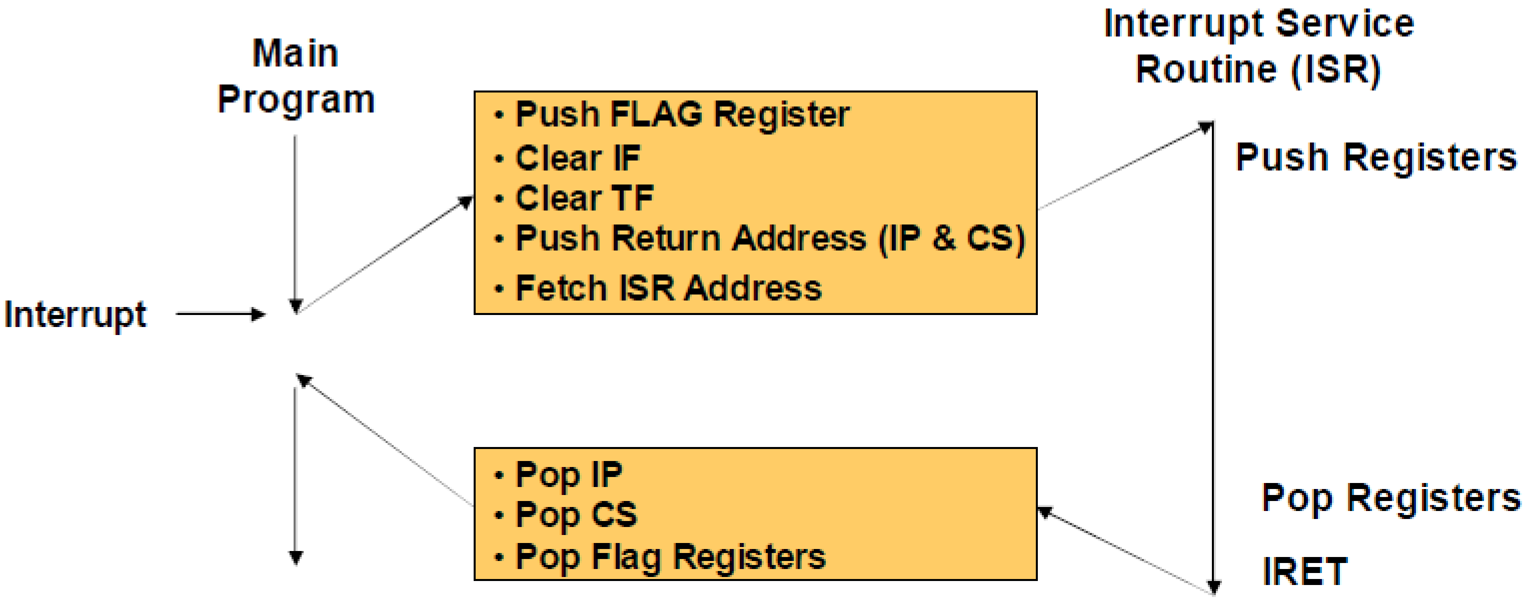
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00000	3C	22	10	38	6F	13	2C	2A	33	22	21	67	EE	F1	32	25
00010	11	3C	32	88	90	16	44	32	14	30	42	58	30	36	34	66
....	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
00100	4A	33	3C	4A	AA	1A	1B	A2	2A	33	3C	4A	AA	1A	3E	77
00110	C1	58	4E	C1	4F	11	66	F4	C5	58	4E	20	4F	11	F0	F4
....	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
00250	00	10	10	20	3F	26	33	3C	20	26	20	C1	3F	10	28	32
00260	20	4E	00	10	50	88	22	38	10	5A	38	10	4C	55	14	54
....	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
003E0	3A	10	45	2F	4E	33	6F	90	3A	44	37	43	3A	54	54	7F
003F0	22	3C	80	01	3C	4F	4E	88	22	3C	50	21	49	3F	F4	65

Interrupt Processing

When interrupt occurs, normal processing is suspended while a special Interrupt Service Routine (ISR) is executed. Normal processing resumes when this routine is completed.


1
Normal processing
2
Interrupt occurs
3
Finish current instruction
4
Push flags, CS, and IP (EIP) onto stack
5
Branch to ISR
6
Execute ISR
7
Pop flags, CS, and IP (EIP) from stack
8
Normal processing resumes

Interrupt Processing



Main Program

Interrupt

- Push FLAG Register
- Clear IF
- Clear TF
- Push Return Address (IP & CS)
- Fetch ISR Address

Interrupt Service Routine (ISR)

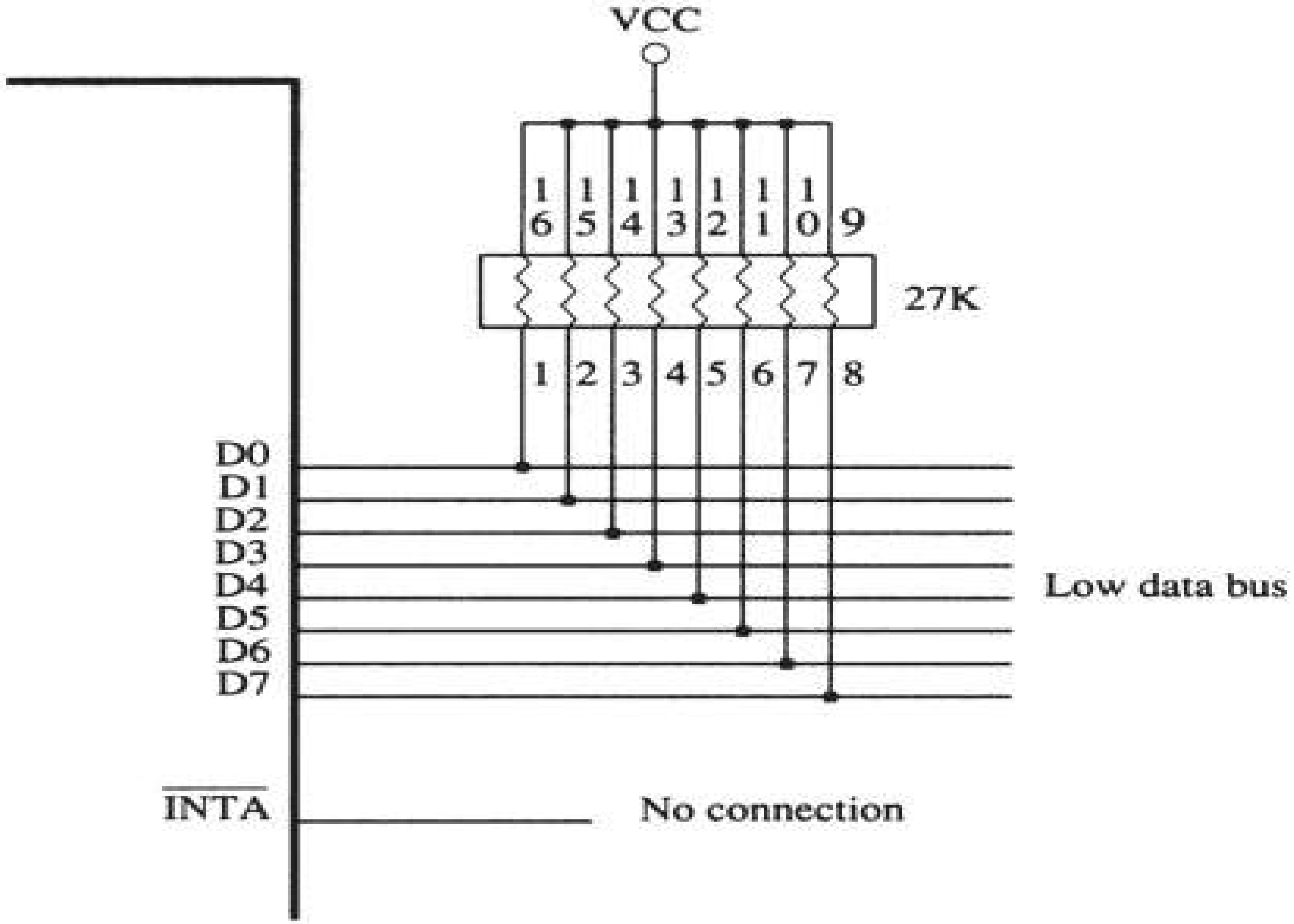
Push Registers

Pop Registers

IRET

- Pop IP
- Pop CS
- Pop Flag Registers

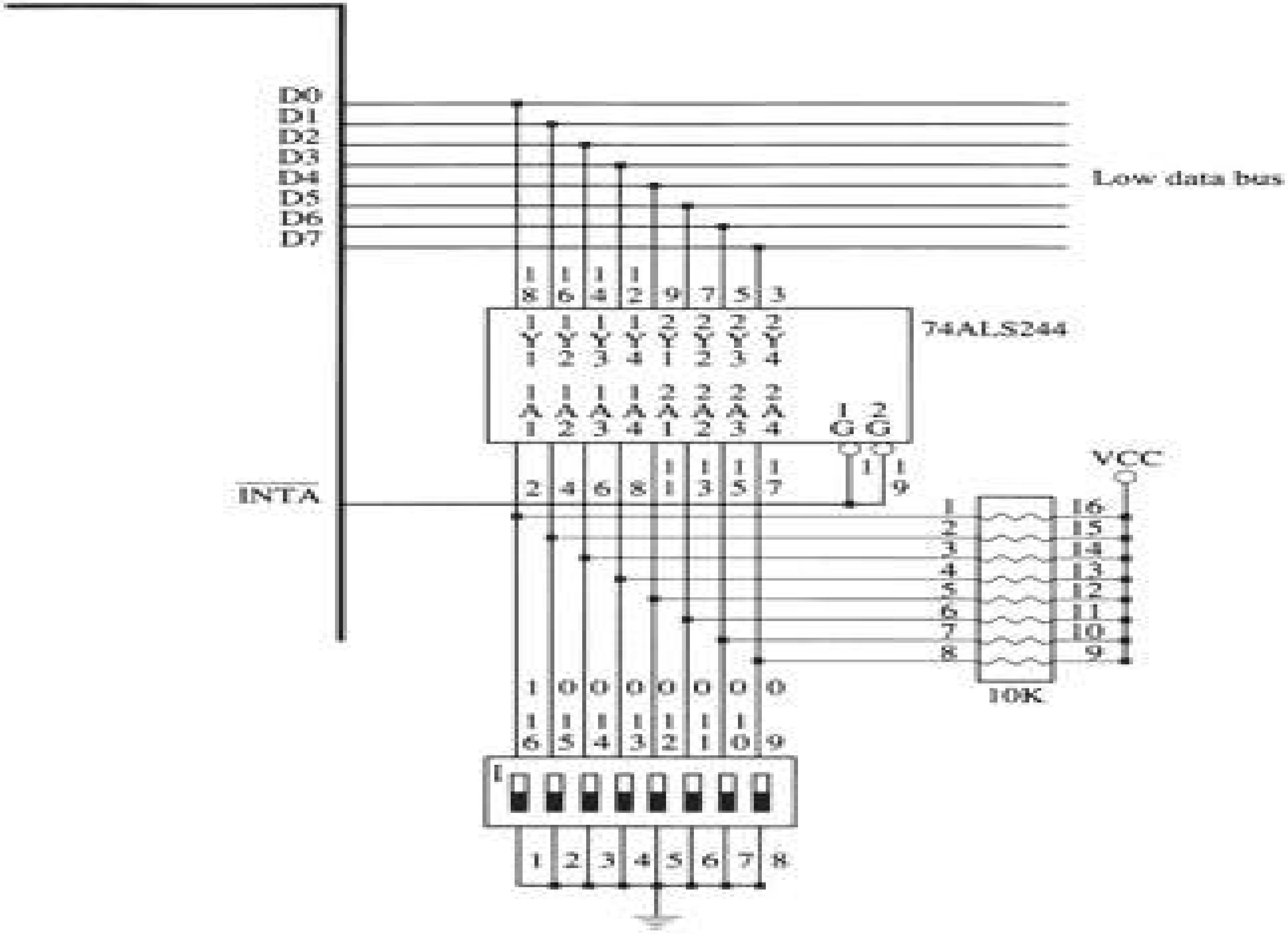
A simple method for generating interrupt vector type number FFH in response to INTR



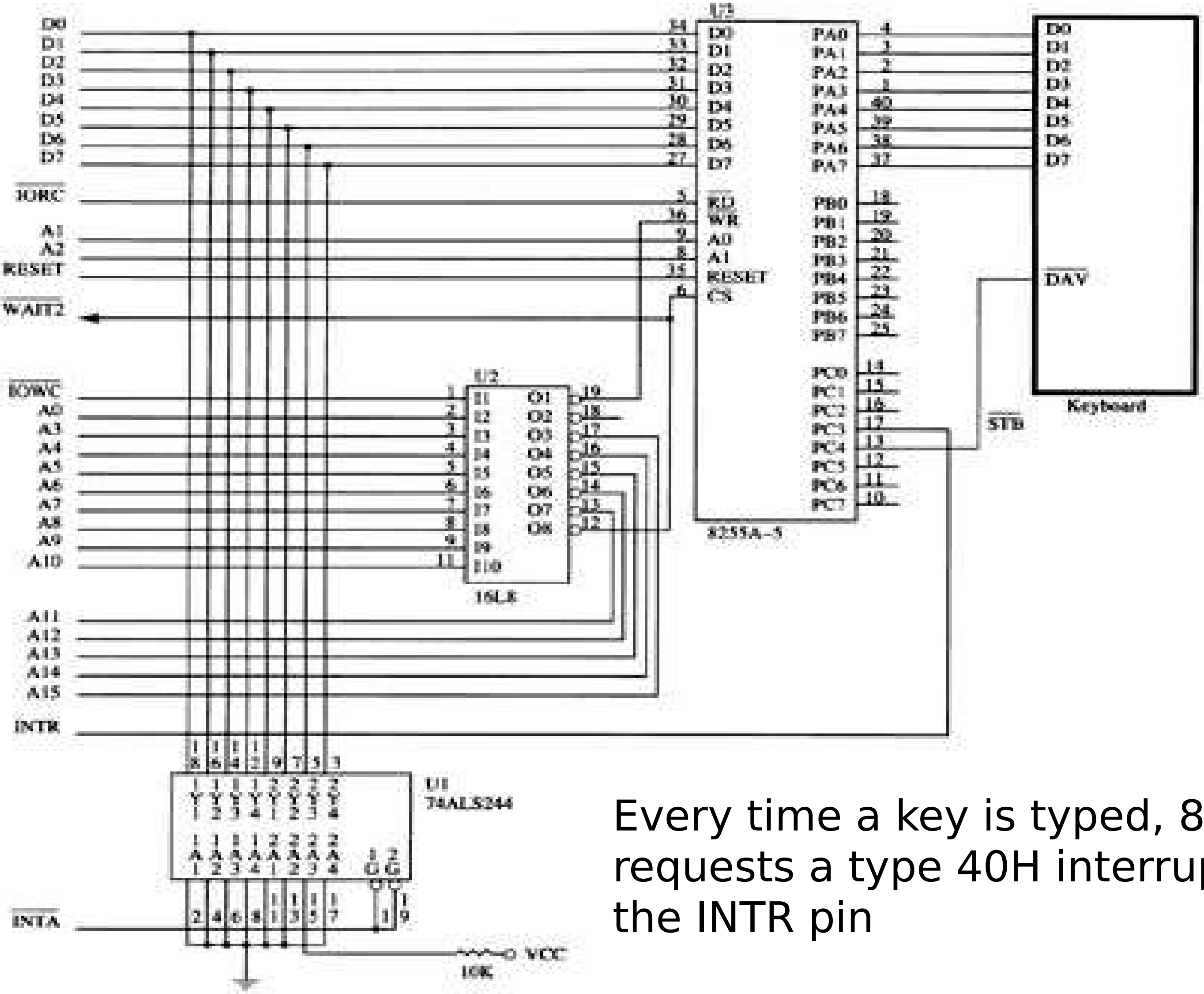
Using a Three-State Buffer for INTA

- Interrupt vector type number 80H is applied to the data bus (D_0–D_7) in response to an INTR.
- In response to INTR, the processor outputs the INTA to enable a 74ALS244 three-state octal buffer.
- The octal buffer applies the interrupt vector type number to the data bus in response.
- The vector type number is easily changed with DIP switches shown in this illustration.

A circuit that applies any interrupt vector type number in response to $\overline{\text{INTA}}$. Here the circuit is applying type number 80H.



82C55 interfaced to a keyboard from the microprocessor system using interrupt vector 40H.

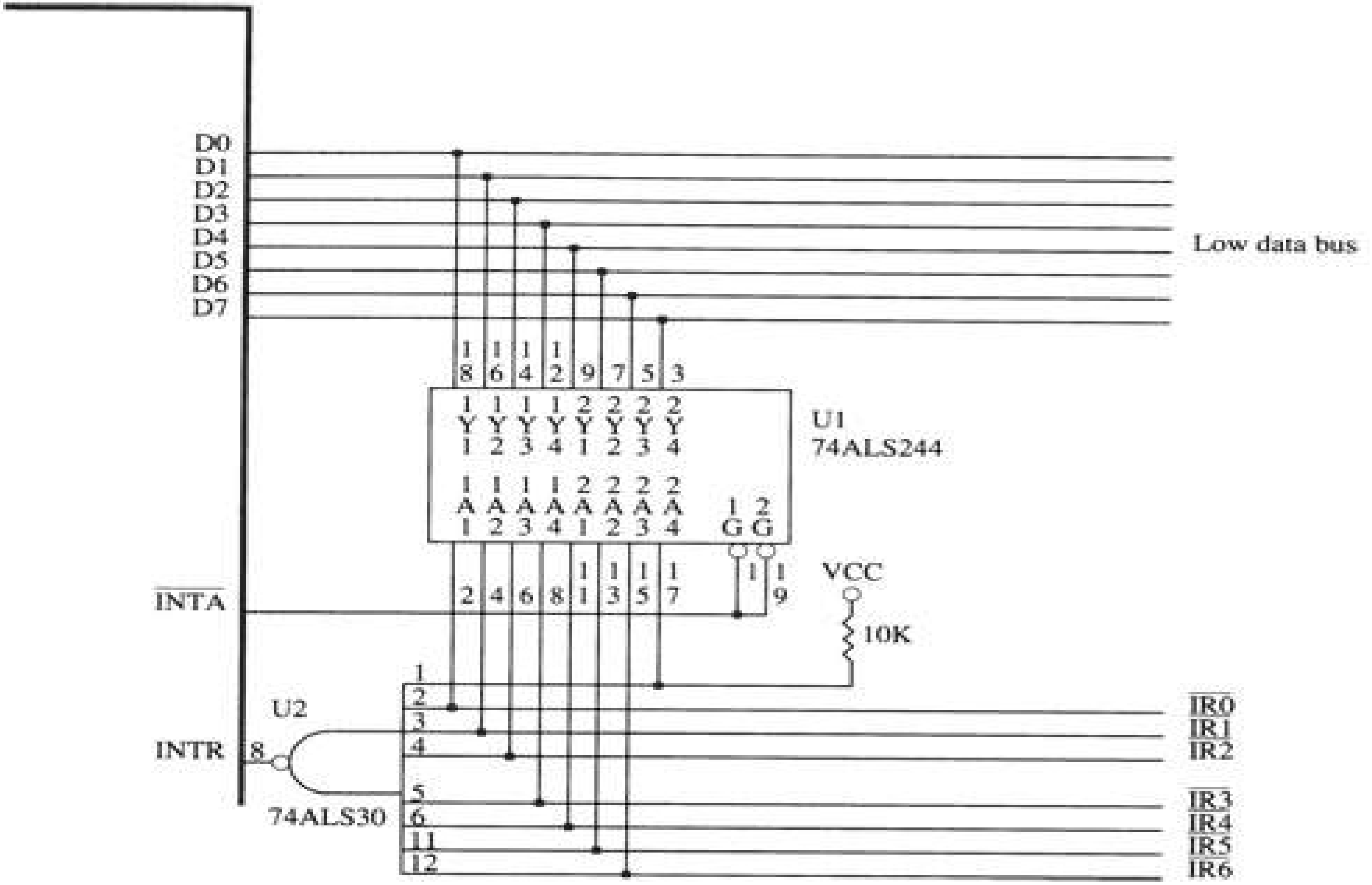


Every time a key is typed, 82C55 requests a type 40H interrupt through the INTR pin

EXPANDING THE INTERRUPT STRUCTURE

- The common methods of expanding the interrupt structure of the processor.
  - Using the 74ALS244 to Expand Interrupts
  - expand the INTR input so it accepts seven interrupt inputs.
- The only hardware change is the addition of an eight-input NAND gate, which provides the INTR signal to the microprocessor when any of the IR inputs becomes active

Expanding the INTR input from one to seven interrupt request lines



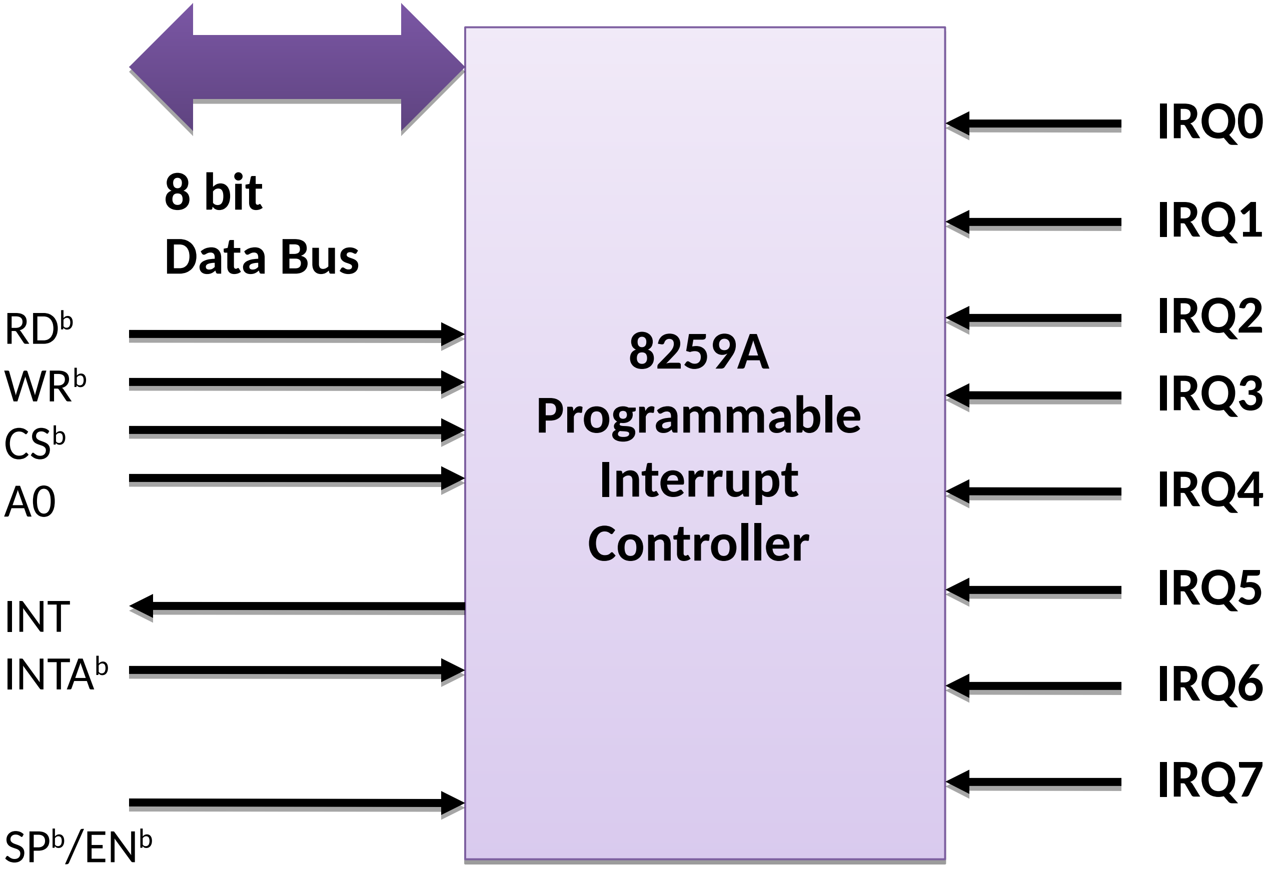
Operation

- If any of the IR inputs becomes logic 0, the output of the NAND gate goes to logic 1 and requests an interrupt through the INTR input.
- The interrupt vector that is fetched during the pulse depends on which interrupt request line becomes active.
  - the interrupt vectors used by a single interrupt request input

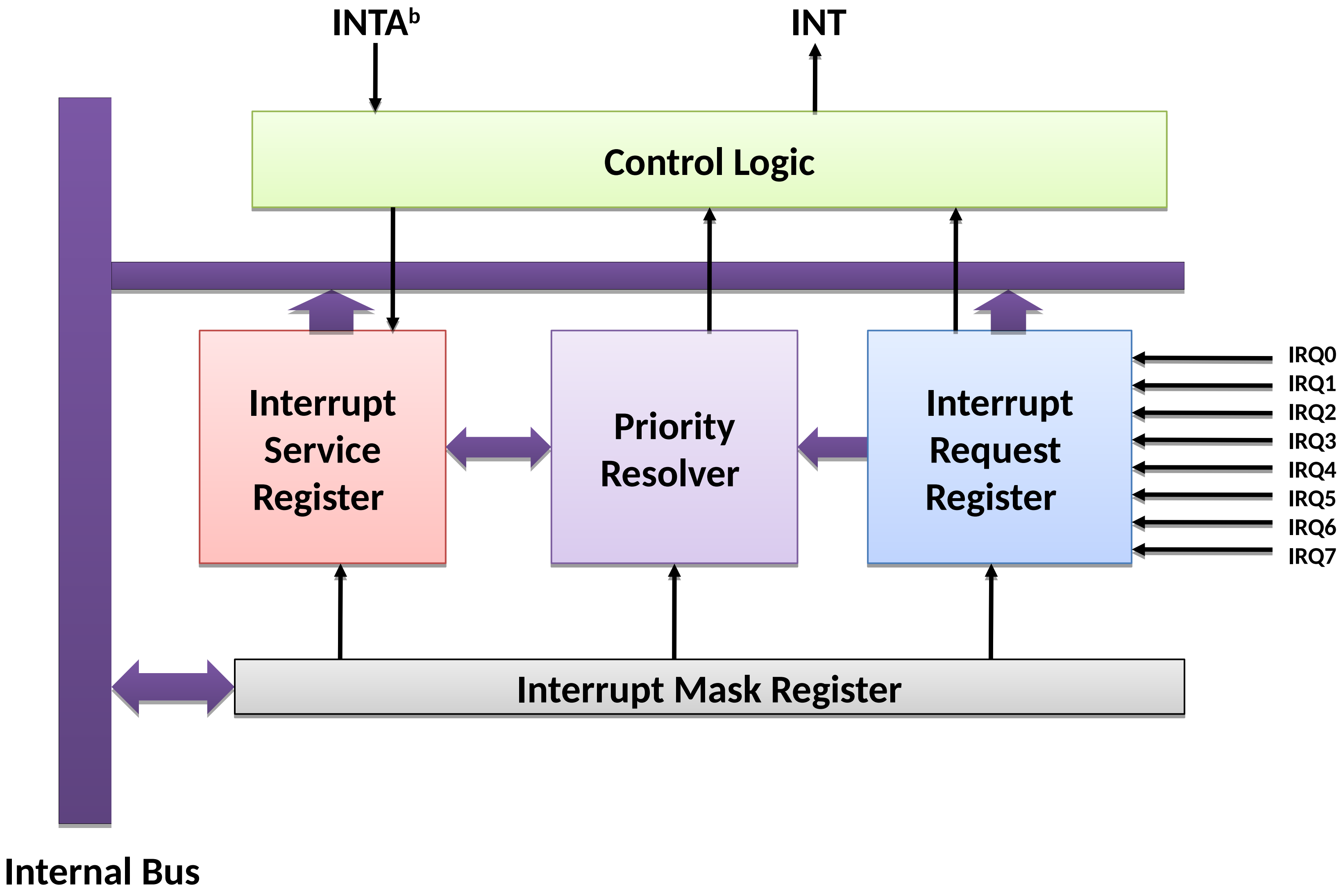
$\overline{\text{IR6}}$	$\overline{\text{IR5}}$	$\overline{\text{IR4}}$	$\overline{\text{IR3}}$	$\overline{\text{IR2}}$	$\overline{\text{IR1}}$	$\overline{\text{IR0}}$	Vector
1	1	1	1	1	1	0	FE h
1	1	1	1	1	0	1	FD h
1	1	1	1	0	1	1	FB h
1	1	1	0	1	1	1	F7 h
1	1	0	1	1	1	1	EF h
1	0	1	1	1	1	1	DF h
0	1	1	1	1	1	1	BF h

What happens if two or more interrupt requests are active, a new interrupt vector is generated?

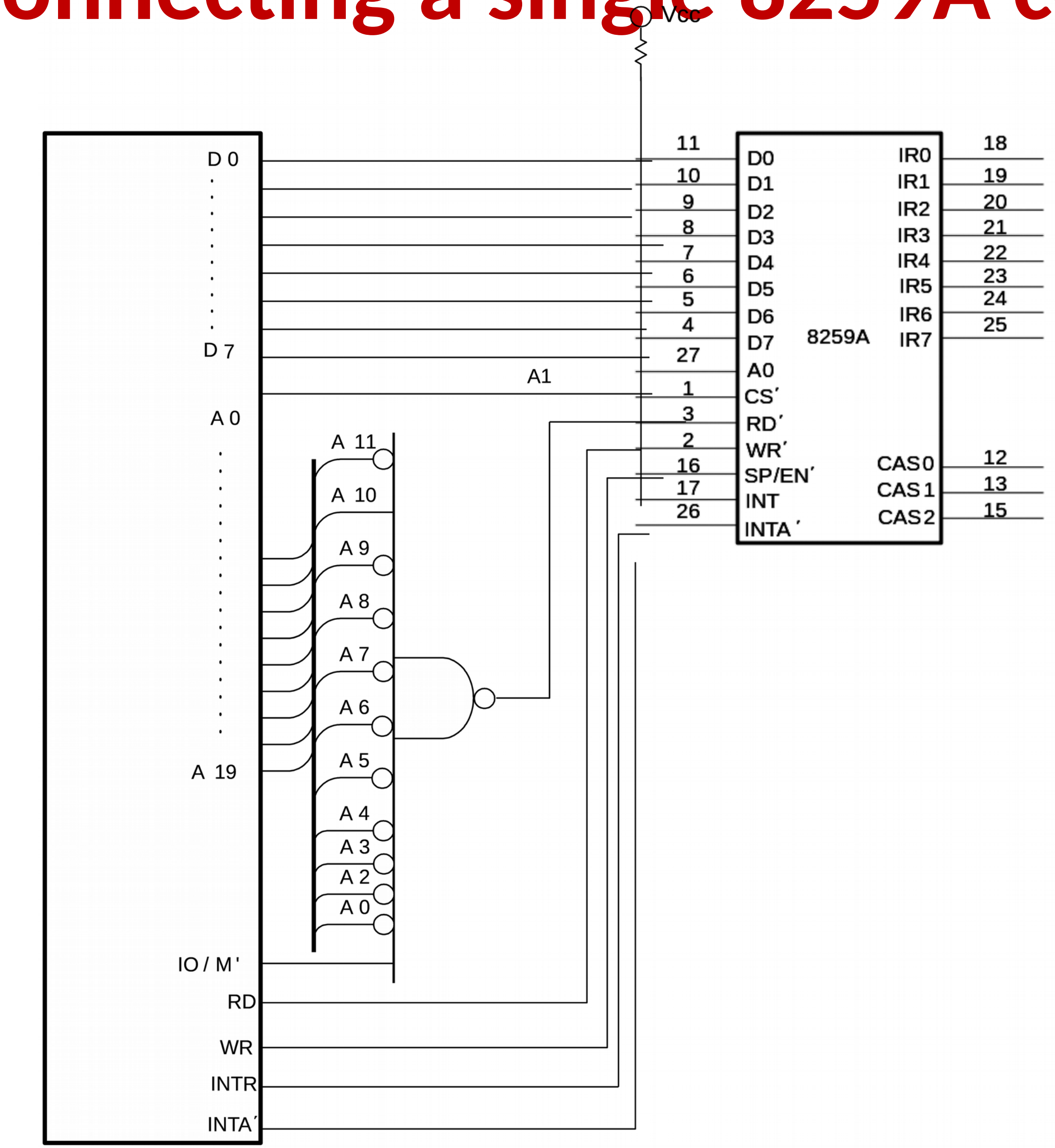
Block Diagram of 8259



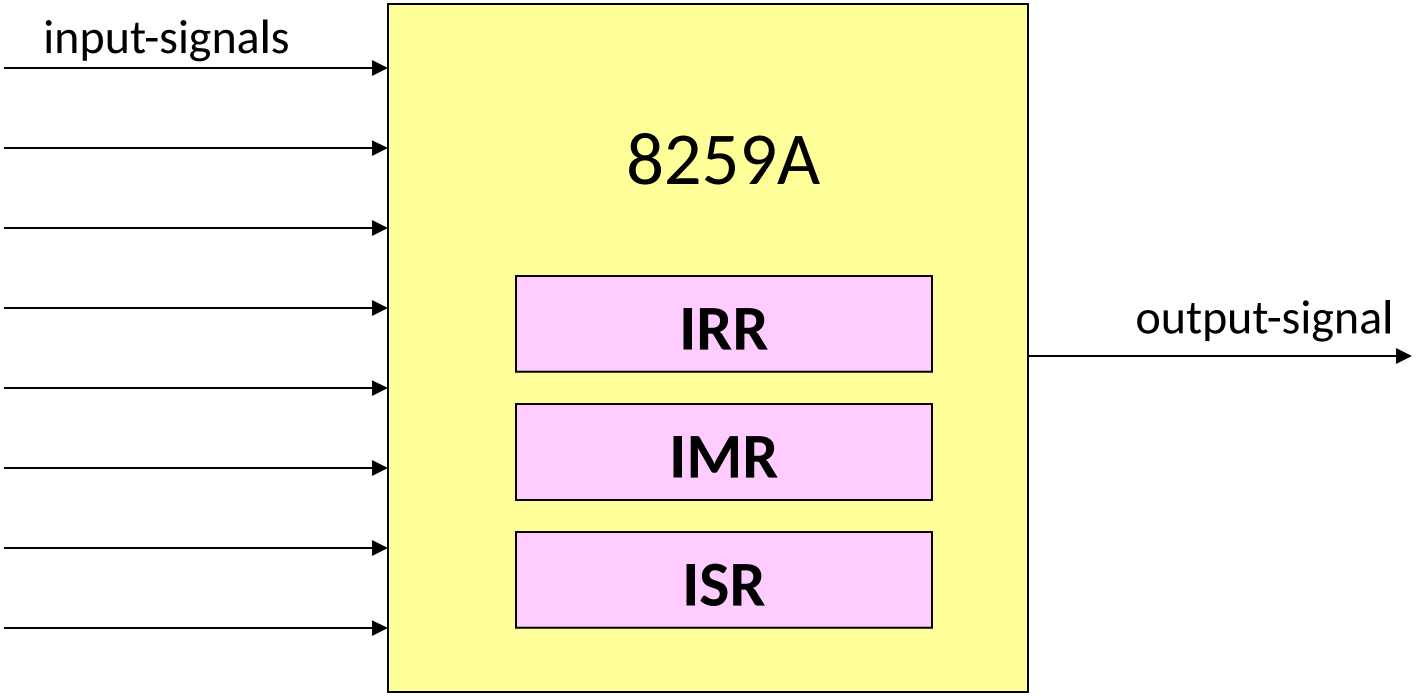
Block Diagram Architecture of 8259



Connecting a single 8259A controller

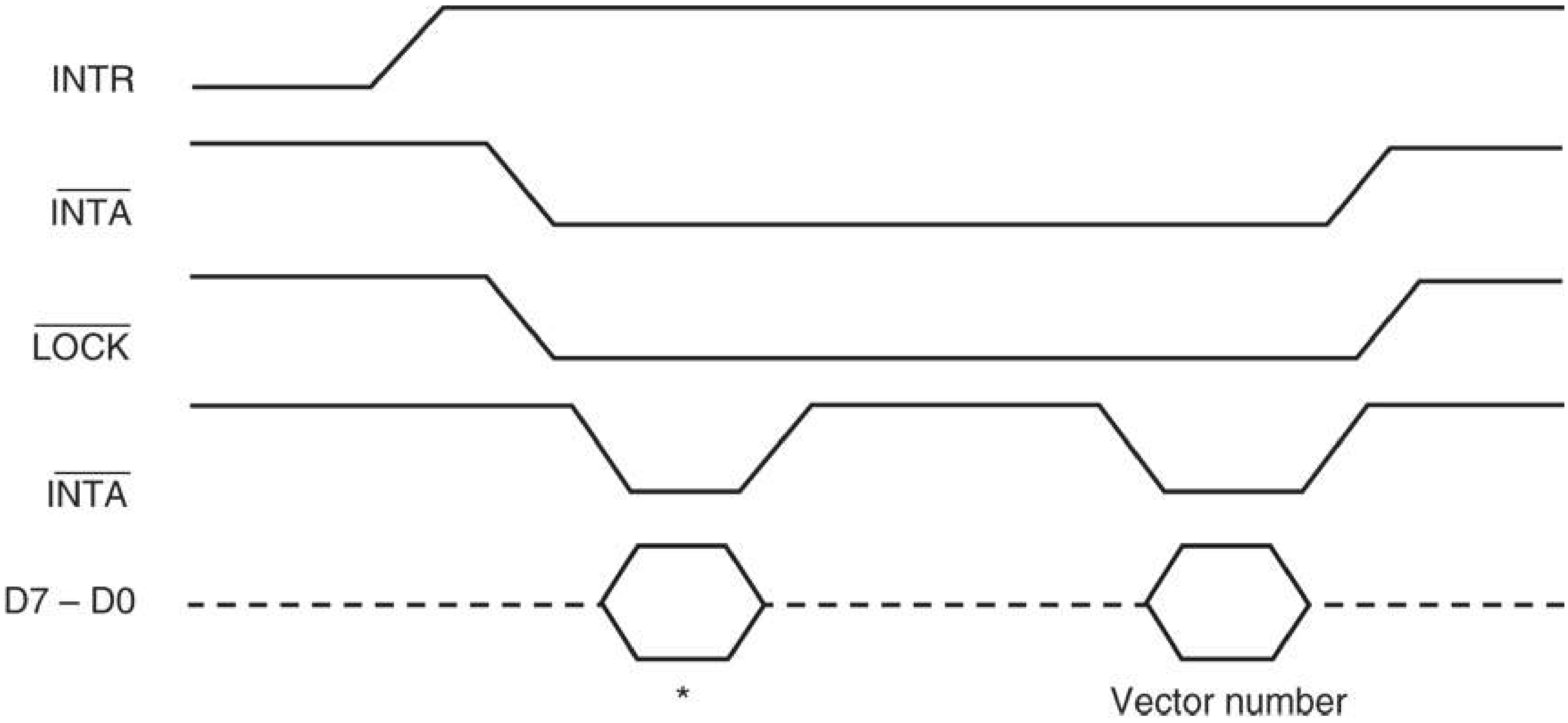


Three internal registers



IRR = Interrupt Request Register
IMR = Interrupt Mask Register
ISR = In-Service Register

- The processor responds to INTR by pulsing INTA output in anticipation of receiving an interrupt vector type number on data bus connections D_7–D_0.
- Two INTA pulses generated by the system insert the vector type number on the data bus.

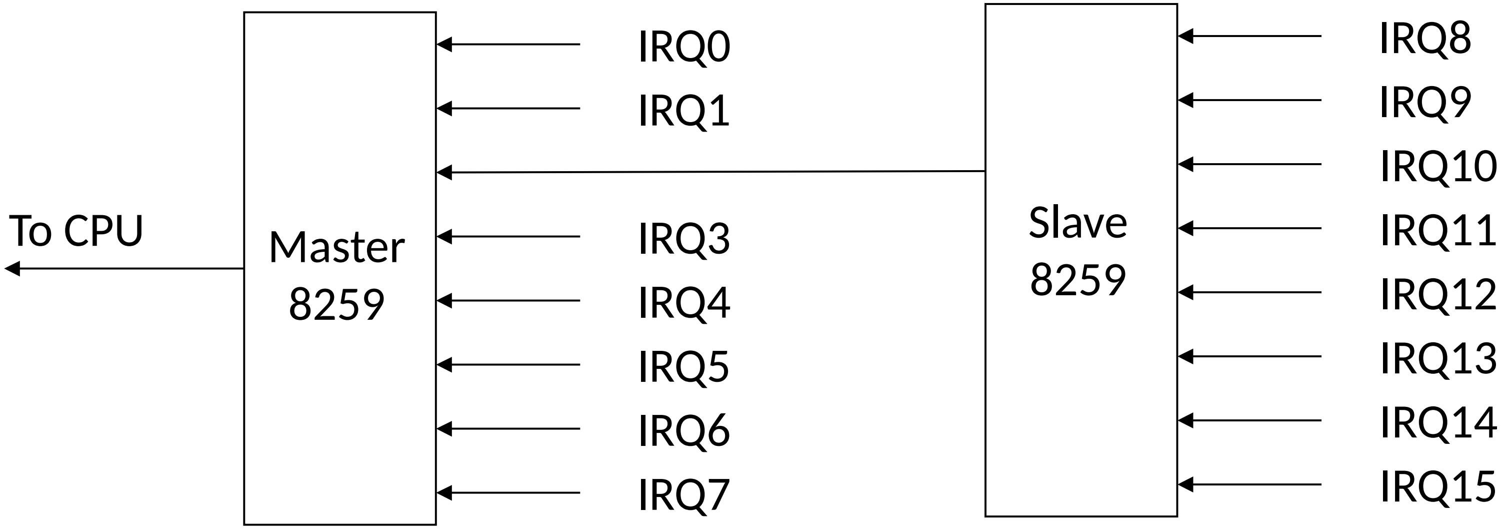


- INTR:
  - The INTR input must be externally decoded
    to select a vector.
  - Intel has reserved interrupts 00H - 1FH for internal and future expansion.
  - Any interrupt vector can be chosen for the INTR pin, but we usually use an interrupt
    type number between 20H and FFH.

- $\overline{\text{INTA}}$:
  - it is an output used in response to INTR input
    to apply a vector type number to the data bus connections D_7–D_0

Adding Interrupts to the Hardware

- Programmable Interrupt Controller: The 8259A chip
  - Each Chip supports eight interrupt request (IRQ) lines
  - Asserts INTR to CPU, responds to resulting INTA# with an 8-bit interrupt vector (“0xdd”) on the data bus
- Industry Standard Architecture (ISA) Bus
  - Priority: highest to lowest order is IRQ0-1, IRQ8-15, IRQ3-7



Priority Modes

- Fully Nested Modes
  - IR are arranged in IR0-IR7 and Any IR can be assigned Highest or lowest priority IR4=0 (high), IR3=7 (low)
- Automatics Rotation Mode
  - A device after being served, receive the lowest priority with value 7 01234567🖧12345670🖧 23456701
- Specific Rotation Mode
  - User can select any IR for lowest priority 06734512🖧67345120🖧 73451206
- EOI: End of interrupt
  - Specific EOI Command
  - Automatic EOI: no command necessary
  - Non-Specific EOI: it resets the ISR bit

Control Word (initialization)

CS	A0	Initialization
0	0	ICW1
0	1	ICW2,ICW3,ICW4
1	X	Not Address

ICW1 & ICW2

AD0	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	1	LTIM	0	SGNL	IC4
	0 for x86				1 for Level Trigger 0 for Edge Trigger		1=single 0=Cascade	

AD0	D7	D6	D5	D4	D3	D2	D1	D0
1	T7	T6	T5	T4	T3	T2	T1	T0
	T7=T0 is the assign to IR0, Vector address for ISR							

Masking and Prioritization

- OCW (operation command word)

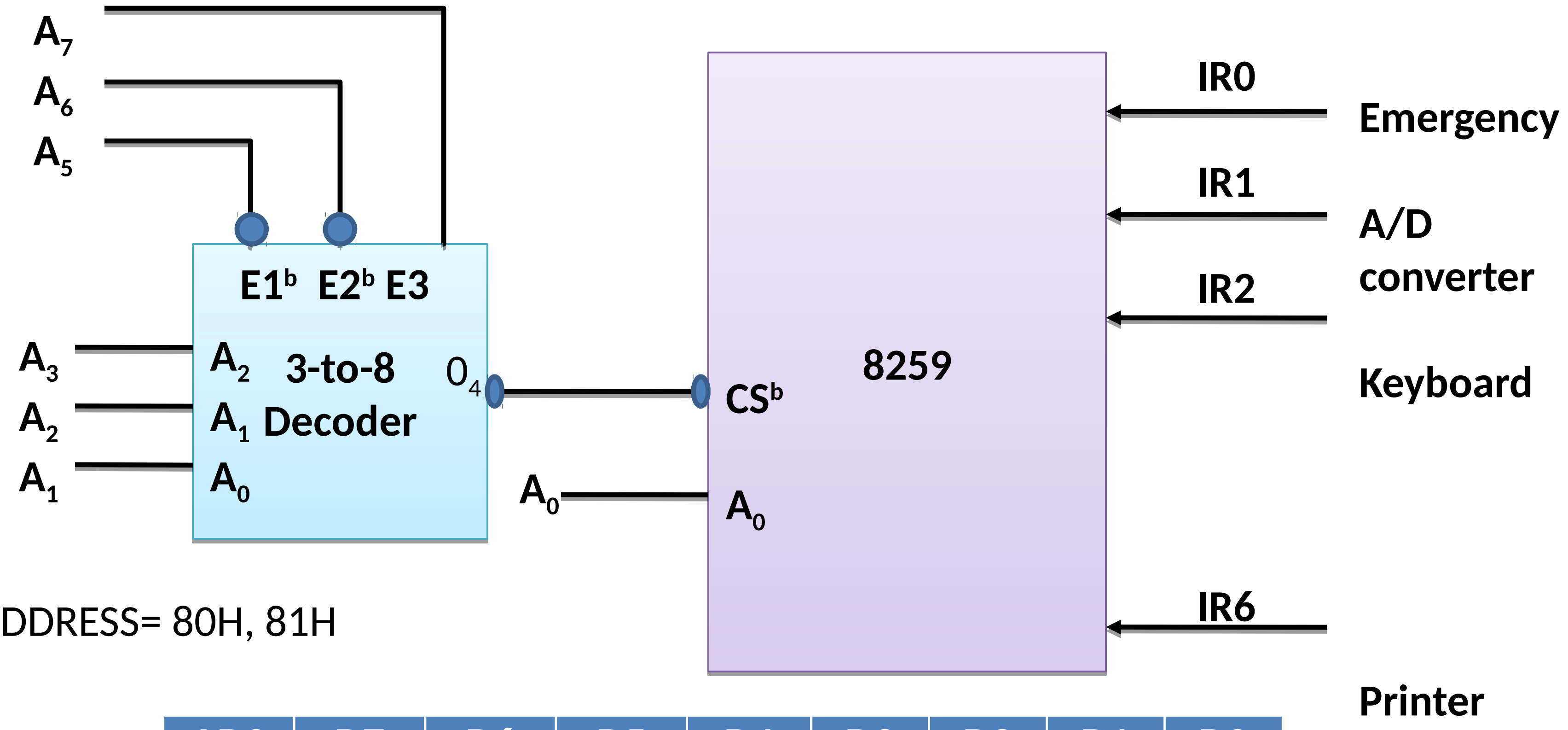
CS	A0	Operation Command Word
0	0	OCW1
0	1	OCW2,OCW3,OCW4
1	X	Not Address

Programming OCWs: OCW1, OCW2

AD0	D7	D6	D5	D4	D3	D2	D1	D0
1	M7	M6	M5	M4	M3	M2	M1	M0
	Interrupt Masks: 1= Mask Set, 0 =Mask reset							

AD0	D7	D6	D5	D4	D3	D2	D1	D0
0	R	SL	EOI	0	0	L2	L1	L0
	Roteate	Specific	EOI			IR Level to be acted Upon (0-7)		

Example: Setting of control word



ADDRESS= 80H, 81H

AD0	D7	D6	D5	D4	D3	D2	D1	D0
1	M7= 0	M6= 1	M5= 1	M4= 1	M3= 1	M2= 1	M1= 1	M0= 1
	Interrupt Masks: 1= Mask Set, 0 =Mask reset							

OCW1=7F

Initialization words (ICW1 & ICW2)

AD0	D7	D6	D5	D4	D3	D2	D1	D0
0	0	1	1	1	0	1	1	0
	A7, A6,A5 Lower address bit of Vector Address				0 for Edge Trigger	Call Address interval =4	1=single 0=Cascade	

76H

AD0	D7	D6	D5	D4	D3	D2	D1	D0
1	T7	T6	T5	T4	T3	T2	T1	T0
	0	0	1	0	0	0	0	0
	T7=T0 is the assign to IR0, Vector address for ISR Lower Byte of call address							

20H

Vector Address 2060, 2064….	**0100 0000**	**0 1 1**	**00000**

Program to initialize

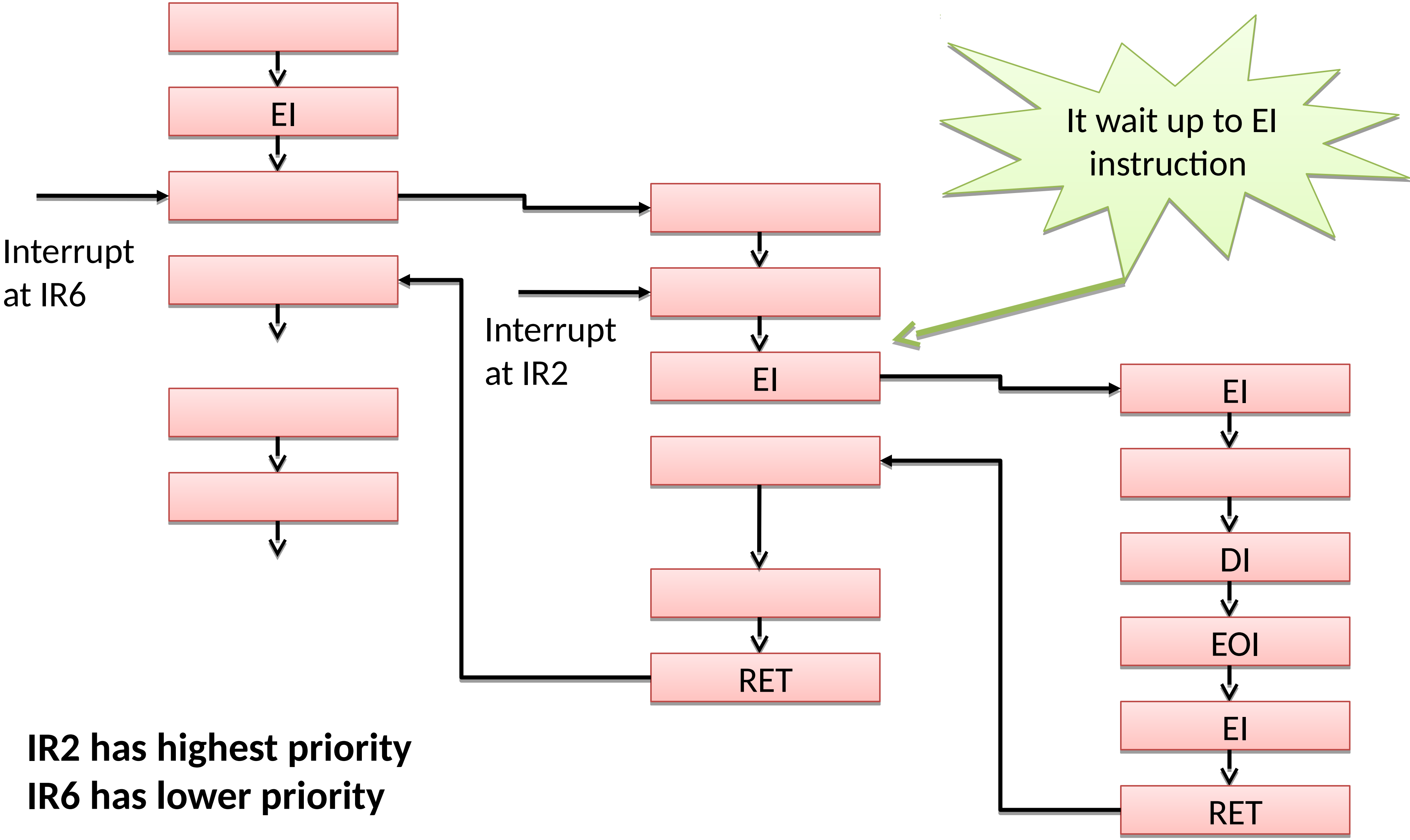
```
DI
MVI A, 76H   ;move ICW1 byte to ACC
OUT 80H   ; initialize 8259A ICW1
MVI A, 20H   ; mov ICW2 byte to  ACC
OUT 81H   ; Initialize 8259A ICW2

MVI A, 7FH   ; Put the OCW1
OUT 80H
```

Nested mode

- By Default 8259 work in Nested modes
  - Unless we put a different OCW
- Suppose IR2 has highest priority and IR6
- IR6 is being serviced
- IR2 can be nested iff IR6 IRS issue an EI command
- Address of IR2=2068, IR2=2074

Nested Interrupt process



IR2 has highest priority
IR6 has lower priority

Maskable Interrupt

- Those interrupt service can be temporarily disable to let the higher priority interrupt ISR to be executed without interruption
- I want IR7 to be Non Maskable

AD0	D7	D6	D5	D4	D3	D2	D1	D0
1	M7= 0	M6= 1	M5= 1	M4= 1	M3= 1	M2= 1	M1= 1	M0= 1
	Interrupt Masks: 1= Mask Set, 0 =Mask reset							

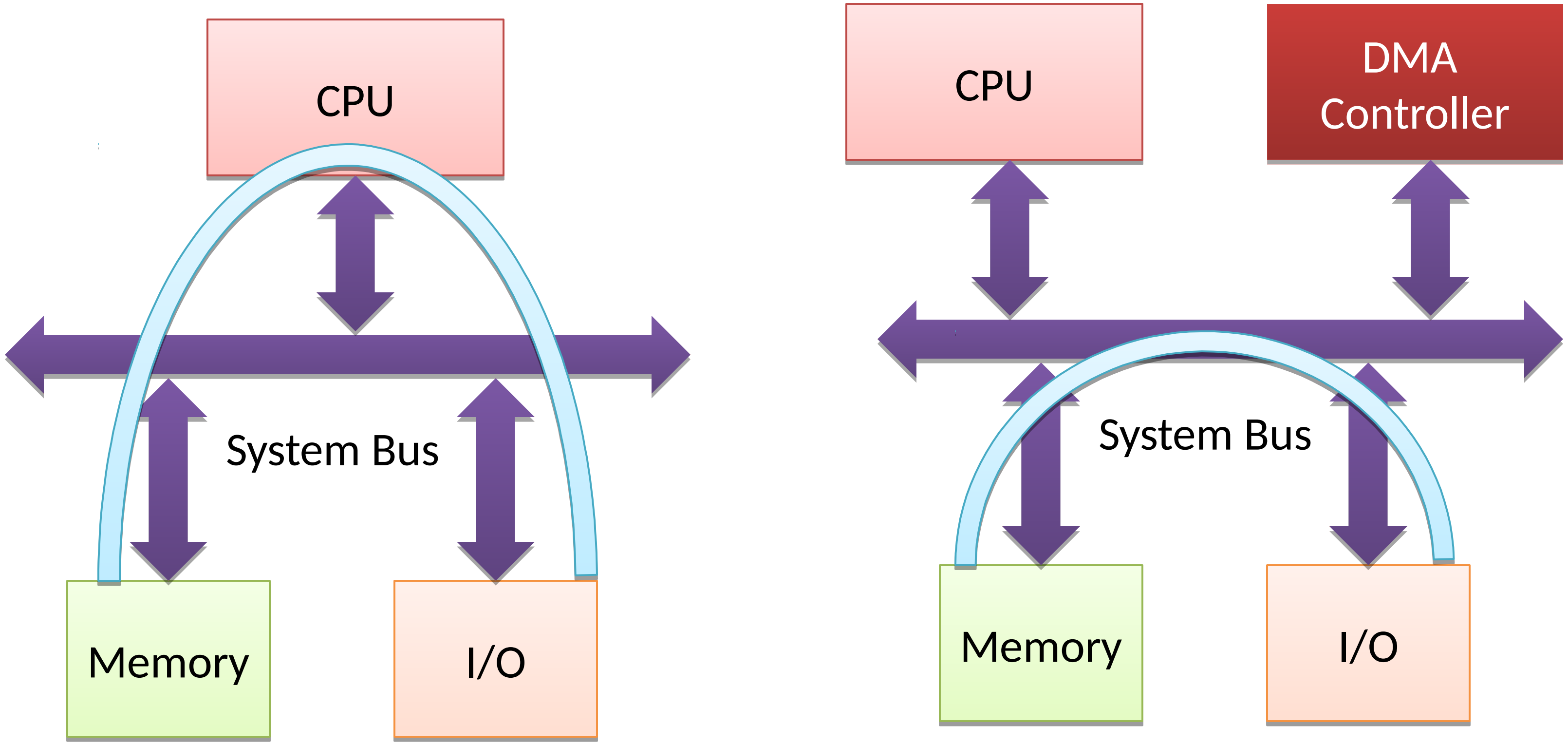
OCW1=7F

DMA Controller (8237)

Outline

- DMA controller
- DMA Architecture
  - Registers
- Introduction to Programming DMA
  - How to configure

Data Transfer DMA mode



Data Transfer : DMA

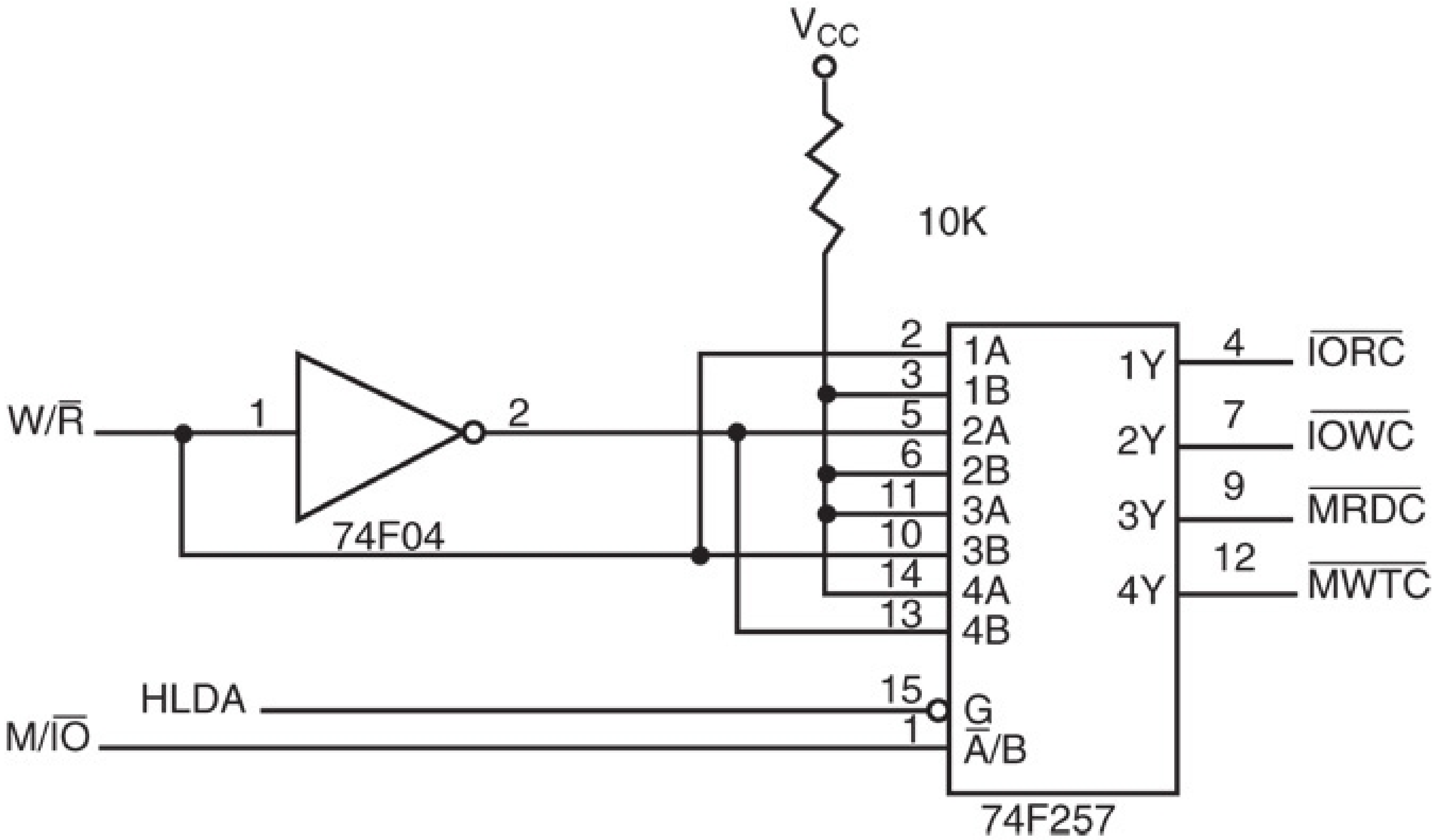
- Problems with programmed I/O
  - Processor wastes time polling
  - Lets take example of Key board
    - Waiting for a key to be pressed,
    - Waiting for it to be released
    - May not satisfy timing constraints associated with some devices : **Disk read or write**
- DMA
  - Frees the processor of the data transfer responsibility

Basic DMA Definitions

- Direct memory accesses normally occur between an I/O device and memory without the use of the microprocessor.
  - a **DMA read** transfers data from the memory to the I/O device
  - A **DMA write** transfers data from an I/O device to memory
- Memory & I/O are controlled simultaneously.
  - which is why the system contains separate memory and I/O control signals

- A DMA read causes the $\overline{\text{MRDC}}$ and $\overline{\text{IOWC}}$ signals to activate simultaneously.
  - transferring data from memory to the I/O device
- A DMA write causes the $\overline{\text{MWTC}}$ and $\overline{\text{IORC}}$ signals to both activate.
- The DMA controller provides memory with its address, and controller signal (DACK) se$\overline{\text{lects the}}$ I/O device during the transfer.
- 8086/8088 require a controller or circuit (next slide) for control bus signal generation.

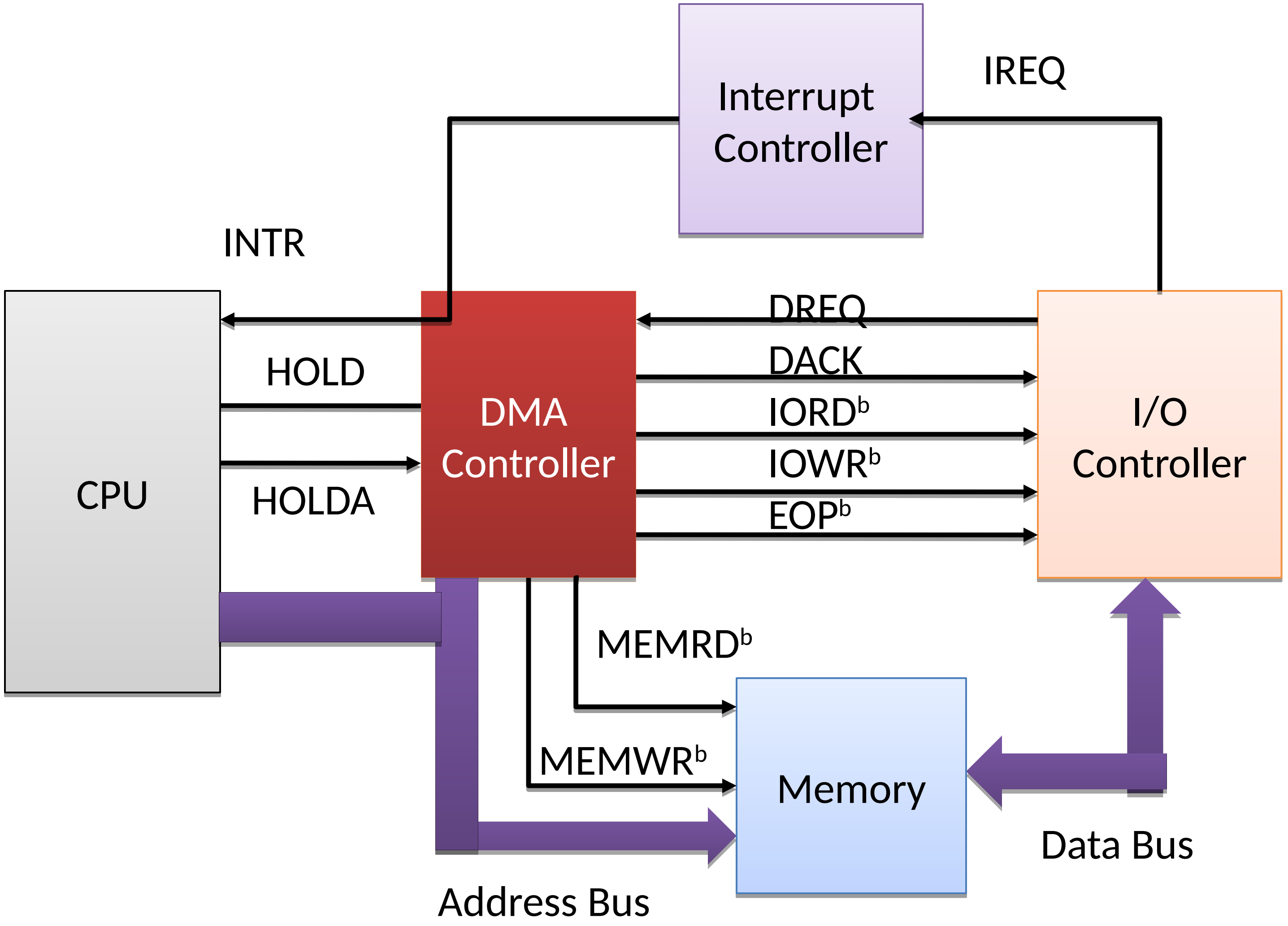
A circuit that generates system control signals in a DMA environment.



- Data transfer speed is determined by speed of the memory device or a DMA controller.
  - if memory speed is 50 ns, DMA transfers occur at rates up to 1/50 ns or 20 M bytes per second
  - if the DMA controller functions at a maximum rate of 15 MHz with 50 ns memory, maximum transfer rate is 15 MHz because the DMA controller is slower than the memory

- In many cases, the DMA controller *slows* the speed of the system when transfers occur.

DMA Controller



DMA Controller

- DMA is implemented using a DMA controller
- DMA controller
  - Acts as slave to processor
  - Receives instructions from processor

Example:

- Reading from an I/O device
- Processor gives details to the DMA controller
  - I/O device number
  - Main memory buffer address
  - Number of bytes to transfer
  - Direction of transfer (memory → I/O device, or vice versa)

DMA: HOLD and HOLDA

- HOLD: DMA to CPU
  - DMA Send HOLD High to CPU
    - I (DMA) want BUS Cycles
- HOLDA
  - CPU send HOLDA
    - BUS is granted to DMA to do the transfer
    - DMA became Slave to Master mode
- HOLD Low to CPU
  - I (DMA) finished the transfer
- Cycle Stealing if One BUS
- Other wise Separate process independent of processing

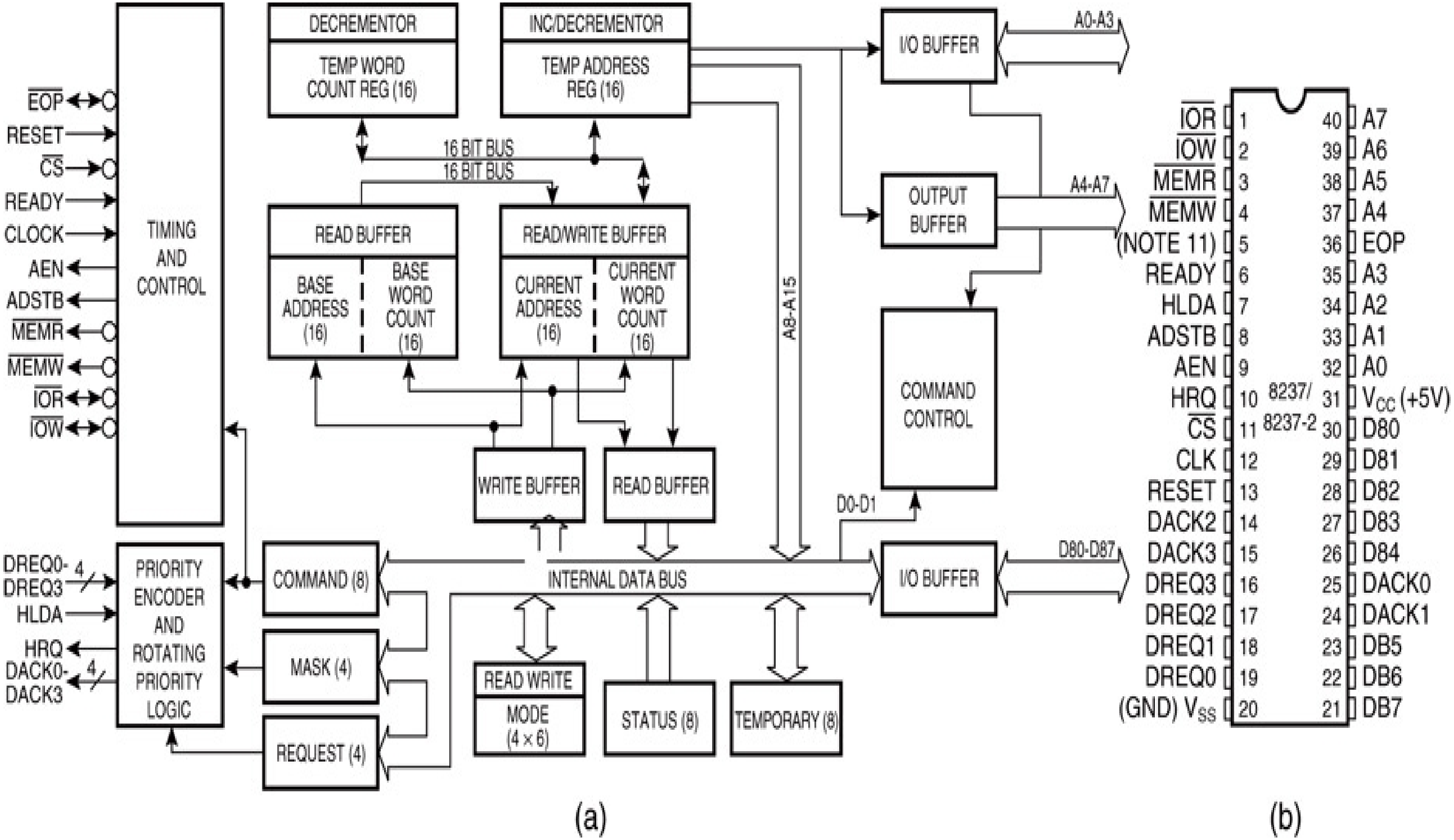
Steps in a DMA operation

- Processor initiates the DMA controller
  - Gives device number, memory buffer pointer, … Called ***channel initialization***
  - Once initialized, it is ready for data transfer
- When ready, I/O device informs the DMA controller
  - DMA controller starts the data transfer process
    - » Obtains bus by going through bus arbitration
    - » Places memory address and appropriate control signals
    - » Completes transfer and releases the bus
    - » Updates memory address and count value
    - » If more to read, loops back to repeat the process
- Notify the processor when done

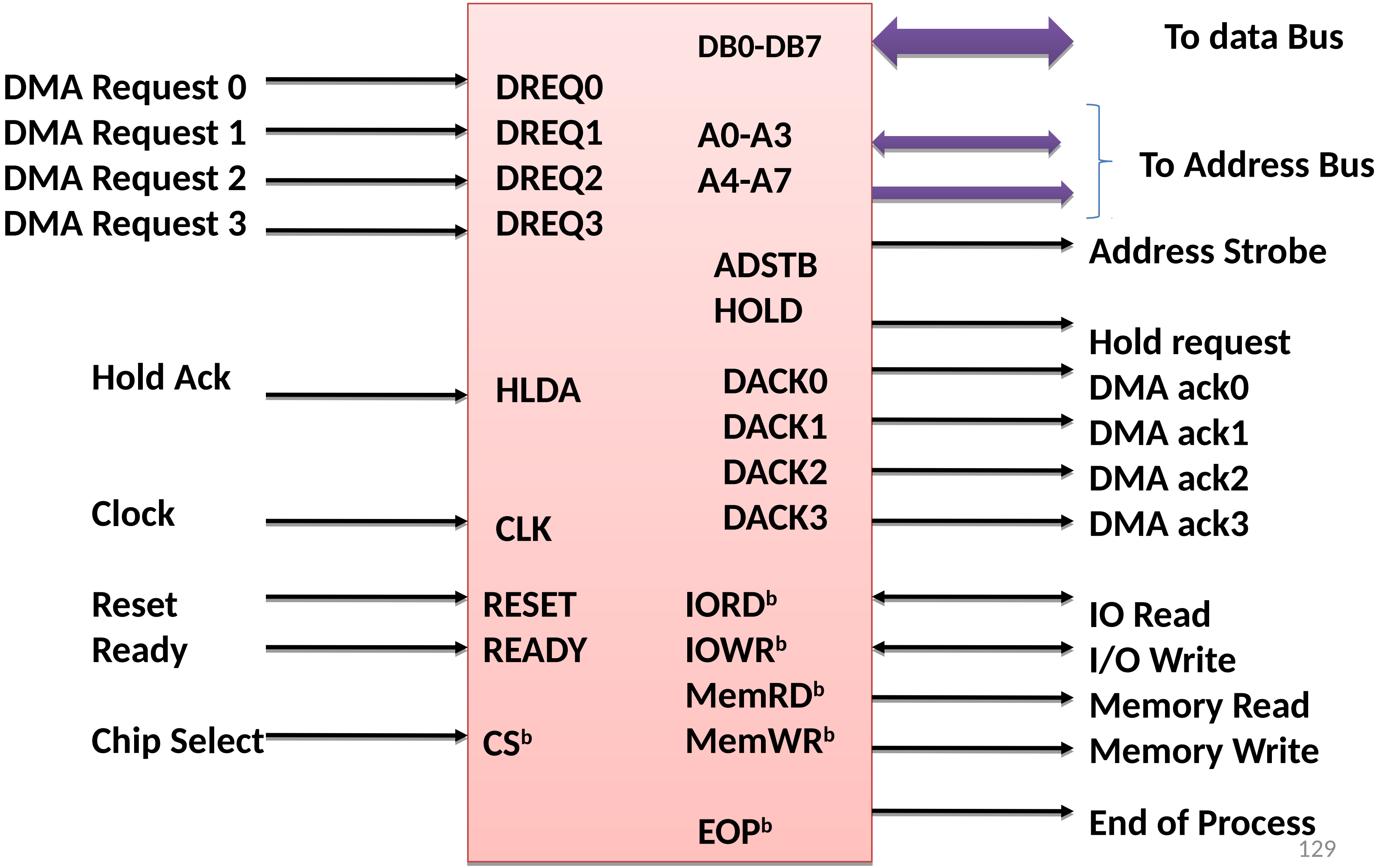
8237 DMA Controller

- Enable/Disable control of Individual DMA Req
- Four Independent DMA Channel
- Independent Auto initialization for all channel
- Memory to Memory transfer
- Memory Block initialization
- Address Increment and Decrement
- Cascade (Directly expandable to any #CHN)
- End of Process input for terminating transfers
- Software DMA requests

The 8237A programmable DMA controller. (a) Block diagram and (b) pin-out. (Courtesy of Intel Corporation.)



8237 DMA Controller



DB0-DB7 ⟷ To data Bus

DMA Request 0 → DREQ0

DMA Request 1 → DREQ1

DMA Request 2 → DREQ2

DMA Request 3 → DREQ3

A0-A3 ⟷ To Address Bus

A4-A7 → To Address Bus

ADSTB → Address Strobe

HOLD → Hold request

Hold Ack → HLDA

DACK0 → DMA ack0

DACK1 → DMA ack1

DACK2 → DMA ack2

DACK3 → DMA ack3

Clock → CLK

Reset → RESET

Ready → READY

IORD^b ⟷ IO Read

IOWR^b ⟷ I/O Write

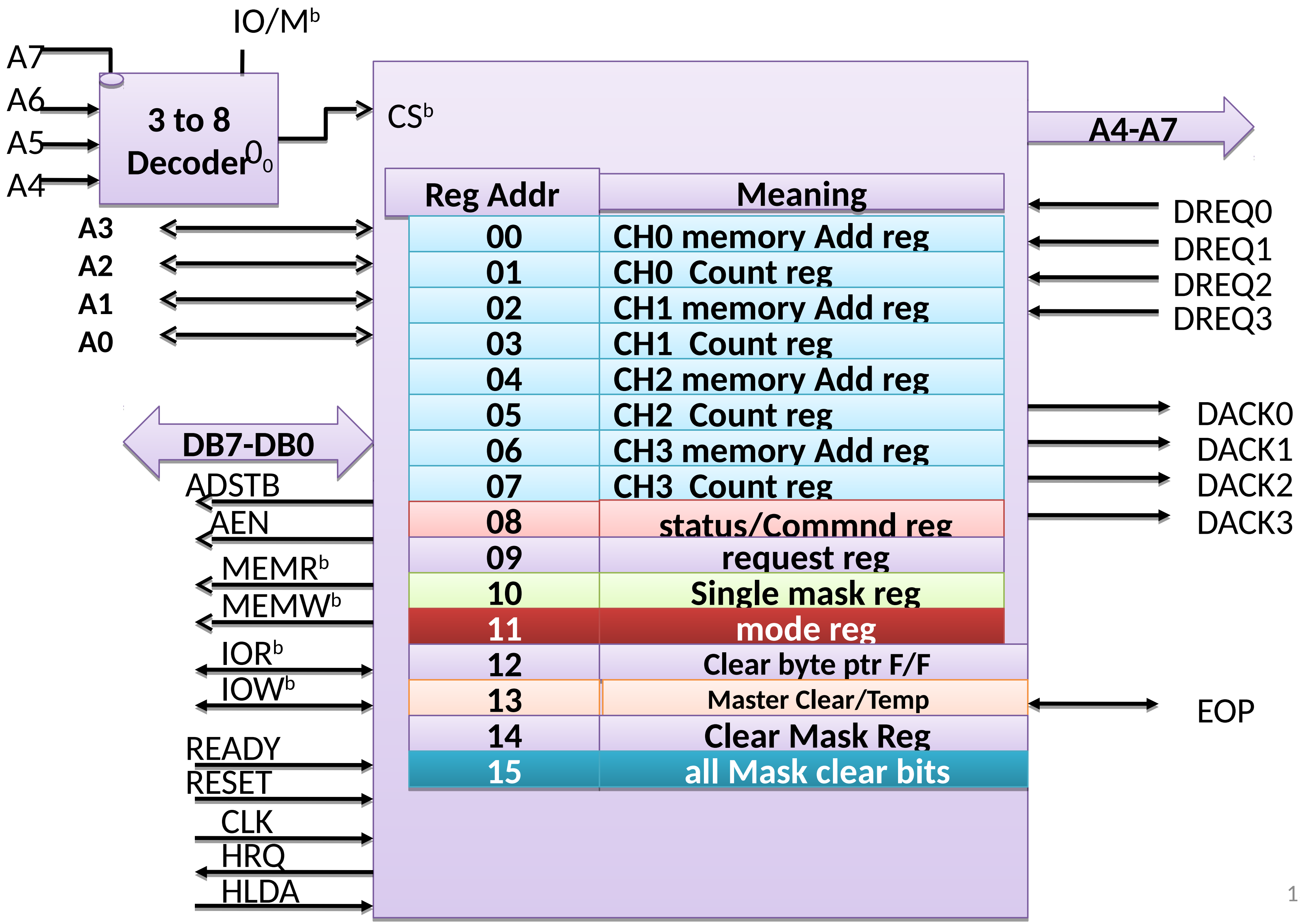
MemRD^b → Memory Read

MemWR^b → Memory Write

Chip Select → CS^b

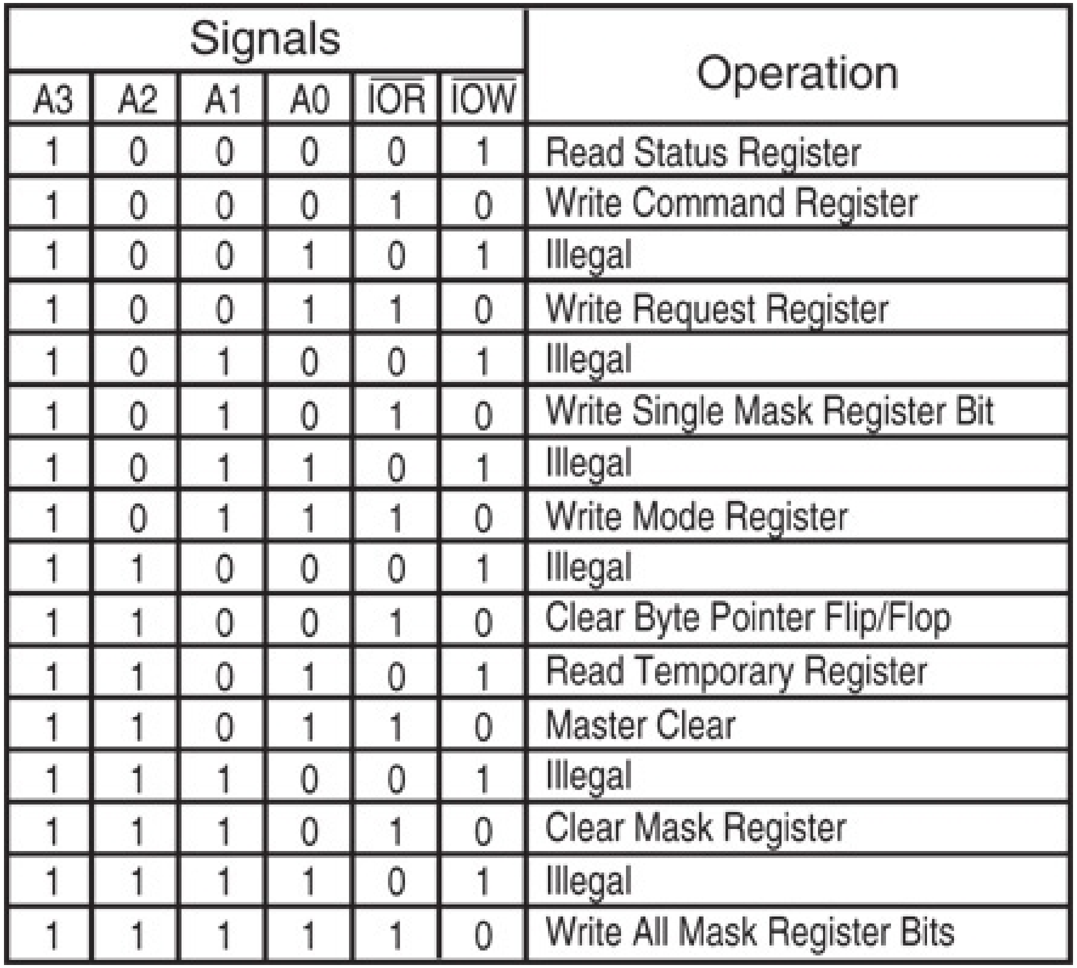
EOP^b → End of Process

DMA Controller with Internal Registers



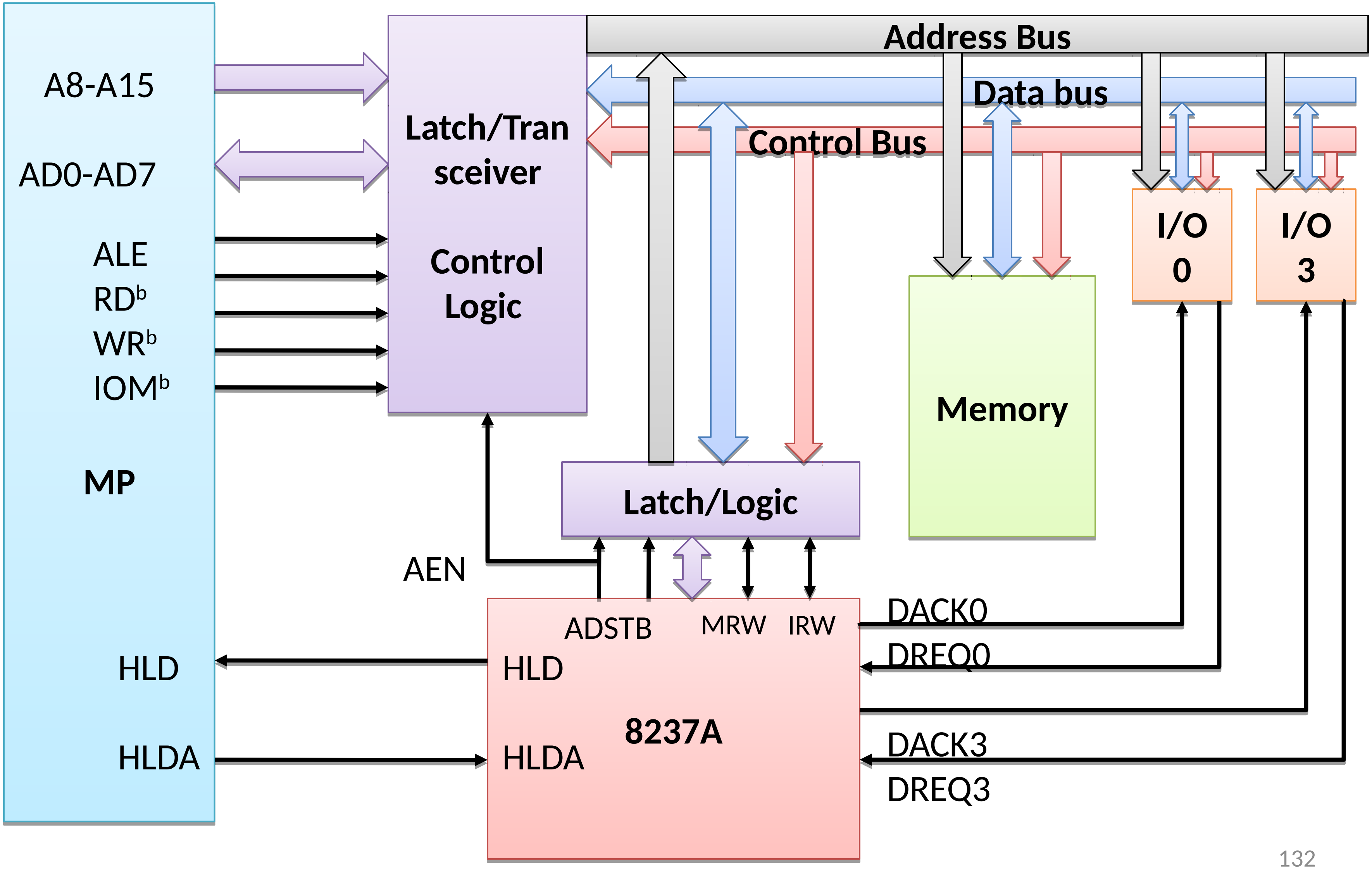
Reg Addr	Meaning
00	CH0 memory Add reg
01	CH0 Count reg
02	CH1 memory Add reg
03	CH1 Count reg
04	CH2 memory Add reg
05	CH2 Count reg
06	CH3 memory Add reg
07	CH3 Count reg
08	status/Commnd reg
09	request reg
10	Single mask reg
11	mode reg
12	Clear byte ptr F/F
13	Master Clear/Temp
14	Clear Mask Reg
15	all Mask clear bits

Figure 13–10 8237A-5 command and control port assignments. (Courtesy of Intel Corporation.)



Signals						Operation
A3	A2	A1	A0	$\overline{IOR}$	$\overline{IOW}$	
1	0	0	0	0	1	Read Status Register
1	0	0	0	1	0	Write Command Register
1	0	0	1	0	1	Illegal
1	0	0	1	1	0	Write Request Register
1	0	1	0	0	1	Illegal
1	0	1	0	1	0	Write Single Mask Register Bit
1	0	1	1	0	1	Illegal
1	0	1	1	1	0	Write Mode Register
1	1	0	0	0	1	Illegal
1	1	0	0	1	0	Clear Byte Pointer Flip/Flop
1	1	0	1	0	1	Read Temporary Register
1	1	0	1	1	0	Master Clear
1	1	1	0	0	1	Illegal
1	1	1	0	1	0	Clear Mask Register
1	1	1	1	0	1	Illegal
1	1	1	1	1	0	Write All Mask Register Bits

DMA interface to I/O Device



8237 DMA channels

- 8237 supports four DMA channels
- Handshake of I/O
  - DREQ3-DREQ0 (DMA Request)
  - DACK3-DACK0 (DMA Acknowledge)
- AEN & ADSTB : Address Enable & Add Strobe
- A3-A0 and A4-A7: Address line
- HRQ and HLDA: Hold Request and Hold Ack
  - Request hold BUS

Registers of 8237A

- Command registers

D7	D6	D5	D4	D3	D2	D1	D0
0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1
DACK Sense Active Low/High	DREQ Sense Active Low/High	Late/Extended Write Selection	Fixed/ Rotating Priority	Normal/ Compressed Timing	Enable/Disable Controller	Dis/En CH0 Addres Hold	Dis/En Mem-mem Transfer

- Mode register

D7	D6	D5	D4	D3	D2	D1	D0
		0/1	0/1				
00=Demand Mode 01=Single Mode 10=Block Mode 11=Cascade mode		INC/DEC Address	Dis/Ena Auto Initialisation	00=verify transfer 01=Write Transfer 10=Read transfer 11=Illegal		Channel Select 00=CH0, 01=CH1 10=CH2, 11=CH3	

DMA Registers

- Request register

D7	D6	D5	D4	D3	D2	D1	D0
XXXXX Don’t Care					Reset/Set REQ bit	00-11 Channel Select 00=CH0, 01=CH1, 10=CH2, 11 CH3	

- Mask register: Used to disable a specific channel

D7	D6	D5	D4	D3	D2	D1	D0
XXXX Don’t Care				0/1	0/1	0/1	0/1
				Clear/Set CH3 Mask Bit	Clear/Set CH2 Mask Bit	Clear/Set CH1 Mask Bit	Clear/Set CH0 Mask Bit

- Status register

D7	D6	D5	D4	D3	D2	D1	D0
0/1, CH3	0/1,CH2	0/1, CH1	0/1, CH0	0/1, CH3	0/1, CH2	0/1, CH1	0/1, CH0
CH# Request				CH# Has reached TC			

Registers

- Temporary register
  - Used for memory-to-memory transfers
- Current address register
  - One 16-bit register for each channel
  - Holds address for the current DMA transfer
- Current word register
  - Keeps the byte count
  - Generates terminal count (TC) signal when the count goes from zero to FFFFH

Type of Data Transfer using 8237 DMA

- Single transfer
  - Useful for slow devices,
- Block transfer mode
  - Transfers data until TC is generated or external EOP^b signal is received
- Demand transfer mode
  - Similar to the block transfer mode
  - In addition to TC and EOP, transfer can be terminated by deactivating DREQ signal
- Cascade mode
  - Useful to expand the number channels beyond four

- Four steps are required to program the 8237:
  - (1) The F/L flip-flop is cleared using a clear F/L command
  - (2) the channel is disabled
  - (3) LSB & MSB of the address are programmed
  - (4) LSB & MSB of the count are programmed
- Once these four operations are performed, the channel is programmed and ready to use.
  - additional programming is required to select the mode of operation before the channel is enabled and started

Programming the 8237

- Write a control word in Mode registers
  - Select Channel, Type of Transfer (R,W, Verify)
  - DMA mode (block, single byte, demand mode)
- Write a control word in Command registers
  - Priority among channel
  - Enable the 8237, DREQ & DACK active level
- Write the starting address of the data block to be transferred in the Channel MAR
- Write the count in the Channel Count Reg

Transfer 1K of memory from/ to Floppy disk at CH3

- Disable DMA controller & begin initialization Instructions
- Initialize CH3
- Starting address of Memory Block (Say 4075H) and subsequent bytes are increasing address order
- Command parameters: Normal timing, Fixed priority, late write, DREQ&DACK both active low
- Set up the demand mode so that DMA can complete the transfer without any interruption

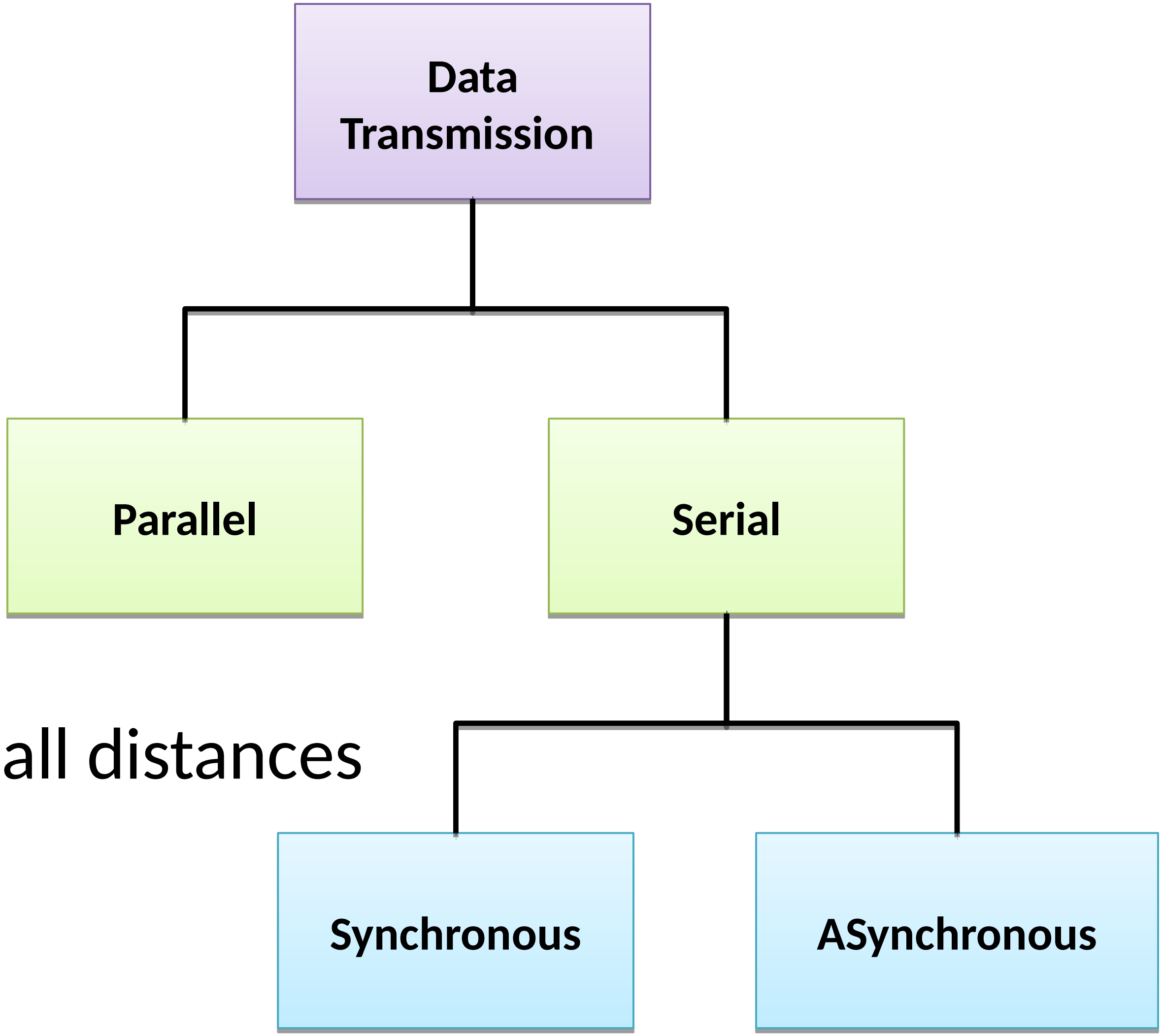
Program to transfer

```
MOV    AL, 0000100B    ; (Disable DMA)
OUT    08H, AL ;  Send to Command Reg
MOV    AL,00000111V   ;00 Demand mode, increAddress,
       0 Disable autoload, 01 write, 11=CH3
OUT    0BH, AL ; Send to Mode Reg
MOV    AL,04H  ; lower Address to
OUT 06H   ; MAR CH3
MOV    AL,75H  ; 7504 Address
OUT 06H, AL
MOV    AL,FFH  ; lower byte of Tc =03FF,  1K Byte
OUT 07H,AL  ; CH3 count Register
MOV    AL,03H  ; higher byte of 03FF
OUT 07H, AL ; CH3 Count Register
MOV    AL, 1000000B    ; Command for DACK High
OUT    08H,AL  ; Send to command Register
```

8251 USART

Data Comm: Serial Vs Parallel

- Serial
  - Cheaper
  - Slower
- Parallel
  - Faster
  - Data skew
  - Limited to small distances

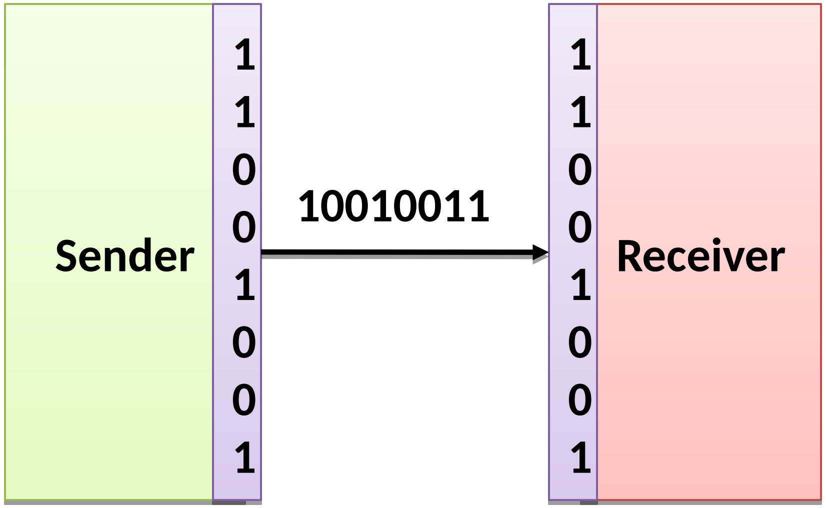


Serial Communication: How ?

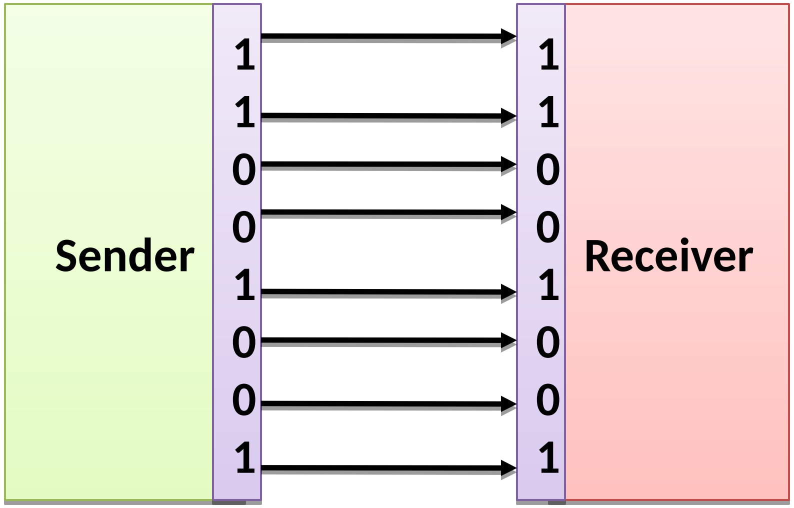
Two basic modes of data transmission

Parallel to serial Conversion

Serial to parallel Conversion



Serial Transmission

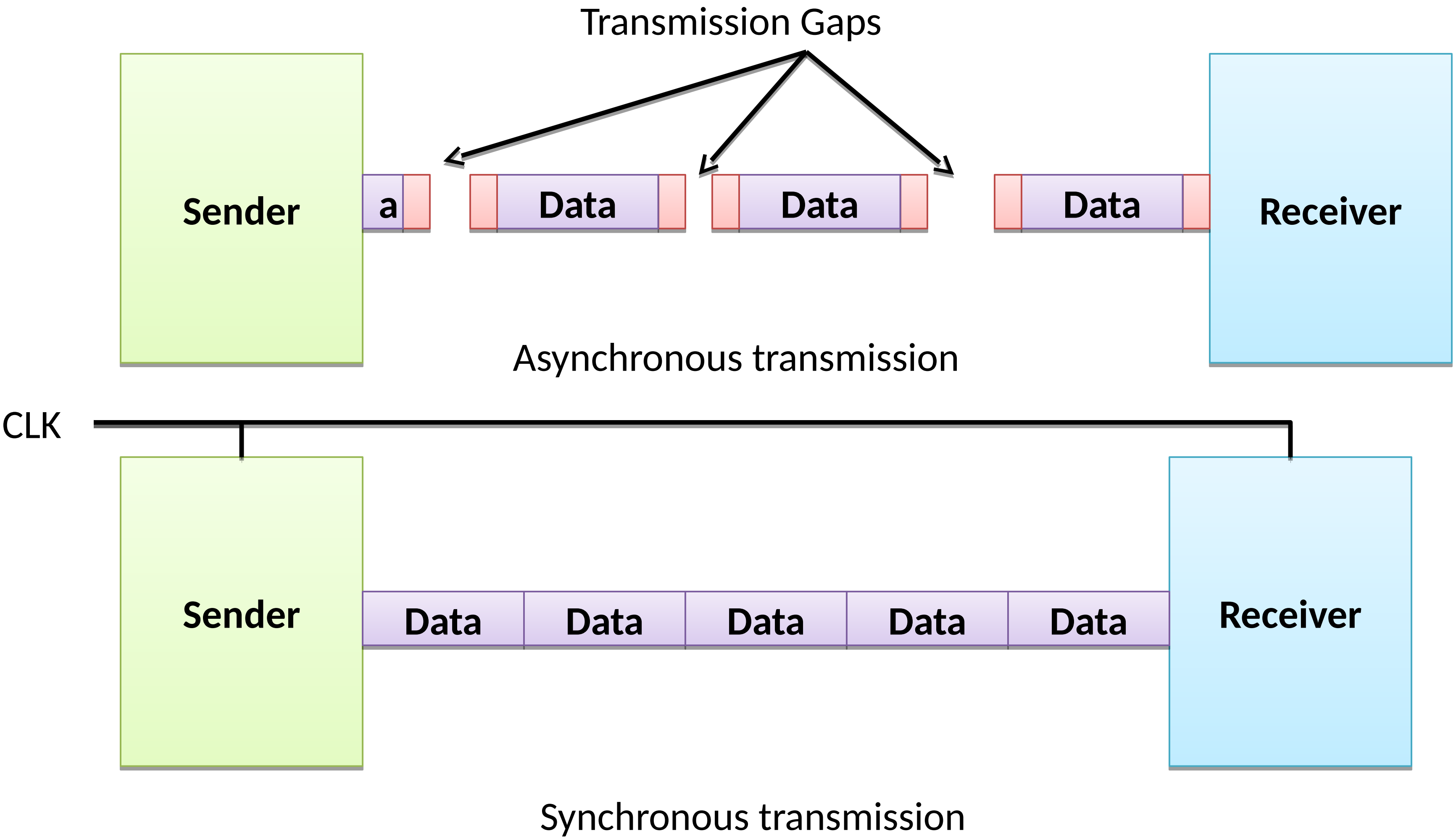


Parallel Transmission

Type of Serial Communication

- Synchronous
  - Sender and receiver must synchronize
    - Done in hardware using phase locked loops (PLLs)
  - Block of data can be sent
  - More efficient : Less overhead than asynchronous transmission
  - Expensive
- Asynchronous
  - Each byte is encoded for transmission
    - Start and stop bits
  - No need for sender and receiver synchronization

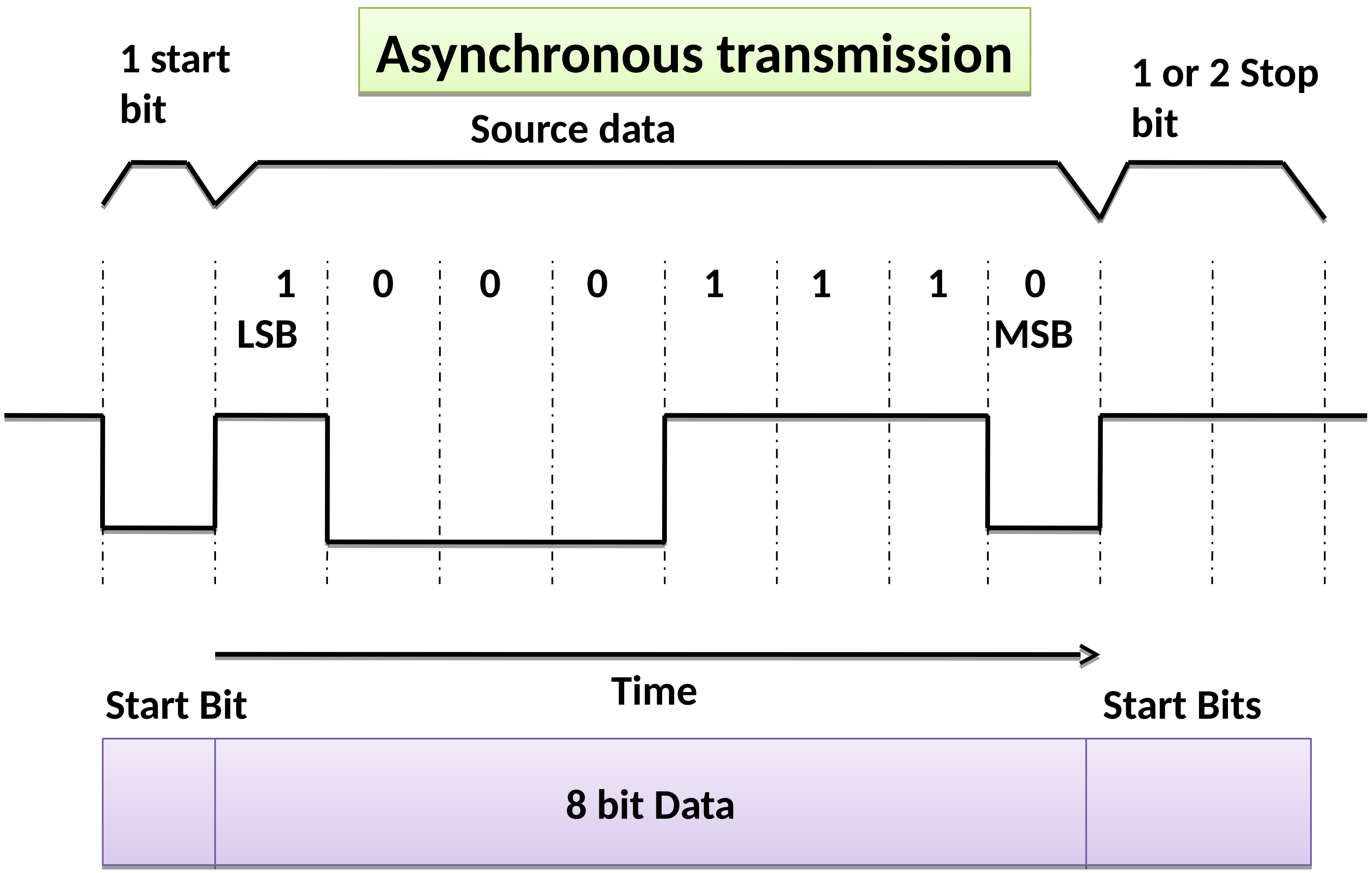
Type of Serial Communication



Asynchronous transmission

Synchronous transmission

Type of Serial Communication



Asynchronous transmission

1 start bit

Source data

1 or 2 Stop bit

1 0 0 0 1 1 1 0

LSB

MSB

Time

Start Bit

Start Bits

8 bit Data

Simplex and Duplex Transmission

- Simplex
  - Data are transmitted in one directions
  - Example: CPU to printer
- Duplex
  - Data flow in both direction
  - Half Duplex (Transmission goes on way at a time)
  - Full Duplex (Both ways simultaneously)

Rate of transmission

- Rate at which bits are transmitted (BAUD)
- Number of signal changes per second
- Bit time: how long the Bit stay On or Off
- Printer, Terminal Baud Adjustable (50-9600)
- 1200Baud means: Bit stay for 1/1200=0.83ms

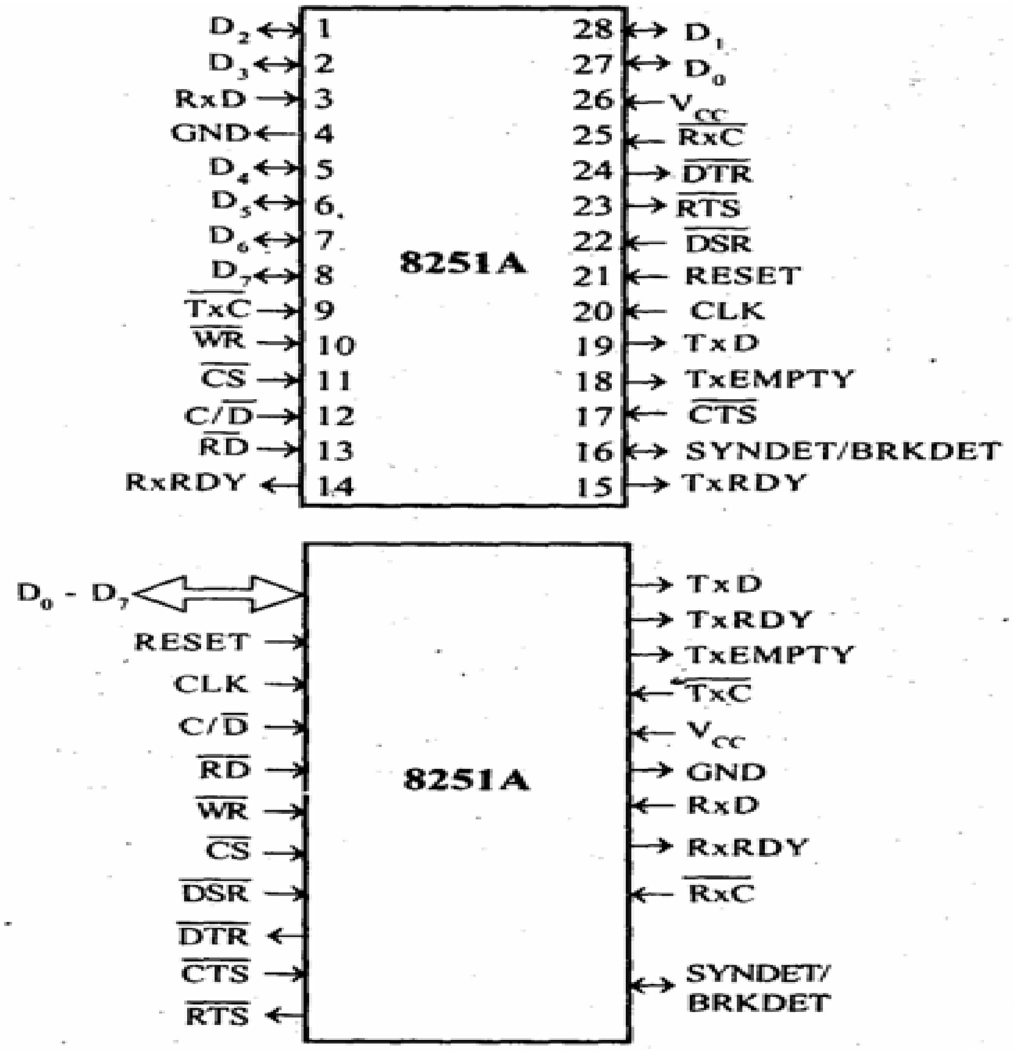
Error Check

- Parity Check
  - Even parity: When odd numbers of 1 make D7=1
    - Send Even number of 1
  - Odd parity: When even number of 1 make D7=1
    - Send Odd number of 1
- Cyclic Redundancy Code (CRC)
  - Synchronous Communication
  - Stream of Data can be represented by Cyclic polynomial that divided by a **constant polynomial**
  - Reminder to set **Bits** and Send out as check for error

8251

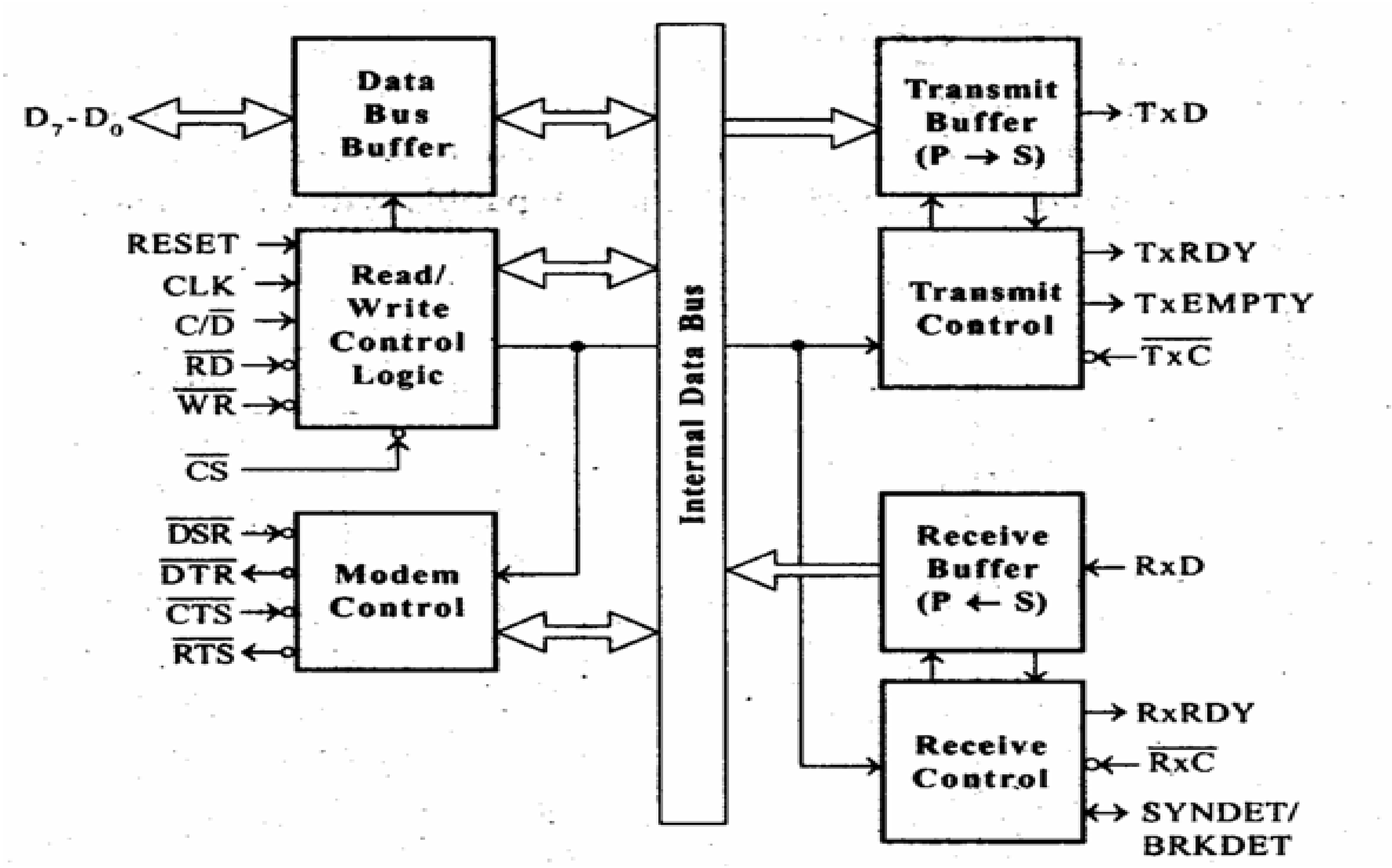
- The 8251A is a programmable serial communication interface chip designed for synchronous and asynchronous serial data communication.
- It supports the serial transmission of data.
- It is packed in a 28 pin DIP.

Pin details



Pin	Description
D_0-D_7	Parallel data
C/$\overline{D}$	Control register or Data buffer select
$\overline{RD}$	Read control
$\overline{WR}$	Write control
$\overline{CS}$	Chip Select
CLK	Clock pulse (TTL)
RESET	Reset
$\overline{TxC}$	Transmitter Clock
TxD	Transmitter Data
$\overline{RxC}$	Receiver Clock
RxD	Receiver Data
RxRDY	Receiver Ready
TxRDY	Transmitter Ready
$\overline{DSR}$	Data Set Ready
$\overline{DTR}$	Data Terminal Ready
SYNDET/ BRKDET	Synchronous Detect / Break Detect
$\overline{RTS}$	Request To Send Data
$\overline{CTS}$	Clear To Send Data
TxEMPTY	Transmitter Empty
V_{CC}	Supply (+5V)
GND	Ground (0 V)

Architecture



Arch - details

- The functional block diagram of 825 1A consists five sections.

They are:

- Read/Write control logic
- Transmitter
- Receiver
- Data bus buffer
- Modem control.

Read/Write control logic

- The Read/Write Control logic interfaces the 8251A with CPU, determines the functions of the 8251A according to the control word written into its control register.
- It monitors the data flow.
- This section has three registers and they are control register, status register and data buffer.
- The active low signals RD, WR, CS and C/D(Low) are used for read/write operations with these three registers.

Read/Write control logic

- When C/D(low) is high, the control register is selected for writing control word or reading status word.
- When C/D(low) is low, the data buffer is selected for read/write operation.
- When the reset is high, it forces 8251A into the idle mode.
- The clock input is necessary for 8251A for communication with CPU and this clock does not control either the serial transmission or the reception rate.

Transmitter

- If buffer register is empty, then TxRDY is goes to high.
- If output register is empty then TxEMPTY goes to high.
- The clock signal, TxC (low) controls the rate at which the bits are transmitted by the USART.
- The clock frequency can be 1,16 or 64 times the baud rate.

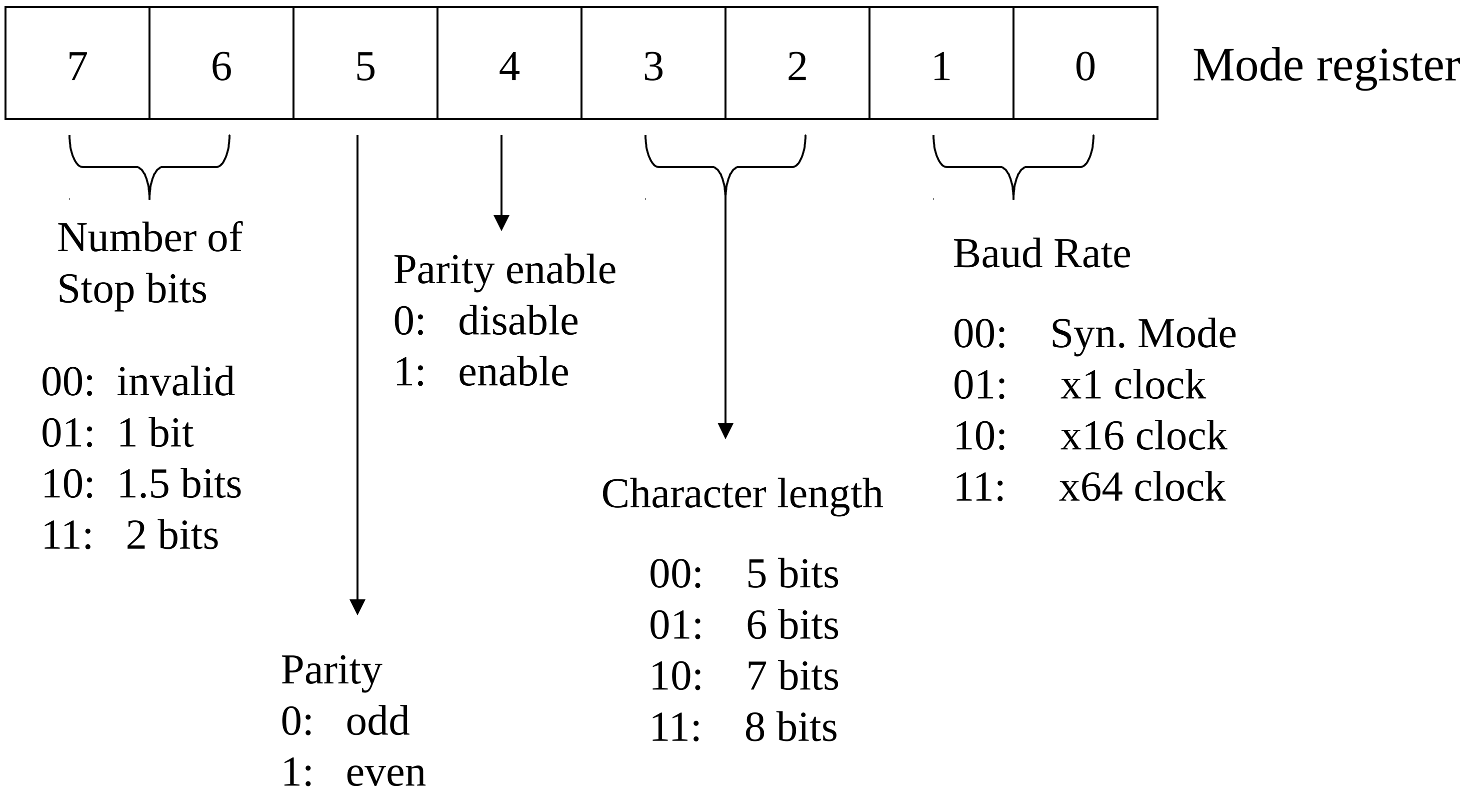
Receiver

- The receiver section accepts serial data and convert them into parallel data
- The receiver section is double buffered, i.e., it has an input register to receive serial data and convert to parallel, and a buffer register to hold the parallel data.
- When the RxD line goes low, the control logic assumes it as a START bit, waits for half a bit time and samples the line again.
- If the line is still low, then the input register accepts the following bits, forms a character and loads it into the buffer register.

Receiver

- The CPU reads the parallel data from the buffer register.
- When the input register loads a parallel data to buffer register, the RxRDY line goes high.
- The clock signal RxC (low) controls the rate at which bits are received by the USART.
- During asynchronous mode, the signal SYNDET/BRKDET will indicate the break in the data transmission.
- During synchronous mode, the signal SYNDET/BRKDET will indicate the reception of synchronous character.

8251 mode register



7	6	5	4	3	2	1	0	Mode register

Number of
Stop bits

00: invalid
01: 1 bit
10: 1.5 bits
11: 2 bits

Parity
0: odd
1: even

Parity enable
0: disable
1: enable

Character length

00: 5 bits
01: 6 bits
10: 7 bits
11: 8 bits

Baud Rate

00: Syn. Mode
01: x1 clock
10: x16 clock
11: x64 clock

8251 command register

EH	IR	RTS	ER	SBRK	RxE	DTR	TxE

TxE:	transmit enable
DTR:	data terminal ready
RxE:	receiver enable
SBPRK:	send break character
ER:	error reset
RTS:	request to send
IR:	internal reset
EH:	enter hunt mode

8251 status register

DSR	SYNDET	FE	OE	PE	TxEMPTY	RxRDY	TxRDY

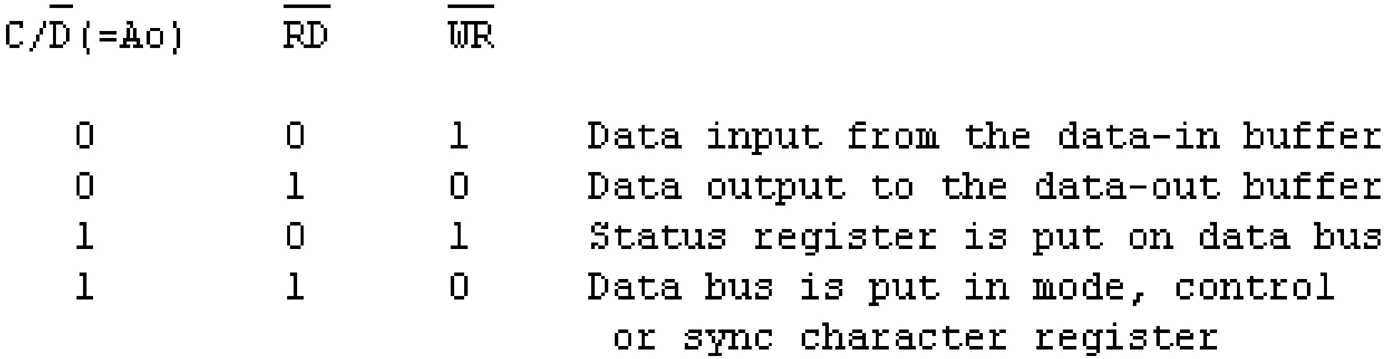
TxRDY:	transmit ready
RxRDY:	receiver ready
TxEMPTY:	transmitter empty
PE:	parity error
OE:	overrun error
FE:	framing error
SYNDET:	sync. character detected
DSR:	data set ready

Initializing the 8251

- To implement serial communication the MPU must inform the 8251 about the mode, baud, stop bits, parity etc. A set of control words must be loaded.
  - Mode Words
    - Specifies general characteristics of the operation.
  - Command Words
    - Enables the data transmission and/or reception
  - Status Word provides the information concerning register status and transmission errors.
- Any control word written into the control register after a mode word is interpreted as a command word; that means a command word can be changed anytime, however 8251 should be reset prior to writing a Mode word.
- 8251 can be reset internally by using the Internal Reset Bit D6.

8251 A Serial Communication Interface

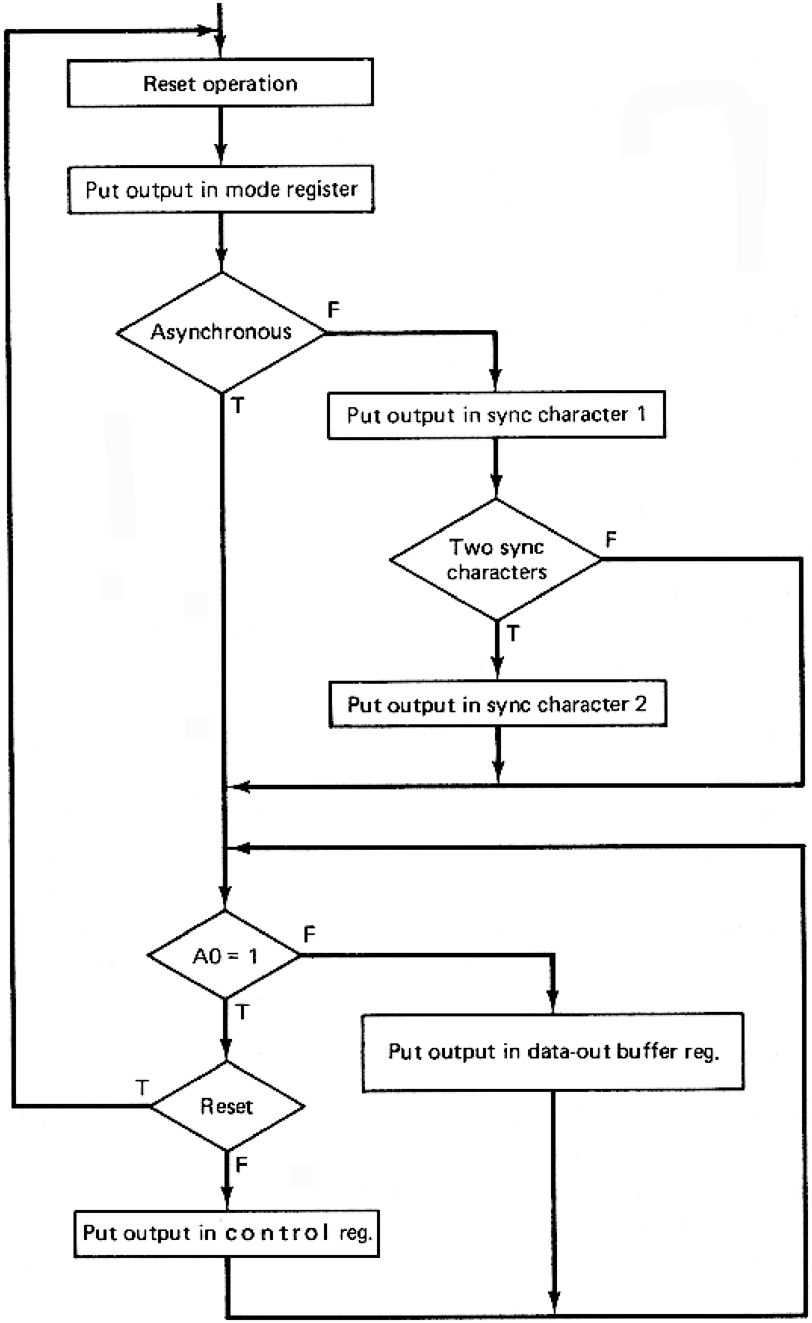
- The 8251A internally interprets the C/D,RD and WR signals as follow:

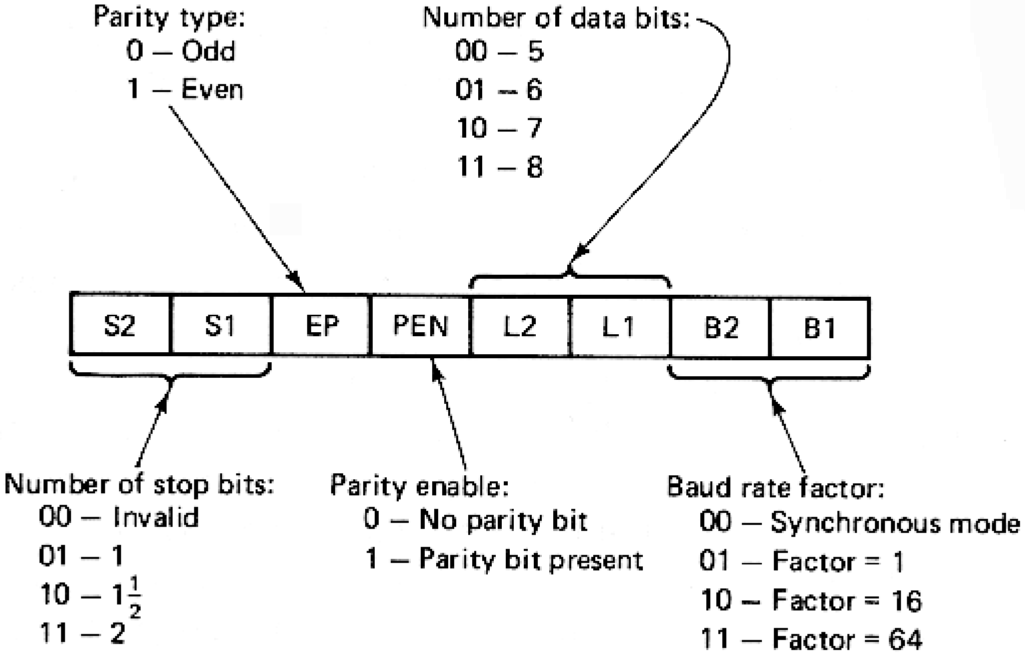


C/$\overline{D}$(=Ao)	$\overline{RD}$	$\overline{WR}$	
0	0	1	Data input from the data-in buffer
0	1	0	Data output to the data-out buffer
1	0	1	Status register is put on data bus
1	1	0	Data bus is put in mode, control or sync character register

- Whether the mode, control or sync character register is selected depends on the accessing sequence.

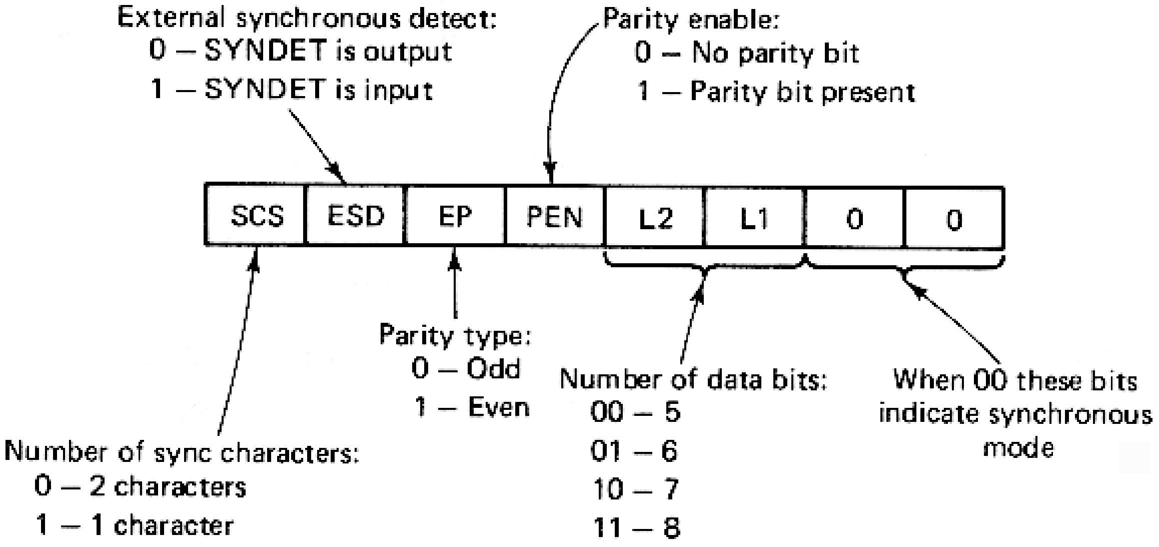
- A flowchart of the sequencing is given in Fig.


Reset operation
Put output in mode register
Asynchronous
F
T
Put output in sync character 1
Two sync characters
F
T
Put output in sync character 2
A0 = 1
F
T
Put output in data-out buffer reg.
T
Reset
F
Put output in control reg.



(a) Asynchronous mode

Summary: Format of the mode register

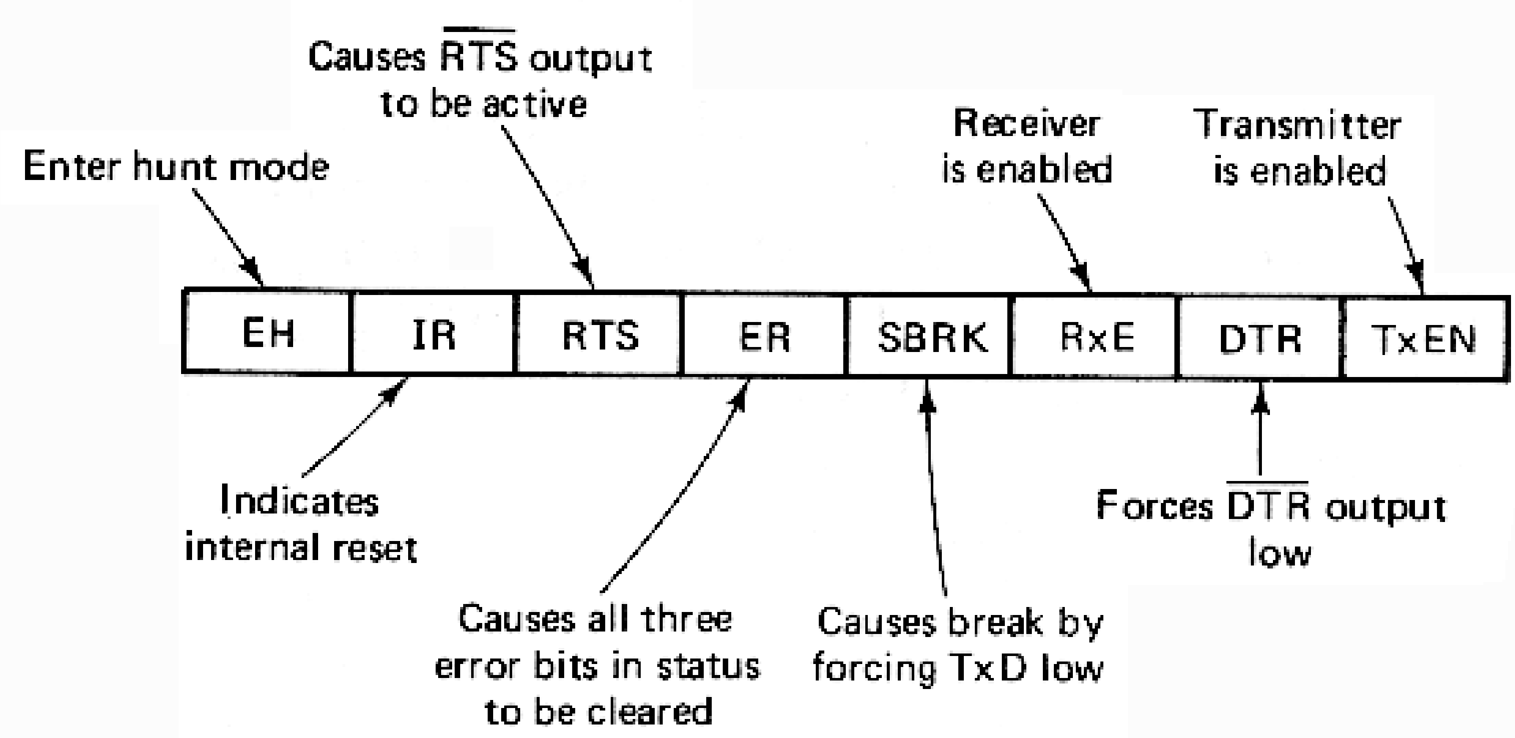


(b) Synchronous mode

What value must be written into the mode control register with baud rate divided by 16, char. Size 16 bits, odd parity, one stop bit ?

01011110 b = 5Eh

Format of the control register


Causes $\overline{\text{RTS}}$ output to be active
Enter hunt mode
Receiver is enabled
Transmitter is enabled
EH
IR
RTS
ER
SBRK
RxE
DTR
TxEN
Indicates internal reset
Forces $\overline{\text{DTR}}$ output low
Causes all three error bits in status to be cleared
Causes break by forcing TxD low

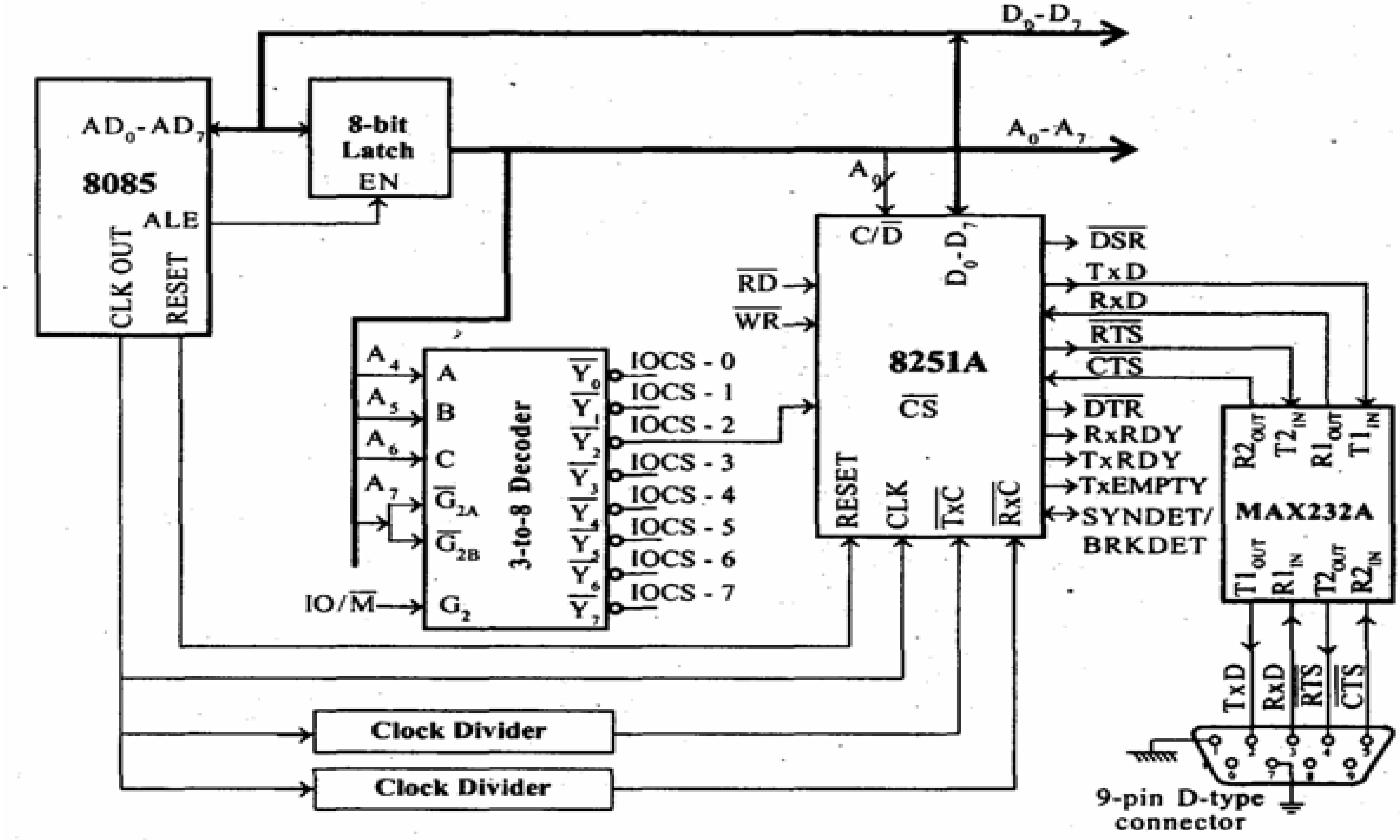
Note: In all cases action is taken when the bit is set to 1.

Example 1

- A program sequence which initializes the mode register and gives a command to enable the transmitter and begin an asynchronous transmission of 7-bit characters followed by an even-parity bit and 2 stop bits is:

```
MOV AL,11111010B
OUT 51H,AL
MOV AL,00110011B
OUT 51H,AL
```

8251 interfaced with 8085



RS-232C standard

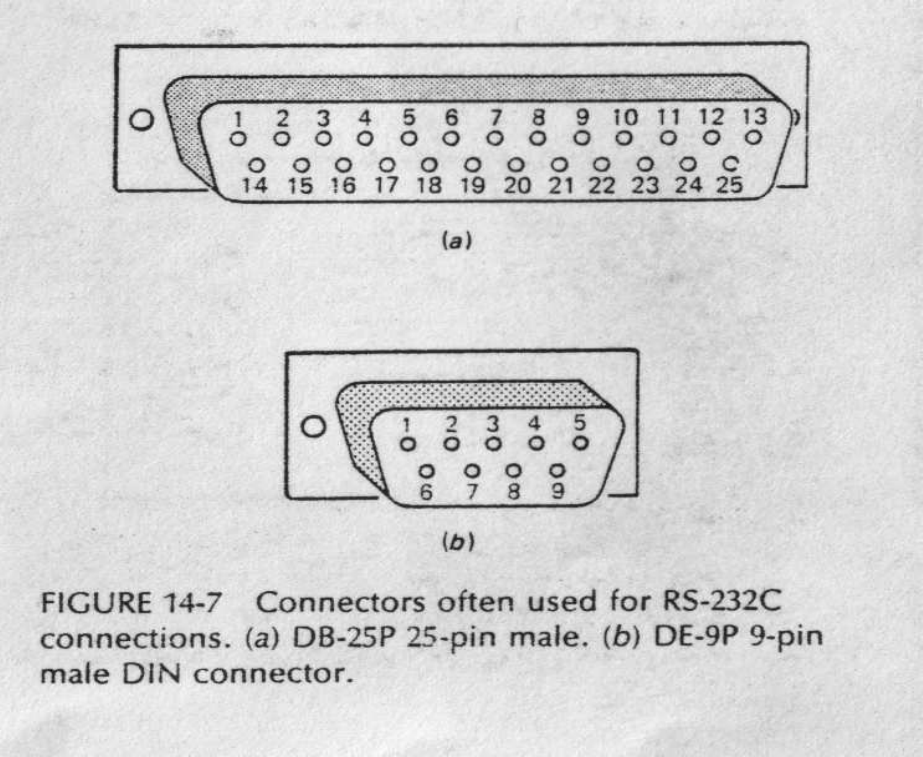


FIGURE 14-7 Connectors often used for RS-232C connections. (*a*) DB-25P 25-pin male. (*b*) DE-9P 9-pin male DIN connector.