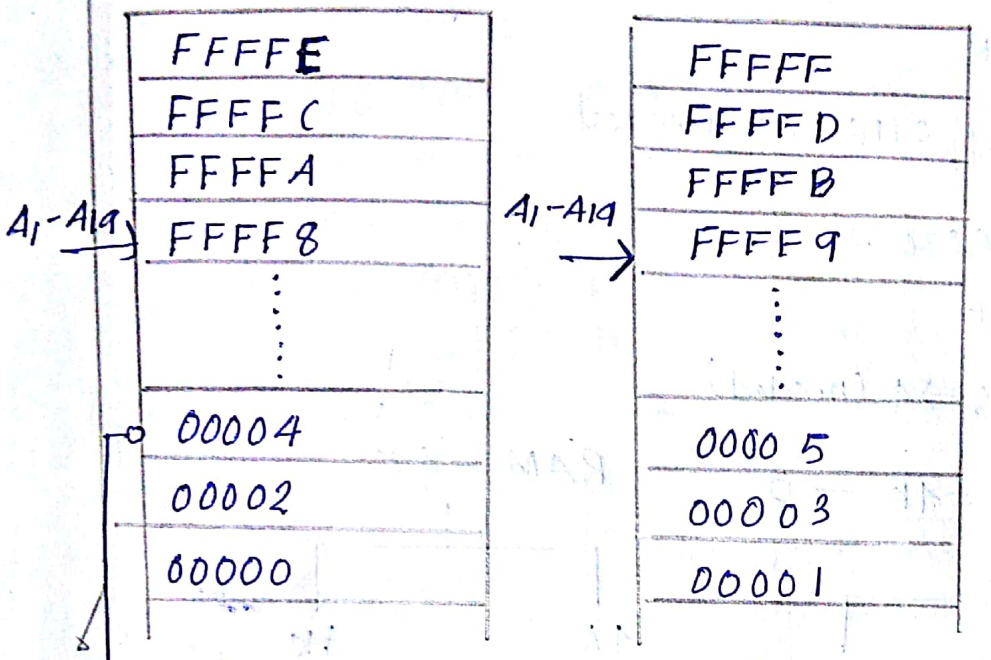
1/10/2017

Module-4

Memory Interfacing in (8086)



A0 Even address (512kb Even)

Odd address (512kb odd)

Example

If m/y chips available are only 2kb each do the interfacing for 8088 1-4K 2716 (ROM) starting at 00000H. 2-8K 6116 (SRAM) Starting at 08000H.

- 2716 $16/8$ = 2K chip.
- 6116 $16/8$ = 2K chip.

how?

1. Find the memory requirements
2. Do the memory mapping
3. Find the address which can be used for decode.

One mb memory is splitted into odd bank & even bank.

1. Memory requirement

a) 2716 → size 2K

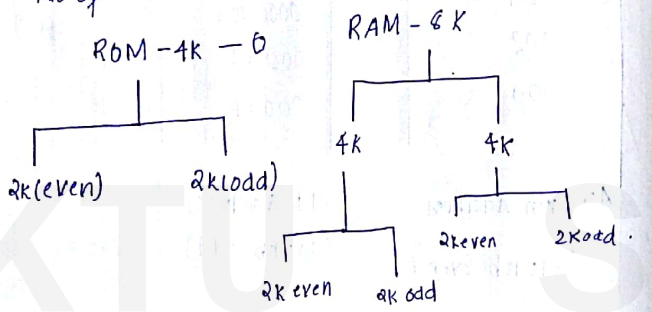
ROM → 4K

Number of 2716 → 2 needed

b) 6116 → size 2K

RAM → 8K

No: of 6116 → 4 needed



2. Memory map

one 4K ROM (00000 - 00FFF)

ROM_{1E} → 00000, 00002, ... 00FFE

ROM_{1O} → 00001, 00003 .... 00FFF

RAM (08000 - 08FFF) (one 4K)

RAM_{2E} → 08000, 08002, ... 08FFE

RAM_{2O} → 08001, 08003, ... 08FFF.

Memory

1K → 00000 - 003FF

2K → 00000 - 007FF

3K → 00000 - 00FFF

4K → 00000 - FFFFF

ROM 1 - (00000 - 00FFF)

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

RAM 1 (08000 - 08FFF)

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0
1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

RAM 2 (09000 - 09FFF)

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0
1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1

2^K chip

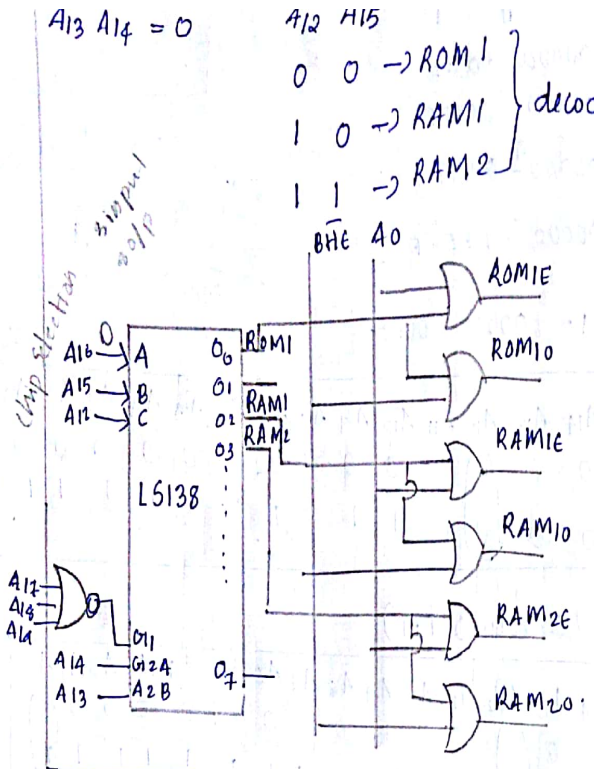
2^{11} - 11 address lines $A_0 - A_{11}$

A_0 used for even

$\overline{BHE}$ used for odd } ie, $A_1 - A_{11}$

available → A_{12} A_{13} A_{14} A_{15}

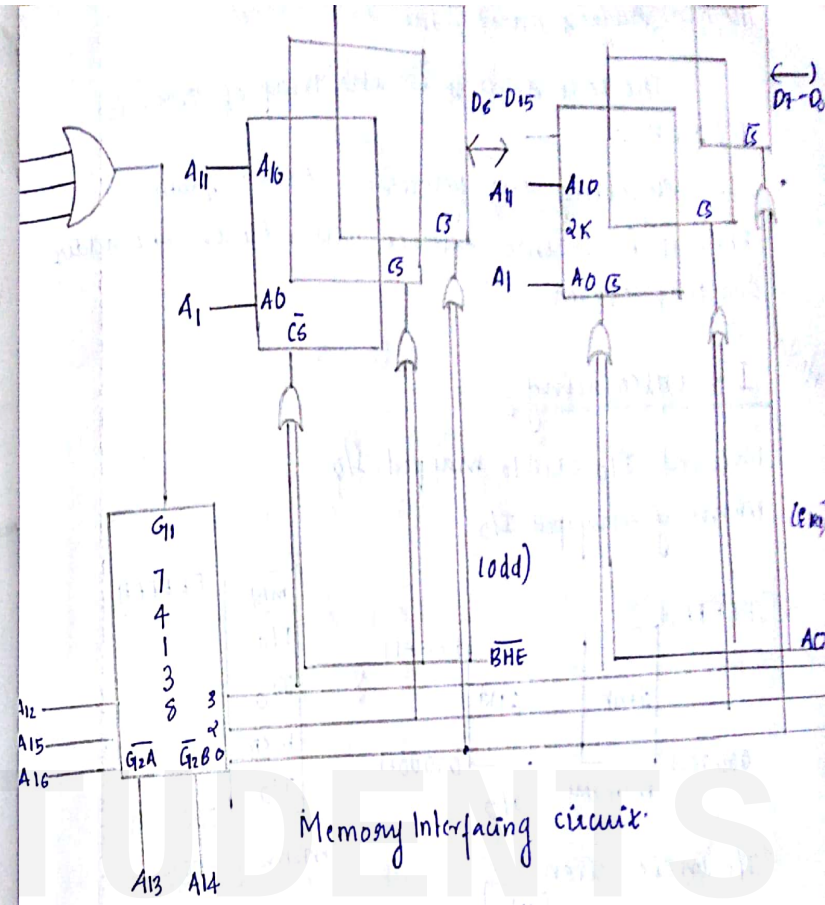
A_{16} A_{17} A_{18} A_{19} (always zero).



000	ROM
010	RAM1
011	RAM2

Address decoder ckt.

If 8086 then odd and even bank



Memory Interfacing circuit.

Q Interface two (4K x 8) EPROM's and 2 two (4K x 8) RAM chips with 8086 select suitable maps

A We know that after reset the IP and CS are initialized to form address FFFF0H Hence this address must lie in the EPROM. The address of RAM may be selected anywhere in the 1mb address space of 8086 But we will select the address such that the address map of the system is continuous.

Q Design an interface b/w 8086 CPU and 2 chips of 16 x 8 EPROM and 2 chips of 32 K x 8 RAM Select the starting address of EPROM suitably.

The RAM address must start at 00000H

The last address in the map of 8086 is FFFFH.

After reseting the processor starts from FFFF0H hence this address must lie in the address range of EPROM

I/o interfacing

Isolated I/o or I/o mapped I/O
memory mapped I/O

I/o instruction

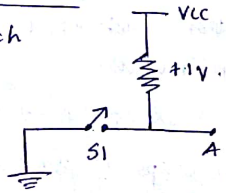
IN AL, P8
IN AX, P8
IN AL, DX
OUT P8, AL
OUT P8, AX
OUT PX, AL

IN
OUT

mov
LD
ST

INPUT DEVICE

Switch



Switch is open A = logic 1
Switch is closed A = logic 0.

output device (LED)

A = 0 LED is off
A = 1 LED is on.

I/o mapped I/o

Seperate mly for I/o device (64 Kb is for that)

Read through i/p port

In memory mapped I/o: only one memory (one mb) no seperate mly. MOV, LD, ST are the instructio are used.

I/o

- Memory address space is different from I/o address space.
- Signals used are I/o read and I/o write for peripherals and mly read and mly write for memory
- Instruction used are IN

Memory mapped I/o

Mly address space is

Signals used are mly read and m/y write both for Memory and peripheral.

IN, OUT instructions are not used instead MOV, ST, LD

Input device must be isolated from the global data bus. Hence else garbage may be transferred to to the data bus

Tri-State Buffers:

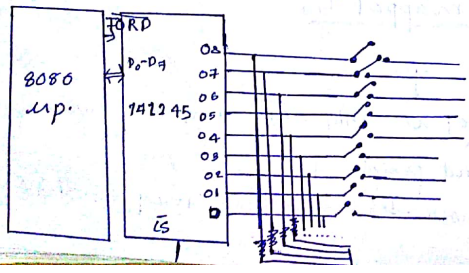
It provide isolation as well as strength the signal

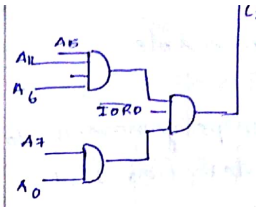
I/o design in 8086:

In any µp based sim when data is send out by µp the data on the data bus must be latched by the receiver or o/p device.

* M/y's have internal latches for storing the data
* Latching sim must be design for ports.
* Data provided by the µp is available only for short period of time. (ans 1000 nano sec)
* Data must be latched else it will be losed. Similarly data comes from a port / m/y data must be i/p through a tri-state buffer.

Q Interface an input 74LS245 to read the status of the switches switch 1- Switch 8. The switches when shorted INPUT A = 1 else INPUT A = 0 to the µp Store the status in register BL The address of the port is 0740H

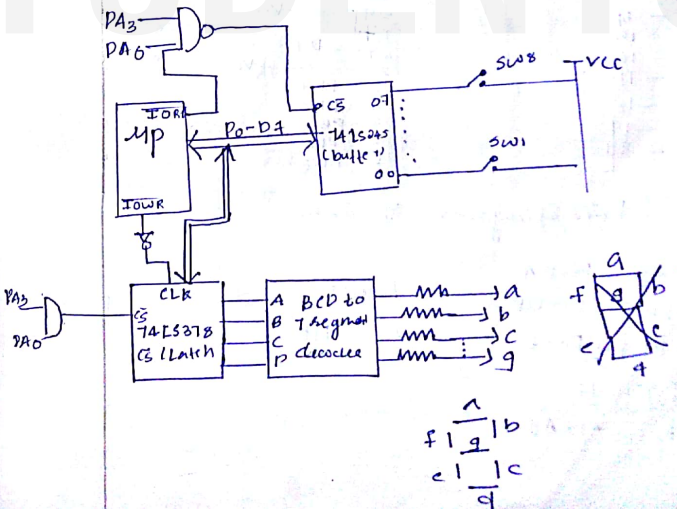




ADDING LCD DISPLAY TO SWITCH

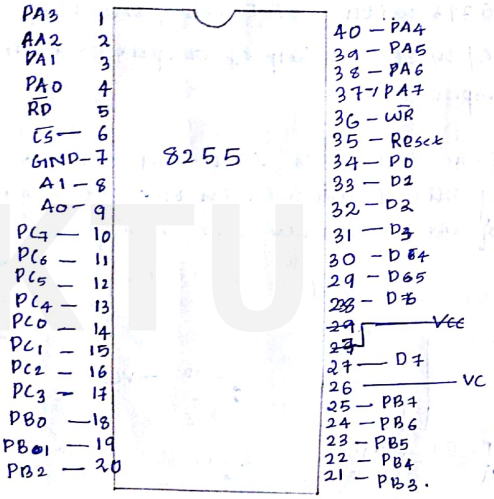
Design PA3 PA0 an interface of input port 74LS245 to read the status of switches. Switch1 - Switch8 and o/p port 74LS373 with 8086. Display the no: of key if is pressed with the help of output port on 7 segment display

Soln: Status of the switch is first read into the AL reg Displaying shorted switch no in the 7 segment display Instead of using to address line it may use only $A_0 - A_3$.



8255 : PPI [Programmable Peripheral Interface]

The 8255 is a general purpose programmable I/O device design to transfer the data from I/O to interrupt I/o under certain conditions are requi- it can be used with any µp. It consist of 3 8 bit bidirectional I/p, o/p port (24 o/p lines.)



Pin diagram of 8255

$PA_0 - PA_7$ (Port A)
$PB_0 - PB_7$ (Port B)
$PC_0 - PC_7$ (Port C)

Power
VCC, GND

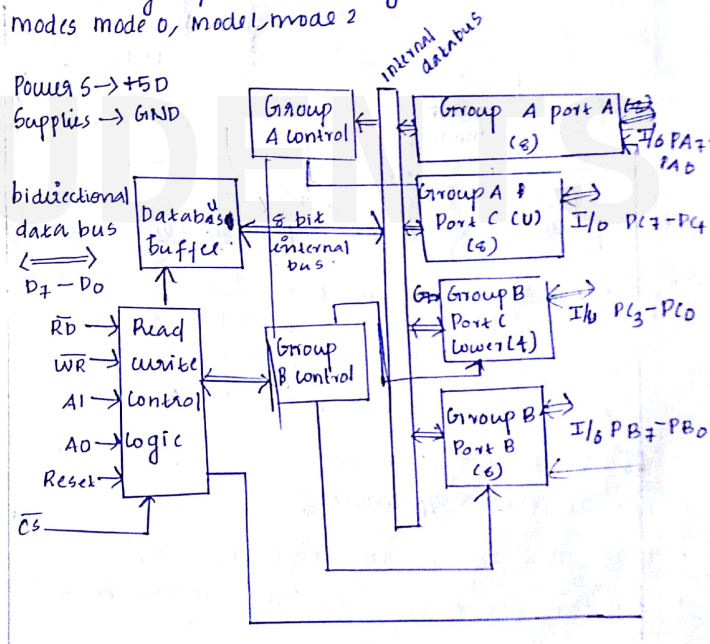
Control Signals
$\overline{RD}$, $\overline{WR}$, $\overline{CS}$, A1, A0.

Ports of 8255

8255 has three ports :-

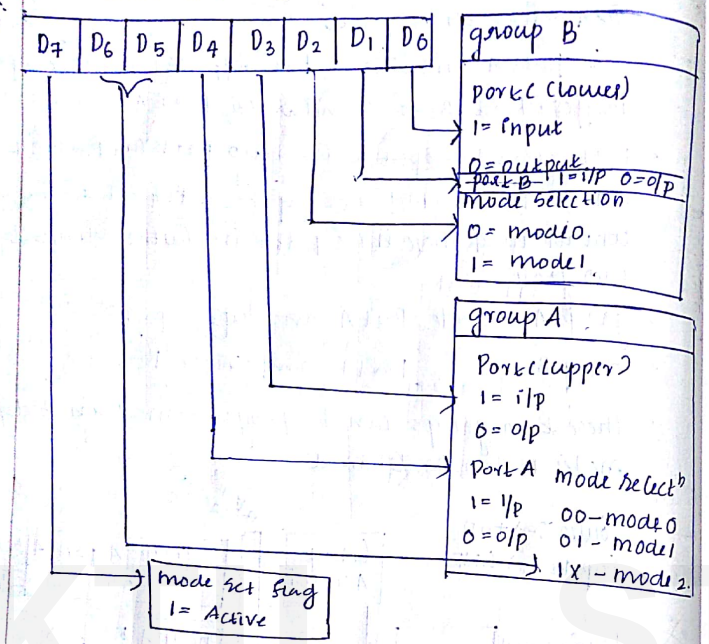
port A contains 8 bit o/p latch or input buffer. Port B is similar to Port A

- Port C can be split into two parts (a) Port C Lower (PC0 - PC3) & (b) Port C upper (PC4 - PC7) by the control word. These three ports are further divided into two group
- group A include Port A and upper port C
- Group B " Port B and lower Port C.
- These two groups can be programmed in 3 different modes mode 0, mode 1, mode 2



9/10/2017

Control word



- configuration of A & B is ~~fix~~ based on control word.
- 3 types of modes mode 0 — all
  1 — B, C
  2 — C
  BSR — Port A
- D_0 - 0, port C - o/p mode
- D_1 — 0, port B - o/p mode
- If we given zero to all ($D_0, D_1, D_2 \dots D_7$)
- 80H can active the 8255 then it will be in o/p mode.

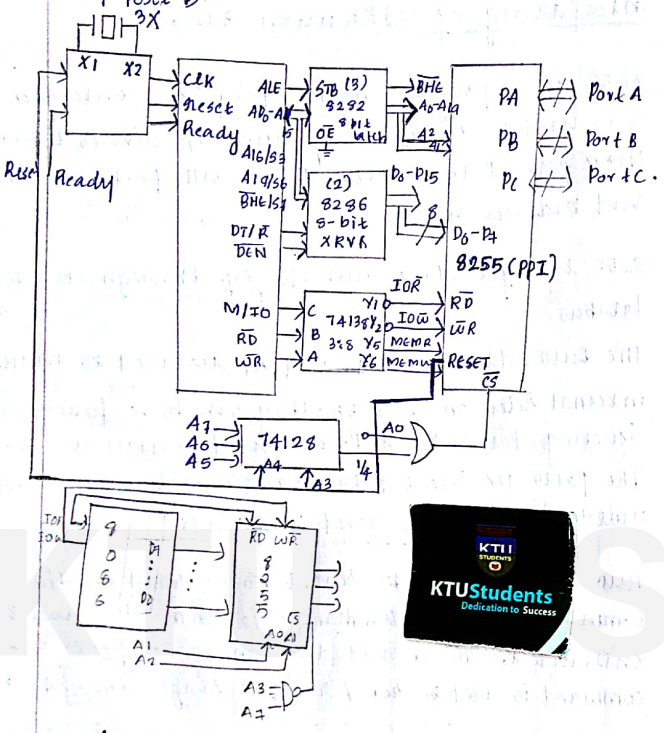
Move 80, X
This value will be stored to control regist...

- control word control the 24 lines.

→ Interfacing of 8255 with 8086

1. 8255 is a programmable peripheral interface it is used to interface up with I/o devices through three port. Port A, Port B, Port C all ports are 8 bit and bidirectional
2. 8255 transfees data with the up through its 8 bit databus
3. The two address lines A_0 and A_1 are used to make internal selection in 8255. They can have four options selecting Port A, Port B, Port C and control word. The ports are selected to transfer data. The control word is selected to send commands.
4. Two commands can be send to 8255 called the I/o command and BSR command. I/o command is used to initialise the mode and direction of the ports. BSR command is used to set / reset a single line of port C
5. 8255 has three operational modes of data transfer
   a. Mode 0 is a simple data transfer mode it doesn't perform handshaking but all 3 ~~two~~ ports available for data transfer
   Mode 1 to perform unidirectional handshaking that ~~to~~ makes transfers more reliable. Port C lines are used by Port A and Port B to perform handshaking
   Mode 2 performs bidirectional handshaking only Port A can operate in mode 2 at that time Port B can operate in mode 1 or mode 0. Port C lines are again

used for performing handshaking for port A and Port B.



	A7	A6	A5	A4	A3	A2	A1	A0	
PA	1	0	0	0	0	0	0	0	80H
PB	1	0	0	0	0	0	1	0	82H
PC	1	0	0	0	0	1	0	0	84H
CW	1	0	0	0	0	1	1	0	86H

I/o map of 8255 at add 80H.

- From 4 20 address line only 2 address lines are selected.
- from 16 address line only 8 data lines are selected.

Application

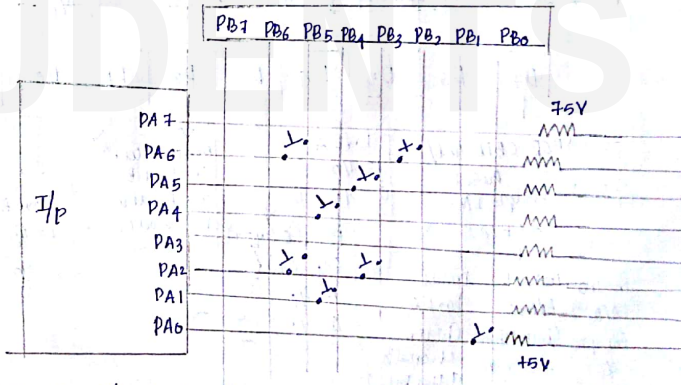
1. Traffic line controlling

			PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
Y_{22}	↓	Y_{11}	0	0	Y_{11}	R_{11}	G_{111}	Y_1	R_1	G_{11}
R_{22}		R_{11}	0	0	0	1	0	0	0	1
G_{122}		G_{111}								
→		←	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
G_{11}	↑	G_{12}	0	0	Y_{22}	R_{22}	G_{122}	Y_2	R_2	G_{12}
R_1		R_2	0	0	0	1	0	0	1	0
Y_1		Y_2								

Set the control word first

```
mov A, CW
out CR, A
mov A, 11
out Port A
mov A, R
out Port B.
```


Keyboard Interfacing:



Initially o/p Port = 0.
I/p Port = 1

```
Inport A
Cmp FF
JZ → no key is pressed.
```

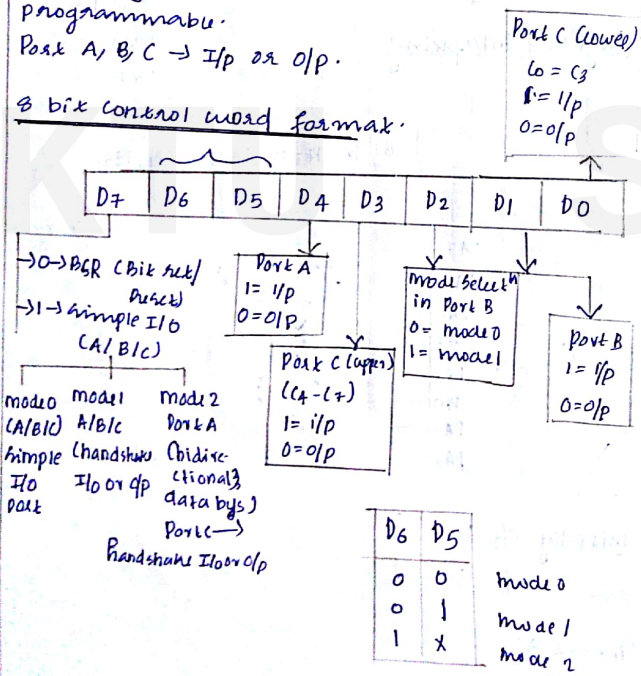
- It is 8x8 matrix keyboard
- A port is the input and B port is the output
- key open value = 1 closed = 0.
- Initially all is 1 (i/p) and output port is zero.
- We can't find out key pressed column using rotation method

10/10/17 Control word format and mode of operation in 8255

→ Control register:

- 8 bit register
- programmable.
- Port A, B, C → I/p or O/p.

→ 8 bit control word format.



D6	D5	
0	0	mode 0
0	1	mode 1
1	X	mode 2

Q) 1. Read input at port B and Port C (lower)
2. O/p at port A and Port C (upper) simple I/O port
3. Simple I/O mode
4. mode 0

Find the control word

→ Using control word we can select the mode, port

BSR - only Port C can be used.
Individually each pin can be set / reset.

Simple I/O - can use Port A, B, C can use.
- It has 3 modes.

→ D6 and D5 to select the mode.

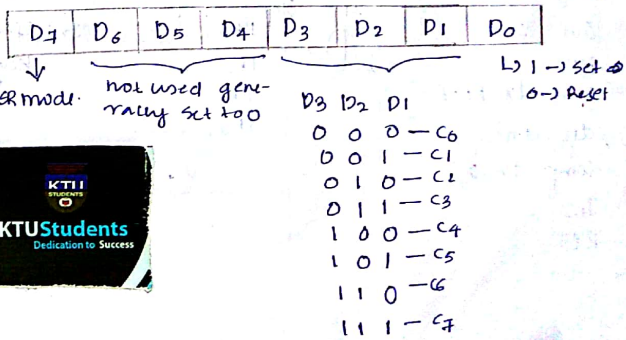
(A)

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	0	0	1	1

8 | 3

8255 BSR mode.

B → bit Port C → O/p Port
S → set It control individual lines of Port C
R → Reset



Q) Operate 8255 in BSR mode & write a ALP to do the following

1. Set lines C_2 & C_5
2. Reset C_2 & C_5 after 0.3 sec.

Set	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
C_2 →	0	0	0	0	0	1	0	1
C_5 →	0	0	0	0	1	0	1	1

Reset

C_2 →

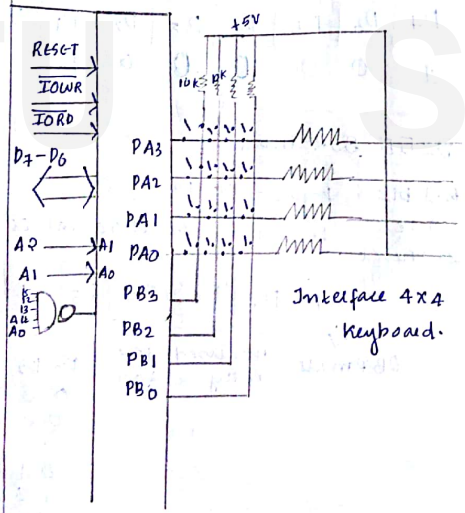
C_5 →

Program :-

```
1000  MOV A, 05
      OUT 83H
      MOV A, 0BH
      OUT 83H
      Call delay(2000)
      MOV A, 04H
      OUT 83H
      MOV A, 0AH
      OUT 83H
      HLT
2000  MOV dx, FFFF
      dec dx
      comp dx, 00
      JNZ
      RET
```

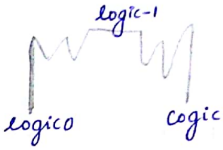
Q: Interface a 4x4 keyboard with 8086 using 8255 and write an ALP for detecting a key closure and return the key code in AL. The debouncing period for a key is 10ms ~~time~~ Use s/w key debouncing technique. ~~det~~

Debounce is available 10ms delay routine.

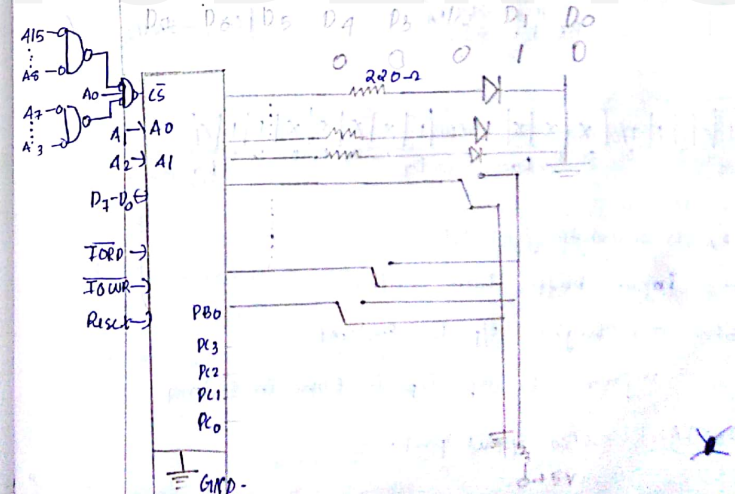


Interface 4x4 keyboard.

→ Key debounce : whenever a mechanical pushed or ~~not~~ released once. The mechanical components of the key don't change the position smoothly rather it generates a transient response.



Q: Interface an 8255 with 8086 to work as an I/O port initialise port A as o/p port, Port B as input port and port C as o/p port. Port A address should be 0140H. Write a program to sense switch position SW0-SW7 connected at port B. The sensed pattern is to be displayed on Port A, to which 8 LED's are connected while the Port C lower displays number of on switches out of the total 8 switches.

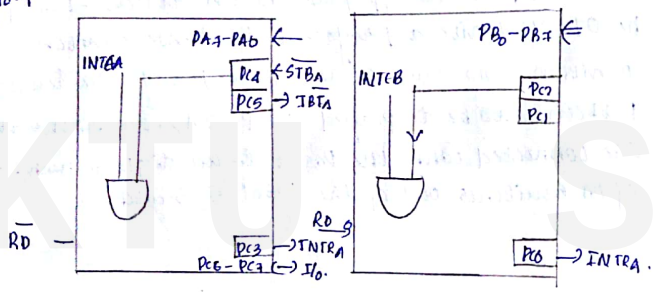


```
mov dx, 0FF617
mov AL, 82
out dx, AL
sub dx, 04 (get address of port B)
IN AL, dx
sub dx, 02
out dx, 02
mov dx, AL
mov bl, 00
mov ch, 06
YY: ROL AL
    JNC XX
    INC bl
XX  dec ch
    JNZ YY
    MOV AL, BL
    add dx, 04
```


MODE 1

group A I/P

group B I/P



CW = | 1 | 0 | 1 | 1 | I/O | X | X | X |
D7 D6 D5 D4 D3 D2 D1 D0

CW | 1 | X | X | X | X | 1 | 1 | X |
D7 D6 D5 D4 D3 D2 D1 D0

$\overline{STB_A}$ is used to give i/p.

$\overline{IBF_A}$ input buffer full. (latch)

$\overline{IBF_A} = 0$ (high) i/p is Entered.

informs the Mp i/p is ~~came~~ Entered.

PA_0 - PA_3 is also input port.

Application

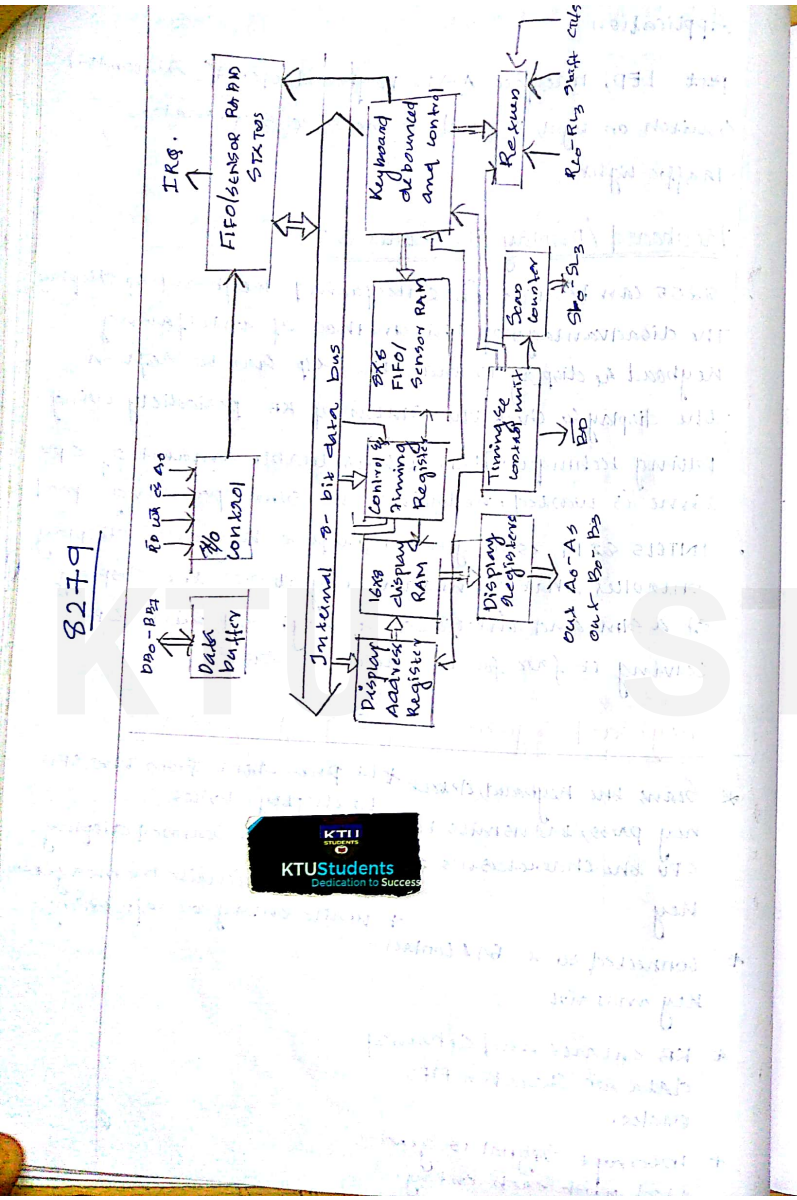
~~Put~~ LED, Interface A→D, keyboard operated, Sequential Switch on light, ~~used~~ DC motor & Stepper motor, Traffic light

Keyboard / Display Controller 8279

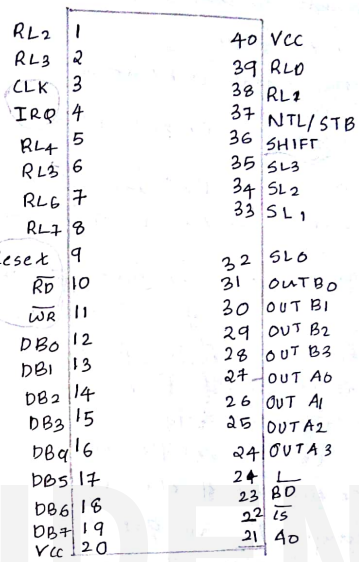
- 8255 can be used in interfacing keyboard & display. The disadvantages of this method of interfacing keyboard & display is that the µp has to refresh the display & check the status of KB periodically using polling technique. Thus a considerable amount of CPU time is wasted, reducing the sim operating speed
- INTEL's 8279 is a general purpose keyboard display controller that simultaneously drives the display of a sim and interfaces a keyboard with CPU, leaving it free for it's routine task

Keyboard Segment	Display Segment
* Scans the keyboard, detects key press, transmits to CPU the characteristics of a key	* It puts data from the CPU to display devices
* connected to a 64 contact key matrix	* 16 character scanned display
* KB entries are debounced & stored in FIFO Stacks.	* 16 x 8 read/write memory (RAM)
* Interrupt signal is generated with each entry.	* write entry or left entry

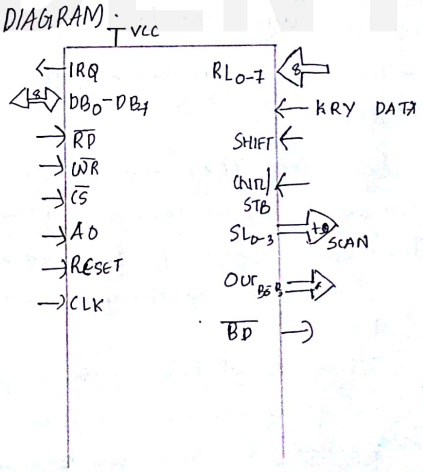
8279



PIN DIAGRAM OF 8279



LOGIC DIAGRAM:



- DB_0-DB_7 these are bidirectional data bus lines The data and command lines to and from CPU are transferred on these lines
- CLK - this is a clock input used to generate internal timings
- RESET - This pin is used to reset 8279
- $\overline{CS}$ - Chip Select a low on this line enables 8279 for normal read/write operation.
- A_0 - A high on the A_0 line indicate the transfer of a command or Status information a low on the line indicates transfer of data. This is used to select one of the internal registers of 8279.

$\overline{RD}/\overline{WR}$ - read or write input pin enable the data buffer to receive/send data over the data bus.

$\overline{IRQ}$ - This interrupt o/p line goes high when there is FIFO in RAM. The interrupt line goes low with each FIFO read operation.

V_{SS}, V_{CC} - These are the ground and power supply lines for the circuit

SL_0-SL_3 : (Scan Lines) These lines are used to scan the keyboard matrix of display digits These lines can be programmed as encoded or decoded using the mode control register

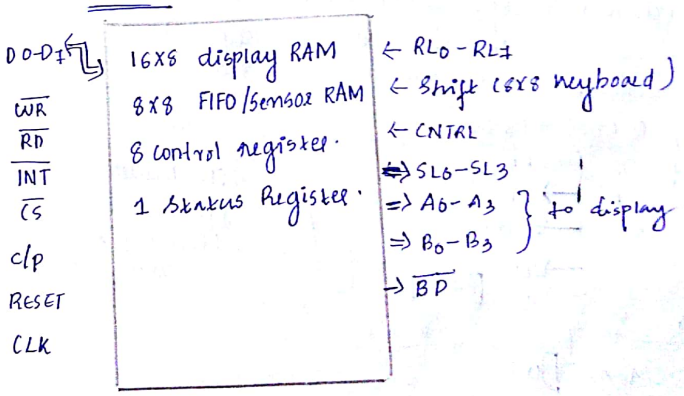
RL_0-RL_7 (Return Lines) These are the input line which are connected to one terminal of key while other terminal of keys are connected to the decoded Scan lines These are normally high. But, Pulled low when a key is pressed

- Shift - the Status of the Shift i/p Line is Stored along with each key code in FIFO in the Scanned keyboard mode
- Control - (CNTL) / STROB STB control / Strobed i/p mode. In the keyboard mode this line is used to control i/p and stored in FIFO on a key closure. The line is a Strobe line that enters the data into FIFO when in the Strobed i/p mode

Blank display - This o/p pin is used to blank the display during digit Switching / by a blanking command

OUT_{B0}-B_0 - These are the output ports for external display Refresh register 2 two 8 16X4

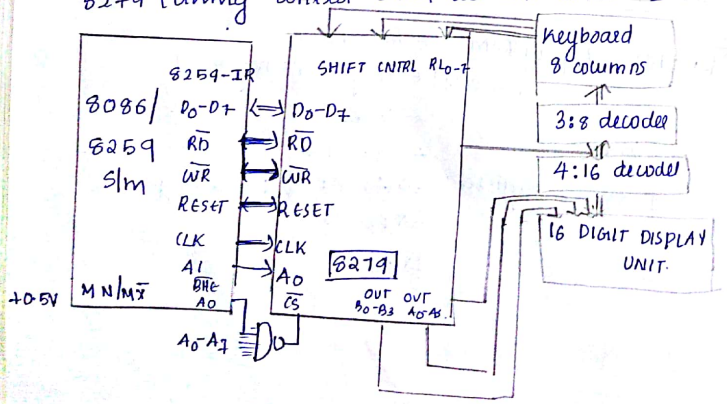
8279



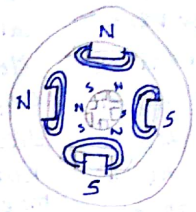
Up and 8279 are connected using the following pins: $\overline{RD}$, $\overline{WR}$, $\overline{INT}$, $\overline{CS}$, RESET, CLK.

To control keyboard - we need to know the row & column of ~~key~~ the pressed key. Details about the key is stored in Sensor RAM (max 8 can be stored). If more than 8 is needed external memory is used.

- SHIFT & CNTL is used to know the status of key & Whenever a key is pressed Sensor RAM interrupts the microprocessor.
- SL_0-SL_3 (Scan line): check the signals coming from keyboard and display periodically.
- Display (8 output) or (2 groups with 4 o/p)
- 16×8 display RAM: 16 i/p and 8 o/p. which port is using...? such information is stored in it.
- Program can be written by program then control 8279 (using control word we can control)



Stepper motor



1 Step angle = 1.8°

Step size = $\frac{360}{N_S \times N_R}$

When input voltage is applied rotor rotates. At some point poles of stator and rotor becomes same and repells at 1.8°.

2-phase scheme:-

Anticlockwise

A_1	A_2	B	B_2	
1	0	0	1	9
0	1	0	1	5
0	1	1	0	6
1	0	1	0	0A

Clockwise

A1	A_2	B_1	B_2	
1	0	1	0	0A
0	1	1	0	6
1	1	0	1	5
1	0	0	1	9

1. Simple scheme
2. 2-phase scheme
3. Half stepping scheme.

Program:-

```
mov al, 80h
out 26, al
mov bx, 32h
mov al, 0A
out 20, al
```

```
call 2000
mov al, 06
out 20, al
call 2000
mov al, 05
out 20, al
```

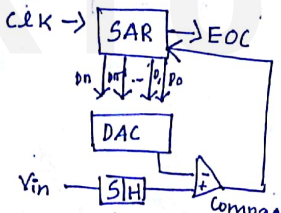
```
call 2000
mov al, 05
out 20, al
call 2000
mov al, 09
out 20, al
call 2000
dec bx
comp bx, 00
Jnc 1004
call 3000
HLT.
```

```
3000: mov al, 80h
      out 26, al
      mov bx, 32
3004: mov al, 09
      out 20, al
      call 2000
      mov al, 05
      out 20, al
      call 2000
      mov al, 06
      out 20, al
      call 26000
```

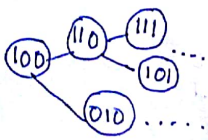
```
mov al, 09
out 20, al
call 2000
dec bx
cmp bx, 00
Jnc 3004
RET
mov al 09
out 20, al
call 2000
dec bx
comp bx, 00
Jnc 1000
call 3000
HLT
```

Digital to analog converters:

SAR → Successive Approximation Register

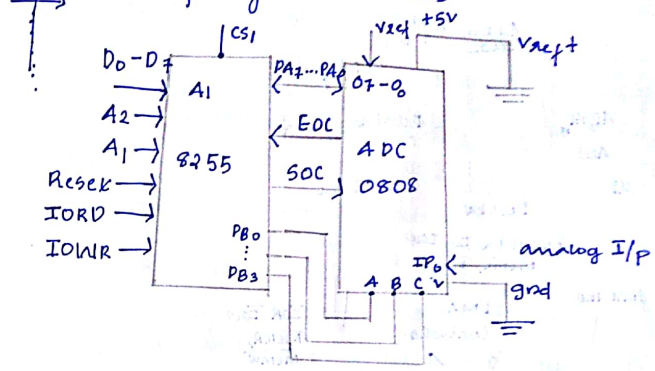


↓ Sample and hold



1. Switch ON ckt. MSB of SAR = 1
2. Vin > Vref → Set the nxt bit = 1
3. Vin < Vref → Analog bit bit to Reset.

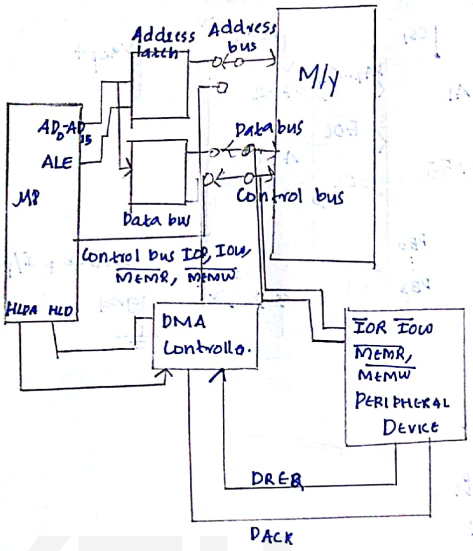
Interfacing ADC to 8255



Program:

```
      mov al, 80h
      out 26, al
1005→ mov al, 00
      out 20, al
      mov al, OFF
      out 20, al
      call 2000
      Jmp 1005
2000: mov cx, 1000
      dec cx
      cmp cx, 00
      Jmp
      RET
```

8257 [DMA Controller]



- I/o devia → up → Memory → location → up
- To speed up this process without accessing up we use DMA
- I/o $\xrightarrow{\text{(request)}}$ DREQ DMA (to access) → HOLD (MP).
  [Initially HOLD = 0 if request arrivals HOLD = 1]
- This information is send to DMA through HLDA
- The DMA handle system bus.
- Now Bus (master) is DMA
- when DMA gets the system bus control it (will be) informed to peripheral devia.
  Then Peripheral devia directly access memory.

- up initialises the no:of bytes to transfer.
- Count decrements during each transfer.

Iv DMA transfer is a hardwared control I/o transfer technique. It is mainly used for high speed data transfer b/w i/o and memory where the speed of the peripheral is generally faster than up.

- In program controlled I/o status/INTR driven I/o the speed of the transfer is slow mainly bcz instruction need to be decoded and then executed for the transfer
- DMA transfer is s/w independant and hence much faster a devia known as the DMA controller (DMAC) is responsible for the DMA transfer

Sequence Of DMA transfer.

1. up initialises the DMAC by giving the starting address and the no:of bytes to be transferred.
2. An I/o devia request the DMA controller to perform DMA transfer through the DREQ line.
3. The DMAC in turn send a request signal to the up through the HOLD line.
4. The up finish the current m/c cycle and release the s/m bus. [disconnected]. It also acknowledges receiving the HOLD signal throug HLDA
5. The DMAC acquires control of the s/m bus The DMA

Send the DMA acknowledgment signal and the DMA transfer begins

7 After every bytes is transferred the address register is increment or decrement and the count register decremented

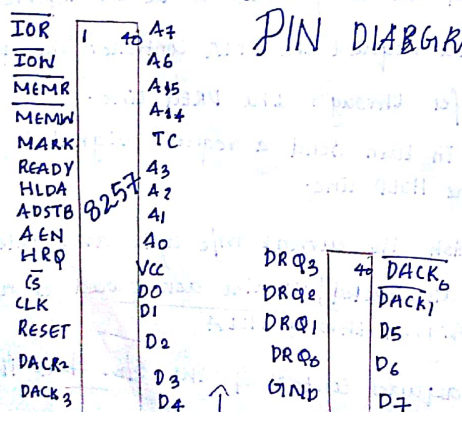
8 Discontinous till the counter reaches zero (terminal count (TC)). Now DMA transfer is completed.

9 At the end of the transfer the sim bus is released by the DMAC Thus up takes control of the sim bus and continous ~~its~~ its operation.

* The DMAC does the DMA ~~controller~~ transfer through its channels. Minimum requirements each channel are

1) address register to store the starting address for the transfer
2) Count register to store the no:of bytes to be transfred
3) DREQ signal from the I/o device
4) D acknowlegment signal to the I/o device.

PIN DIABGRAM OF 8257

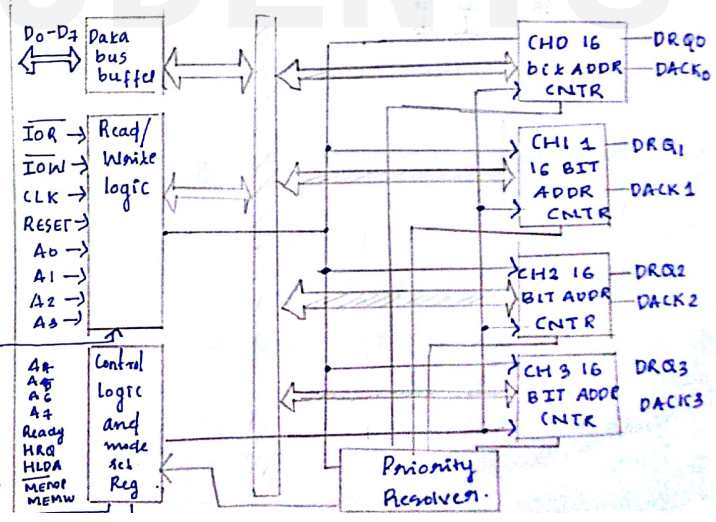


23/10/17

8257 [DMA Controller) Ancl

Features of 8257

* It has four channel which can be used of over four I/o devices
* Each channel has 16 bit address and 14 bit counter
* Each channel can transfer data upto 64KB.
* Eac be programmed independently perform read tranfer, write transfer and verify transfer operations. It generate mark signal to the Peripheral device that 128 bytes have been transfer
* It requires a single pulse clock.
* It's frequency changes from 250Hz – 3MHz
* It operates in two mode ie master & slave mode.



INTERFACING 8259 WITH 8086