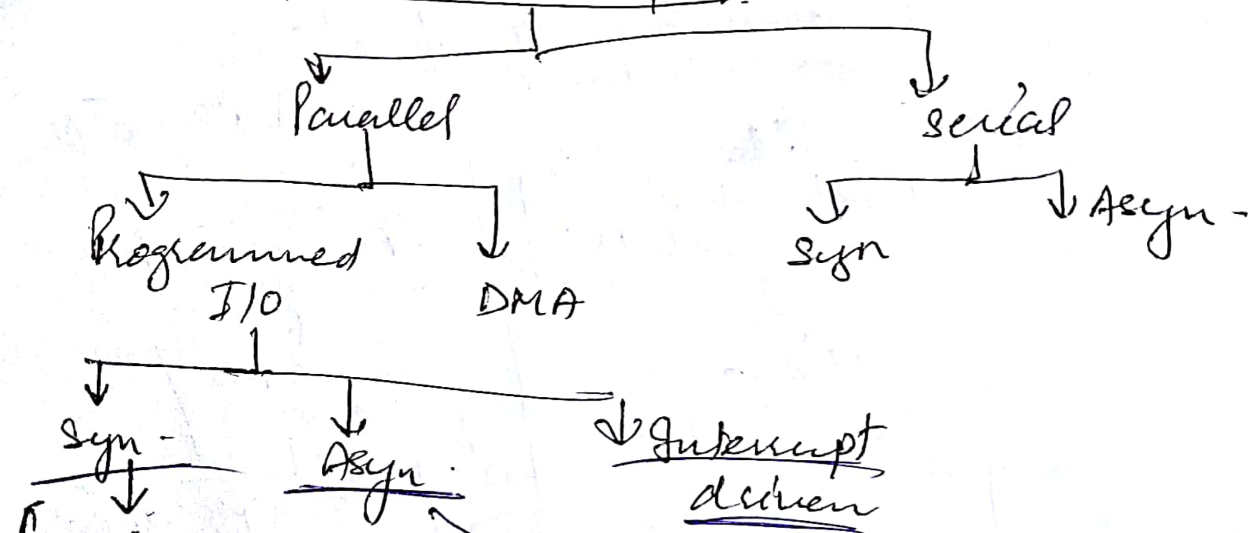
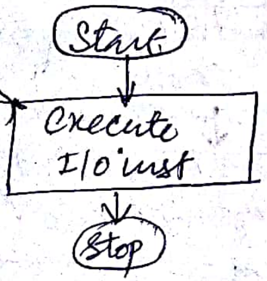
* Modes of Data transfer



[can performed only when I/o device is compatible in speed i.e memory ~~spee~~ devices)

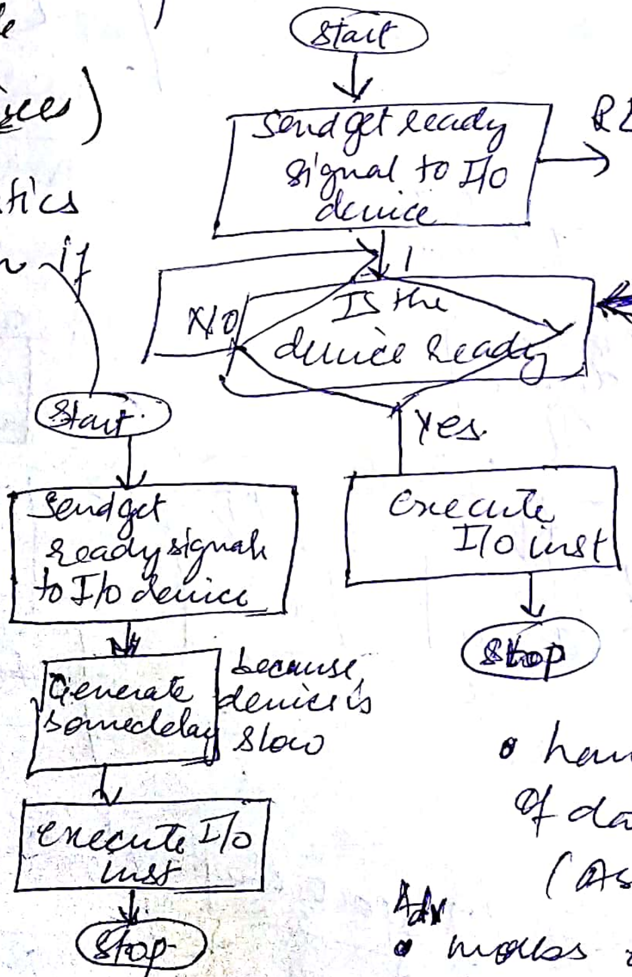
if • Speed characteristics is precisely known. if



compatible in speed.

• Status of I/o device is not checked in syn. mode of So, it is not suitable when speed charac. is not known.

• Suitable when timing characteristic are not known.



REQ. to I/O device

I/o send ACK to i/p

These are usually called as handshaking signal.

because device is slow

• hand shaking mode of data transfer (Asyn.)

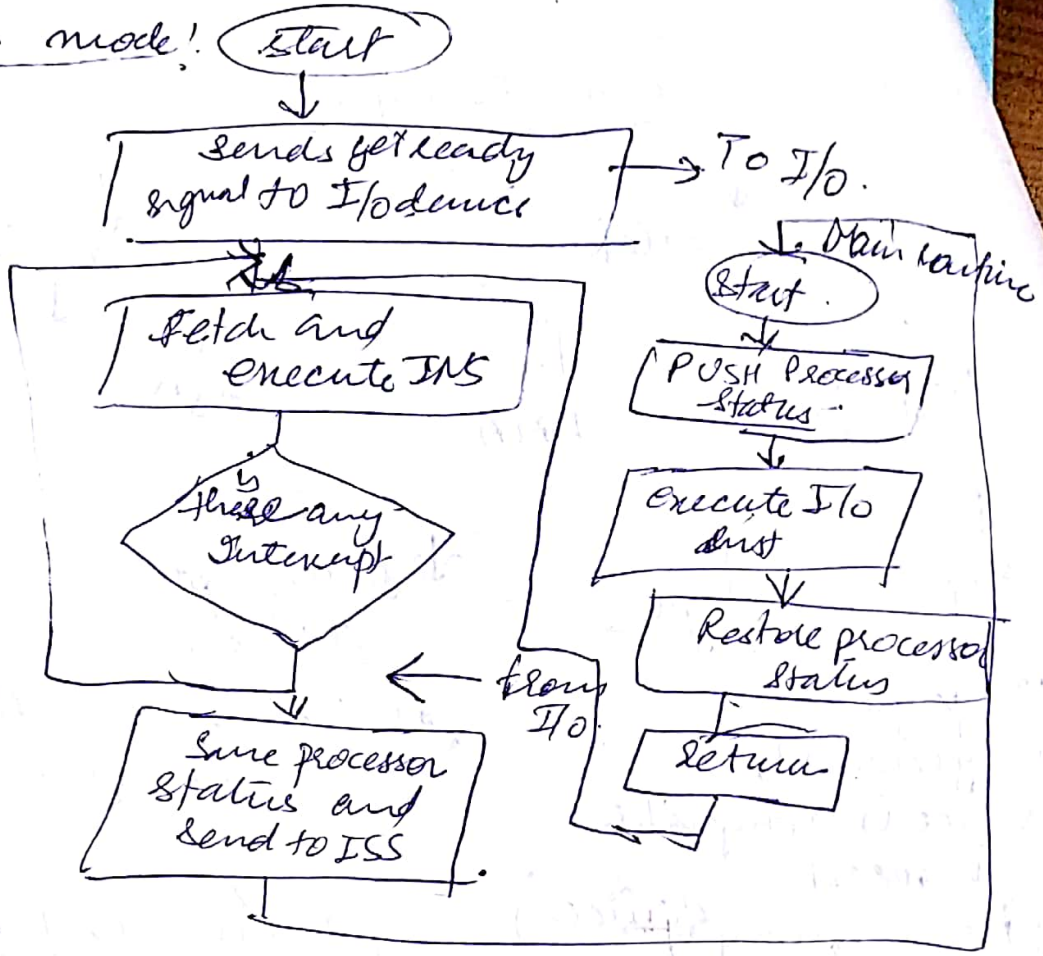
Adv

• works when speed chara. are not known.

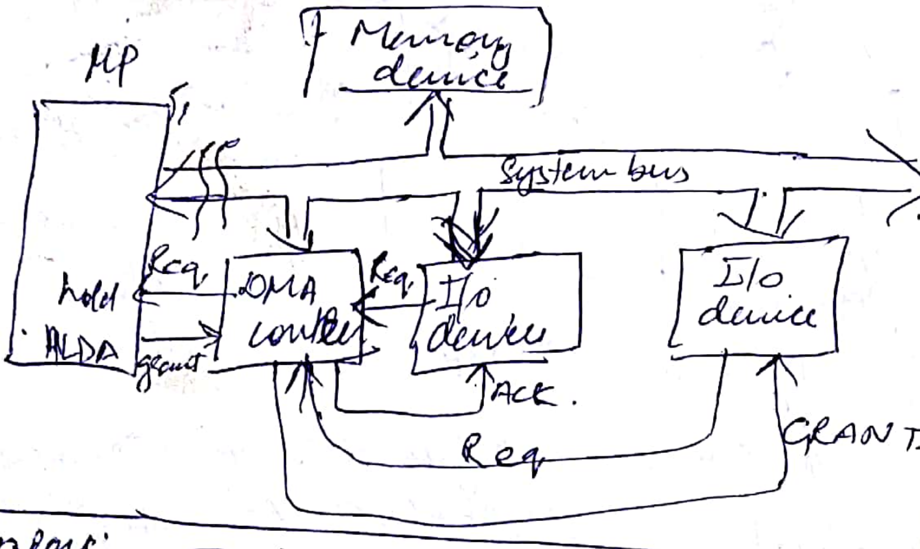
Disadv

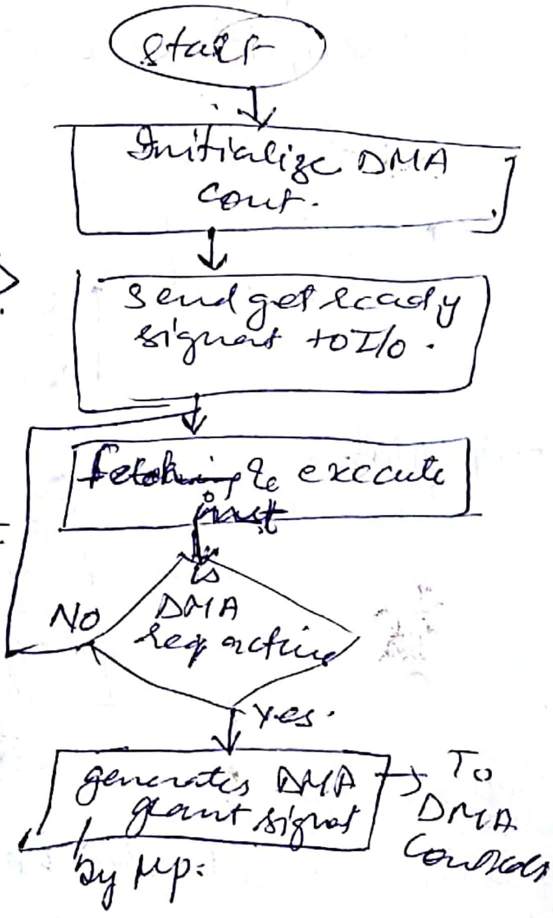
• Precious CPU time is wasted.

Interrupt driven mode:

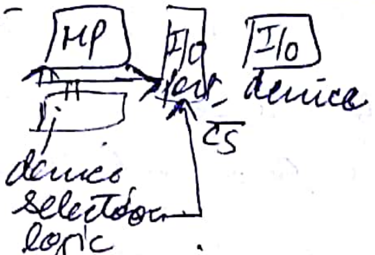


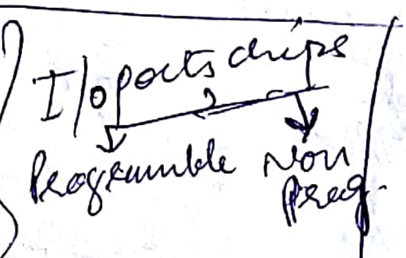
* DMA - mode of data transfer:




Start
Initialize DMA cont.
Send get ready signal to I/o.
~~fetching~~ & execute inst
is DMA req active
NO
yes.
generates DMA grant signal by MP.
To DMA Controlr

I/o port:-





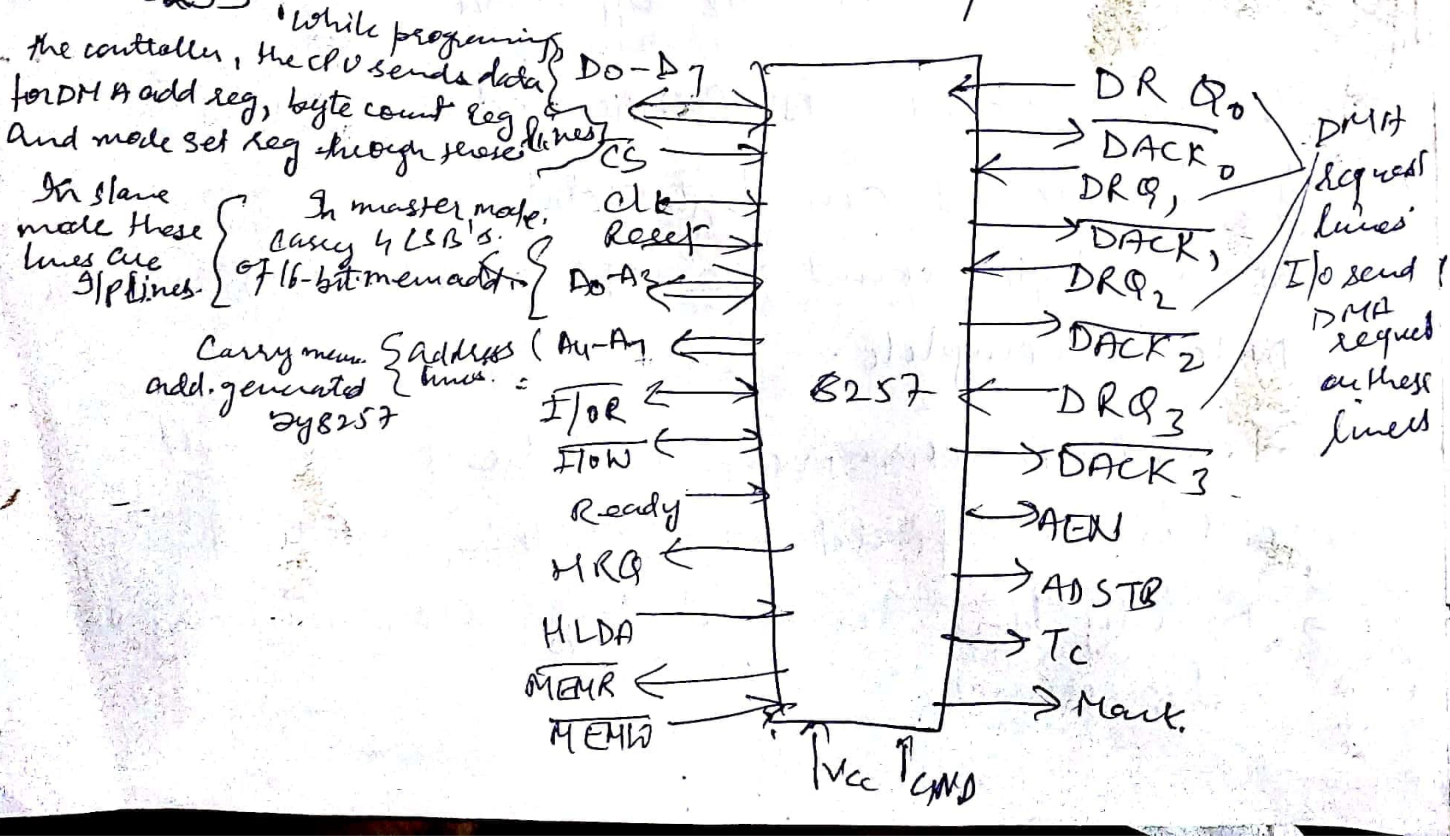
- I/o port provide bus compatibility.
- Tri state buffer • Device selection logic
- Data buffer • control register. • handsking signals, interrupt signals.

# DMA controller: data transfer from fast I/O → memory or from memory → I/O devices

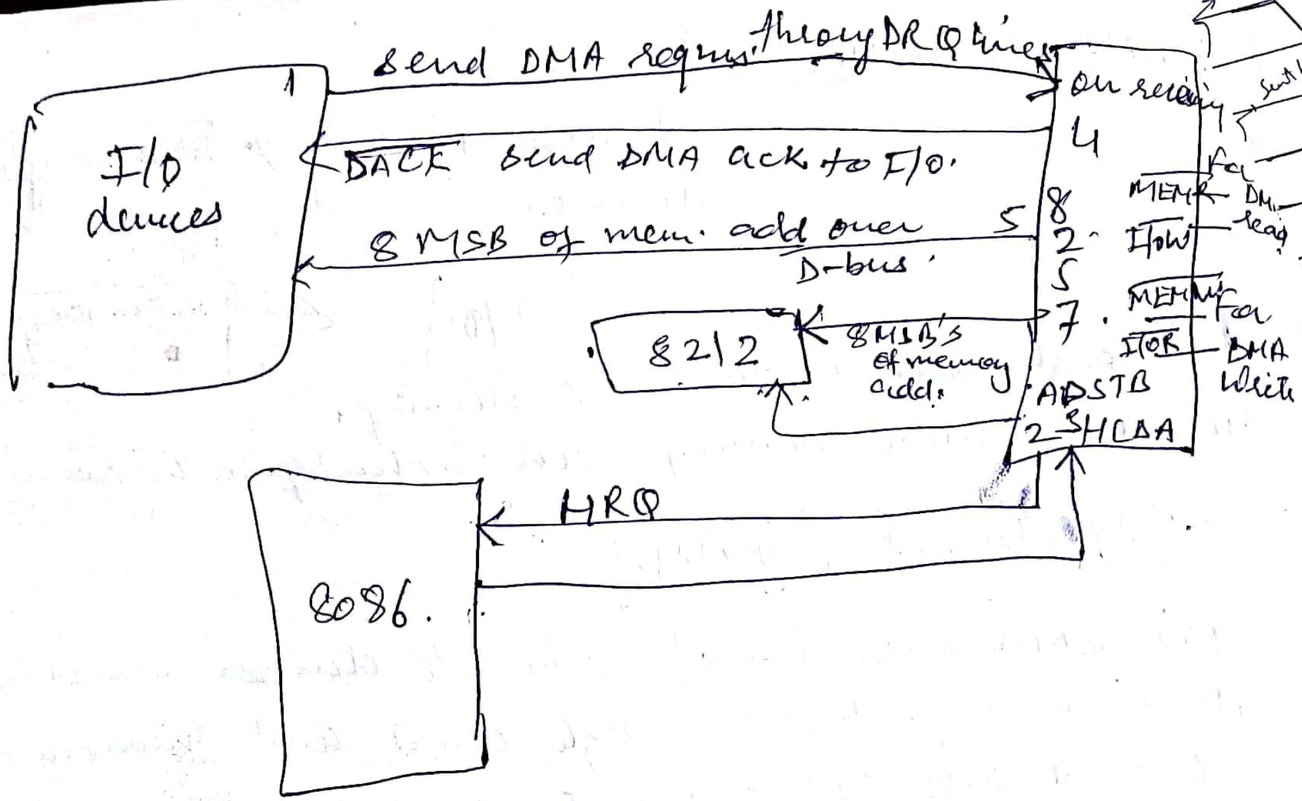
through acc. is time consuming. for this direct memory access technique is required.

I/O device ⇄ to RAM.

for DMA data transfer, the I/o devices must have its own reg. to store byte count and memory add. It must also able to generate control signal required for DMA data transfer. Generally such facilities are not available with I/o devices, hence prog. chip DMA controller have been developed by several manufacturers. for interfacing I/o devices to μp for DMA data transfer. 8257 or 8253



While programing the controller, the CPU sends data for DMA add reg, byte count reg & And mode set reg through these lines
D0-D7
CS
clk
Reset
In slave mode these lines are I/P lines
In master mode, carry 4 LSB's of 16-bit mem addr.
A0-A3
Carry mem. add. generated by 8257
address lines
A4-A7
I/OR
I/OW
Ready
HRQ
HLDA
MEMR
MEMW
8257
Vcc
GND
DRQ0
DACK0
DRQ1
DACK1
DRQ2
DACK2
DRQ3
DACK3
AEN
ADSTB
Tc
Mark
DMA request lines
I/o send DMA request on these lines



$\overline{\text{MEMR}}$ enable the addressed mem for reading data from it.

$\overline{\text{I/OW}}$ enable I/o to accept data.

$\overline{\text{MEMW}}$ " " " writing data to it.

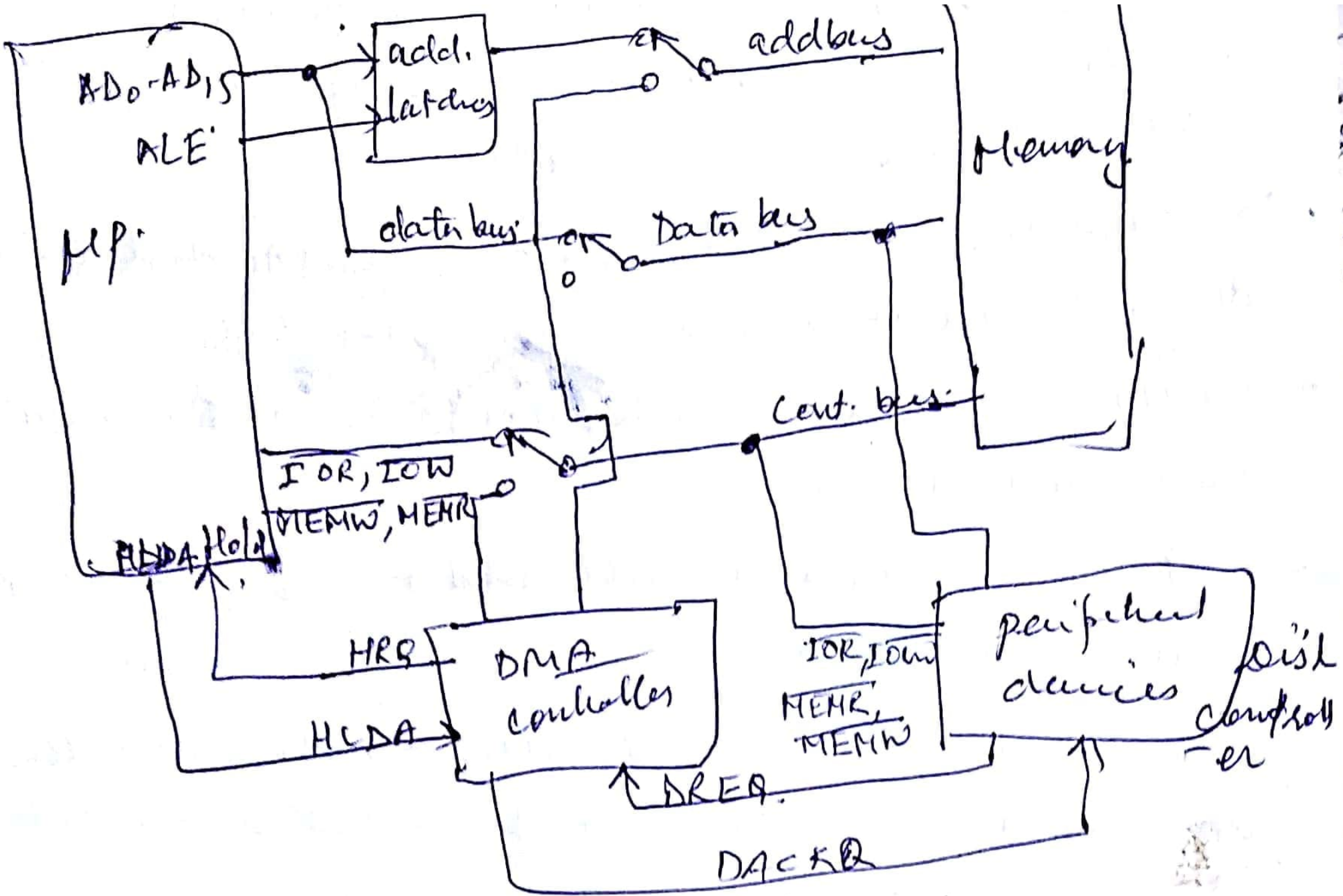
I/oR " " to o/p data.

The byte count is decremed by one after the transfer of one byte of data.

When byte count =0, TC goes high, indicate DMA is complete.

→ four DMA channels can be progremmed either in fixed or rotating mode of operation

→ Ready line is used by slow memory or I/o devices.



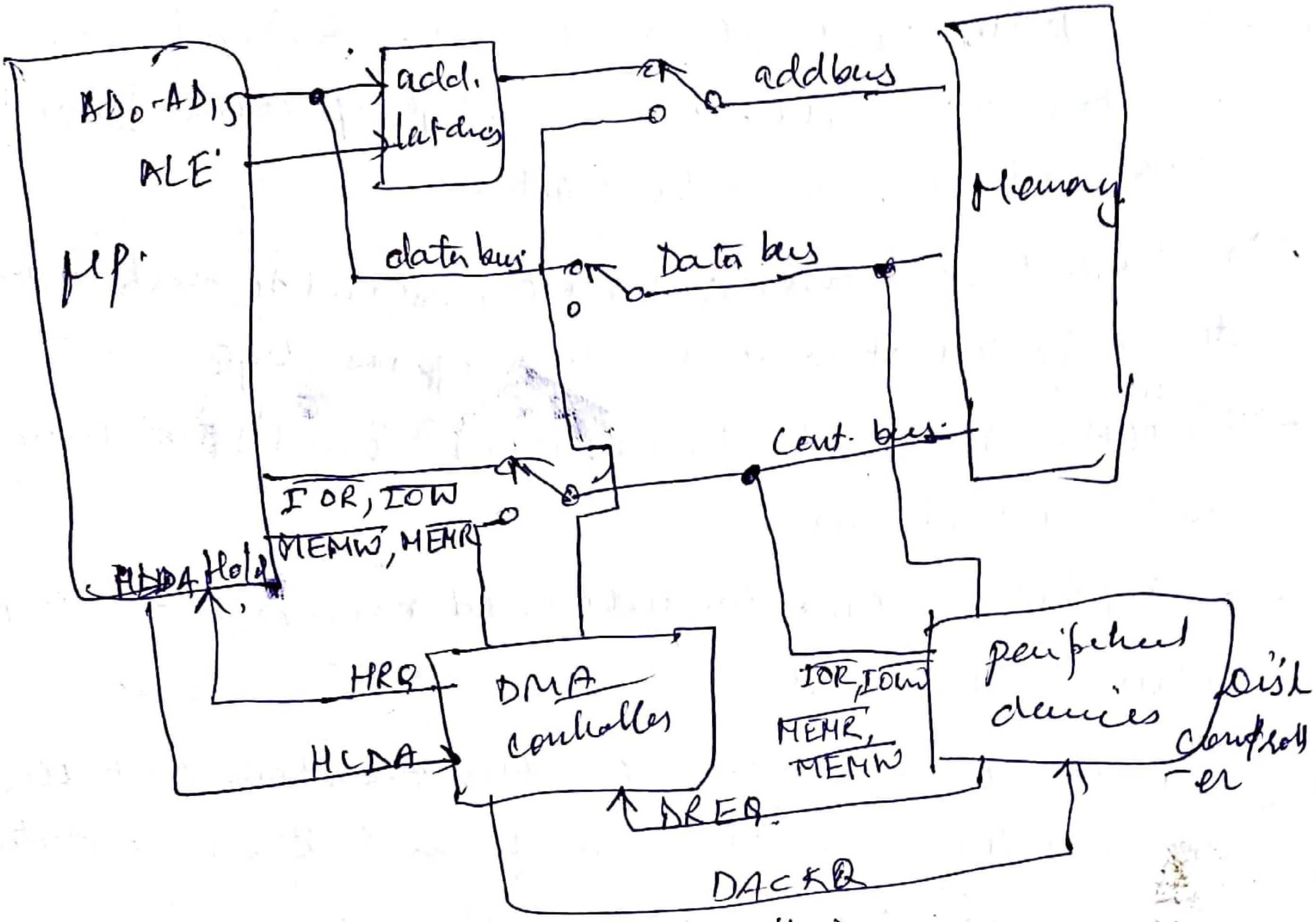
→ Initially, switches are in up position,

→ So buses are connected from μp to sys memory and peripherals.

→ To read a disk file we send a series of commands to the smart disk controller device, telling it to find and read the desired block of data from the disk.

→ When the disk controller has the 1st byte of data from the disk block ready, it sends a DMA request DREQ, to DMA controller,

→ then DMA controller → HRQ to μp → μp responds to this I/p by floating its buses and sending out HLDA to the controller → When DMA cont. receives the HLDA signal, it will send out a control signal which throws the three bus switches down to their DMA position. It disconnect μp from buses and



→ Initially, switches are in up position,

→ So buses are connected from μp to sys memory and peripherals.

→ To read a disk file we send a series of commands to the smart disk controller device, telling ~~it~~ to find and read the desired block of data from the disk.

→ when the disk controller has the 1st byte of data from the disk block ready, it sends a DMA request DREQ, to DMA controller,

→ then DMA controller → HRQ to μp → μp responds to this I/p by floating its buses and sending out HLDA to the controller → when DMA cont, receives the HLDA signal, it will send out a control signal which throws the three bus switches down to them DMA positions. It disconnect μp from buses and

Connects DMA cont. to the buses.

→ When DMA cont. gets cont. of the buses, it sends out the mem. add. where 1st byte of data from the disk controller is to be written.

→ DMA Cont. sends a DACK0, signal to disk controller device to tell it to get ready to o/p the byte.

→ finally, DMA cont. asserts both $\overline{MEMW}$ and $\overline{IOR}$ lines on the control bus.

→ $\overline{MEMW}$ enables the addressed memory to accept data written to it.

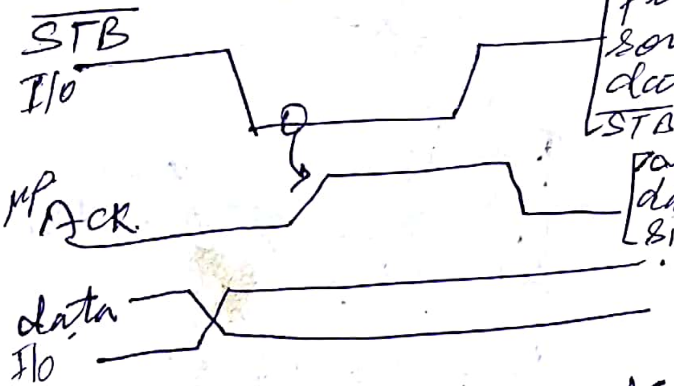
→ asserting $\overline{IOR}$ signal enables the disk controller to o/p the byte of data from disk on the data bus.

→ This data is transferred from disk controller to memory location without passing through the CPU or DMA controller.

* for low rates of data transfer, such as from a keyboard to µp, a simple strobe transfer works well.
For higher speed it does not work, because there is no signal which tells the sending device when it is safe to send the next data byte.
To prevent this prob, a handshake data transfer scheme is used.

Single handshake I/O:

data transfer from I/O to µp.

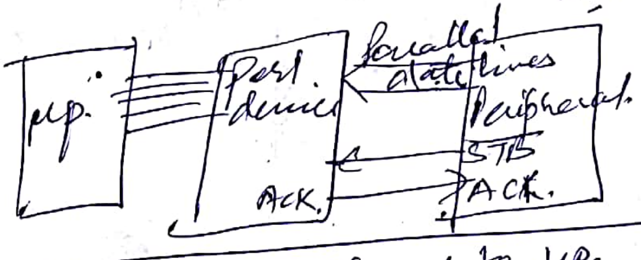


[peripheral o/p's some parallel data and sends STB signal to µp.]

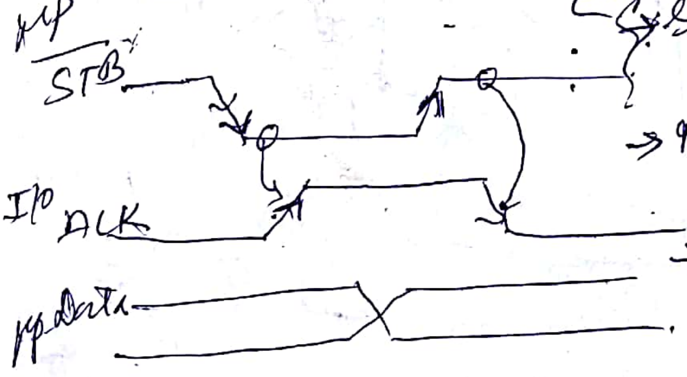
[after reading data, µp send ACK signal to I/O to send another byte of data].

Double handshake data transfer

for data transfere when more coordination is required b/w sending and receiving system double handshake is used.



From a peripheral to µp.



Sending device asserts its STB line low to ask, are you ready.

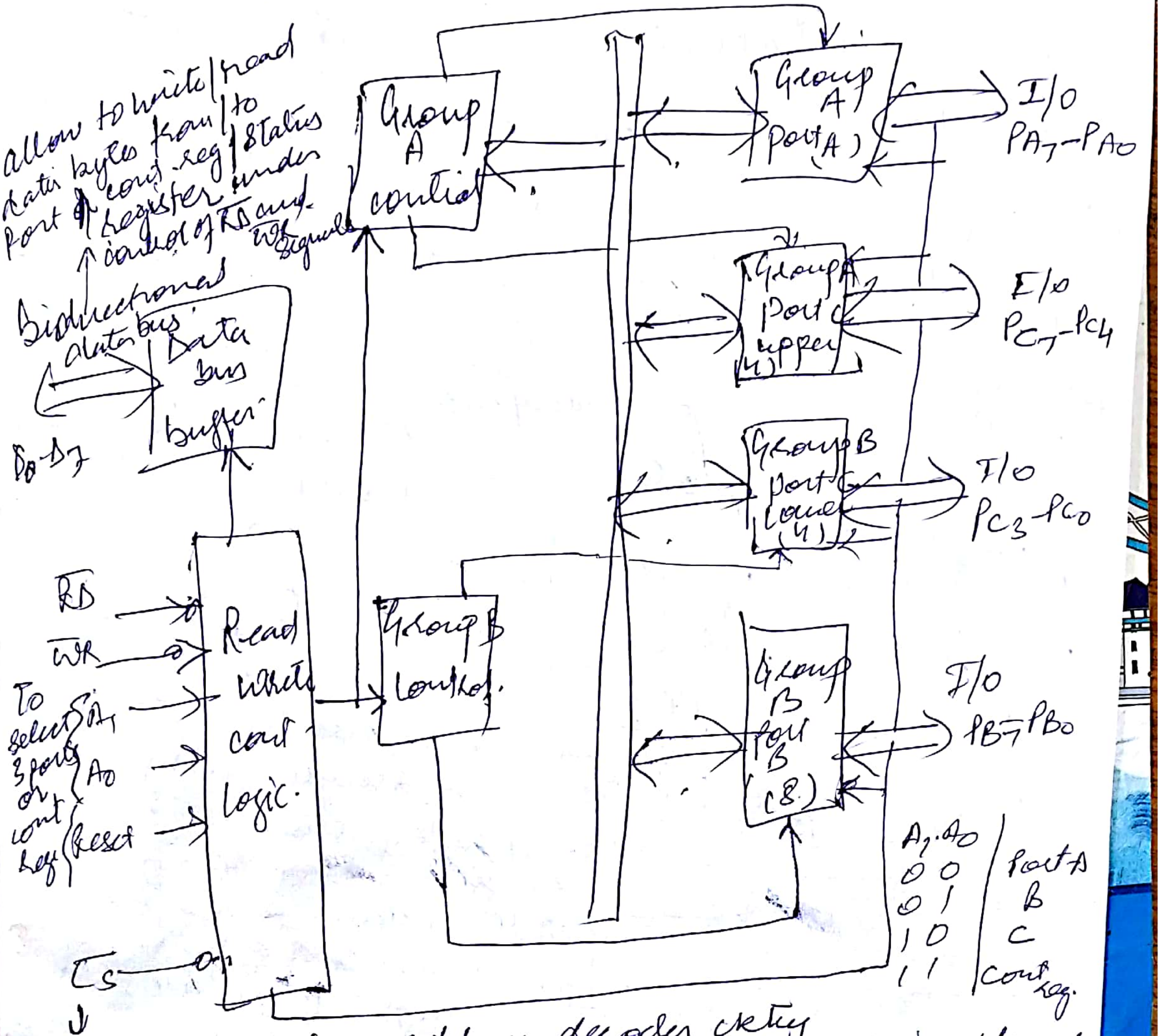
→ receiving system raises its ACK line high — I am ready-

→ peripheral device sends the bytes of data and raises its STB line high to say, here is some valid data for you.

→ After reading data, receiving system drops its ACK line low

To implement handshake data transfer, STB and ACK signals can be produced on a port pin by instruction in the program. but this method usually uses too much time, so parallel port devices such as 8255A have been designed to automatically manage the handshake operation at proper times.

8255A. Prog. Parallel port:



A1	A0	
0	0	Port A
0	1	B
1	0	C
1	1	cont reg.

connected to o/p of address decoder ckty to select the device when it is addressed.

Operational Modes and Initialization:

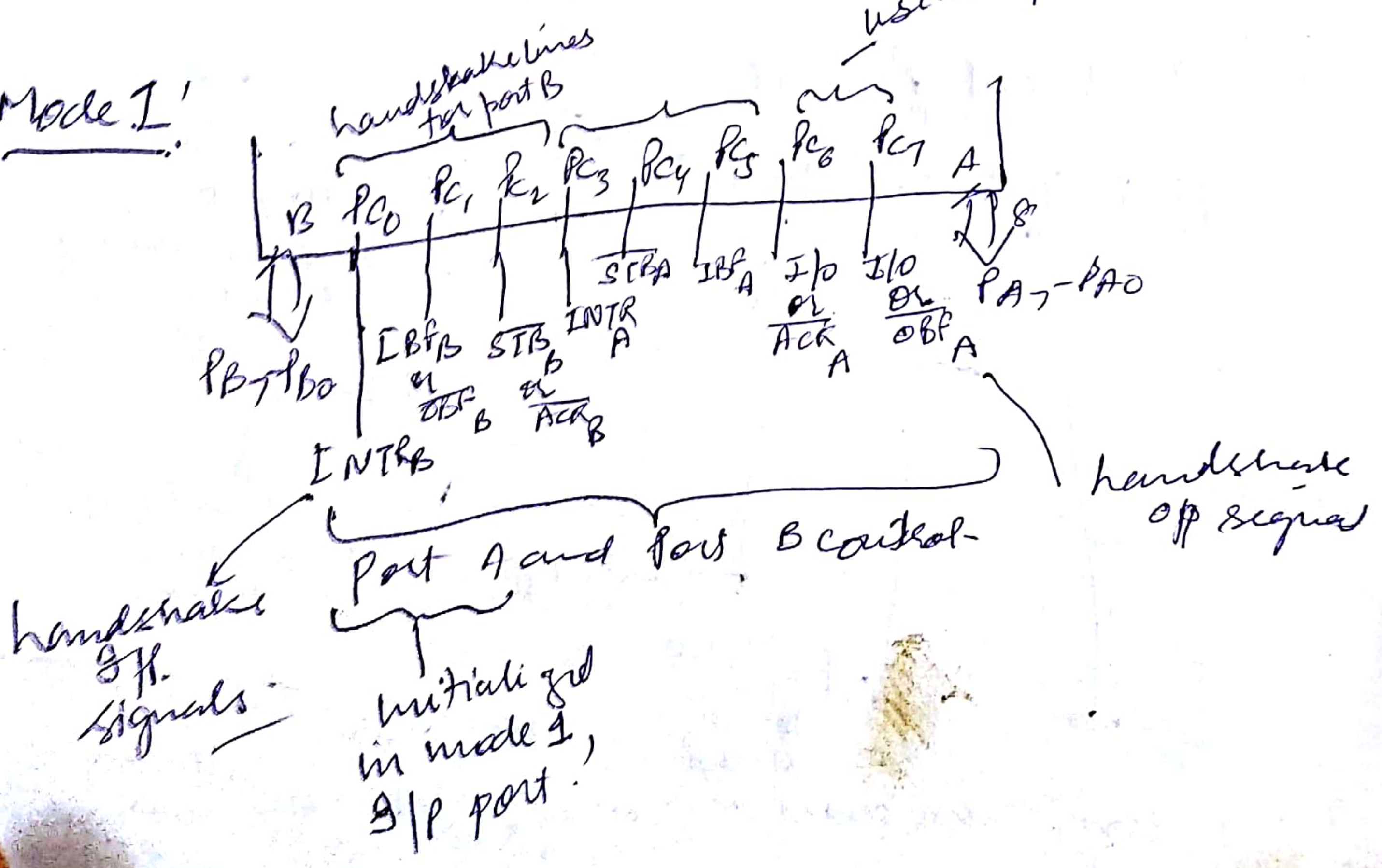
Mode 0. When use a port for simple I/P or O/P without handshaking. initialize that port in mode 0.

If both port A and B are initialized to mode 0, then two halves of port C can be used together as addition 8 bit port or as two 4 bit ports.

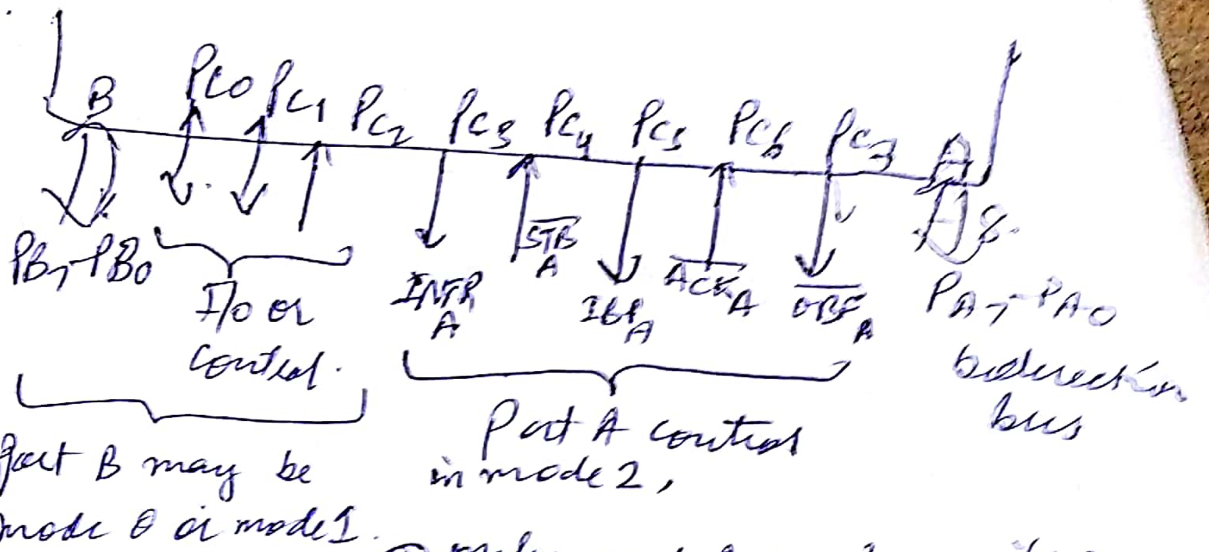
→ When used as O/P:- port C can be set or reset by sending control word to control reg. address.

→ port C — can be initilised as I/P. (1/2)
→ as O/P. (1/2)

Mode 1:



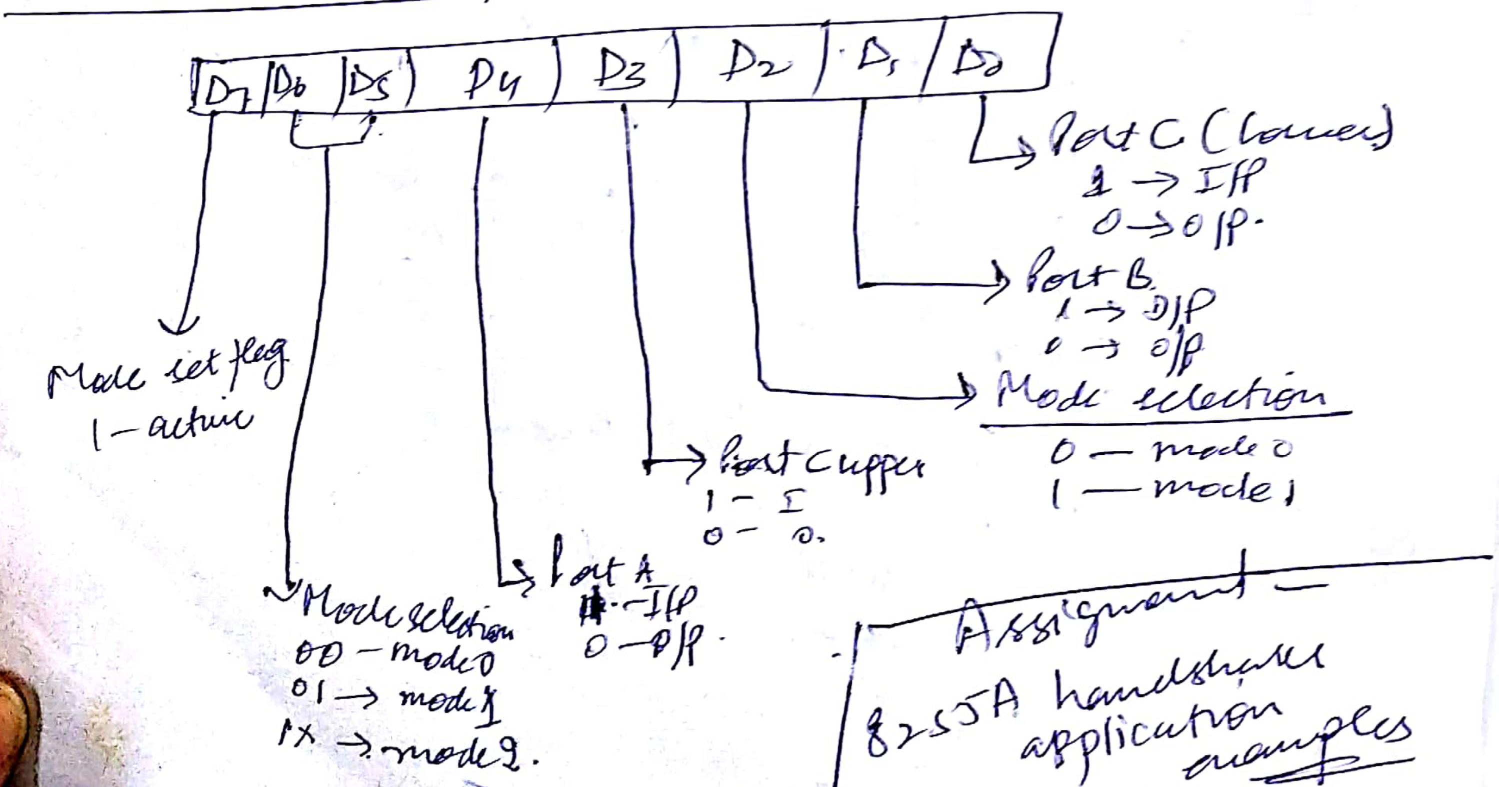
Mode 2


B
PC0 PC1 PC2 PC3 PC4 PC5 PC6 PC7
A
PB7-PB0
I/O or control.
INTR A
STB A
IBF A
ACK A
OBF A
PA7-PA0
bidirection bus
port B may be mode 0 or mode1.
Port A control in mode 2,

① only port A can be initialized to mode 2.

② In mode 2, port A can be used for bidirectional handshake data transfer, means data can be o/p or I/p on same eight lines.

Control word format



Assignment —
8255A handshake application examples