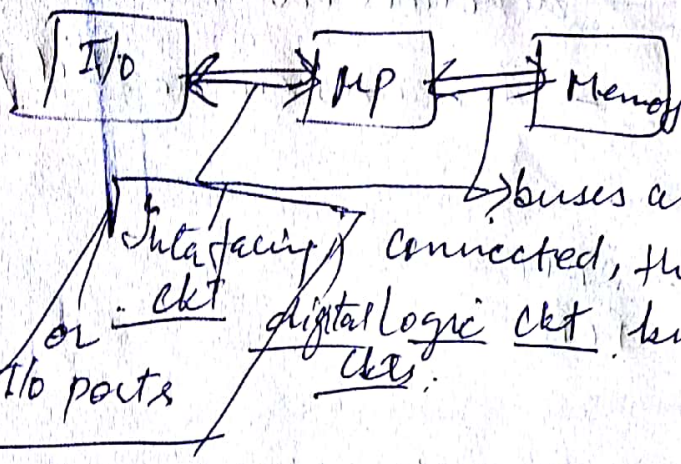
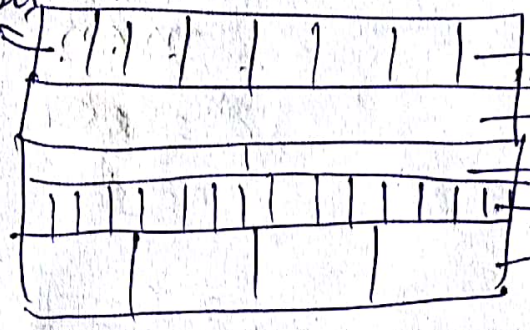
Interfacing:



1 gf can store 1-bit M/M.



If we want to connect M to µp :-

$2^n = N$ (We are taking 8 bit/1 byte in each m.L)

If $n = 16$, $2^{16} = 2^6 \times 2^{10} = 64 \times$ KB

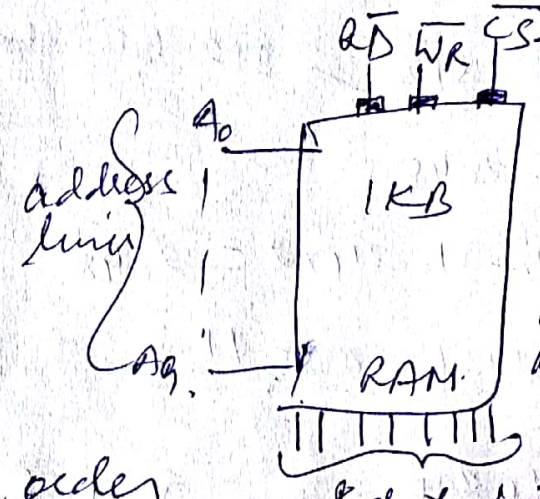
n → no. of address lines

$2^{10} = 1$ KB

$2^{20} = 1$ MB

$2^{30} = 1$ GB

$2^{40} = 1$ TB



if 1 chip can store 4 bit we need 2 chip's to make 8-bit memory, and CS is used to do RD and WR on particular chip) store (it can 8 bit in each M.L)

in 8085, lower order add. bus is used as data bus.

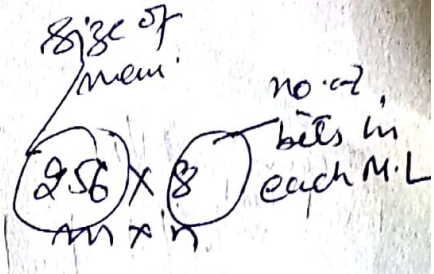
① Add lines :- depends on mem. location.

② data lines :- depends on no of bit / M.L. in each.

③ $\overline{CS}$ / $\overline{CE}$

④ $\overline{RD}$ / $\overline{WR}$ → RAM
$\overline{RD}$ ROM

* Representation of M/M chip :-



$m \times n$

↓

no. of m.L

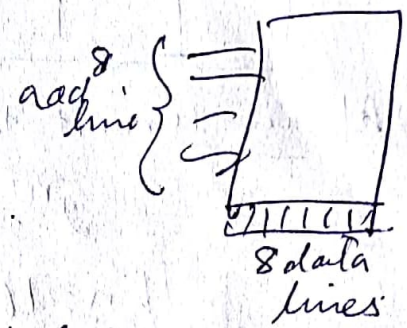
$2^n = N = m$

$2^{16} = 64KB$ mem. size (m)

n = No. of bits in each m.L.

$2^8 = 256$.

$n = 8$.



eg find the length of add and data.

2048×512 ← 512 bits stored in each M.L.
So, 512 data lines

$2^n = 2048$

$2^1 \times 2^{10}$ $1024 \times 2 = 2048$

no. of add lines = 11

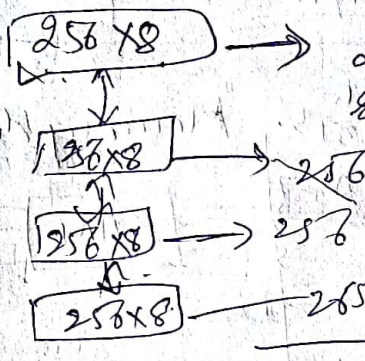
(iii) 1024×1024 — no. of data lines = 1024
2^{10} / 1024 — no. of add lines

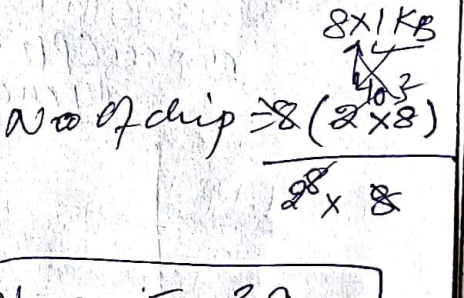
(2)

Calculating No. of chips required to design a memory?

$$\text{No. of Chip} = \frac{\text{M/M to be designed}}{\text{No. of add lines}}$$

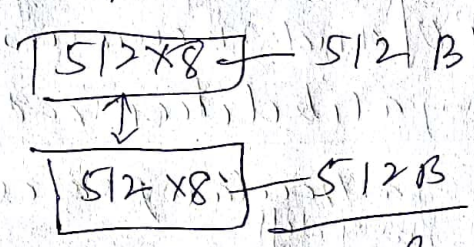
find no of 256×8 ROM chip required to desig ~~8 KB~~ 8 KB of m/m. (a) 4. (b) 16. (c) 32.





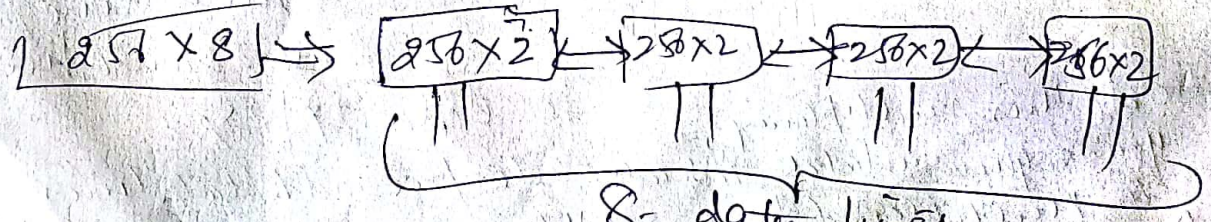
No. of chips = 32

eg 512×8 Chip — design 1KB m/m.



$$\text{No. of chip} = \frac{2^{10} \times 8}{2^9 \times 8}$$

eg 256×2, design 256 B m/m, how many chip

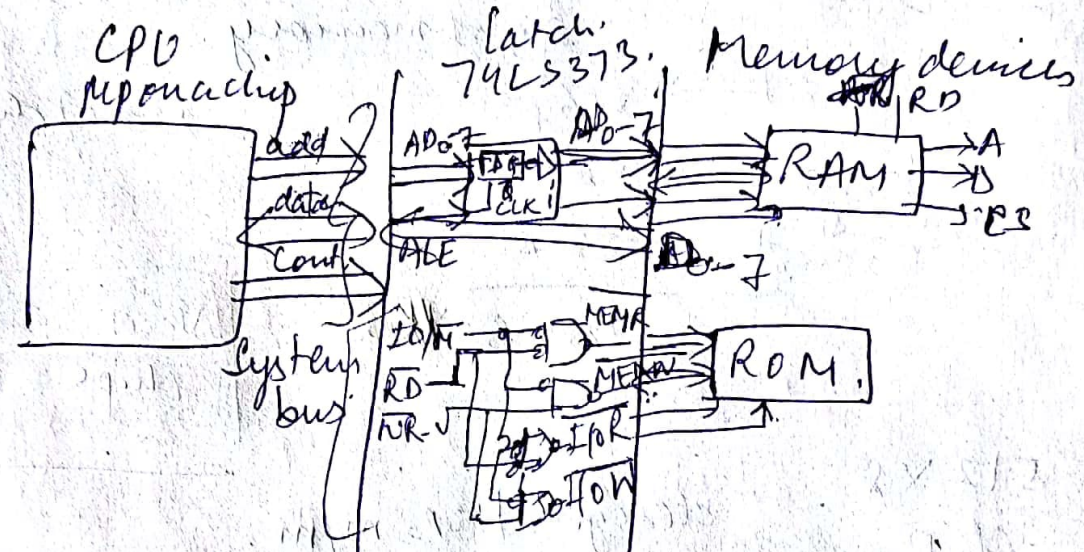


Q. find no. of (1024 × 2) mem. chips required to design 16 KB

$$\text{No. of chips} = \frac{16 \times 2^{10} \times 8}{2^{10} \times 2}$$

$$= 64 \text{ chips}$$

* Incompatibilities in m/m interfacing:



* Signal generated by CPU and required by memories are not same, so incompatibilities occur:-

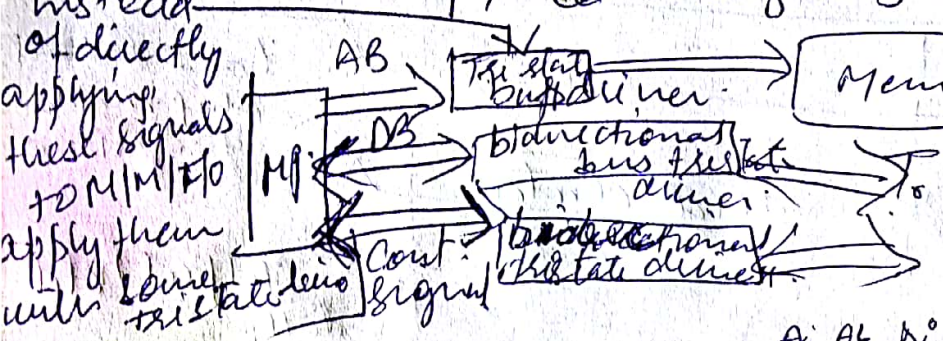
(i) bus incompatibility:- particular processor bus is not comp-with bus of diff memory devices

eg. In 8085, TDM is done to form Add/data bus. generate add on same line then use same lines for data tx/Rx.

to do this multiplexing, you require a latch which interface add/data lines in multiplexed form to m/m.

* Removed by using tristate buffers - 74LS373 or 8282

* Electrical incompatibilities: — CPU is provided with elec parameters (V, I, current) when CPU is driving a large no. of chips it may not be driving all the chips simultaneously, so this is overcome by using bus drivers.



Ai Ai Di

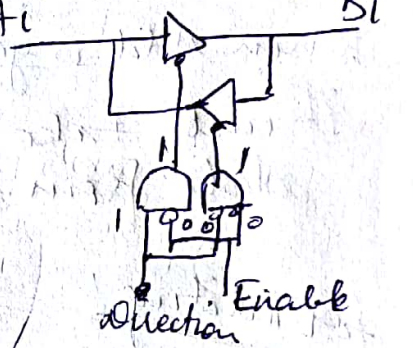
Tristate unidirectional bus driver.

O/P

Active High/Low enable line

74LS244 (unidirectional) tristate buffer

74LS245 (bidirectional tristate buffer)



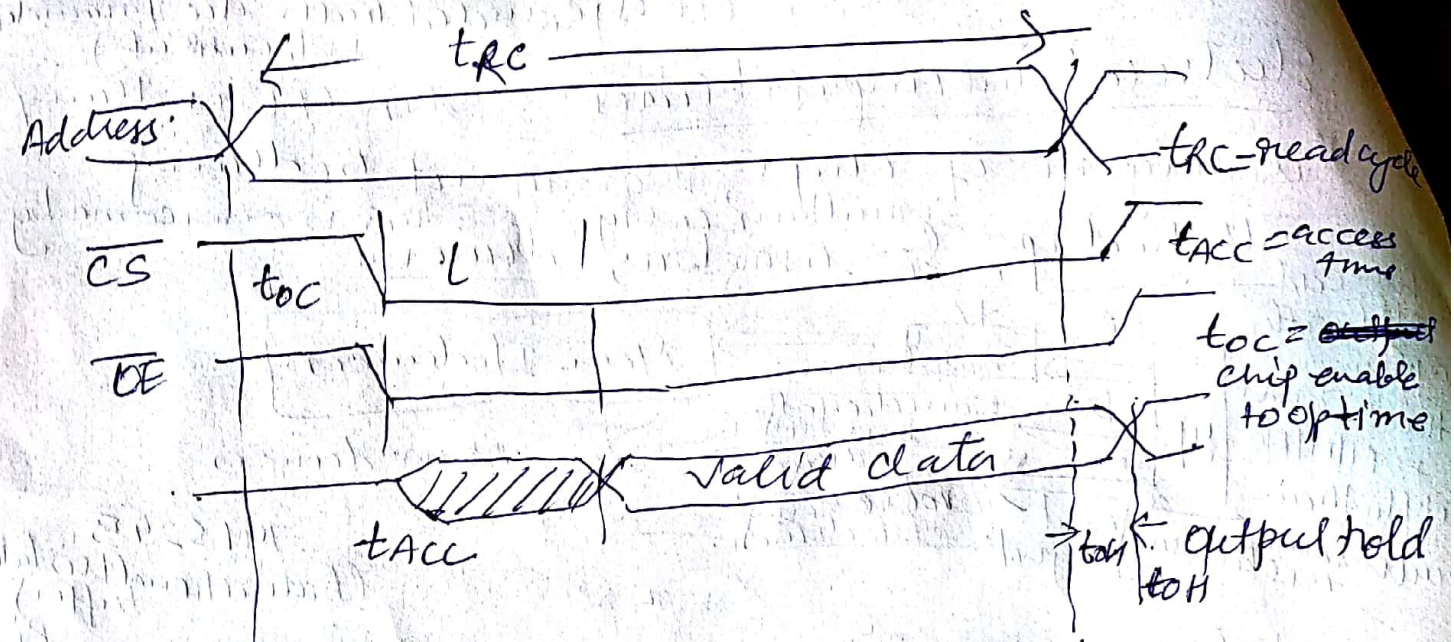
bidirectional tristate bus driver.

DIR	EN	
0	0	B → A
1	0	A → B
	1	Tristate

Why tristating require?

When CPU connected with DMA or I/O port, to perform high speed data transfer it has to relinquish system bus, so it tristated the buses.

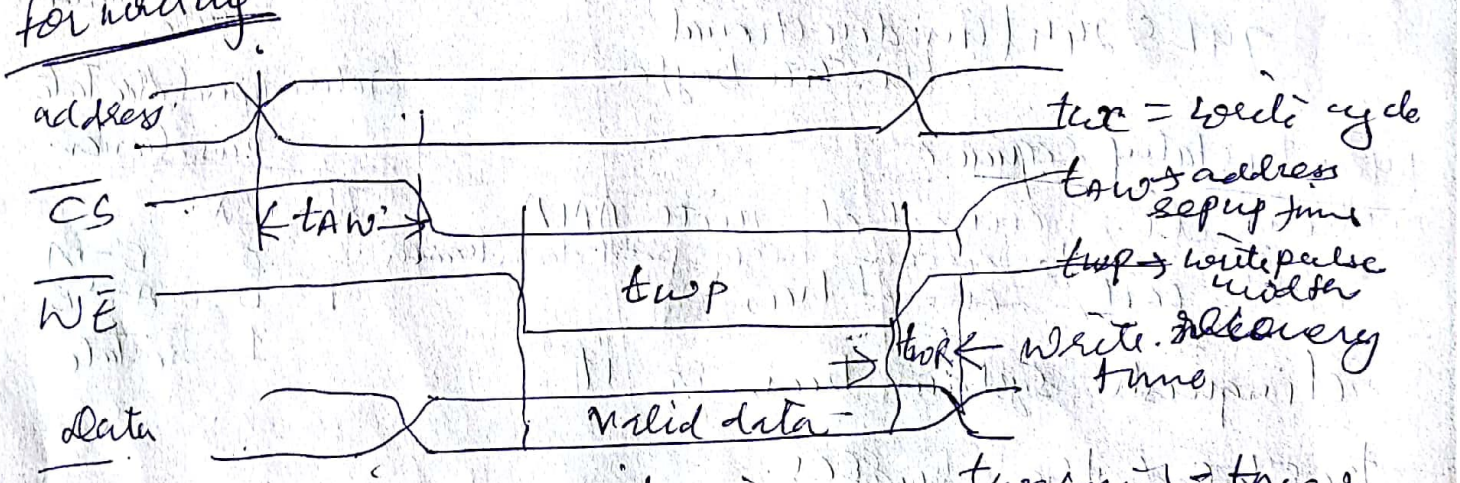
* Time/speed incompatibility:



When Add is generated by CPU it will take some time to ACC valid data from memory.

$t_{RC(Min)} = t_{ACC}$

for writing!



$t_{WC(min)} = t_{AW} + t_{WP} + t_{WR}$

On the other hand µp take its time for R/W cycles is assumed.

$t_{add} + t_{ds} = 225$ nsec (8085)

t_{add} ↓ address setup time. t_{ds} data setup time. clk freq = 320 ns

When appropriate address do µp, it take sometime to this address come on address bus, it is called address set-up time.

So for reading operation, µp takes 2½ cycles,

∴ 2½ × 320 = 800 nsec (total time)

∴ $t_{ad} + t_{ds} + t_{acc} = 800$ nsec

∴ memory acc. time $t_{acc} \leq 800 - 225$

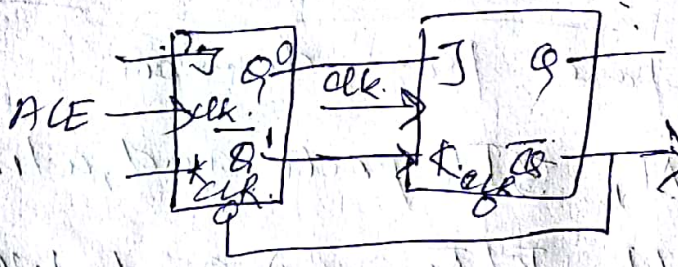
$t_{acc} \leq 575$ ns.

$t_{acc} \leq$ 575 ns, otherwise it will not get correct data from memory,

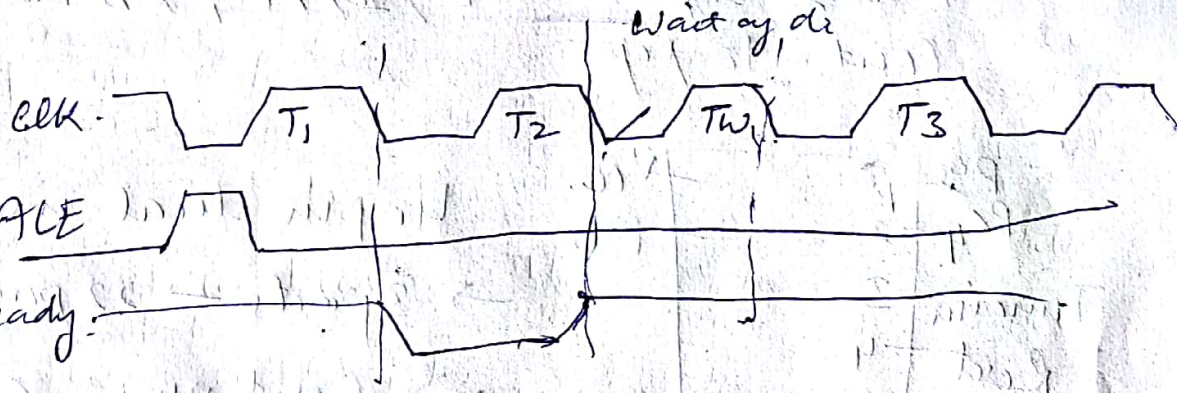
if it is not ! — How do you interface slow memory devices ?

→ (1) Slow down clk freq (but not good sol, overall speed will reduce)

→ Use of wait cycles (8085) (in b/w T_2 and T_3), provide some wait cycle.



J = K = 1,
clk = 1
Q = 1
$\overline{Q}$ = 0



if this applied to Ready signal of µp,

* 8155 (Programmable Peripheral interface) (Parallel Interface)

→ To get data, μp read or send data to I/O devices.

→ accepting / sending data from I/O devices

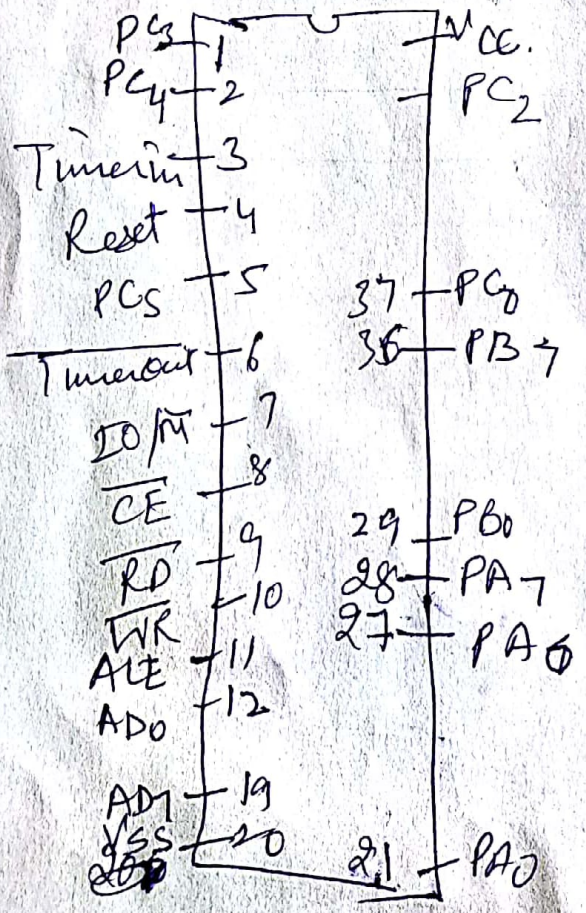
→ I/o devices are supporting devices.

* μp has more speed than I/o devices and vice versa.

→ It is a multipurpose Prog. device

→ It has 256 B RAM, 3 I/o ports and, a timer.

→ 4 cont. signal ALE, IO/M, RD and WR. cont. to μp.



40 pin dual in line package

→ 3 ports — 22 pins

→ Cont pins — 6

→ Add/data.

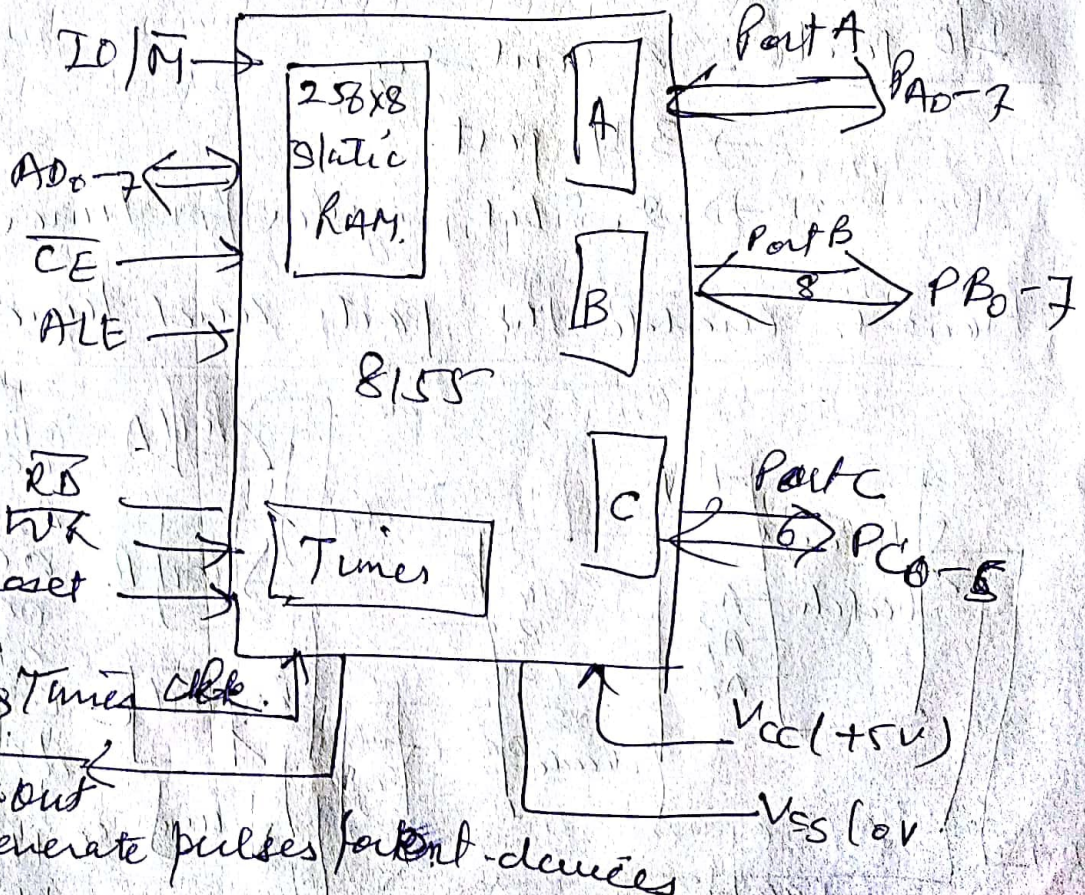
→ Timer — 2 pins

It has. Two section

→ 256 byte of R/W memory

→ Prog: I/O
- Two 8 bit parallel I/O ports (A and B)
- one 6 bit port (C)
- 14 bit
- 16 Timer
  - Lower B
  - Higher byte

Block diagram



It will generate pulses for baud-devices

Control Logic

→ designed to eliminate the need of ext. ckt for demux and generating the seperate cont. signal for memory and I/O.

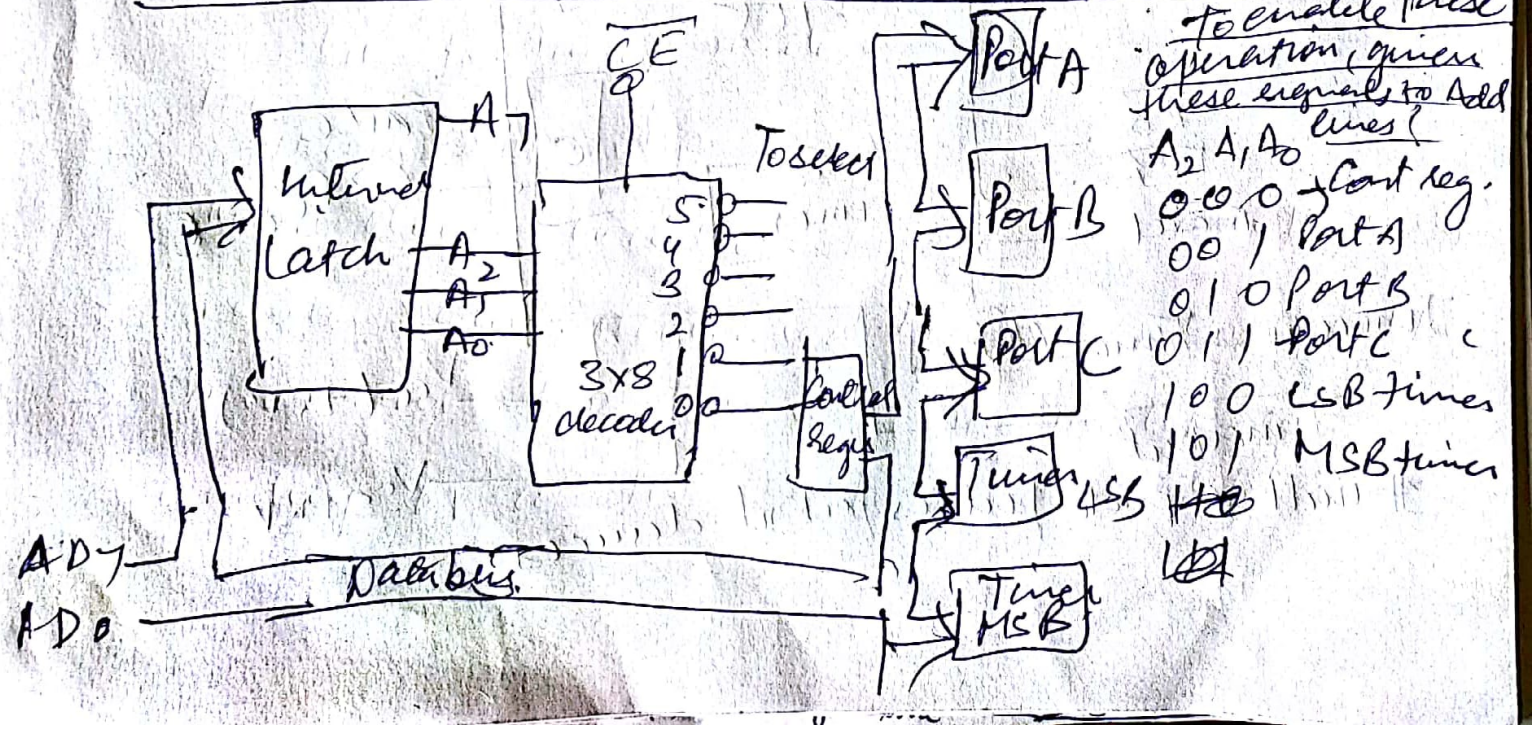
→ Six cont. signal. CE, IO/$\overline{M}$, ALE

After making proper interfacing with I/O we can send/rex the data by sending inst.

IO/$\overline{M}$ → 1 → I/O operation
→ 0 → Memory operation

I/O ports → A cont. reg.
→ 3 I/O port.
→ 2, 8 bit register for timer

How to enable the ports and timer?



Port address

address lines

$A_7\ A_6\ A_5\ A_4\ A_3\ A_2\ A_1\ A_0$

0 0 1 0 0 0 0 0 → 20H control reg.

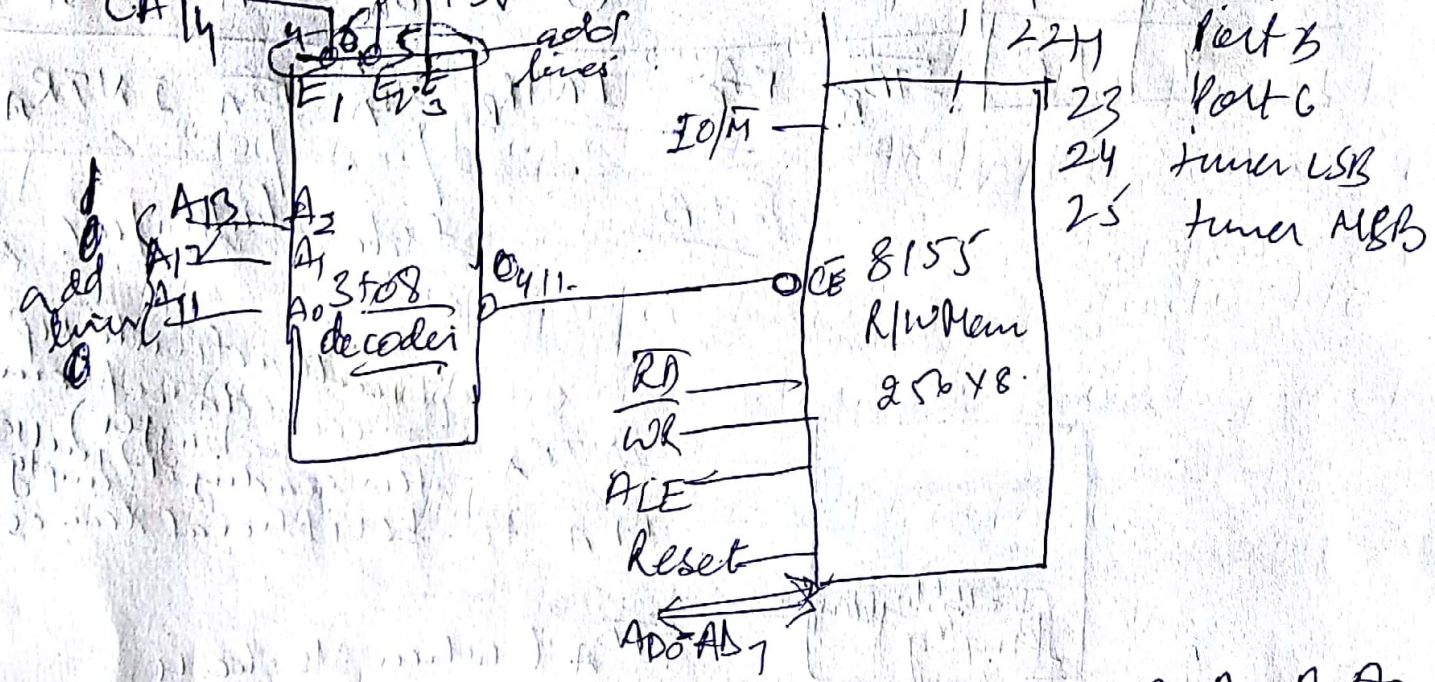
0 0 1 → 21H Port A

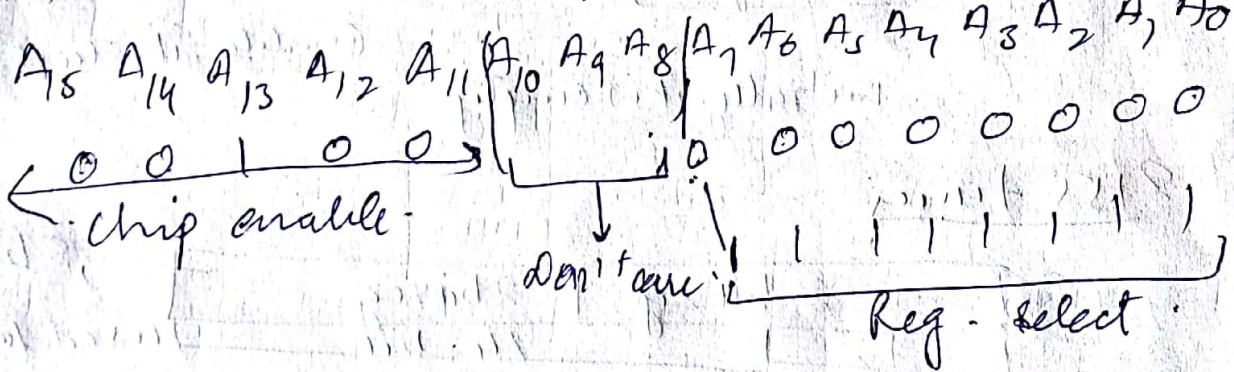
22H Port B

23 Port C

24 timer LSB

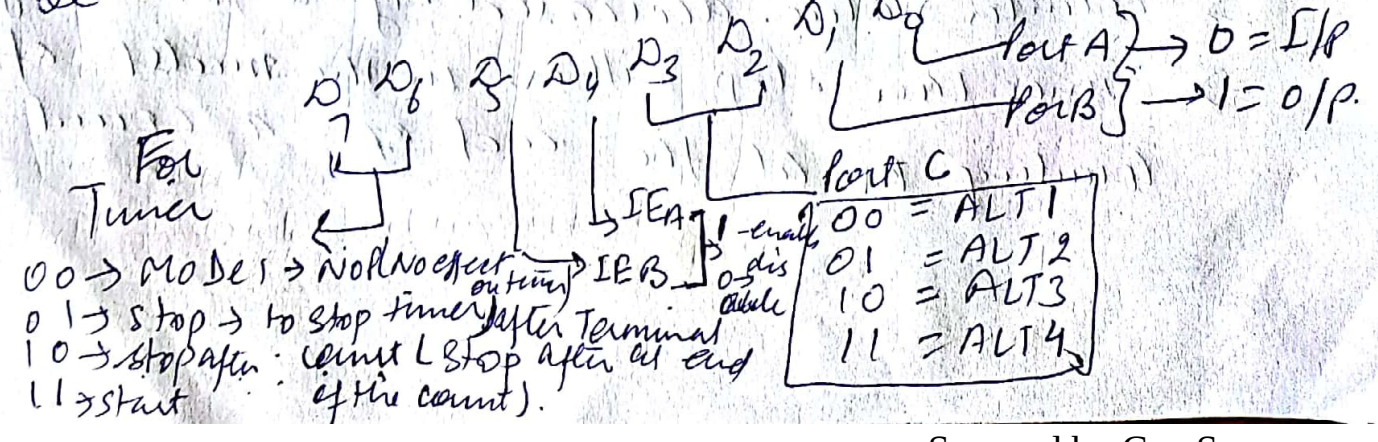
25 timer MSB





---

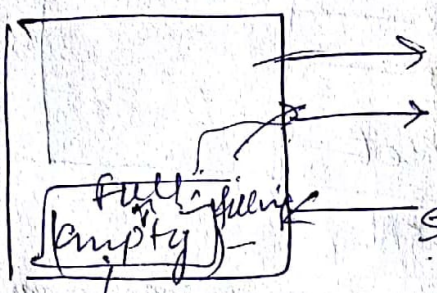
* **Control word.** 8-bit contents of control register. The contents will define which ports can be used as I/O.



ALT#	D_3 D_2	PC_5	PC_4	PC_3	PC_2	PC_1	PC_0
ALT1	0 0	I	I	I	I	I	I
ALT2	1 1	0	0	0	0	0	0
3	0 1	0	0	0	STB_A	BF_A	$INTR_A$
4	1 0	STB_B	BF_B	$INTR_B$	STB_A	BF_A	$INTR_A$

handshake signal for port A

Port B — Strobe, buffer full, Port A



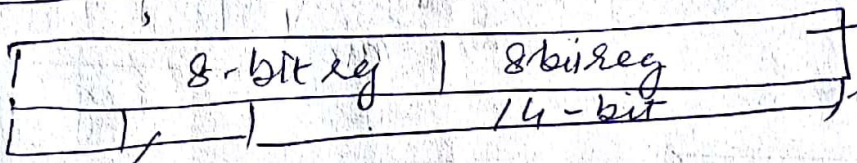
$INTR_A$ (it will activate MP to read data from the buffer) after reading, it will again empty

BF_A to peripheral devices

STB_A (when strobe is enable, it start filling

initially it is empty

8155 timer



8 bit reg. used for comb.

2 bits for timer mode

→ requires clk as I/P

→ 14 bit down counter operates in 4 diff modes

→ counter value decreased by 2 for every clk pulse

→ Timer can stop either in the midst of counting or at the end of a count.

Register format

- MSB

D_{15}, D_{14} ... D_9 D_8

M_2	M_1	T_{13}	T_{12}	-- --	T_9	T_8

- LSB

D_7 ... D_0

T_7		T_0

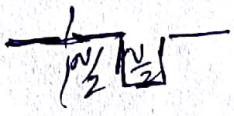
Mode 0 → 00 → Single pulse mode.

1 → 01 → square wave mode.

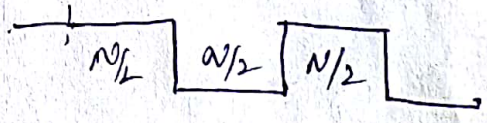
10 → one shot terminal count

11 → for every count is given small pulse.

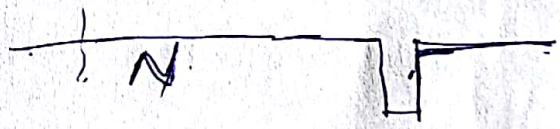
Mode 0 : 00



1 : 01



2 : 10



3 : 11

#