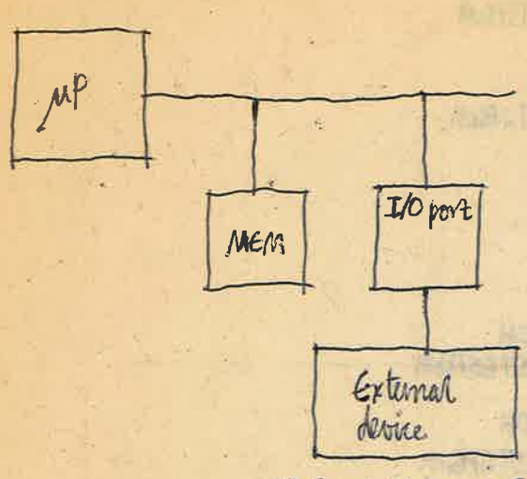
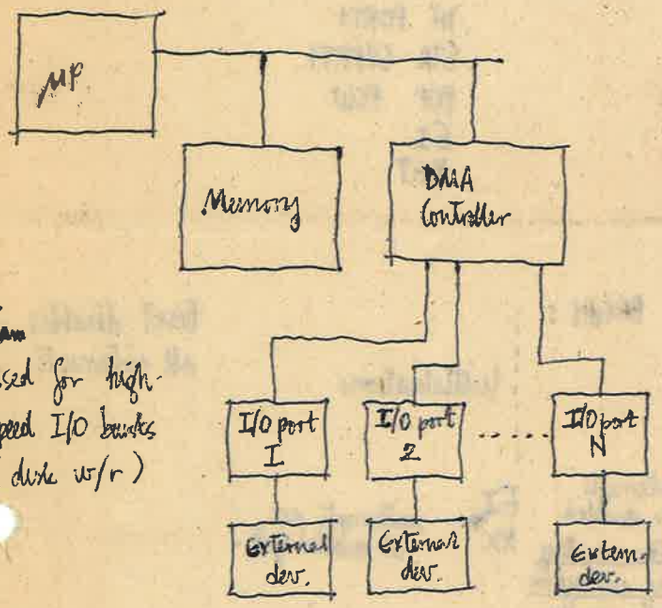
EE514 18/11/83

Textbook: µComputer Interfacing
Harold S. Stone



i) Port initiated interrupts
ii) Processor initiated periodic status checking.

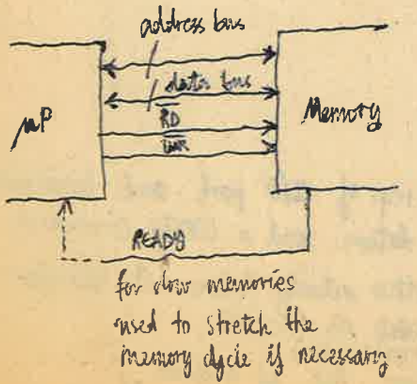


Can used for high-speed I/O devices (disk w/r)

DMA Controller like a second processor, dedicated to I/O connects ports directly to memory.

DMA active ⟹ Processor idle

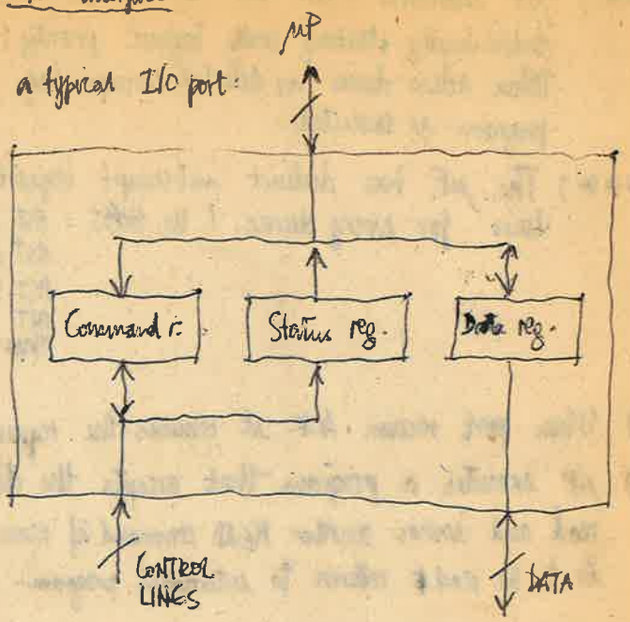
Processor - Memory Interface



for slow memories used to stretch the memory cycle if necessary

I/O Interface

a typical I/O port:



Program Controlled I/O: (no interrupt) Processor initiates

I/O Read

i) Test Status and wait for ready
ii) Load Command Reg. with READ Command
iii) Test Status and wait for ready
iv) Read DATA, go to step 2 or 3

```
WAIT: IN STATPORT
      ANI 00000010B
      JZ WAIT
      MVI A, COMWRD
      OUT COMMPORT
WAIT1: IN STATPORT
      ANI 00000001B
      JZ WAIT1
      IN DATAPORT
```

I/O Write

i) Test Status & wait for ready
ii) Transfer output DATum
iii) Transfer a WRITE Command.

Example: A serial port (110 baud)
= 10 char/sec
take 100 msec for every transfer.

Interrupt driven I/O "overhead is larger"

I/O Read interrupt

i) The port A is enabled by the µP.
ii) The I/O port completes data transfer and asserts INTERRUPT REQ.
iii) After certain time µP answers, µP must begin device identification.

There are three ways:

* µP transmits INT ACK to whole I/O system. The highest priority device with pending interrupt request, answers (responds by placing an identifier on I/O bus,)
Then µP executes a program related to this port. (Daisy Chain Structure)

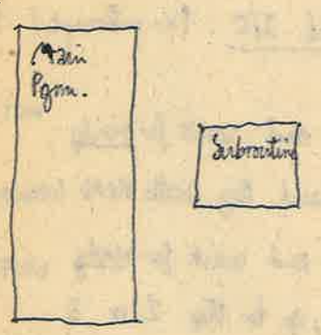
**) μP transmits INT ACK to each port individually starting with highest priority; When active device is detected corresponding program is executed.

***) The μP has distinct interrupt request lines for every device. (In 8085 : RST 7.5
RST 6.5
RST 5.5
INT
TRAP)

iv) When port receives ACK it removes the request

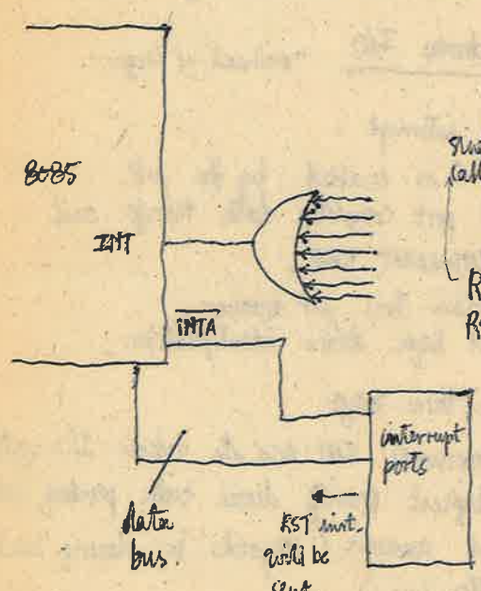
v) μP executes a program that accepts the datum read and issues another READ command if more data is to be read & returns to interrupted program.

Ex



μP's typically have interrupt mask bit. It changes state when interrupt is acknowledged. Therefore during polling it will not be interrupted.

8085



single byte call instr.

RST n (0-7)

RST 1 ≡ CALL 0008H
RST 2 ≡ CALL 0010H

v) Another datum & WRITE command, go to step 2

```
ORG  Ø
JMP  BEGIN

ORG  8
JMP  RST1PGM

ORG  10H
JMP  RST2PGM
 :
 :
ORG  2CH
JMP  RST55PGM

ORG  30H
JMP  RST6PGM
```

```
RST1PGM :  PUSH PSW
           IN   PORT1
           STA  SVPRT1
           POP  PSW
           EI
           RET
```

```
BEGIN :
        :
        Initializations
        :
        EI      ← interrupts are not enabled yet.
        XX
```

RESET disables all interrupts

interrupts are enabled after executing the instruction following the EI

```
RST2PGM :
        :
        EI      ← nested interrupts can be handled.
        :
        :
        RET
```

I/O Write

i) μP enables int. req. of I/O port and transmits to port output datum and a WRITE command

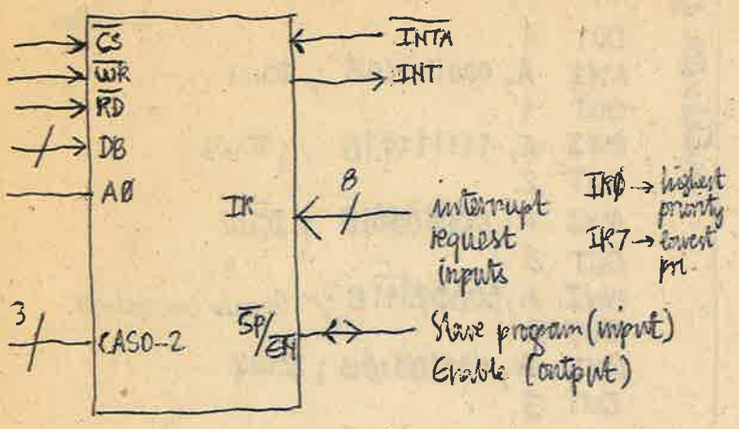
ii) μP returns to other activity when data transfer is done, an INT REQ is received. (PORT sends)

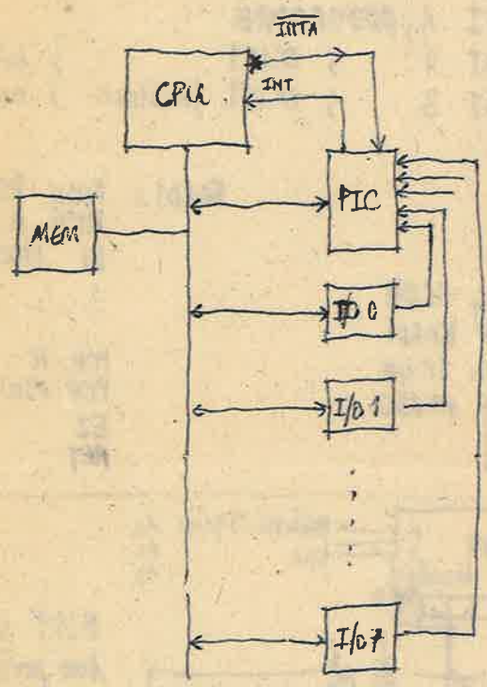
iii) After certain time ACK is received,

iv) Device removes request

514 21/11/83

8259 Programmable Interrupt Controller "PIC"



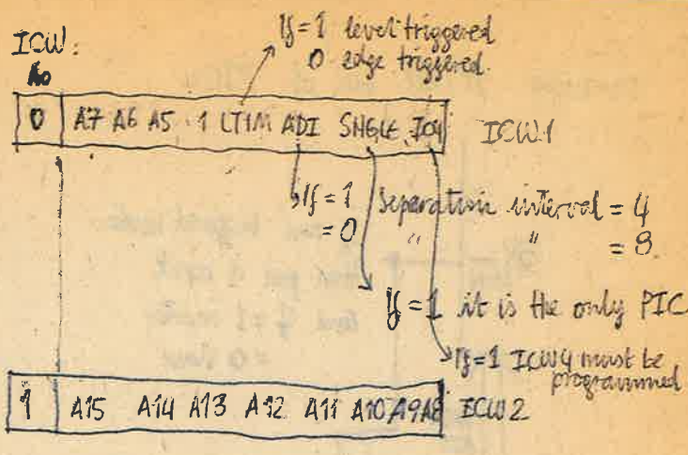


i) One or more IR are raised
ii) 8259 sends INT to CPU
iii) CPU sends $\overline{INTA}$
iv) 8259 sends ϕCDH to DB
↑ op-code of CALL instruction
v) CPU sends two more $\overline{INTA}$
vi) 8259 sends two more address bytes (Subroutine address)

Subroutine address for each level are separated by 4 or 8 bytes.

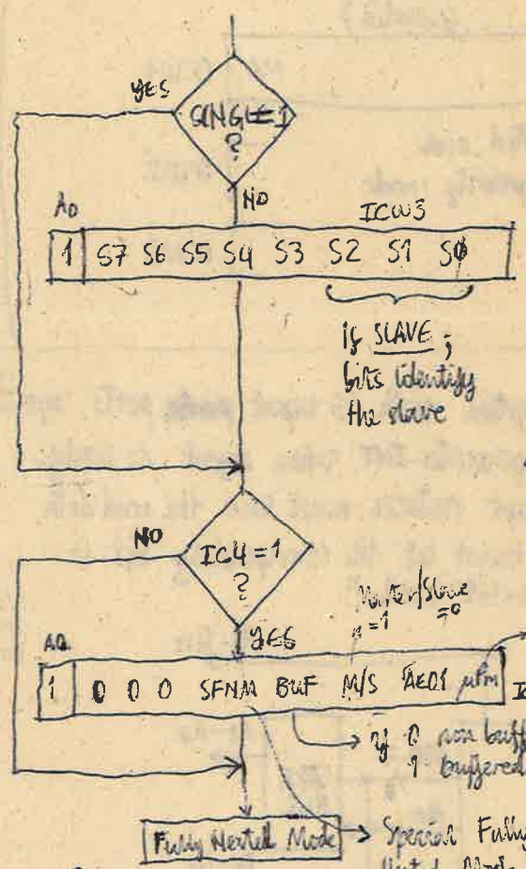
Interrupt controller must be programmed by ICW's (initialization command words)
OCW's (Operation Command Words)

ICW:



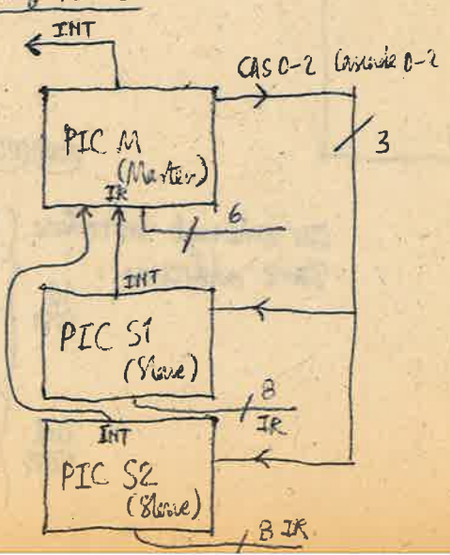
If interval = 4

A7 A6 A5 1 1 0 0 0 IR7
A7 A6 A5 1 1 0 0 0 IR6
⋮
A7 A6 A5 0 0 0 0 0 IR0

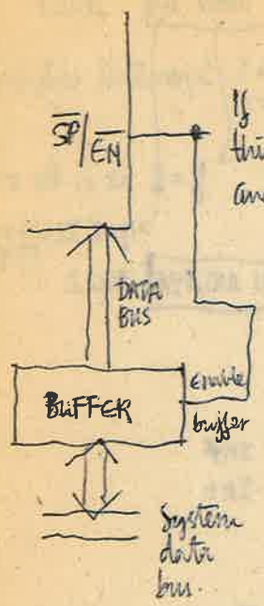


If MASTER; IR
Denotes which pins of master are connected to slaves.
Sx = 1 denotes a connection to slave.
The corresponding slave will be released through cascade lines to put address bytes of CALL instruction.
= 8080/85 mode
= 1 8086 mode

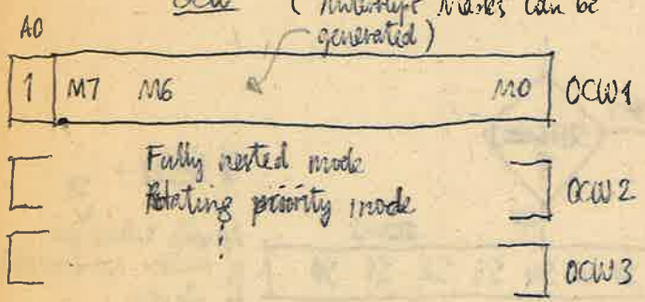
Cascading PIC's:



remember $\overline{SP}/\overline{EN}$ pin of PIC:



OCW (Interrupt Masks can be generated)



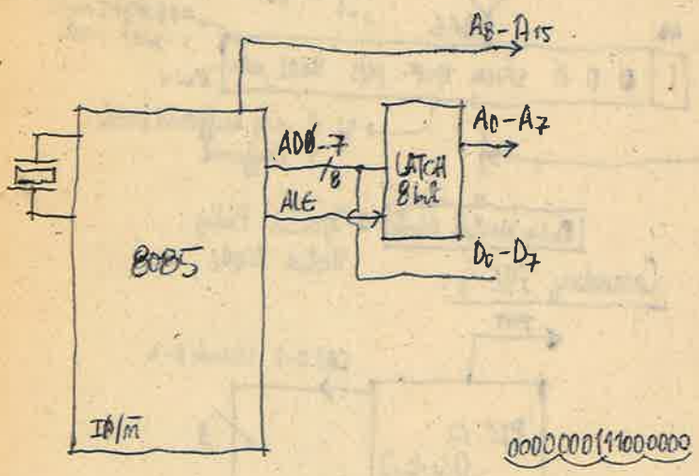
Initialization of 8259's

```
MVI  A, 11ϕ1111ϕ1B  ; ICW1   level triggered; interval = 4; not single
OUT  ϕ              ; Port ϕ  ICW4 will come,
MVI  A, ϕϕϕϕϕϕϕ1B   ; ICW2
OUT  1              ; Send to Port 1
MVI  A, 1ϕϕϕϕϕϕϕB   ; ICW3   S7 is connected to slave.
OUT  1
MVI  A, 00010010B   ; ICW4
OUT  1
MVI  A, 11111ϕ1B    ; ICW1
OUT  2
MVI  A, ϕϕϕϕϕϕϕ1B  ; ICW2
OUT  3
MVI  A, ϕϕϕϕϕ111B  ; Cascade line address
OUT  3
MVI  A, ϕϕϕ10ϕ1ϕB  ; ICW4
OUT  3
MVI  A, ϕϕϕϕϕϕϕϕB
OUT  1              ; OCW1          } interrupts
OUT  3              ; OCW1 for slave } non-masked
EI
```

```
READ1: PUSH PSW
       PUSH H
       IN   PORT1
       ⋮
       POP  H
       POP  PSW
       EI
       RET
```

```
ORG 1CϕH
JMP READ1
ORG 1CϿH
JMP READ2
```

EXAMPLE " A system with 13 input ports. IO/M Each generates INT when input is ready: Interrupt routines must save the read data and must set the corresponding bit in a 16-bit word."



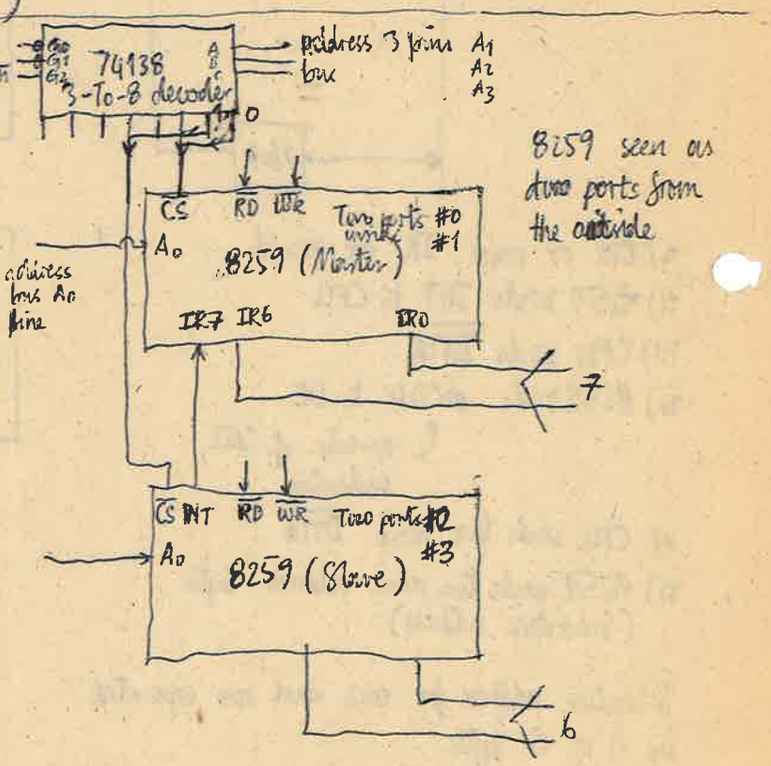
0000001111000000

Our interrupt subroutine start addresses:

1st 8259: 01Cϕ, 01C4, 01C8, ..., 1DC

13

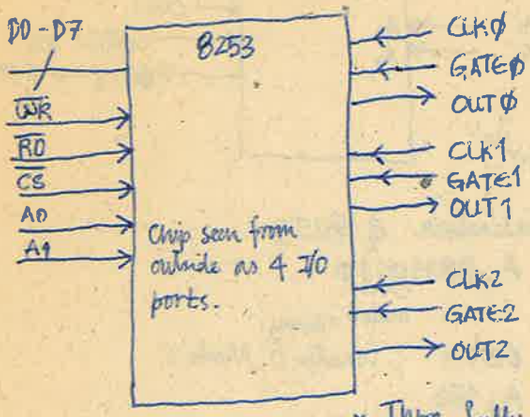
2nd 8259: 1Eϕ, ...



314 25/11/83

Clock Software Programmable Clock:

8253 Programmable Interval Timer:



A_1	A_0	
0	0	Counter 0
0	1	Counter 1
1	0	Counter 2
1	1	Mode Word

* Three fully independent 16-bit presettable BCD or binary DOWN Counters.
* Any counter can be read at any time
* Control words must be sent to each counter of 8253

Control Word

SC1	SC0	RL1	RL0	M2	M1	M0	BCD

SC1	SC0	
0	0	Select Counter 0
0	1	" " 1
1	0	" " 2
1	1	Illegal

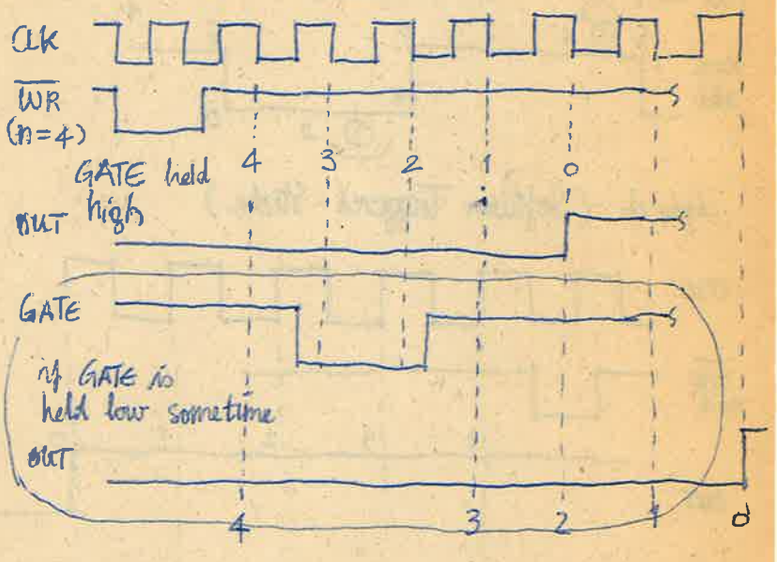
M2	M1	M0	
0	0	0	Mode 0
0	0	1	Mode 1
X	1	0	Mode 2
X	1	1	Mode 3
1	0	0	Mode 4
1	0	1	Mode 5

RL1	RL0	
0	0	Latch counter (To read the counter contents without disturbing the counting)
0	1	Read/load most sig. byte
1	0	" " least " "
1	1	" " least first, then most

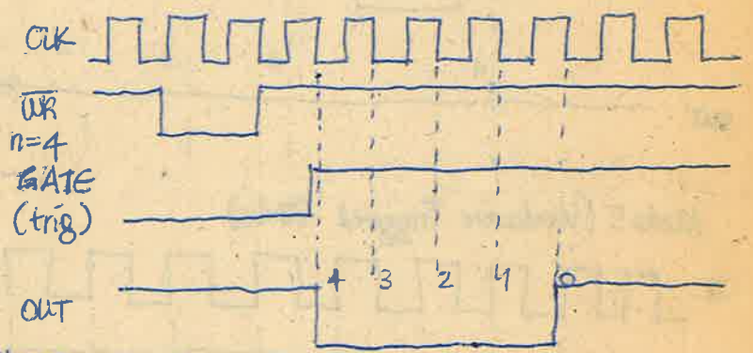
Typical initialization sequence of Counter:

Six different modes of operation:

Mode 0 (Interrupt on Terminal Count)

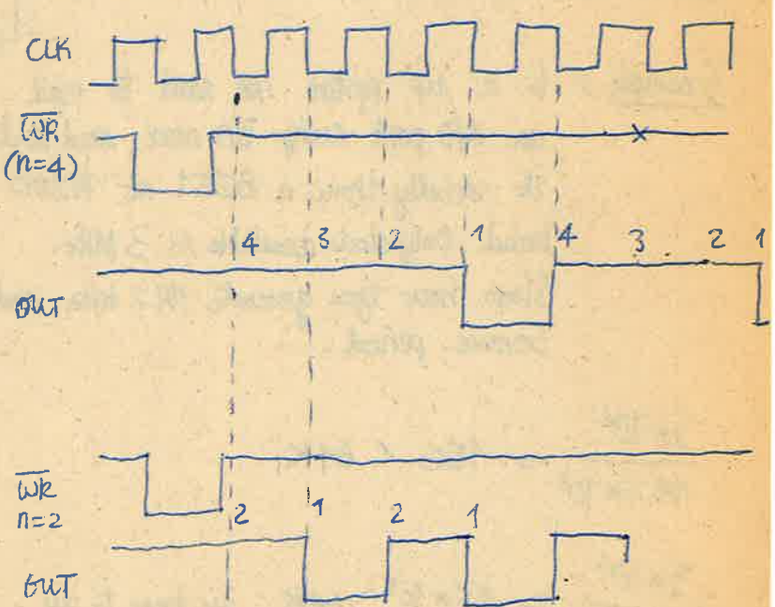


MODE 1 (Programmable One-Shot)

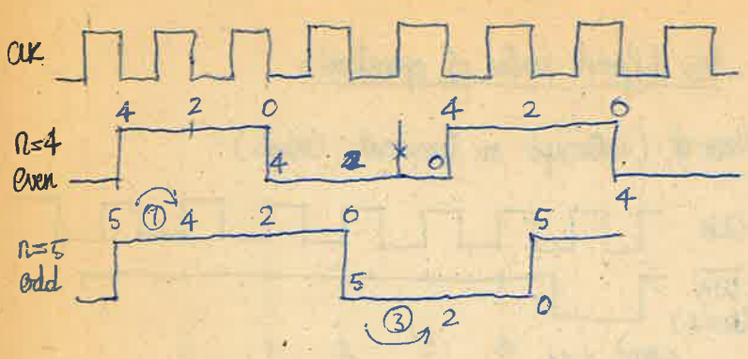


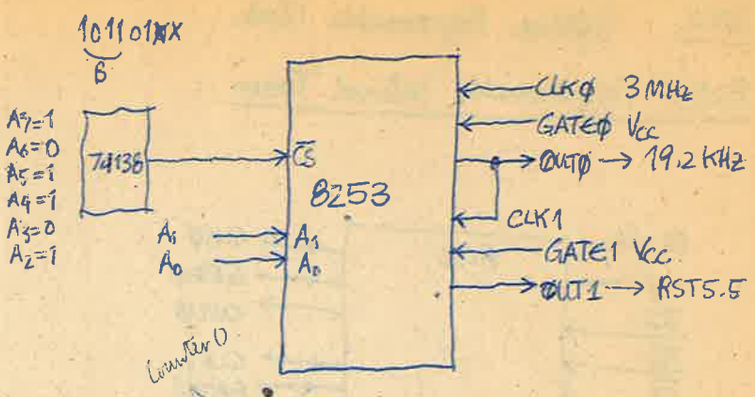
"Retriggerable"

MODE 2 (Rate Generator)

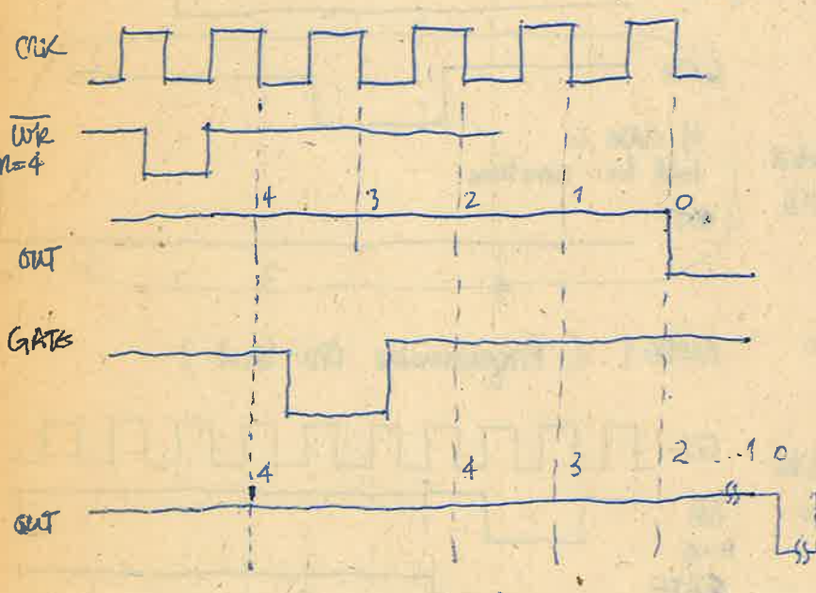


Mode 3 (Square Wave Rate Generator)





Mode 4 (Software Triggered Strobe)

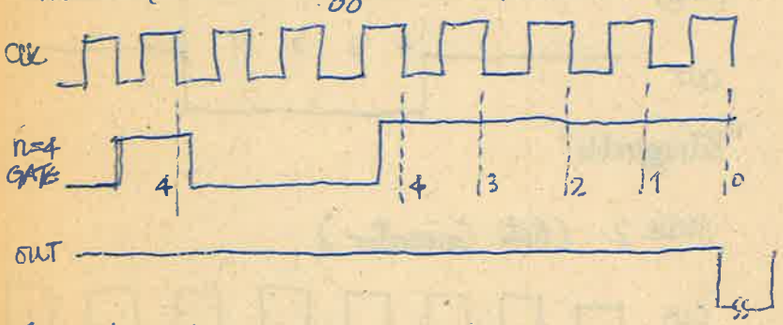


* Initialization of 8253

```
MVI   A, 00110110     ; mode 3 binary.
OUT   0B7H  ; Counter 0 Mode 3
MVI   A, 156
OUT   0B4H
MVI   A, 0
OUT   0B4H  ; Counter ready to go.
MVI   A, 01110002 ; Counter 1, mode 0
OUT   0B7H
MVI   A, 194
OUT   0B5H
MVI   A, 3
OUT   0B5H
EI
```

962 = 3 | 194

Mode 5 (Hardware Triggered Strobe)




```
ORG   2CH
MVI   A, 194
OUT   0B5H
MVI   A, 3
OUT   0B5H
EI
RET
```

To reinitialize Counter 1 in Mode 0

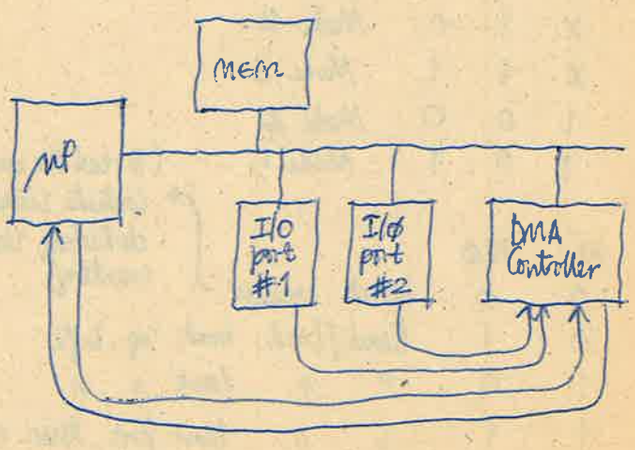
Example: In a µP system we need to read an A/D port every 50 msec. and send it serially thru a 8251 at 19200 baud. Only clock available is 3 MHz. Show how you generate 19.2 kHz and 50msec. period.

$$\frac{3*10^6}{19{,}2*10^3} = 156 < 64K$$

$$\frac{3*10^6}{\frac{1}{0.05}} = 1.5*10^5 > 64K$$ we have to use 2 counters cascade.

$$\frac{1.5*10^5}{156} = 962 \qquad 1.5*10^5 = 156*962$$

DIRECT MEMORY ACCESS

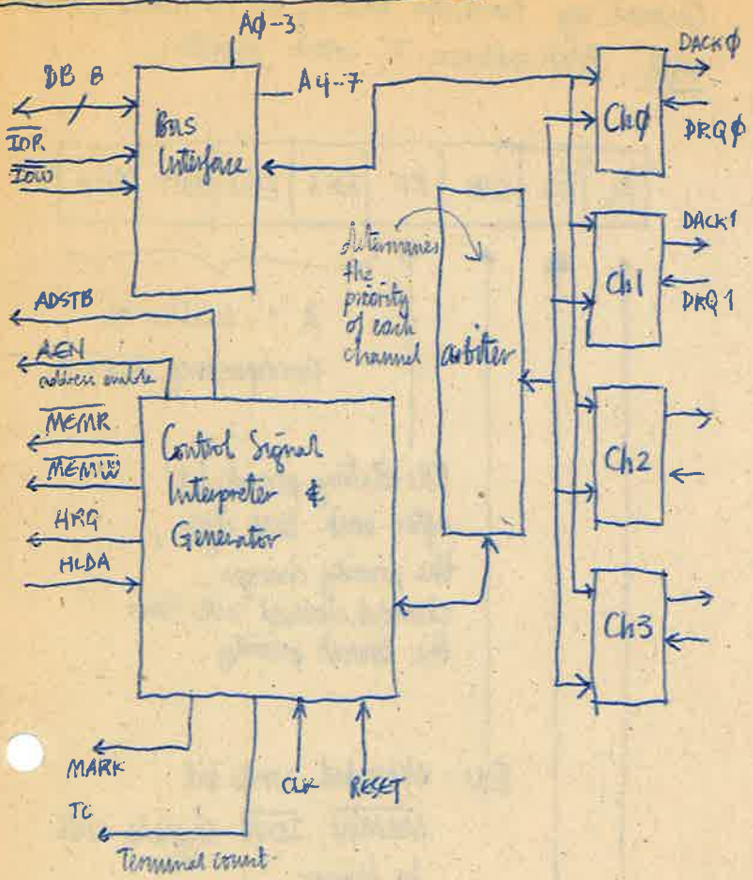


DMA interface

1. μP sends to DMA controller
   a. Beginning address
   b. Block length
   c. Direction of Data flow (Mem → I/O / I/O → Mem)
   d. Port ID.
   e. End of block action (INT or not)
2. μP returns to other activities.
3. DMA controller accesses ~~To memory~~ the bus at idle periods of μP.
   a. forcing an immediate disable
   b. request a HALT and wait for acknowledgement
   c. time the access to an idle cycle of μP.
4. DMA controller must generate the same signals produced by μP.
5. At the completion it causes an INT req. or shows it in its status register.

28/11/83 514

8257 Programmable DMA Controller



Upon receiving a DMA request

i) Takes control of system bus.
ii) DACK is sent
iii) LSB on $A_0 - A_7$, MSB is $D_0 - D_7$ (ADSTB)
iv) Generates appropriate I/O write-read and MEMR/MEMW signals to cause data transfer

3 kinds of operation:

i) DMA read : Memory → output port
ii) DMA write
iii) DMA verify

Each channel has two 16-bit registers

i) Address register address of first mem. loc.
ii) Terminal count reg.

LS → 14 bits specify the # of DMA cycles.

bit 15	bit 14	
0	0	Verify
0	1	DMA write
1	0	DMA read
1	1	Illegal

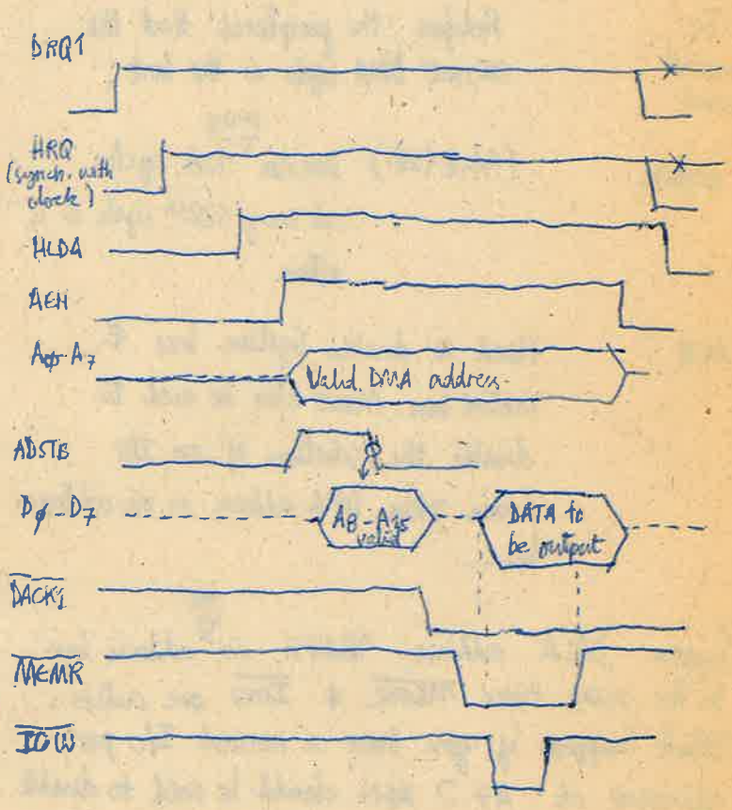
If N is the required number, load N-1 into

In normal mode DRQ0 has the highest priority DRQ3 the lowest.

DACK acts as chip selects for the peripheral requesting service.

When ADSTB has high-low transition, DB has $A_8 - A_{15}$ on it.

During a DMA read, I/O W MEMR are active
" " DMA write I/O R MEMW " "



Single byte transfer DMA cycle

4 clock cycles are used to transfer a byte.

2 MHz clock ⇒ 500 kbytes/sec.

Normal software data transfer:

```
    LXI D, COUNT
    LXI H, BEGADD
BACK: MOV A,M      7
      OUT PORT     11
      INX H        5
      DCX D        5
      MOV A,D      4
      ORA E        4
      JNZ BACK     10
                   ---
                   56 clock cycles
                   for one byte
```

$\overline{IOR}$ & $\overline{IOW}$ in master mode outputs
" slave " inputs

$\phi - A_3$ in master mode outputs
" slave " inputs
to select 9 internal registers

$\overline{CS}$ In master mode disabled
To prevent the chip from selecting itself

TC Terminal Count — Notifies the peripheral that the current DMA cycle is the last.

MARK (Mod 128) Denotes every 128 cycles.
At every 128th cycle it is active

AEN : Used to disable System bus & Control bus. Must also be used to disable the selection of an I/O device when DMA address is on address bus.

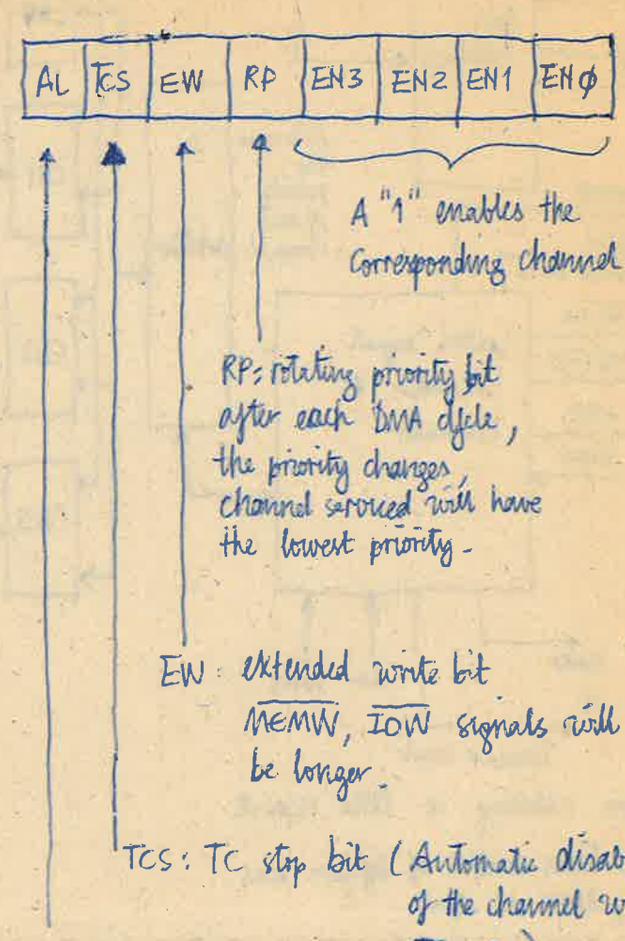
Suppose DMA address 1BA7H is on address bus. In the mean time $\overline{MEMR}$ & $\overline{IOW}$ are active. What happens if you have a normal I/O port addressed at A7 ? AEN should be used to disable it.

Register Address

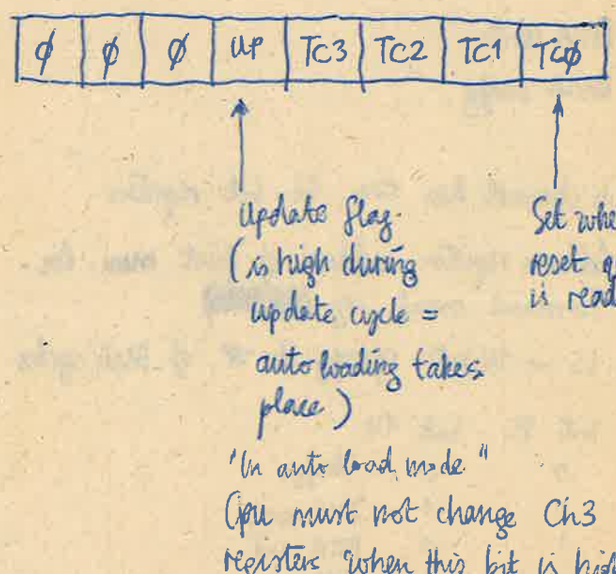
A_3	A_2	A_1	A_0		
0	0	0	0	Ch0	DMA address port
0	0	0	1	Ch0	TC port
0	0	1	0	Ch1	"
0	0	1	1	Ch1	"
0	1	0	0	Ch2	"
0	1	0	1	Ch2	"
0	1	1	0	Ch3	"
0	1	1	1	Ch3	"
1	0	0	0	Mode Set Reg (write) / Status Port (read)	

LSB first
MSB last
8257 has a F/L flip flop. This flip flop resets by hardware reset or by mode set.

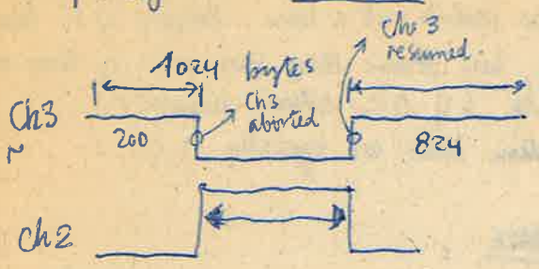
Mode Set Register :
Cleared by hardware RESET; is normally written after DMA address, TC count registers.



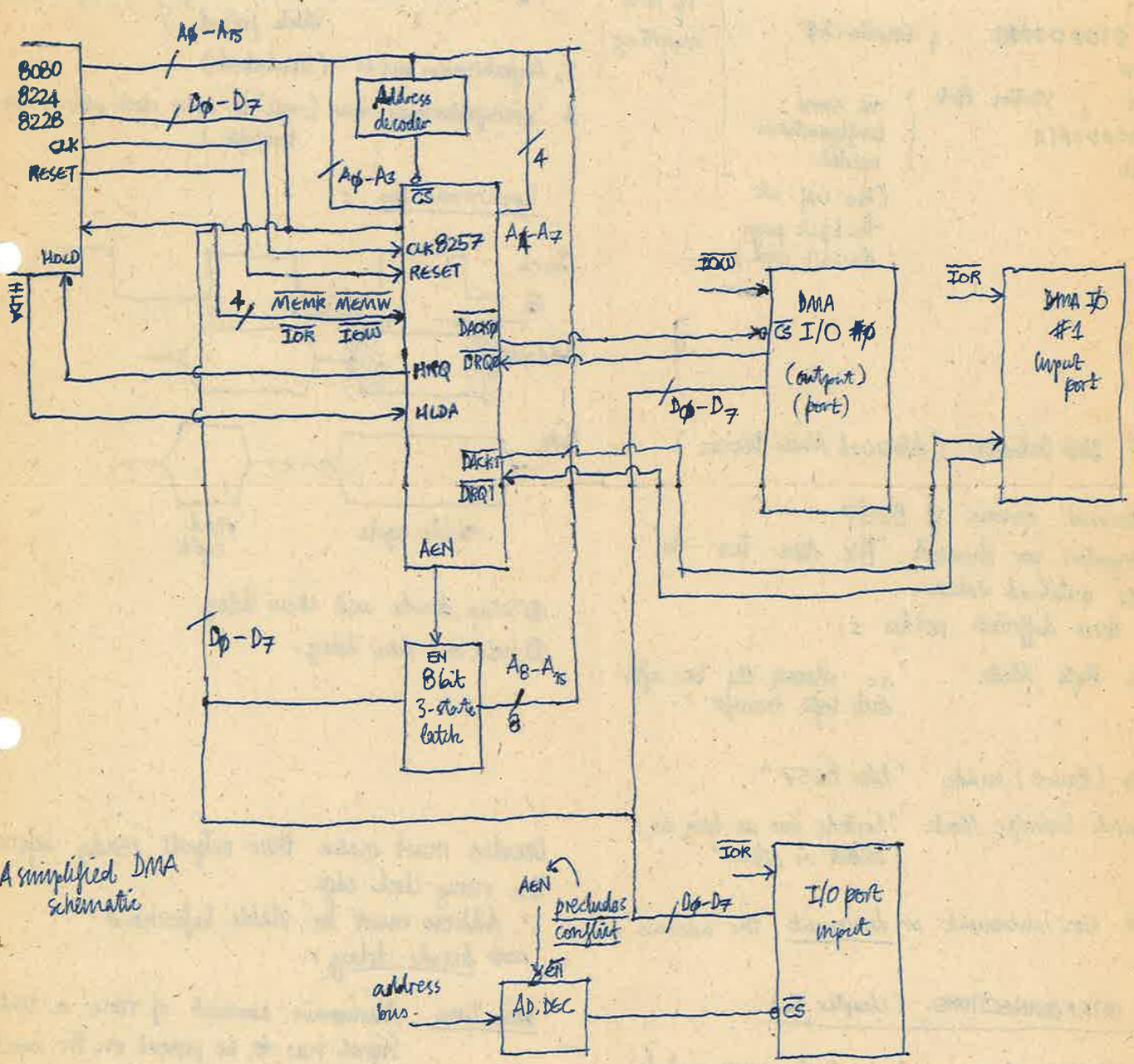
Status Register



If more than one channel requests service highest priority is recognized during the next transfer. Lower priority will be overridden



2/12/83



A simplified DMA schematic

Example: Initialize a 8257 in order to read 256 bytes from the I/O port into locations 2000H - 20FFH. Suppose that 8257 ports have addresses 10H - 1BH.

```
B: DI
   MVI A,00H   ;  LSB of beginning address
   OUT 10H     ;  for Ch 0
   MVI A,20H   ;  MSB of beginning address
   OUT 10H     ;
   MVI A,00H   ;  LSB of 00FFH (Count)
   OUT 11H
   MVI A,1     ;  MSB of 00FFH
   OUT 11H
   EI
   MVI A,01000000B   ;  Enable Ch 0
   OUT 1BH
BACK: IN 1BH   ;  Status Port
   ANI 00000001B
   JZ BACK
   .
   .
   .
   .
   .
   JMP B
```

(256 bytes of DMA inputting; in some configurations needed. (the conf. at the back page doesn't need))

AM 9517 DMA Controller (Advanced Micro Devices)

An enhanced version of 8257
Four registers per channel. The other two to keep the autoload values.
It has three different modes:

i) Single Byte Mode "i.e. releases the bus after each byte transfer"

ii) Block (Burst) mode "like 8257"

iii) Demand transfer Mode "Controls bus as long as DREQ is active"

9517 can increment or decrement the address.

BUS INTERCONNECTIONS (Chapter 3)

Bus: A collection of signal lines that carry module to module communications.
Performance and reliability suffer as bus length increases

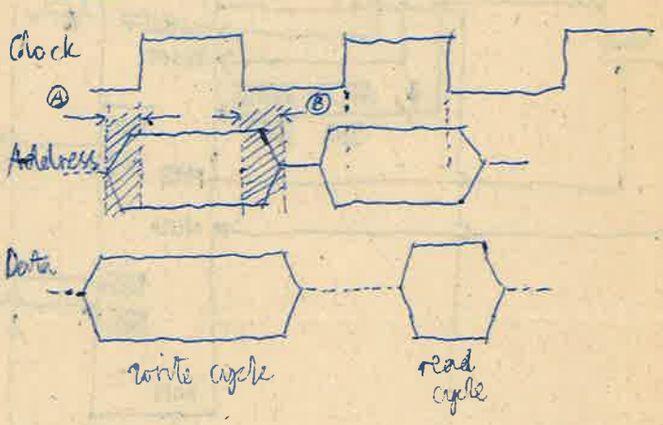
In general three groups:

1. Basic information
   a) Address (memory & I/O)
   b) Data
   c) Commands (Read, Write)
2. Data transfer control (data handshake) lines
3. Arbitration lines (Arbitrate which module gains control of the bus) If two modules attempt to transmit simultaneously a bus conflict may occur. Only one module at a time. Because of a bus conflict, bus drivers may burn out. If they are tri-state. (If O.C. nothing happens)
   Arbitration lines are typically O.C.

Bus HANDSHAKE:

1. Synchronous buses (clocked transfer, one transfer per clock period.)
2. Asynchronous buses (unclocked)
3. Semisynchronous bus (one or more clock periods per transfer)

Synchronous bus:



A Setup, decode and skew delay
B Hold and skew delay.

Decoders must make their outputs ready before the rising clock edge.
∴ Address must be stable beforehand.
⇒ decode delay.

Setup time: Minimum amount of time a control signal has to be present on the input before the clock triggers the transfer.

Hold time: The minimum time data has to be held stable on the inputs after a clock change triggers the transfer.

Skew: is due to varying logic delays in a system, due to varying gate thresholds, or different rise and fall times
To compensate for skew addresses must be stable a maximum skew time earlier.

Example : Initialize a 8257 in order to read 256 bytes from the I/O port into locations 2000H - 20FFH. Suppose that 8257 ports have addresses 10H - 1BH.

```
B : DI
    MVI A, 00H   ;  LSB of beginning address
    OUT 10H      ;  for Ch 0
    MVI A, 20H   ;  MSB of beginning address
    OUT 10H      ;
    MVI A, 00H   ;  LSB of 100H (Count)
    OUT 11H
    MVI A, 1     ;  MSB of 100H
    OUT 11H
    EI
    MVI A, 01000008B   ; Enable Ch 0
    OUT 18H
BACK: IN 18H     ;  Status Port
    ANI 00000001B
    JZ BACK
    .
    .
    .
    .
    .
    JMP B
```

256 bytes of DMA inputting

in some configurations needed. (the conf. at the back page doesn't need)

AM 9517 DMA Controller (Advanced Micro Devices)

An enhanced version of 8257
Four registers per channel. The other two to keep the autoload values.
It has three different modes :

i) Single Byte Mode "i.e. releases the bus after each byte transfer"

ii) Block (Burst) mode "like 8257"

iii) Demand transfer Mode "Controls bus as long as DREQ is active"

9517 can increment or decrement the address.

BUS INTERCONNECTIONS (Chapter 3)

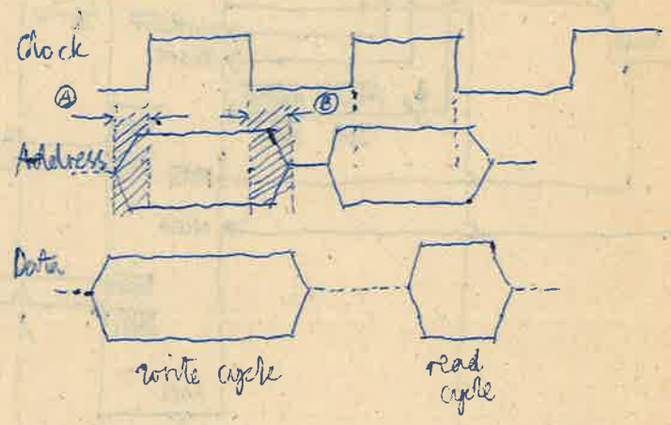
Bus : A collection of signal lines that carry module to module communications.
Performance and reliability suffer as bus length increases
In general three groups :

1. Basic information
   a) Address (memory & I/O)
   b) Data
   c) Commands (Read; Write)
2. Data transfer control (data handshake) lines
3. Arbitration lines (Arbitrate which module gains control of the bus) If two modules attempt to transmit simultaneously a bus conflict may occur. Only one module at a time. Because of a bus conflict, bus drivers may burn out. If they are tri-state. (if O.C. nothing happens)
   Arbitration lines are typically O.C.

BUS HANDSHAKE :

1. Synchronous buses (clocked transfer, one transfer per clock period.)
2. Asynchronous buses (unclocked)
3. Semisynchronous bus (one or more clock periods per transfer)

Synchronous bus :



A Setup, decode and skew delay
B Hold and skew delay.

Decoders must make their outputs ready before the rising clock edge
∴ Address must be stable beforehand.
⇒ decode delay.

Setup time : Minimum amount of time a control signal has to be present on the input before the clock triggers the transfer.

Hold time : The minimum time data has to be held stable on the inputs after a clock change triggers the transfer.

Skew : is due to the varying logic delays in a system, due to varying gate thresholds, or different rise and fall times
To compensate for skew addresses must be stable a maximum skew time earlier.

Clock period is determined by the slowest device in the system. If another device is faster, no advantage can be gained.

Advantage is that synchronous bus is a simple system, requires minimum hardware.

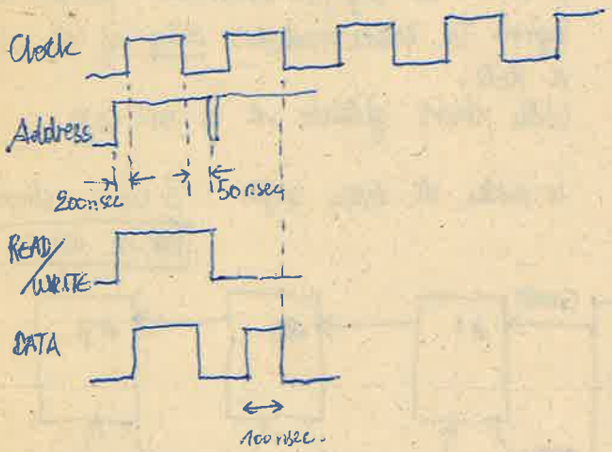
Minimum Bus Cycle time =

$$T_{setup} + T_{decode} + 2T_{skew} + \max(T_{Hold}, T_{round\ trip\ propagation})$$

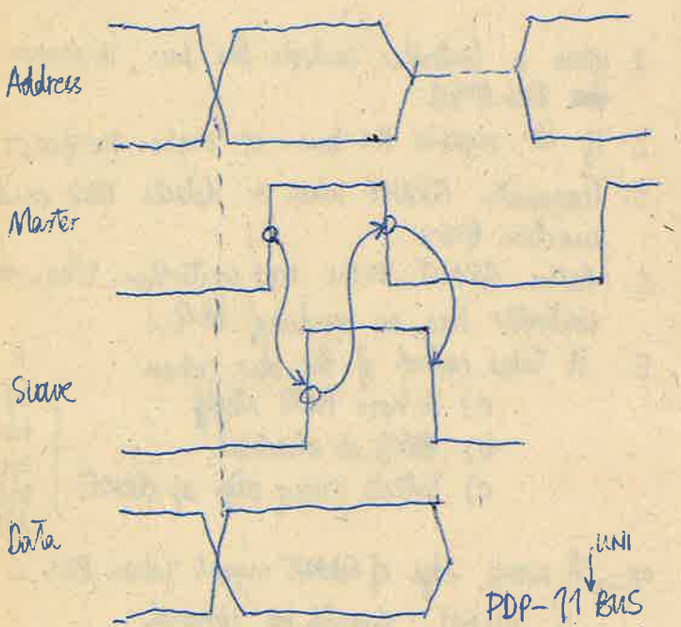
5/12/83

Synchronous Bus :

Ex. MC6800 Bus.



Asynchronous bus :



UNI

PDP-11 BUS
Fully locked
Asynchronous
Bus.

Write

i) Master Places address & data
ii) After a delay ($T_{skew} + T_{decode} + T_{setup}$)
MASTER is raised (get it)
iii) When slave finishes reading
SLAVE is raised (I've got it)
iv) MASTER goes low (I see you've got it)
v) SLAVE " " (I see you see I've got it)

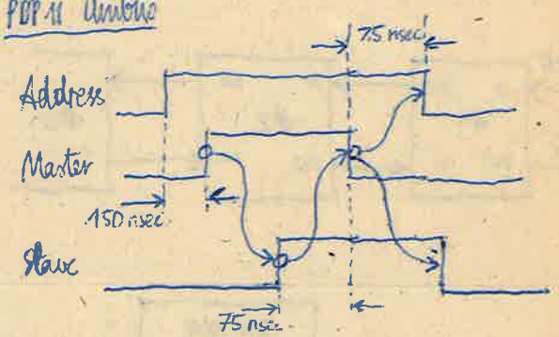
Read

i) Master Places address.
ii) After delay MASTER ↑ (send it)
iii) After data is ready SLAVE ↑ (read it)
iv) After reading data MASTER ↓ (I've got it)
v) SLAVE ↓ (I see you got it)

1. Very reliable
2. Can deal with slow devices
3. But slower than synchronous bus.

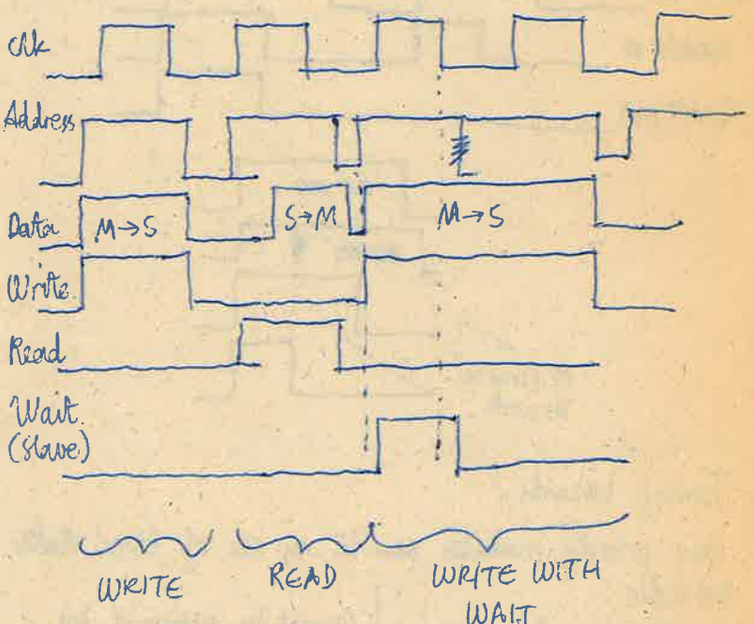
Minimum cycle time = $T_{setup} + T_{decode} + T_{hold} + 2T_{skew} + 2T_{round\ trip\ prop.\ delay}$

Example.
PDP 11 Unibus



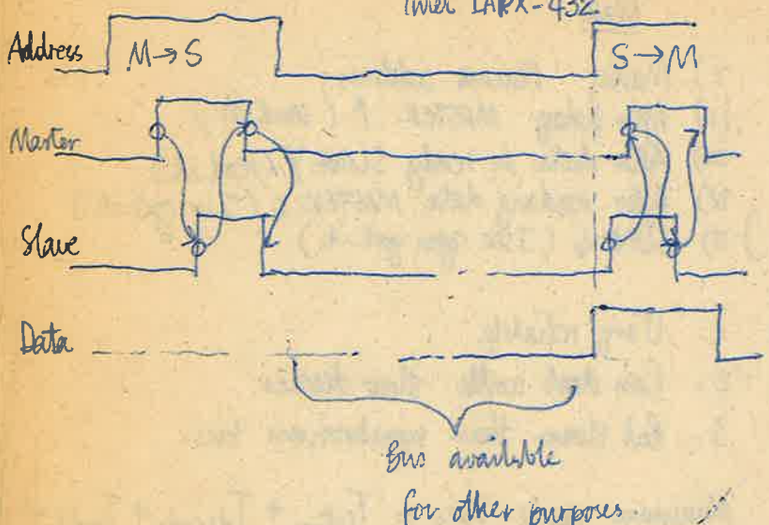
Semisynchronous Bus (Hybrid bus)

8080
8085
Z80



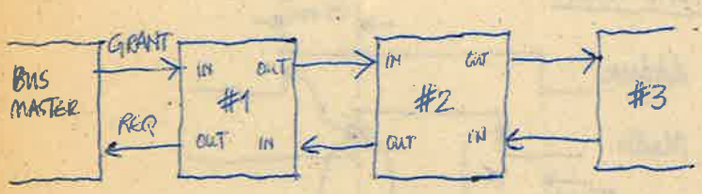
Split Cycle Bus

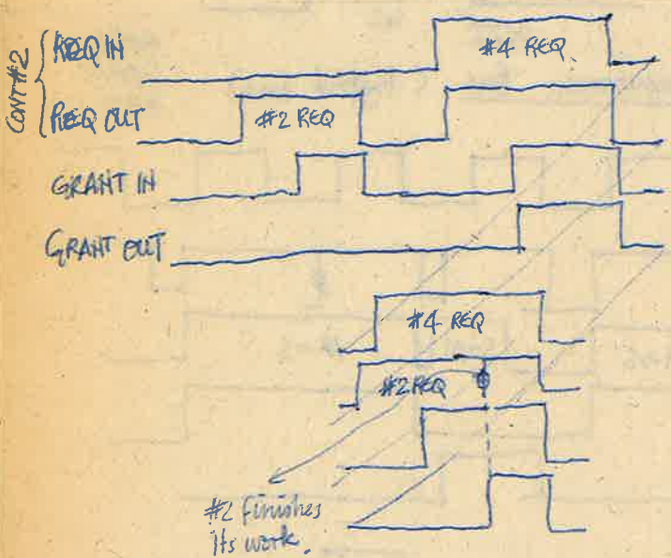
"Suitable for multiprocessor environment"
asynchronous bus. VAX-780 system
Intel iAPX-432



ARBITRATION

Daisy Chain





Timing hazards

Low priority modules can be in one of three states
1* idle
2* has a pending REQ
3* Controlling the bus.
} cannot be informed by single bit;
So two bits are needed.
REQ & GRANT are used
no timing hazard.

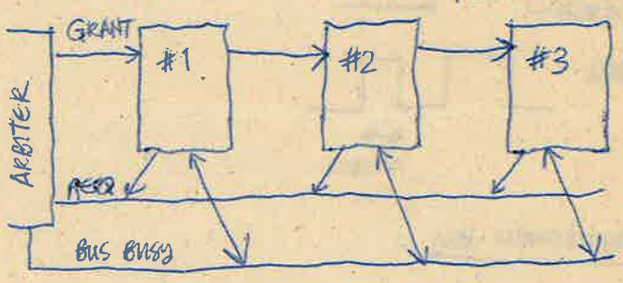
1. If GRANT is low, REQ is high ⇒ There is pending REQUEST but before of the low priority modules has control.
2. GRANT & REQ high ⇒ a low priority module is controlling the bus.

- After REQ is raised a low-to-high transition on GRANT is to be expected before taking control.

Suppose a module passes GRANT onto the next module whether or not there is a request assume #1 & #3 make REQ just after #2 finishes, #1 & #3 may try to control the bus at the same time.

To make it safe, a controller passes GRANT signal to lower modules only if it sees a REQ.
With short glitches it is not safe.

To make it even safer: 3 wire method
PDP-11 UNIBUS.

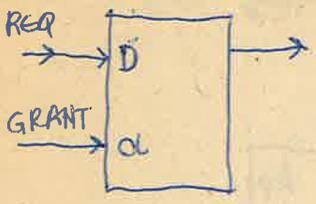


1. When a controller controls the bus, it makes ~~true~~ BUS BUSY.
2. If it requires the bus it makes REQUEST
3. Transmits GRANT when it detects REQ and inactive BUSY
4. Passes GRANT to the next controller when the controller has no pending REQ.
5. It takes control of the bus when
   a) It has REQ itself
   b) BUSY is inactive
   c) Detects rising edge of GRANT
   } Safer than two wire system with glitches.

ex. → A rising edge of GRANT signal when BUS is busy is ILLEGAL; should be ignored.

PROBLEMS WITH ASYNCHRONOUS SYSTEMS

In a flip flop setup time must be satisfied



If setup time ~~is~~ is violated

① Flip-flop may enter a metastable state for (neither 1 nor 0), an undetermined time, then relaxes to 0 or 1

② Q and $\bar{Q}$ may oscillate in phase for an undetermined time, then relaxes to complementary values.

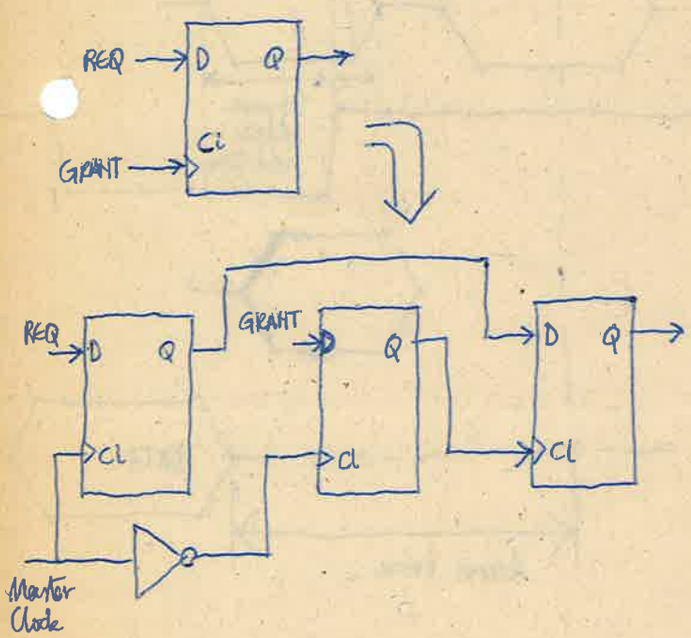
③ Transition time ~~Time~~ Spec may be violated. A worst case

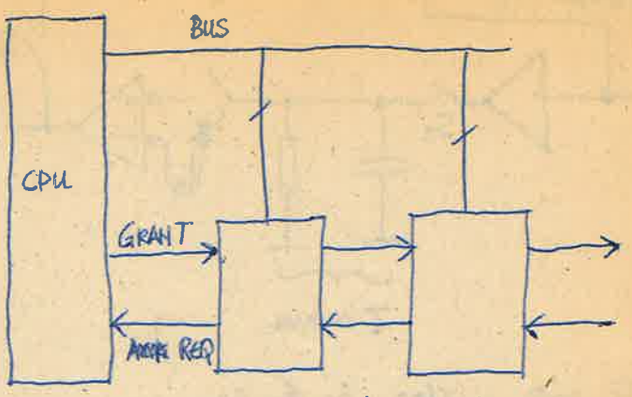
④ A very short pulse may be transmitted

9/12/83

Solutions to problems of asynchronous systems:

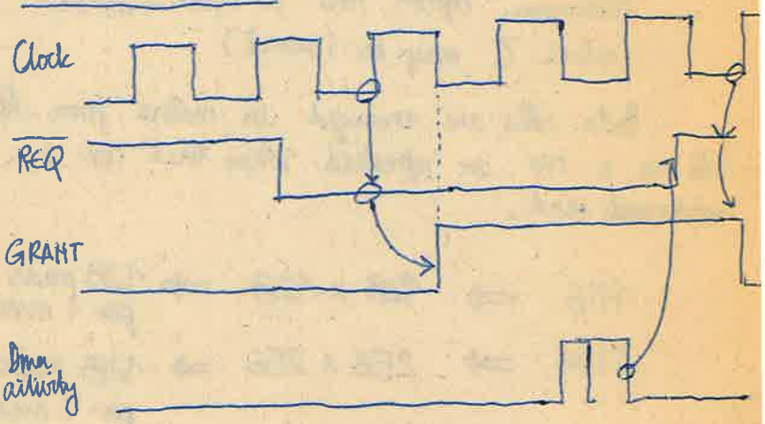
Use a single master clock:





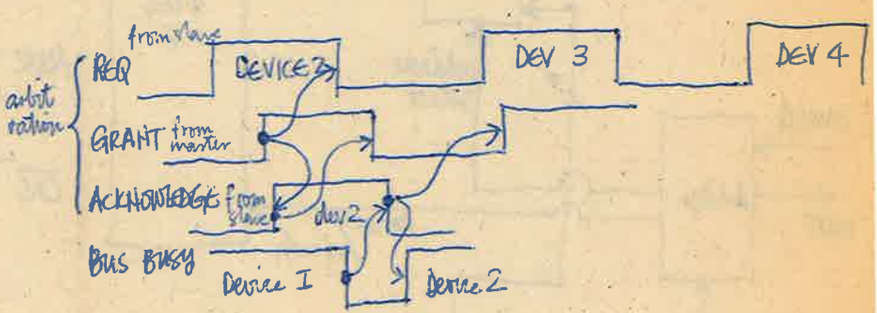
Interrupts thru daisy chain

6800 arbitration



DEC PDP-11 Unibus

Data transfer for first transaction overlaps the arbitration for second.

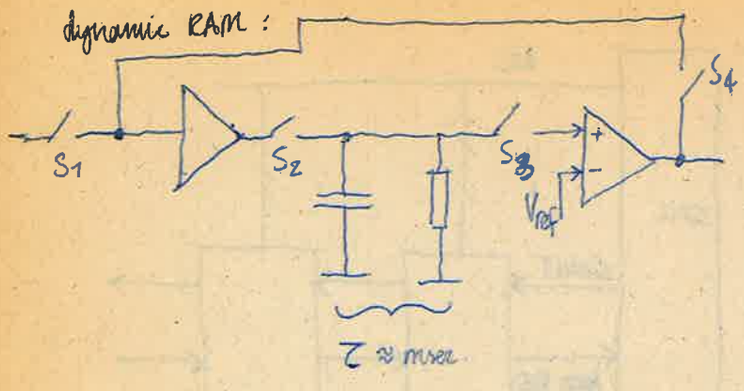


DYNAMIC RAM INTERFACING

Dynamic Ram: 4164 (64K bits)
4116 (16K bits)
↑ outdated

if you need <16K use static
>16K use dynamic

Dynamic RAM :



To write : Close S_1 & S_2.

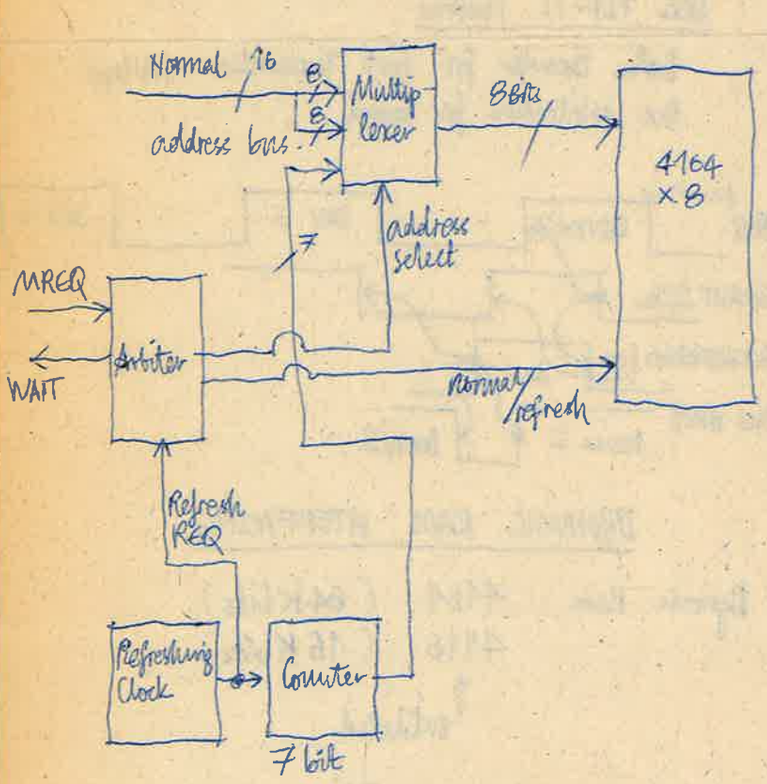
To read & refresh : Close S_3, S_4 & S_2

Minimum refresh rate : 1msec - 2msec.
(actual τ may be 1 second)

Data cells are arranged in matrix form. All bits in a row are refreshed when that row is ~~addressed~~ read.

4116 ⇒ 128 x 128 ⇒ 128 reads per 1 msec.

4164 ⇒ 256 x 256 ⇒ 128 reads per 1 msec.



Different ways of refreshing :

i) Special chips Intel 3242, MC3480

ii) Use no arbiter, refresh during idle periods 68XX or 65XX use first half of the clock.

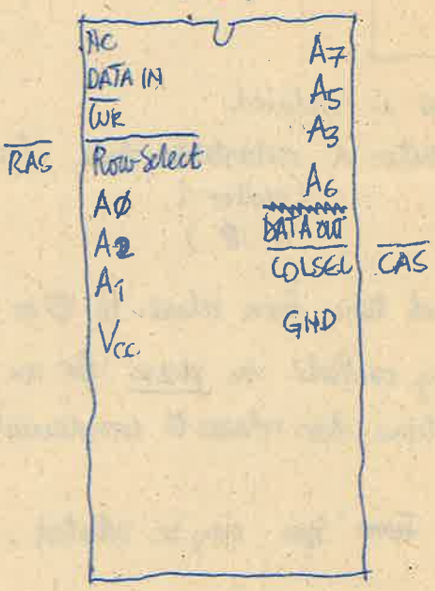
iii) Software : A clock interrupt every 2 msecs µP reads all rows (128). Whole thing 128 msec.

Can be used if µP is not busy.

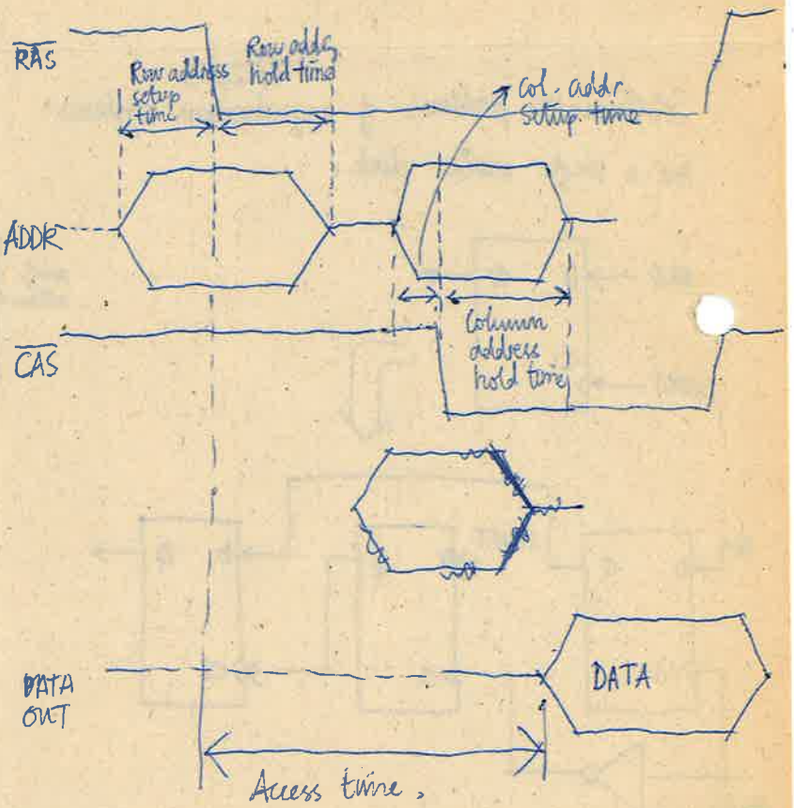
iv) Z80 Produces refresh address & control during the inactive times of µP.

v) Dynamic RAM chip itself may do the refreshing.

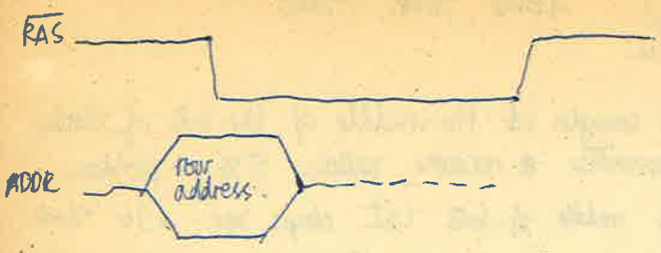
4164

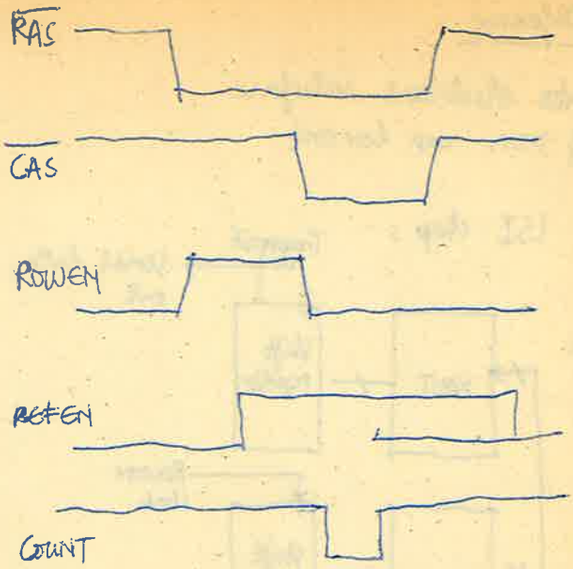


READ CYCLE

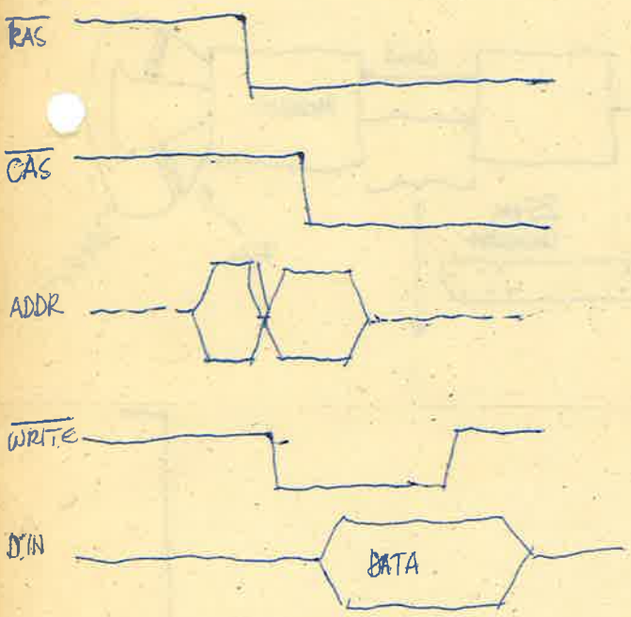


REFRESH CYCLE

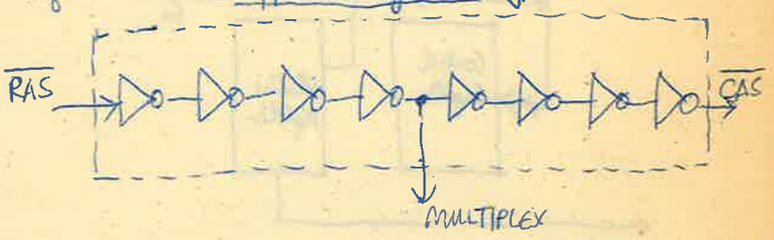


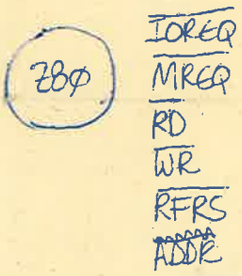


WRITE CYCLE



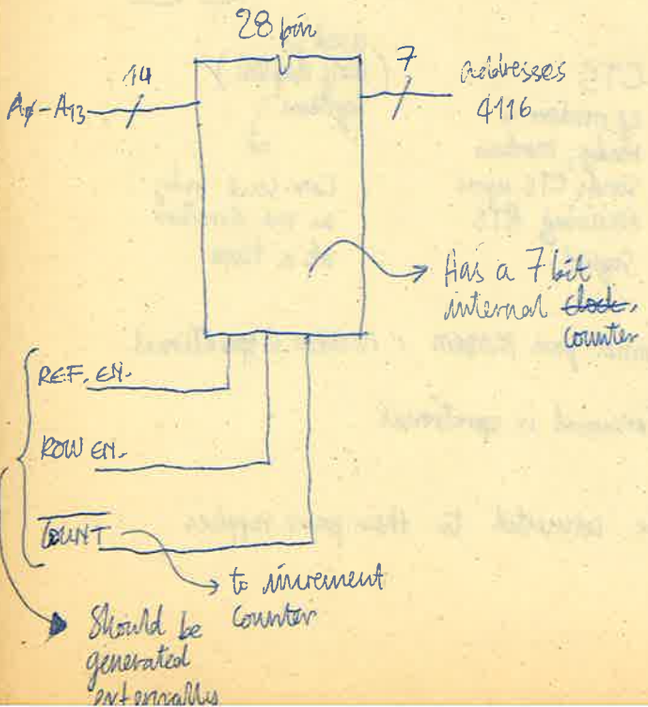
To generate $\overline{CAS}$ and Address multiplexing signal use a tapped delay line

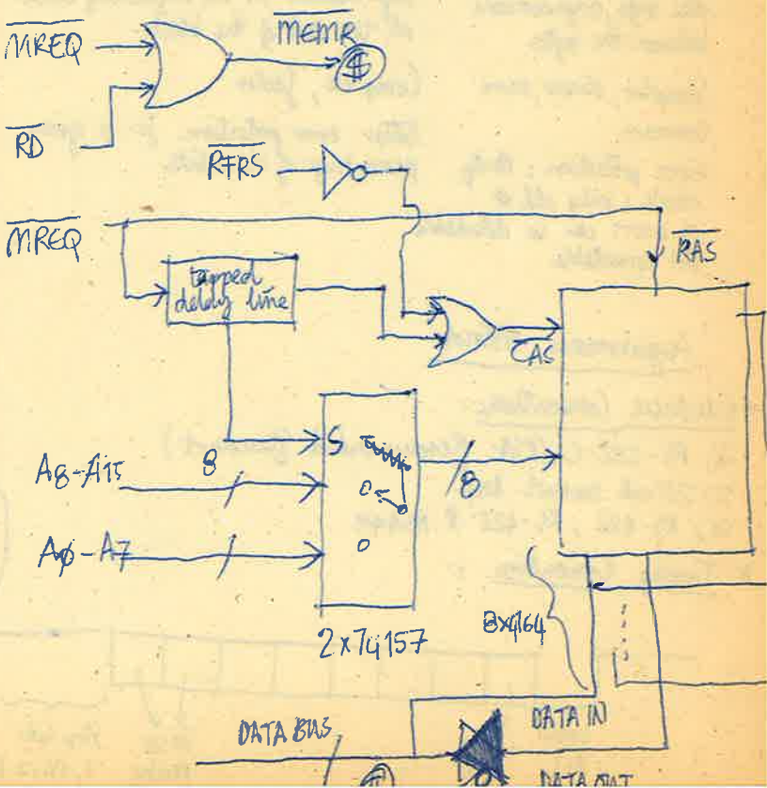




ADDR contains 7 bit counter value when $\overline{RFRS}$ is low (A_0 – A_6)

Intel 3242





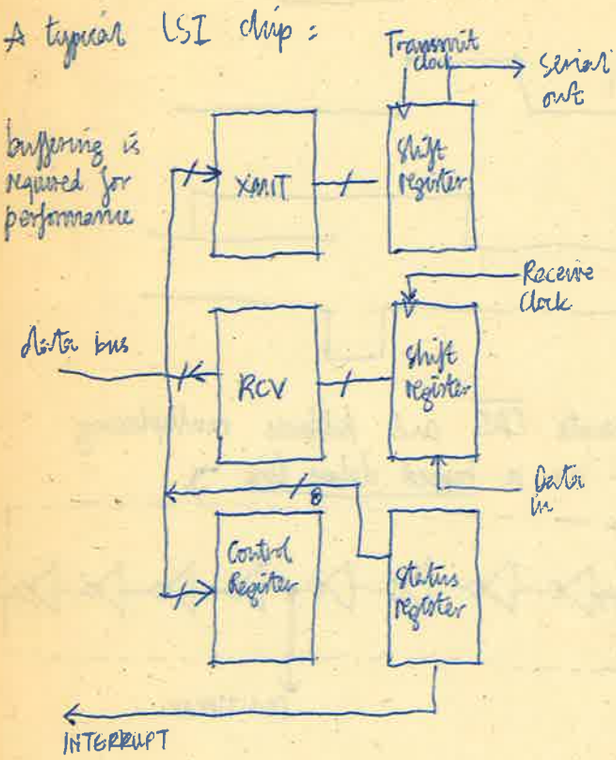
12/12/83

Serial Interfacing

Bit times : 110, 150, 300, 600, 1200, 2400
bits/sec. 4800, 9600, 19200

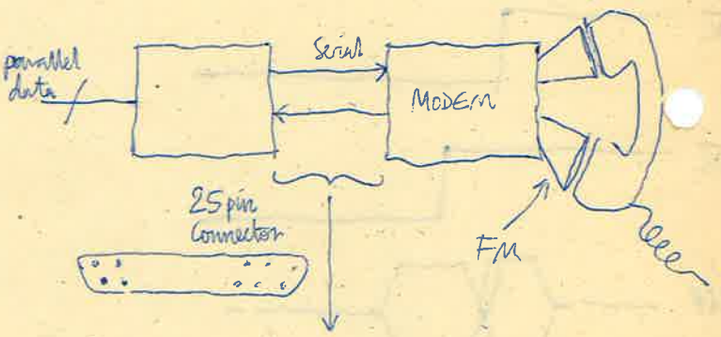
Least complex electrical interface.
Few # of wires ⇒ low cost

A typical LSI chip :



Receiver samples at the middle of the bits if clocks of transmitter & receiver within 5% no problem.
To find middle of bits LSI chips use x16 clock.
i) When high-to-low transition occurs, counter is cleared
ii) Increments for each tick of x16 clock.
iii) When it reaches 8, middle of start bit -
if low ⇒ counter cleared
iv) Each time counter reaches 16 bit is sampled and counter is cleared.
v) If stop bits are high, character is loaded into buffer.

RS-232-C Interface



Serial I/O

Asynchronous

Successive data at arbitrary times

Synchronous within the byte asynchronous between the bytes

Simpler, slower, more common

Error protection : Parity check ; only odd # of errors can be detectable. NOT correctable

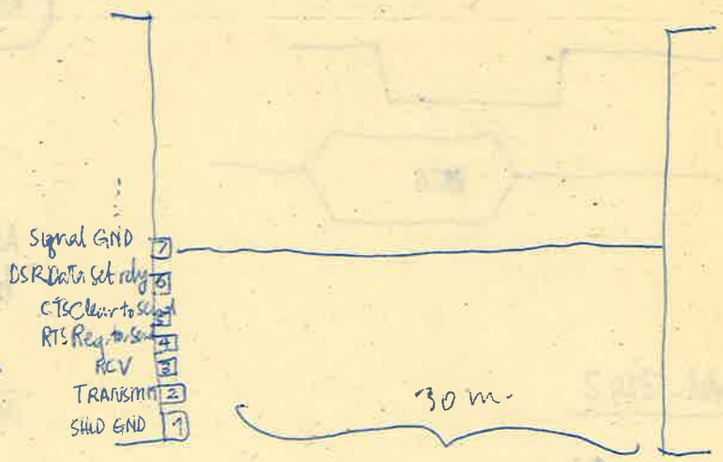
Synchronous

Specific time intervals determined by master clock.

Groups of bytes with control information at the beginning and at the end of the block.

Complex, faster

Better error protection for a given percentage of checkbits.



RTS ⇒ CTS
if modem is ready, modem sends CTS upon receiving RTS signal.

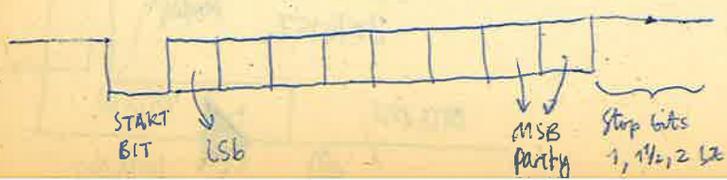
(used in Half duplex) systems
Can send only in one direction at a time

Asynchronous Protocols

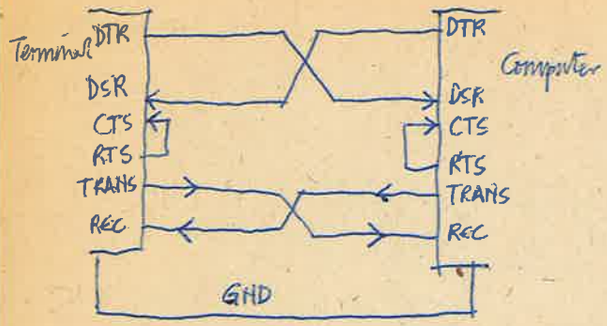
* Electrical Conventions :
i) RS-232-C (RS: Recommended Standard)
ii) 20 mA current loop
iii) RS-422, RS-423 & RS449

DSR : comes from MODEM : MODEM is operational
DTR : : Terminal is operational
→ may be connected to their power supplies

* Timing Convention :



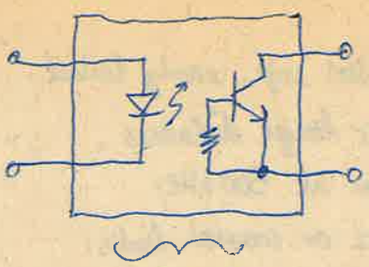
To connect terminals and computers without a MODEM :



logic 1 < −3 V.

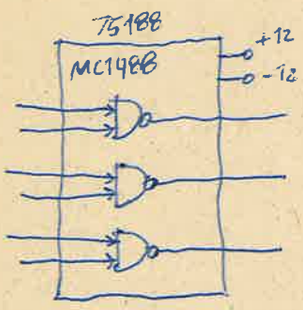
logic 0 > 3 V.

OPTOCOUPLER :



2kV - 3kV isolation can be achieved

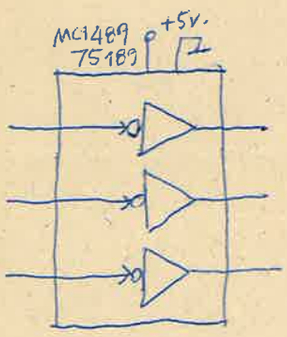
TTL



RS-232

±12 for noise immunity

RS-232

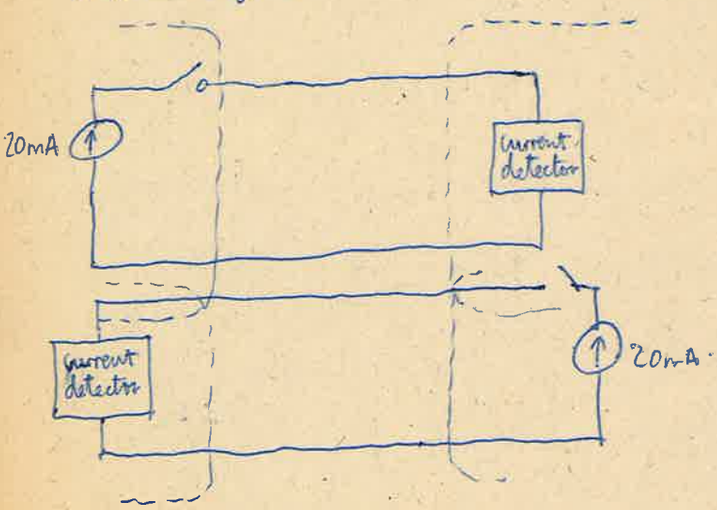


TTL

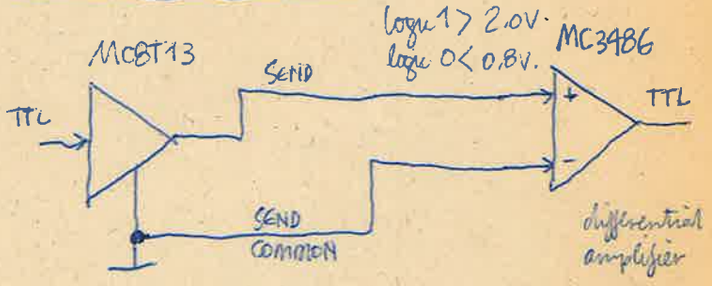
20 mA current loop

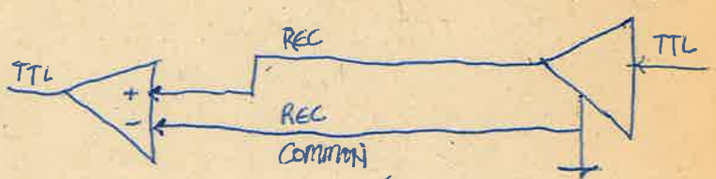
20mA → logic 1

0 mA → logic 0

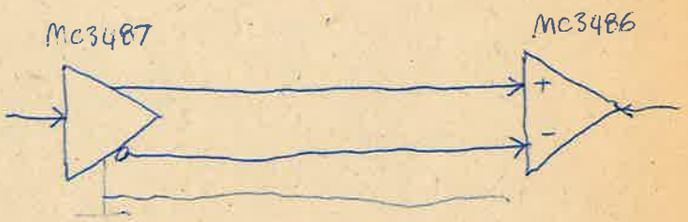


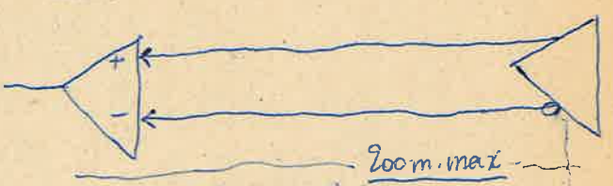
RS422 & RS 423





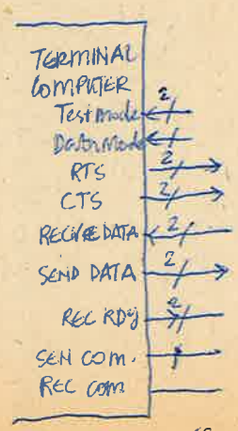
RS423 60 m. max





RS422

RS-449 Connector standart



49 pin connection

SYNCHRONOUS INTERFACES

i) No need for start & stop bits ⇒ increased data rate

ii) Block oriented ⇒ control info. easily linked.

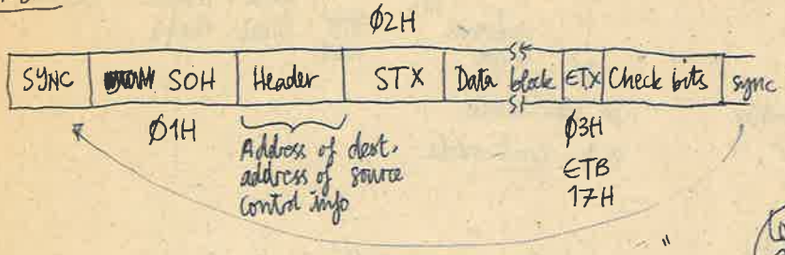
iii) Higher data rate over longer distances
Sync. link can run at 500 kHz.
Must use RS-422 or coaxial links.

16/12/83

Synchronous interfaces :

i) BISYNC (Binary Sync. Comm.) used by IBM
ii) DDCMP (Dig. Data. Comm. Mess. Prot.) by DEC
iii) HDLC (High level Data Link Cont) used in new designs

BISYNC



SYNC. characters can be inserted anywhere.

To differentiate control characters from data : DLE (10H)
If a control symbol is to be treated as data, send a DLE before it.

" data link escape "

SYNC, SOH, DESADD, SOURCADD, CONCAR, STX, 2DH, 3F, 10H, 03H, 10H, 02H, 10H, 10H, 03H, CHKBYT, SYNC, SYNC ...
(03H = ETX)

Select a SYNC pattern which cannot be obtained by rotating itself

Problem : Automatic insertion of SYNC characters between DLE and the following character if it is supplied late.
Solution = Don't supply the DLE char to interface until you're sure that the following char. is ready.

HDLC Protocol

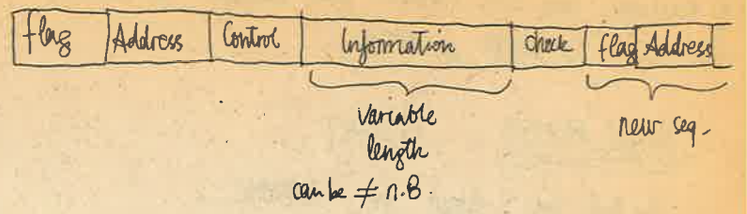
A unique pattern is used for sync. (Can never occur in bit stream)

FLAG PATTERN
01111110 six successive 1's no data sequence can contain this.

When a data contains five successive 1's, a false 0 is inserted

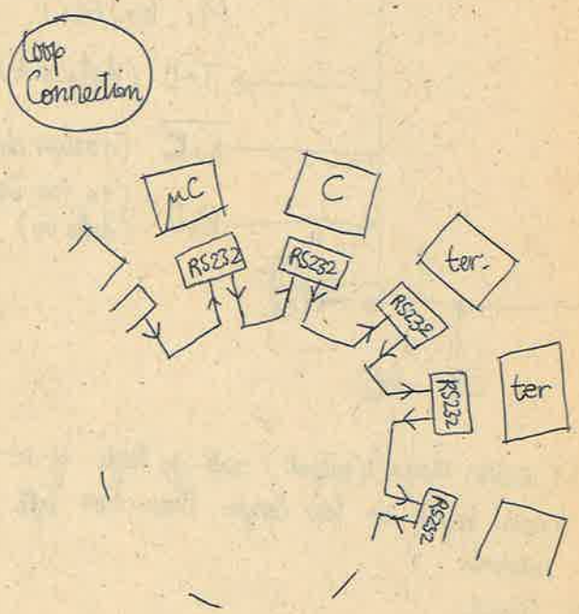
1111 1111
ABORT SEQUENCE

Receiver removes a 0 following five 1's.



No idle char. allowed.

If µP cannot service in time ; block will be aborted by abort seq.



1. Receiver retransmits what comes in if it is not the destination address.
   It keeps it, if it is the dest.
2. Remove the message from the loop, after it travels around the loop once.
   Destination point alters the message slightly to mark successful receiving.

NEWHALL LOOP : Transmitter sends all messages it has, then control passes to next.
⇒ waiting time may be high.

PIERCE LOOP : Small packets of message.
⇒ reassembled at receiver

Serial interfaces :

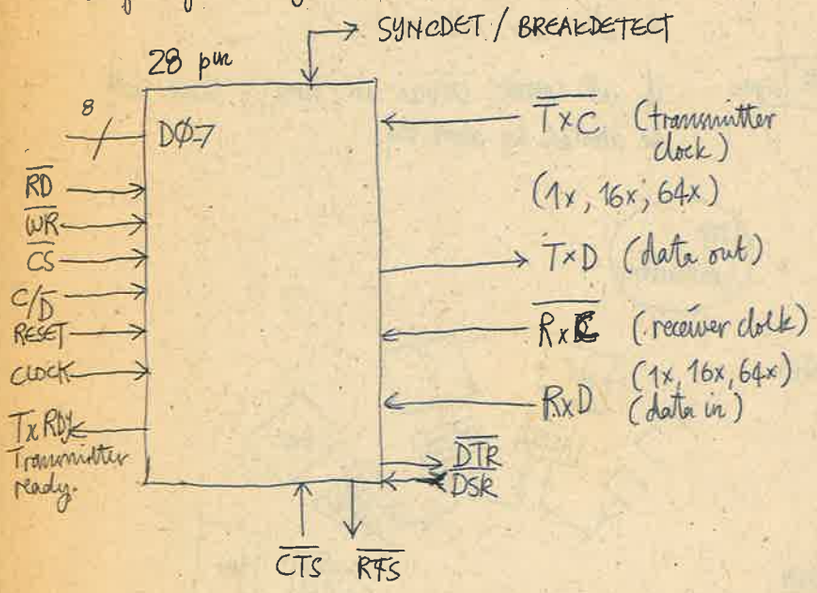
Crosstalk = Coupling between adjacent wires.

Problem in an asynch. system :
If transmitter clock is faster than receiver clock.
on transmitter : # of stop bits : 2
on receiver # of stop " : 1

If receiver can not respond fast enough, it should use CLR TO SEND or RTS.

INTEL 8251 : USART

Useful for Asyn or BISYNC :



BreakDet = (ASYNC Mode)(output) will go high if receiver input is low longer than two byte intervals.

Break : Low signal

Syndet = (SYNC Mode) : If external synchronization is required, used as an input.

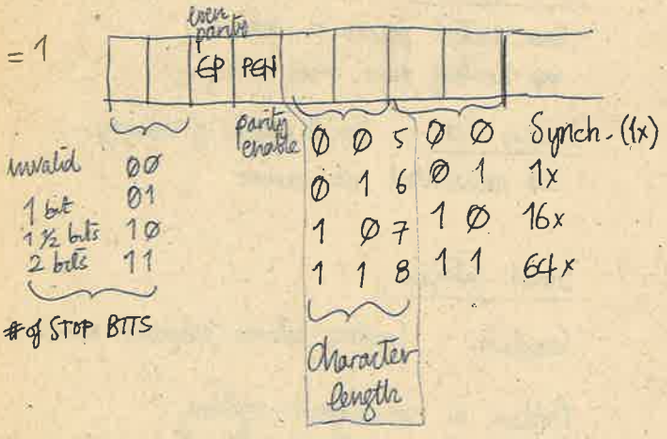
Programming 8251 :

Mode Instruction, Command instruction, command inst,

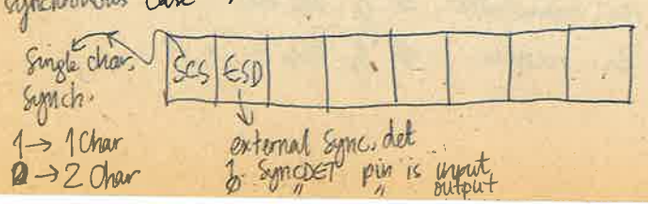
FIRST

after reset.

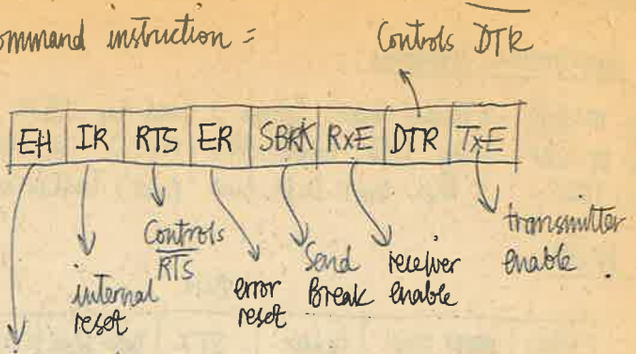
With C/D = 1



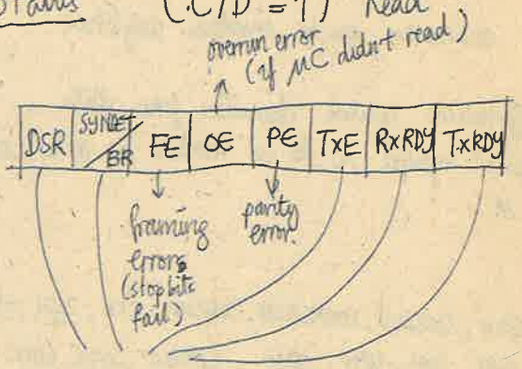
In synchronous case :



Command instruction :



Status (C/D = 1) Read



like external pins.

8251 Programming

Read from serial port, output to PORT1
Read from PORT1, output to serial port.

addressed at 80H & 81H

↑ data
command port

Async. Programming : (input clock 3.072/10 = 307.2 kHz
19200 Hz.

```
INIT : MVI A, 11111010B ; 2stop bits, even parity, 7 bit, 19200 Hz., async.
       OUT 81H          ; mode inst.
       MVI A, 00100111B ; RTS & DTR active, REC, TRANS enabled.
       OUT 81H
```

```
BACK: IN 81H    ; read 8251 status
      MOV B,A
      ANI 00000001B ; TxRDy bit.
      JZ REC
      IN PORT2
      OUT 80H   ; data port
REC:  MOV A,B   ; status info.
      ANI 00000010B ; RxRDy bit.
      JZ BACK
      IN 80H    ; data port.
      OUT PORT1
      JMP BACK
```

Sync. Programming : Data between 1000H - 10FFH must be sent at 30.72 kHz.

C.S.

SYNC

```
INIT : MVI A, 10001100B ; single sync. char, internal sync., no parity,
                        ; 8 bit/char.
       OUT 81H          ; mode inst.
       MVI A, 00010110B ; sync. character.
       OUT 81H
       MVI A, 00000011B ; trans enable, DTR active
       OUT 81H
```

```
        MVI  C,01H    ; SOH
        CALL SEND
        MVI  C,SRCADD ;
        CALL SEND1
        MVI  C,DESADD
        CALL SEND1
        MVI  C,MESNO
        CALL SEND1
        MVI  C,02H    ; STX
        CALL SEND
;
        LXI  H,1000H  ; pointer
        LXI  D,0      ; checksum.
        MVI  B,0
BACK    MOV  C,M
        ORA  A        ; clear carry.
        XCHG          ; DE⇔HL
        DAD  B        ; HL ← HL+BC
        XCHG          ; DE ← checksum.
        CALL SEND1
        INX  H        ; pointer increment
        MVI  A,0
        CMP  L        ; if L is zero it is finished
        JNZ  BACK
;
        MVI  C,03H    ; ETX
        CALL SEND
        MOV  C,E      }
        CALL SEND1    } checksum.
        MOV  C,D      }
        CALL SEND1    }
        MVI  C,16H    ; sync. char.
        CALL SEND
```

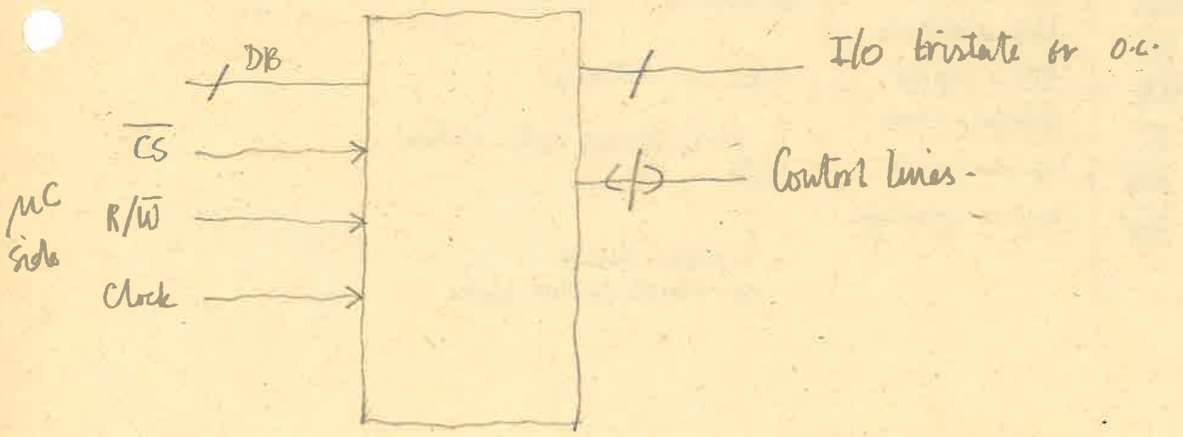
```
SEND1:  MOV A,C
        CPI 10H     ; is it DLE?
        JZ  SNDDLE
        DCR A
        CPI 03H     ; if less. it is CNTR char.
        JNC NOPROB
SNDDLE: CALL TRDY   ; wait until TR ready.
        MVI A,10H   ; send DLE.
        OUT 80H
SEND:
NOPROB: CALL TRDY
        MOV A,C
        OUT 80H
        RET
TRDY:   IN 81H      ; status
        ANI 00000001B
        JZ TRDY
        RET
```

Control characters:
01H, 02H, 03H, 10H
↑ D·L·E character

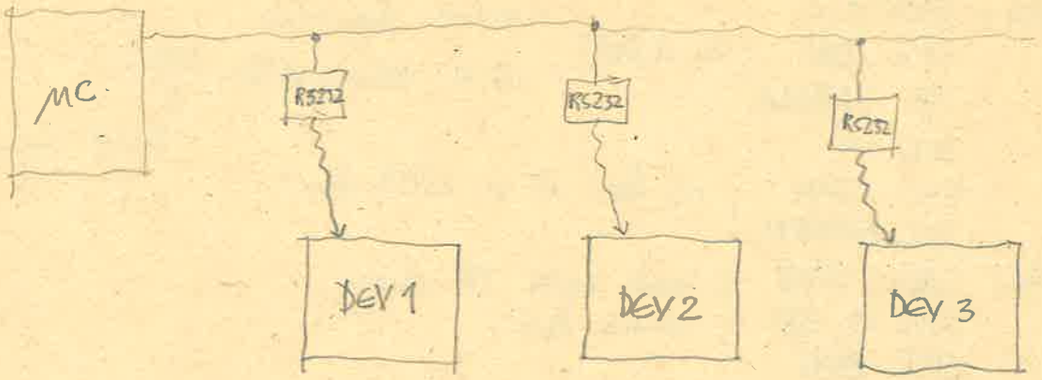
Parallel Interfacing

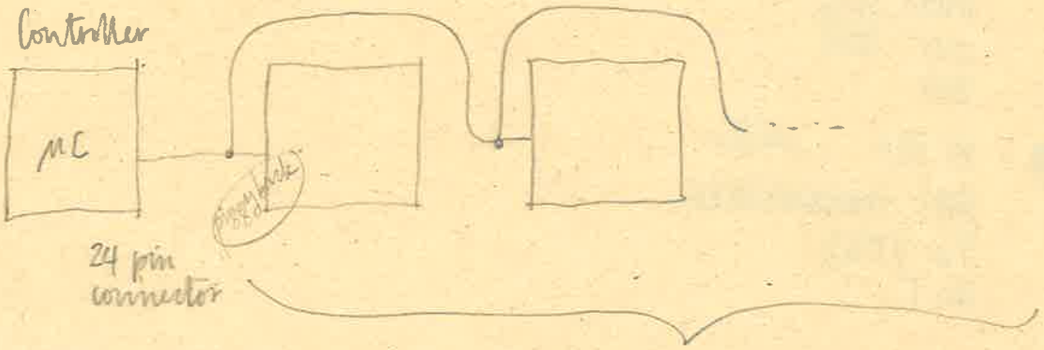
8155
8255 PIO

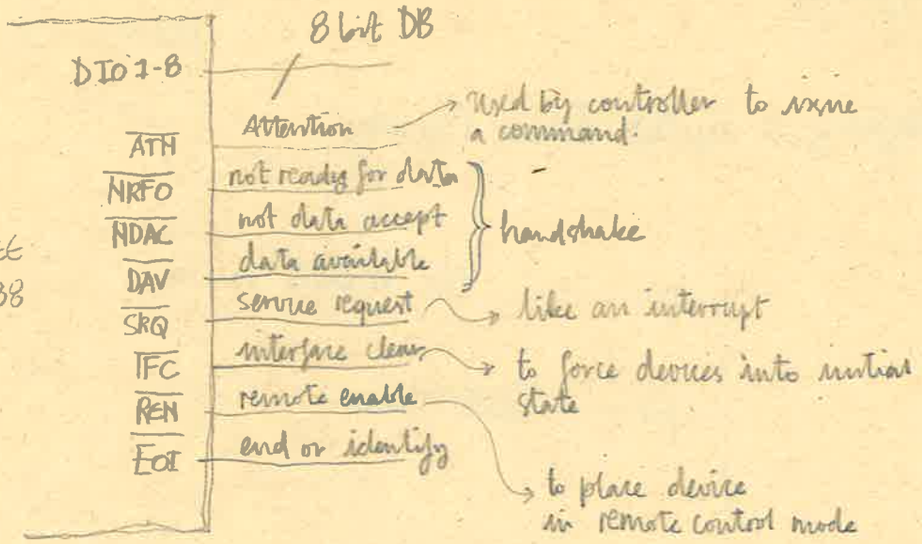
High speed,
Simplifies interfacing to digital devices (nearby)

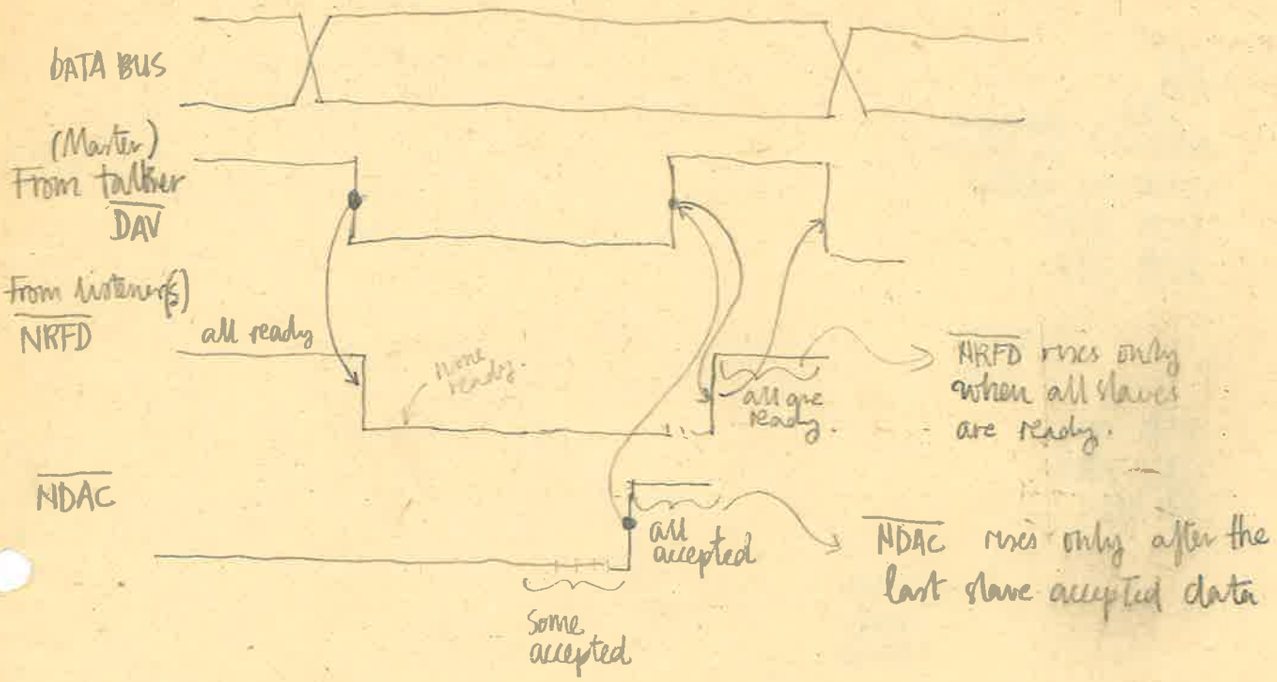


IEEE-488 Interface (HPIB) (GPIB).


µC
RS232
RS232
RS232
DEV 1
DEV 2
DEV 3


Controller
µC
24 pin connector
up to 31 devices.


8 bit DB
DIO 1-8
ATN
Attention
used by controller to issue a command.
NRFD
not ready for data
NDAC
not data accept
DAV
data available
handshake
IEEE 488
SRQ
service request
like an interrupt
IFC
interface clear
to force devices into initial state
REN
remote enable
EOI
end or identify
to place device in remote control mode

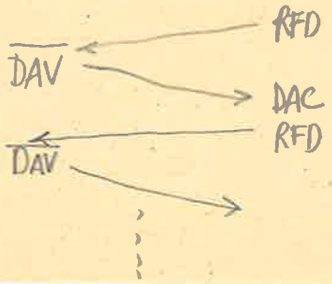


Rising edge of $\overline{NDAC}$ triggers rising edge $\overline{DAV}$
Rising edge of $\overline{NRFD}$ " falling edge "

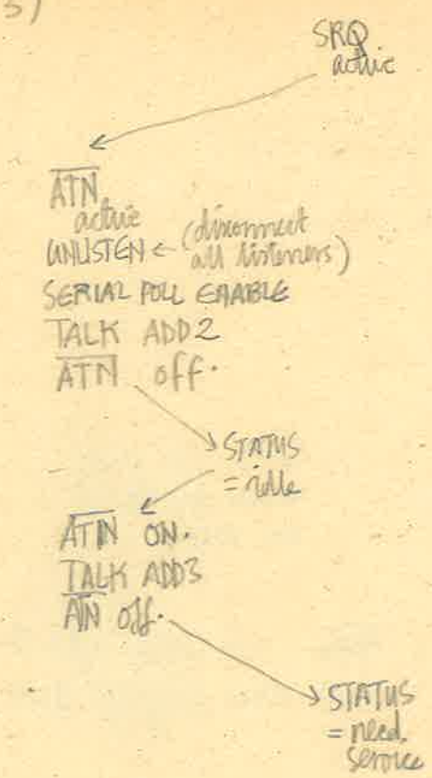
Transmitter (talkers)
Receivers (listener)
One controller in the system
Controller starts the data transfer by commanding a talker to talk & a listener to listen.

1) $\overline{ATN}$ made active
Talker address (#2)
Listener address (#4)
$\overline{ATN}$ passive.

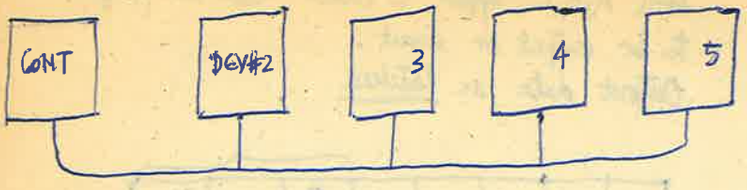
2)



3) Cont dev#2 dev#3 dev#4 dev#5



514 23/12/83



Cont

ATN on
TALK ADD3
LIST ADD5
ATN off → 3 & 5 transfer data
⋮
END (EOI) ← Reinitiation of interrupted transfer (2&4)

ATN on
UNLISTEN
SERIAL POLL EN
TALK AD 3
ATN off → STATUS: transfer complete

ATN on
Serial Poll enable
Talk ad 2
LISTEN ad 4
ATN off → DEV 2 & 4 continues

Parallel Polling

ATN on
EOI on → "end or identify"

→ Data byte is sent: One bit is assigned for each device (up to 8 devices is possible)

If more than 8 ⇒ devices can share the data bus bit, must be followed by a serial poll.

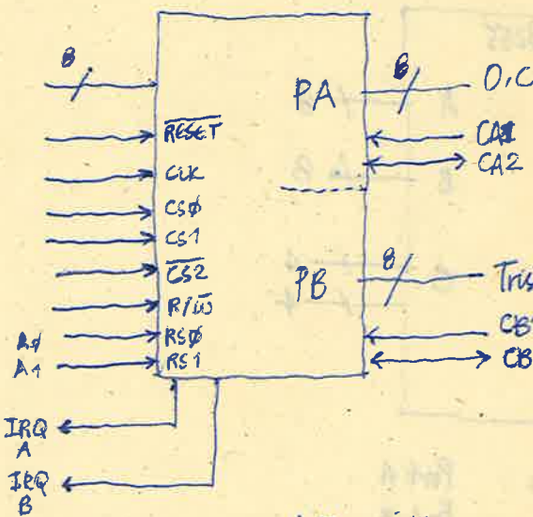
Interface functions :

Source handshake	:	Send DAV, sense RFD & DAC (ready for data, data accepted)
Acceptor "	:	Sense DAV, send RFD & DAC
Talker "	:	Respond to talker ad. transmit streams of data; return status in response to serial poll.
Listener	:	Respond to listener ad, receive streams of data return status in response to serial poll,
Service req.	:	Send SRQ to notify Controller
Remote/local	:	With REN active, accept commands from bus.
Parallel Poll	:	Respond with status when EOI is active.
Device Clear	:	Change to initial state.
Device trigger	:	Perform a prespecified action
Controller	:	Issues commands on the bus.

Intel 8291 ← Talker/Listener
8292 ← Controller

Motorola - AMI - 6488
Fairchild 96LS488

6821 PIA "programmable interface adapter"



Each port has data register
data direction register ← each bit defines whether that bit is input or output.

Control/status register.

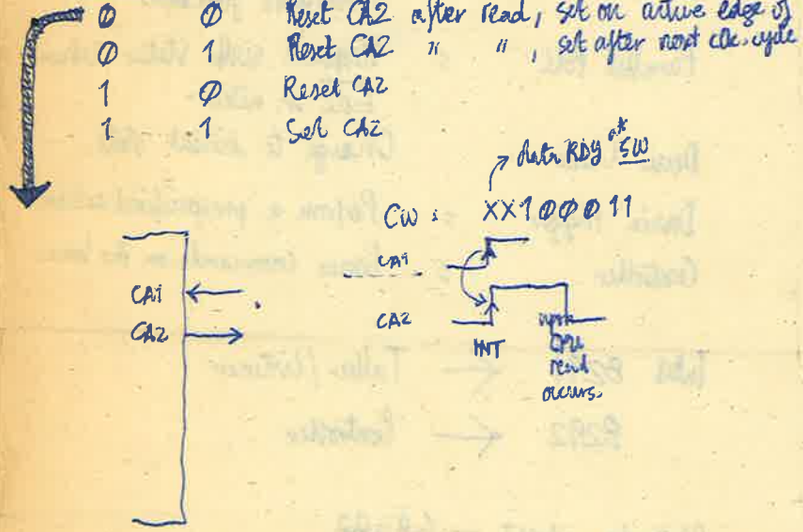
Bits 1 & Ø of control word define

Bit 1	Bit Ø	(6) Bit 7 of CW	IRQ
Ø	Ø	At ↓ CA1 (CA2) High	disabled
Ø	1	At ↑ CA1 (CA2) High	low
1	Ø	At ↓ CA1 (CA2) High	disable
1	1	At ↓ CA1 (CA2) High	low.

CA2 is input if bit 5 of CW is low.

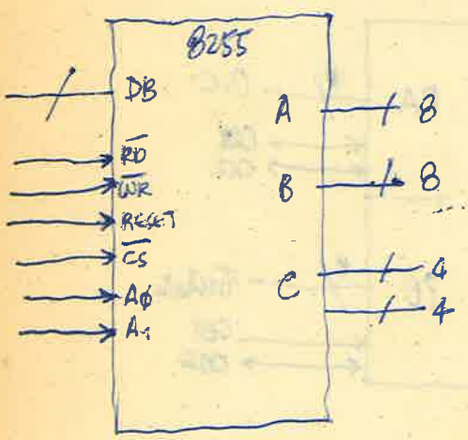
If bit 5 of CW is high CA2 is output

BIT 4	BIT 3	
Ø	Ø	Reset CA2 after read, set on active edge of CA1
Ø	1	Reset CA2 " " , set after next clk. cycle
1	Ø	Reset CA2
1	1	Set CA2



8255 Intel — 24 I/O pins.

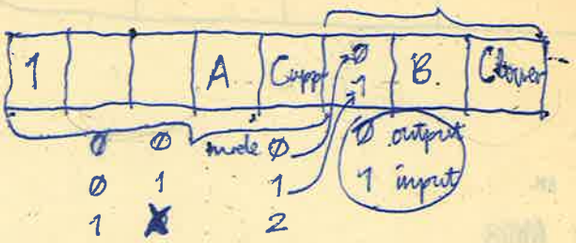
Three modes : i. Basic I/O without handshaking (Mode Ø)
ii. Unidirectional with " (Mode 1)
iii. Bidirectional with " (Mode 2)



4 registers : Port A
Port B
Port C
Control word.

Mode Ø

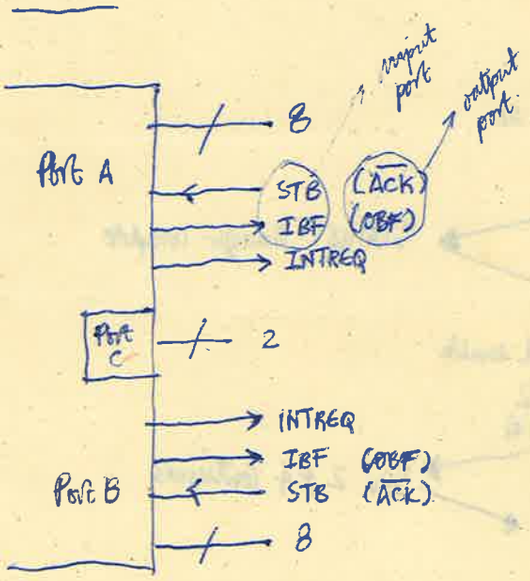
Ports A, B, C upper, C lower can be programmed to be output or input.
Output data is latched.

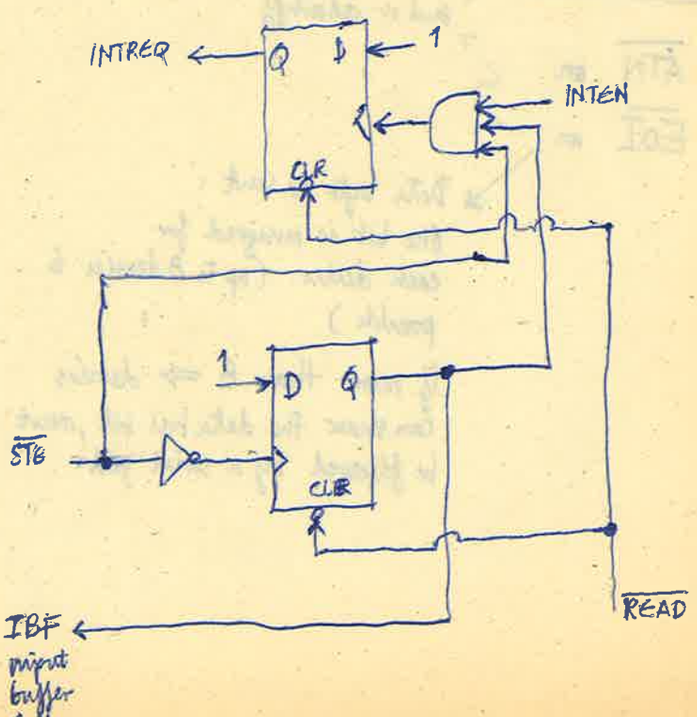


ex : define A output, B input, C upper in, C lower out (in mode Ø)

CW → 1ØØØ1Ø1Ø

Mode 1 :





↓STB ⇒ IBF → 1

↑STB ⇒ INTREQ ↑ ⇒ CPU $\overline{\text{READ}}$ ⇒ IBF → 0
INTREQ → 0

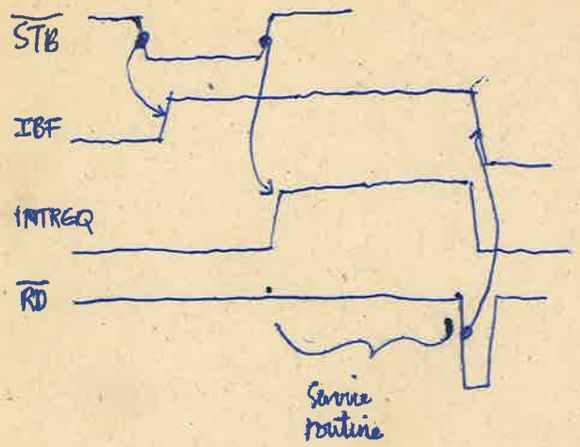
(master) $\overline{\text{STB}}$

READ

IBF

INTREQ

$\overline{\text{RD}}$



$\overline{\text{WR}}$

WRITE

INTREQ

~~$\overline{\text{OBF}}$~~

$\overline{\text{ACK}}$