Embedded Networking :-

Networking (A group of systems interconnected people & things)

Unit-5

(U-5) Wireless Embedded networking → Energy Efficient (MAC Protocol & robust routing)

- Wireless sensor networks
- Applications
- N/w topology
- Time Synchronization

Unit-4 we have n/w to pass information → message.

↳ Embedded Ethernet.

↳ Exchange of information using → UDP, TCP Protocols

(User Datagram Protocol)
(Transmission Control ")

↳ Serving web pages with Dynamic Data / responds to user i/p email.

↳ Keeping Devices & n/w secure.

Protocol :- Specify interaction b/w the communicating entities.

Unit-3

Ethernet Basics :-

↳ Elements of n/w
↳ Building of n/w → Hardware options → Cables, Connections & n/w speed, Design choices.

↳ Ethernet Controllers → Local & Internet Protocols.

Unit-2

USB & CAN Bus

↳ USB Bus (Universal Serial Bus)
- Speed
- States
- Communication → Packets, Dataflow types, Enumeration
- USB Interfaces → PIC 18 MC

↳ CAN Bus (Controller Area N/w Bus)
- frames of ~~USB~~
- Bit stuffing
- Types of errors
- nominal Bit timing
- Interface → PIC MC (CAN)

Unit-1

Embedded Communication Protocols.

↳ Embedded N/w → Serial/Parallel Communication
↳ Serial Comm Protocols → RS232 standard / RS 485 - synchronous serial Protocols
↳ Interfaces (SPI/I2C)
↳ PC Parallel Port Programming → (ISA/PCI Bus Protocols).

Unit-1

Embedded Communication Protocols :-

1) Embedded Networking :-

→ Some Computers are independent units, with little need to exchange information with other computers near (or) far.
→ Computers may use local interfaces such as USB (or) RS-232 to communicate with printers (or) other devices.
→ But with a n/w connections a computer can reach beyond its local interfaces to send & receive information of any kind, over distance large & small, via wires & air.
→ Computers of different types can communicate using n/w protocols supported by all.
→ In n/w of embedded system each system can communicate with the other systems on the n/w, sharing information and sending and responding to requests as needed.

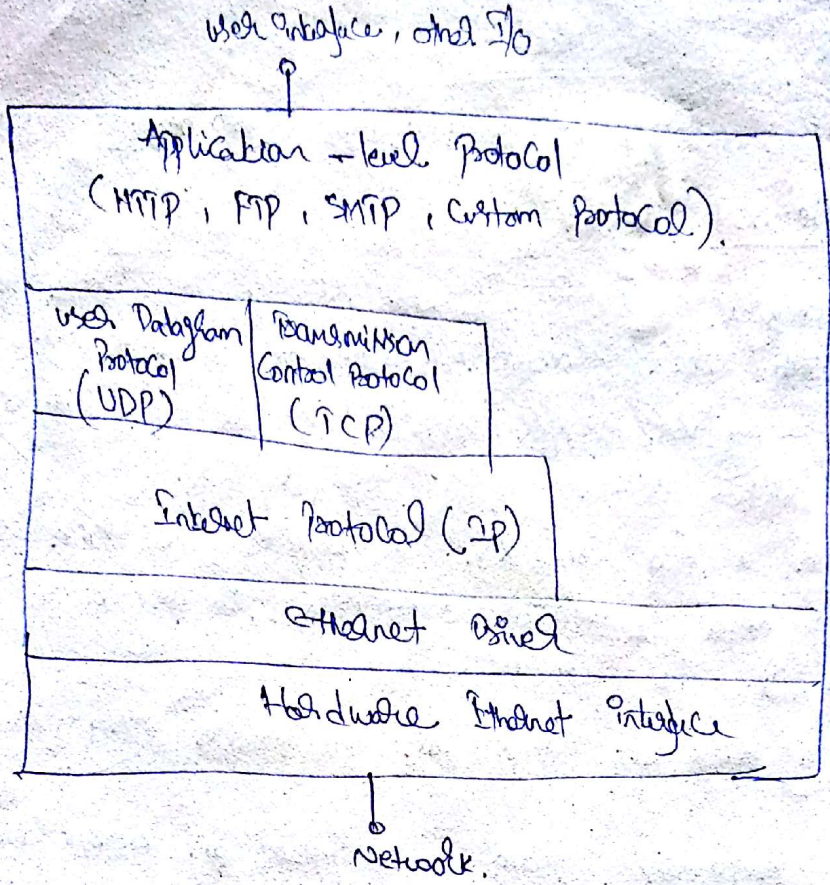
2) Elements of a Network :-

Components :-

① Two (or) more computers that need to communicate with each other. i.e at least one of computer is an embedded system, which is a device that contains a computer dedicated to a specific task (or) a series of related tasks.

② A defined physical interface, to ensure that the dp of transmitting computer is compatible with the ip of receiving computers. For Ethernet n/w, the Ethernet standard specifies this interface.

③ Cables (or) wireless transrecievers to connect the computer, Ethernet n/w have several options for cables. An Ethernet interface may also connect to a device called a wireless access point, which enables the embedded system to access a wireless n/w.



HTTP - Hyper text transfer Protocol
FTP - file transfer Protocol
SMTP - simple mail transfer Protocol

→ Learn to understand the need of Communication in Embedded systems
→ Parallel Vs serial Communication.
→ Synchronous & Asynchronous transmission methods
→ Serial data transmission modes.

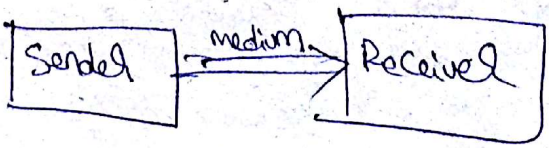
(*) An Integrated System with some dedicated function formed with micro processor units/ micro controller units interconnected with Electronics/ mechanical/ optical components/ peripherals.

(*) Em.S Contains 2 - main elements → E.m.Sys Hardware
→ Em.Sys firmware

(*) in general Communication is a activity of conveying information.

(*) means sending (or) receiving data information b/w peripherals (or) b/w MCU/MPU and peripherals.

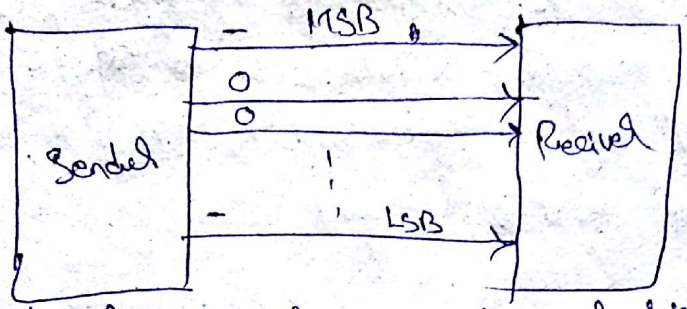
(*) Data Communication:-



Medium - wired / wireless.

Parallel ← serial.

Wired Communication – Parallel:-

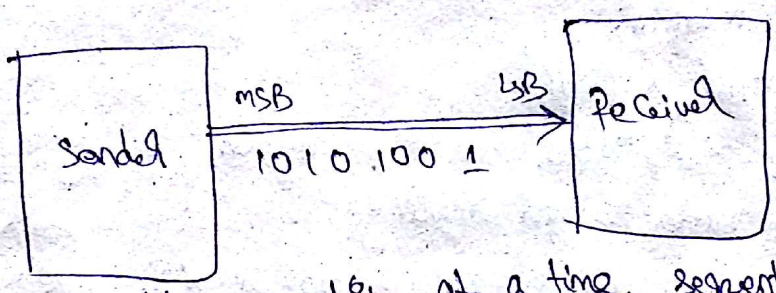


(*) method of sending several streams of data simultaneously along multiple channels.

(*) Parallel channel may requires additional signals.
- → clock signal to pace the flow of data
- → Direction signal to contol the direction of data flow
- → Handshaking signal

→ Fast and suitable for short distance only.

Wired Communication – Serial:-



→ method of sending data one-bit at a time sequentially over a single channel. Can operate on as little as one wire, usually over more than four wires.

→ much more common, particularly over longer distances, improved signal integrity, and transmission speeds in serial making USB & use common even at shorter distances.

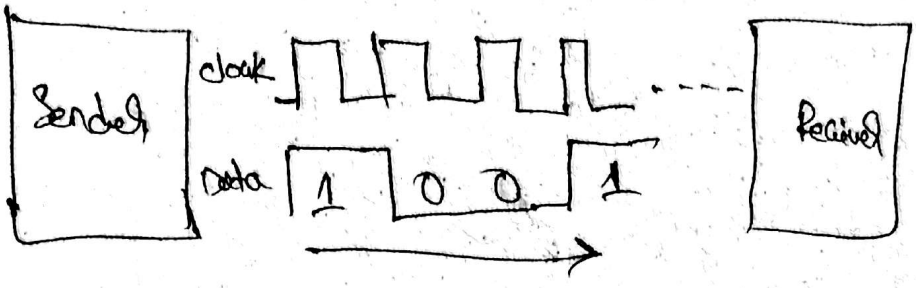
(*) Data transmission methods:-
- → need of synchronization.
- → Synchronous transmission method
- → Asynchronous " "

→ Synchronous Transmission Method:-

→ Data transmission is synchronized by clock

→ 2 - signals are required - a pulse on one signal clock/strobe indicates when another bit of information is ready on other signal DATA

→ lower overhead and greater throughput



→ Asynchronous Transmission Methods:-

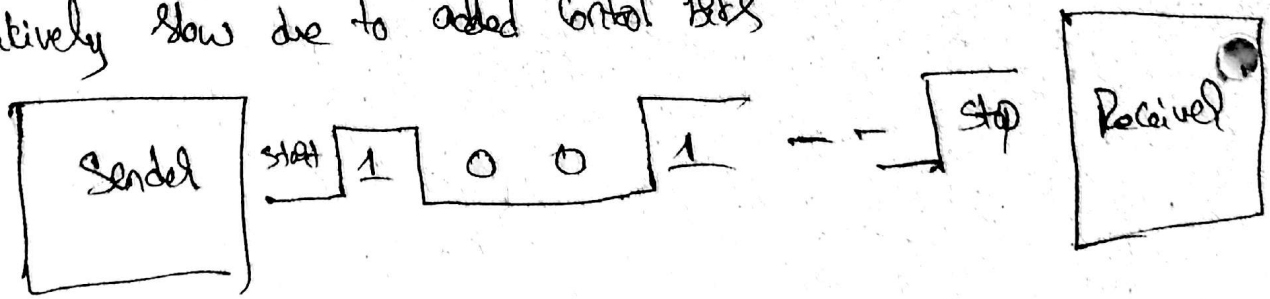
① Data tx is not synchronized by clock, but special signals are used along tx-medium.

② one signal line is enough.

③ Receiver uses transitions on this signal to figure out the transmission bit rate (baud rate) and timing.

④ well suited for applications where messages are generated at irregular intervals

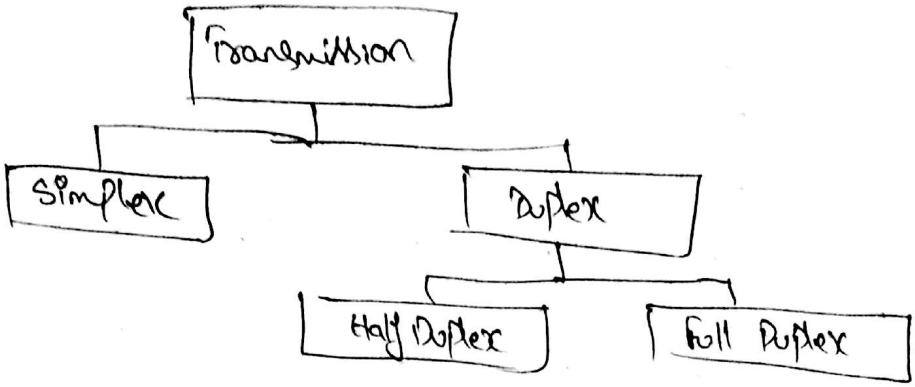
⑤ Relatively slow due to added control bits



1 - start bit
2 - stop bits.

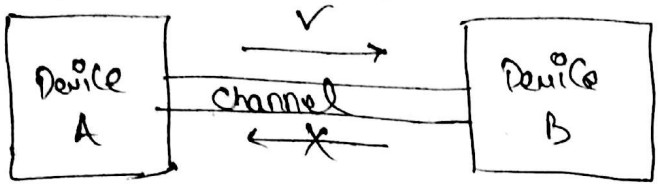
*) Serial Data transmission modes:-

Modes directs the direction of flow of data from two communication devices.



Simplex transmission mode:-

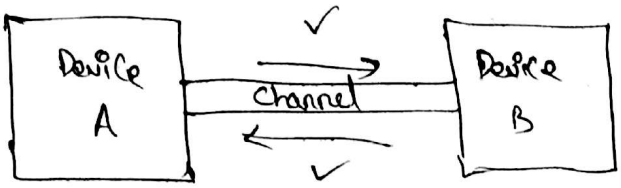
→ Data Communication is unidirectional.
→ Receiver cannot send the reply back to sender.



Eg:- Mouse & Keyboard only send signal to CPU but never receive signals.

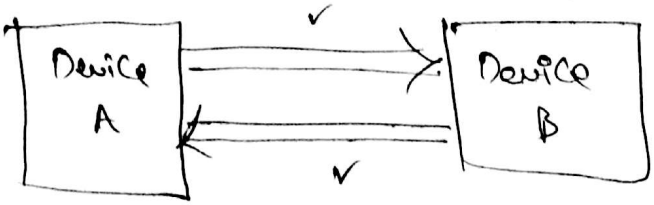
Half Duplex transmission Mode:-

Data Communication is bidirectional but only one direction is allowed at time, means the medium/channel is alternatively used by the devices.



Full Duplex transmission mode:-

Data Comm - is bidirectional and both directions are allowed at same time.
→ Data can send as well while receiving the data



*) Need for Communication interfaces :-

i) E.S needs to send data to a host (a Pc (or) a workstation). The host will analyse of data and present the data through a graphical user interface.

ii) The E.S may need to communicate with another embedded system to transmit/receive data. Providing a standard communication interface is preferable rather than having a proprietary interface.

iii) A no:- of E.S may need to be networked to share data. N/w interfaces needs to be provided in such a case.

iv) An E.S may need to be connected to the internet so that anyone can access the embedded system.

Eg:- Real time weather monitoring system. weather monitoring system can be internet enabled using TCP/IP protocol stack and HTTP server.

v) mobile device such as cell phones & Palmtops needed to interact with other devices such as Pc's and laptops for data synchronization.

vi) For some E.S the software may need upgradation after it is installed in the field. The software can be upgraded through communication interface.

*) N/w Introduction :-

→ Embedded & RTOS could be standalone (or) connected.

→ A Real time is often composed from a number of Periodic (time triggered) and Sporadic (event triggered) task which communicate their result by passing messages.

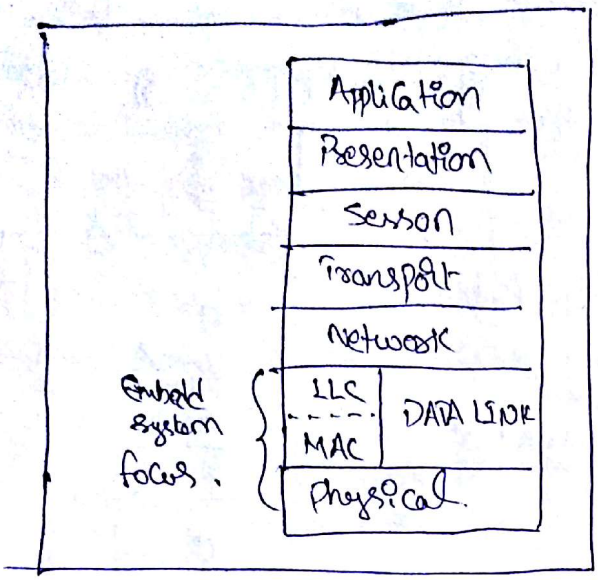
→ In a distributed real time systems these messages are sometimes sent between processors across a communication devices.

→ To guarantee that the timing requirements of all tasks are met the communication delay b/w a sending task and receiving tasks being able to access that message must be bounded.

Eg:- Control system b/w sensors and actuators via Central Computer.

Multi processors b/w processes tasks communicating.

*) Open system interConnection :-



i) Intended for computers
ii) Designed to solve compatibility problem
iii) Layers provide standard interface and services
(iv) Embedded system use some standard. duction ideas
v) Higher layer requires lower layer to work.

→ Layers :-

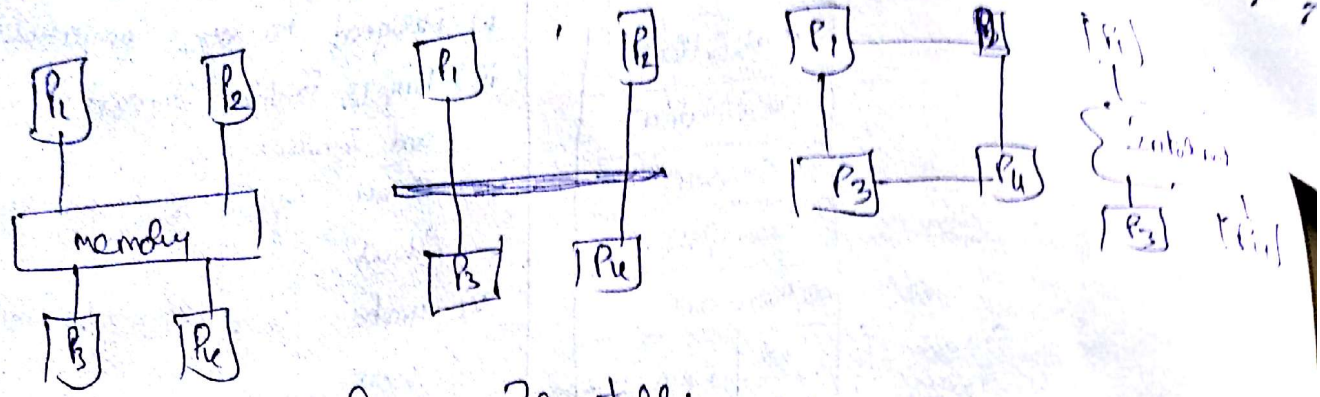
Application - user interface Eg:- Internet Explorer
Presentation - data formating Eg:- Compression & Encoding
Session - handle overall Connection Eg:- OS, Scheduling programs.
Transport - Ensure data transfer, Error checking Eg:- TCP
Network - logical addressing, routing Eg:- IP (from TCP/IP).
Data link - Prepare data for transfer, physical addressing such as Media Access Control (MAC).
Physical - wires and cable, hubs, repeaters.

→ Embedded Communication :-

① Point to Point n/w
→ Each node connected to every node.
→ simple and reliable
→ Dedicated links make it easy to meet real time deadlines
→ costly due to many wires required.

② Shared media networks :-
→ nodes are connected via a bus (or) other topologies
→ less wiring and hence cheaper
→ easily extendable by adding new nodes to n/w
→ Complex n/w protocol
→ Being the system focus from now on.

Basic n/w Architectures:-



*) N/w resources & Qualitative Parameters:-

→ N/w resources:- Bandwidth
Buffer space
Protocol efficiency (data bits / bandwidth) depends on
→ message overhead
→ media Access overhead
Determinacy (Ability to calculate worst case response time)
Robustness
Cost.

*) Event based System:-
→ efficient use of n/w resource.
→ needs high reliability (Event based data comes once in a while)
→ may need Acknowledgement
→ Hard to predict delay in case of overloading (eg:- alarm).

*) State based System:-
→ message ~~based~~ sent at predefined, regular intervals.
→ less efficient due to regular occupation of communication channel by nodes
→ more tolerance. missed message may be ok, since the next one will be coming
→ transient data problem. sending node has to keep data long enough for other to see.
eg:- button pressed may need to be repeated.

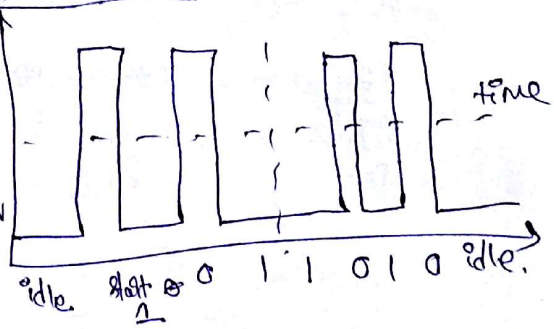
Serial Communication Protocols - RS 232 standards :-

→ Serial Comm is the most simplest form of Communication btw two devices.
→ RS232 is a standard by which two serial devices communicate.
→ The Connection must be no longer than 50 feet.
→ Tx voltages are -15V and +15V.
→ It is designed aboard transmission of characters (of 7-bit of length).
→ RS232 is an Asynchronous form of communication.
→ Asynchronous Communication is important because it is efficient if no data is to be sent, the connection is idel.

logic voltages → { logic 1 is -15V DC, logic 0 is +15V DC } when the Connection is idle, the hardware ties the connection to logical.

(1) → How Can you transmit data :-
→ RS-232 Communication is dependent on a set timing speed at which both pieces of hardware communicate.
→ In other way the hardware knows how long a bit should be high (or) low.
→ RS232 specifies 'start' & 'stop' bits.

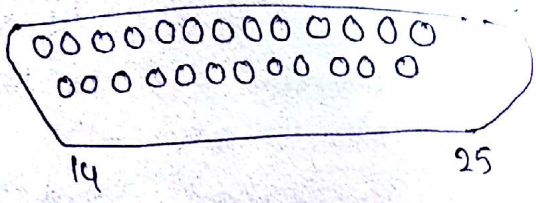
(2) → Sending one character :-
→ This Communication is dependent on the fact that both devices are sampling the bits at same rate!
→ The start bit is a logical - 0' sent on a line to tell other device to start sampling.
→ The stop bit is always sent (per RS232 standards). logic - '1'
→ The transmission rate of serial devices is called "baud rate'. It is number of changes in signal per second.



(3) → Line sampling & framing :-
→ RS232 hardware samples the line multiple times during a single bit transmission.
→ If the samples don't have same voltages a framing error occurs.
→ A framing error should only occur if one device is sending faster than the other device is set to received.
→ An intentional frame error can be caused by sending a "BREAK."

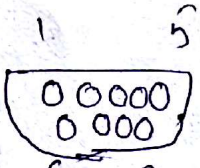
④ RS232 DB25 Pin out:-

DB25	function	Abbreviation
Pin-1	Chassis / frame Ground	GND
P-2	Transmitted data	TD
P-3	Receive Data	RD
P-4	Request to Send	RTS
P-5	Clear to Send	CTS
P-6	Data Set Ready	DSR
P-7	Signal Ground	GND
P-8	Data Carrier Detect	DCD (or) CD
P-9	Transmit + (current loop)	TD+
P-10	Transmit - (Current loop)	TD-
Pin-18	Receiver + (current loop)	RD+
P-20	Data terminal Ready	DTR
P-22	Ring Indicator	RI
P-25	Receiver - (Current loop)	RD-



⑤ RS-232 DB-9 Pin out

Pin-1	Data Carrier Detect	CD
P-2	Receiver Data	RD (or) RX (or) RXD
P-3	Transmitted Data	TD (or) TX (or) TXD
P-4	Data Terminal Ready	DTR
P-5	Signal Ground	GND
P-6	Data Set Ready	DSR
P-7	Request to Send	RTS
P-8	Clear to Send	CTS
P-9	Ring Indicator	RI.



⑥ Connector types:- 2 types

Computer Terminal Equipment (CTE)
Computer transmit on Pin-2 & Receiver Pin 3

Data Terminal Equipment (DTE)
modem will transmit on Pin-3 receiver on Pin-2.

A popular way to transfer commands and data between a personal computer and a microcontroller is the use of standard interface, like the one described by protocols RS232 (older) or USB (newer). This chapter is devoted to communication conforming to RS232 protocol, the hardware for such interface is provided on board. An example will be presented showing the processing of commands received through RS232 interface, and sending of a string of numbers using the same interface.

The protocol RS232 defines the signals used in communication, and the hardware to transfer signals between devices. The time diagram of the typical signal used to transfer character 'A' (ASCII: 65_{10} or 0x41) from device A to device B is given in Fig. 1, and would appear on the upper line TX -> RX between devices.

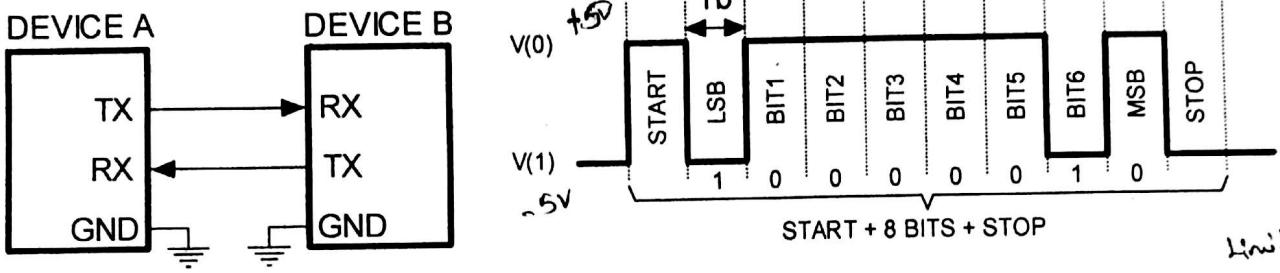


Figure 1: A signal conforming to RS232 standard

The standard defines voltage levels V(0) to be at least +5V at the transmitting end of the line TX, and can be degraded along the line to become at least +3V at the receiving and of the line. Similarly voltage level V(1) must be at least -5V at TX, and at least -3V at RX. The standard also defined the upper limit for these voltages to be up to ±15V. Logic high is transferred as V(0). The microcontroller cannot handle such voltage levels, so typically a voltage level translator is inserted between the microcontroller and the connector where the RS232 signals are available. The connectors are typically so-called D9 connectors, and the electric wiring in between two connectors at devices A and B is shown in Fig. 2, for two female type connectors at both devices.

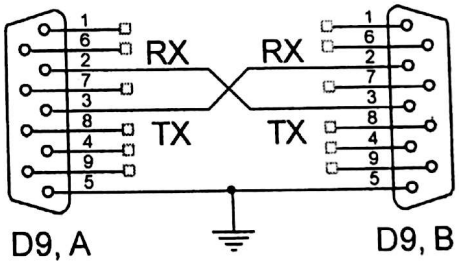


Figure 2: Wiring for RS232 communication

The standard defines the number of bits to be transferred within one pack, Fig. 1 right, as eight for regular transmission, and nine for special purposes. The duration *Tb* of each bit defines the speed of transmission and is called the baud-rate. The typical baud-rate is 9600 bits per second (Baud, Bd), and the time *Tb* equals 104.16μs. Other baud rates are also common: 19200 Bd, 38400 Bd, 57600 Bd,

(9) Typically Baud rate :-

Time Equal (T_b)	Complete Tx of Packet rate is
9600 bits/sec (Baud/Bd) → 104.16 μs.	1.0416 ms

other Bd used very frequently:
- 19200 Bd
- 38400 Bd
- 57600 Bd
- 115200 Bd.

→ The oscillator in RS232 circuitry operates @ 1.8432 MHz and it is divided by 600 to obtain 115200 data rate.

(10) Data bits:- no: of bits Tx for each character. character can have 5(or)6(or)7(or)8 bits.
If you send ASCII character the no: of bits is '7'.

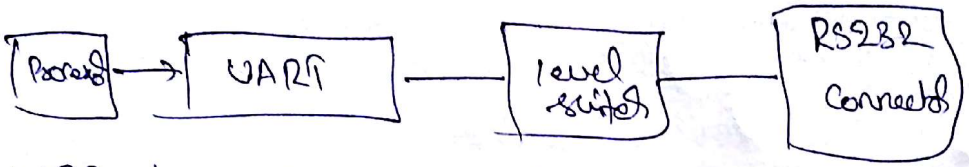
(11) Parity:- Bit appended to the character for error checking → Even, odd.
Ex:- 1010110 (Even '1')
⇒ P=0 if even parity is used
⇒ P=1 if odd parity is used
i.e device will calculate no: of bits received.

(12) Flow Control:- 1-devices sends data at very fast rate and other device can not absorb the data at that rate.
→ Flow control is a protocol to stop/resume data transmission
→ This protocol is known as hand shaking.
The hand shaking in RS232 using two signals → RTS (Request to send), CTS (clear to send)
→ To have software hand shaking:-
→ A device can request to suspend data transmission by sending the character control S (0x13).
→ The signal to resume data transmission is sent using the character control Q (0x11)
→ The software hand shaking is also known as (XON/XOFF).

(13) UART :- Universal Asynchronous Rx Tx chip converts Parallel data received from the processor into serial format and send it to RS232 level shifter.

→ It also takes serial data from RS232 level shifter and converts it to parallel format.

→ The processor process data in 16 bit format not in serial format.



UART has 2 sections (Receiver & Transmit section)

Receiver data in serial format convert to parallel format

Transmitter section → Receives parall convert to serial.

→ The UART chip also adds the start bit, stop bit and parity bit.

→ RS232 specifies a distance 19.2 meters, however can achive distance upto 100 meters using RS232 cables.

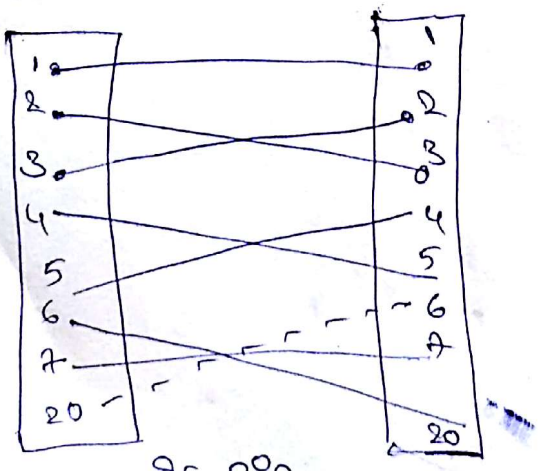
→ data rate supported will be dependent on UART chip and clock used.

→ most of processors including DSP have on-chip UART. ICs using MAX 3222, MAX 3241.

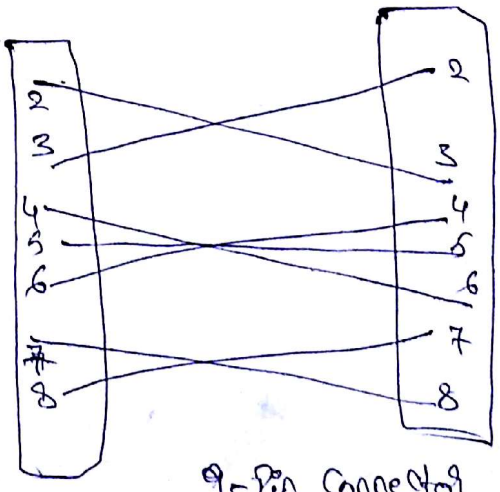
Maxim used as level shifters.

(14) Null modem Cable Connection :-

If you want to Connect 2 - DTE (data terminal Equipments) as two PC's you need to interconnect the two RS232 ports using a null modem cable.

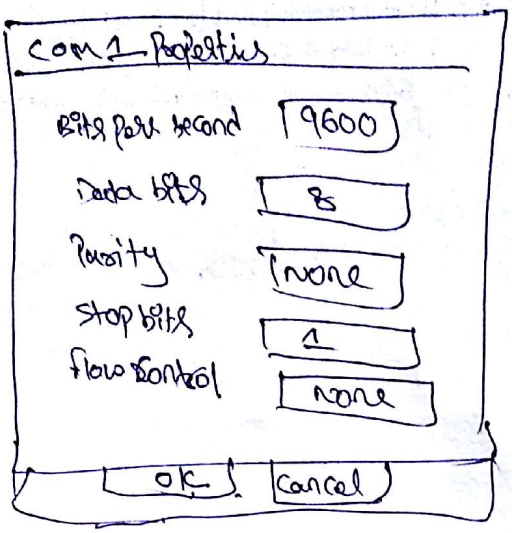


25-pin



9-pin Connector.

By using Hyper terminal Program on windows, Serial comm. Parameters are set.



Programming:-

```
#include <stdio.h>
#include <string.h>
   "      <unistd.h>
   "      <fcntl.h>
   "      <errno.h>
   "      <termios.h>.

int Port_Open (void)
{ int fd;
  fd = open ("/dev/ttyS0", O_RDWR | O_NOCTTY | O_NDELAY);
   if (fd == -1)
   { Perror ("Unable to open the port : /dev/ttyS0");
   } Else
     { fcntl (fd, F_SETFL, 0);
     }
     return (fd);
   }
   void Port_Config (int fd)
   {
     Struct termios settings;
   tcgetattr (fd, & settings);
   cfsetispeed (& settings, B9600); // Set baud rate to 9600
   cfsetospeed (& settings, B9600);
```

```
Settings.c_cflag |= (clocal | CREAD); // set to local mode
Settings.c_flag &= ~PARENB; // no Parity bit
Settings.c_flag &= ~CSTOPB; // two stop BPt
Settings.c_cflag &= ~CSIZE; // bit mask for data Bits.
Settings.c_cflag |= CS8; // 8 data Bits
tcsetattr (Fd, TCSANOW, &settings);
void write_data (int fd, char *c)
{
int n;
n= write (fd, c, strlen(c));
if (n<0)
{ fputs ("write () of data failed! \n", stderr);
}}
void read_data (int fd)
{ char array (255);
char *ptr;
int nbytes;
Ptr = array;
while ((nbytes = read (fd, Ptr, array = sizeof (array)-Ptr-1))>0)
{ Ptr += nbytes;
if ((Ptr[-1] == '\n' || (Ptr[-1] == '\r'))
break;
} *Ptr = '\0';
Printf ("Received string : %s", array);
} void Post_close (int fd)
{ close (fd);
}
int main(void)
{int fd;
fd = Post_open ();
Post_config (fd);
write_data (fd, "Hello world");
read_data(fd);
Post_close (fd); return 0; }.
```

RS422, RS423 & RS485 :-

→ In some modern equipment RS232C has been replaced by other standard that could be used at longer distances and at higher speed.

→ All these can use only half duplex & works upto 1200 meters.

→ RS423 → is an unbalanced allows 1 - t_x & 10 Rx at maximum distance of 1200 meters

→ Baud rate : - 110 - 115,200.

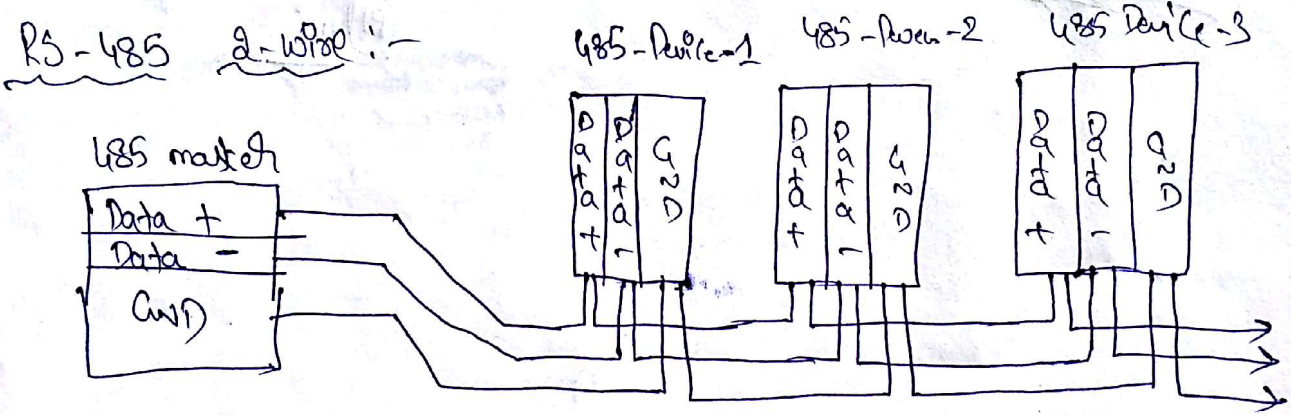
Extended upto 8000 feet / 2,400 meter. by using Repeater

RS 485 Protocols :-

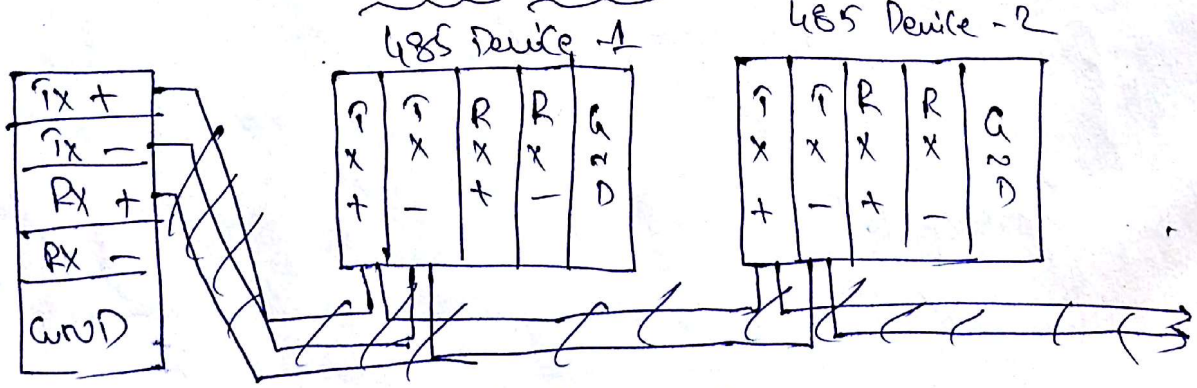
① Modbus RTU ② Optomux ③ PROFIBUS DP

④ DH-485

RS-485 2-wire :-



RS-485 4-wire



Rx + → Tx +
Rx - → Tx -

Tx + → Rx +
Tx - → Rx -
(GND) common

Bus Interfaces

Different types of buses:

- ISA (Industry Standard Architecture)
- EISA (Extended ISA)
- VESA (Video Electronics Standards Association, VL Bus)
- PCI (Peripheral Component Interconnect)
- USB (Universal Serial Bus)
- AGP (Advanced Graphics Port)

ISA is the oldest of all these and today's computers still have a ISA bus interface in form of an ISA slot (connection) on the main board.

ISA has 8-bit and 16-bit standards along with the 32-bit version (EISA).

All three versions operate at 8MHz.

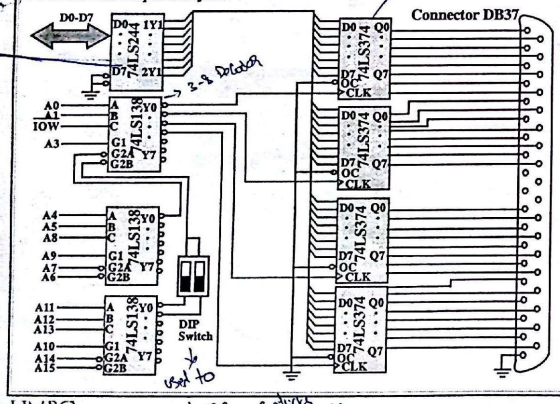
8-Bit ISA Bus connector

Pin #			
	1	GND	IO CHK
	2	RESET	
	3	+5V	
	4	IRQ9	
	5	-5V	D0-D7
	6	DRQ2	
	7	-12V	
	8	OWS	
	9	+12V	
	10	GND	IO RDY
	11	MEMW	AEN
	12	MEMR	
	13	IOW	
	14	IOR	
	15	DACK3	
	16	DRQ3	
	17	DACK1	
	18	DRQ1	
	19	DACK0	
	20	CLOCK	
	21	IRQ7	
	22	IRQ6	A0-A19
	23	IRQ5	
	24	IRQ4	
	25	IRQ3	
	26	DACK2	
	27	T/C	
	28	ALE	
	29	+5V	
	30	OSC	
	31	GND	

ISA Bus Connector Contains

- 8- bit Data Bus
- Demultiplexed 20-bit address Bus
- I/O and Memory Control Signals
- Interrupt Request Lines (IRQ2->IRQ9)
- DMA channels 1-3 Control Signals
- Power, RESET and misc. signals

8-Bit ISA Bus Output Interface



8-Bit ISA Bus Output Interface

4, 8-bit latches interfaced using an ISA interface for 32 bit parallel data.

74LS244 buffers used to ensure only **one** lower power TTL load on the bus. Loading is important as many cards can be connected on the bus.

The DIP switch can be used to change the address thus avoiding address conflicts with other cards in the system.

See text for examples of output interface using a PLD and also an ISA bus input interface for A-to-D converters.

16-bit ISA bus has an additional connector attached behind the 8-bit connector.

Although 8 additional data bits, D_8-D_{15}, are available, the features most often used are the additional interrupt request and DMA request signals.

74LS244 → Octal Tri-State Buffer/Line Driver/
Line Receiver

74LS374 → 3-state octal D-type Transparent latches & Edge Triggered
flip-flops

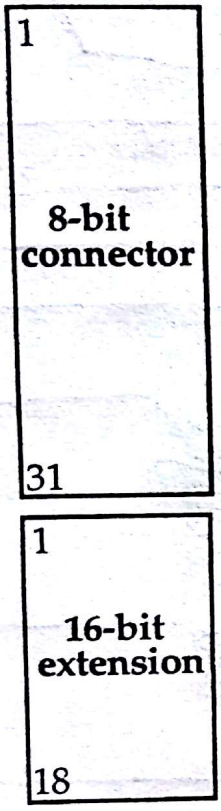
Pin - Config:-

1 → GND - Ground
2 → RESET → Active high to Reset (or) initialize System logic
3 → 5V → +5V DC
4 → IRQ9 → Interrupt Request
5 → -5V → -5V DC
6 → DRQ2 → DMA Request @ 2
7 → -12V → -12V DC
8 → /NOWS → not wait state

11 → /SMEMW → system memory write
12 → /SMEMR → " " Read
13 → IOW → I/O write
IOR → " Read.
DACK3 → DMA ACK 3
DRQ3 → DMA Req 3
DACK1 → DMA ACK 1
DRQ1 → DMA Req 1
DACK 0 → DMA ACK 0
CLOCK → System clock (67 ns; 8-8.33 MHz; 50% duty cycle)
T/C → Terminal Count; Pulses high when DMA term; Count reached
OSC → High Speed clock (70 ns, 14.31818 MHz, 50% duty cycle)

16-Bit ISA BUS

Back of computer



16-bit connector

1	$\overline{\text{MCS16}}$	BHE
2	$\overline{\text{IOCS16}}$	A23
3	IRQ10	A22
4	IRQ11	A21
5	IRQ12	A20
6	IRQ15	A19
7	IRQ14	A18
8	$\overline{\text{DACK0}}$	A17
9	DRQ0	$\overline{\text{MEMR}}$
10	$\overline{\text{DACK5}}$	$\overline{\text{MEMW}}$
11	DRQ5	D8
12	$\overline{\text{DACK6}}$	D9
13	DRQ6	D10
14	$\overline{\text{DACK7}}$	D11
15	DRQ7	D12
16	+5V	D13
17	MASTER	D14
18	GND	D15

Peripheral Component Interconnect (PCI) Bus

PCI is the most common bus found in computers today due to plug-and-play characteristics and ability to function with 64-bit data bus.

A PCI interface contains a series of registers, located in a small memory device, that contain information about the board.

The information in this registers allow the computer to automatically configure the PCI card (Plug-and-Play ***PnP*** feature).

The microprocessor connects to the PCI bus through an integrated circuit called a **PCI Bridge** thus making the PCI bus independent of processor type and architecture.

PCI functions with either a 32-bit or 64-bit address and data bus.

The address and data buses are multiplexed to reduce the size of the edge connector.

32-bit and 64-bit cards.

Newest versions run at 66 MHz (twice the older 33 MHz version).

PCI Parallel Bus:-

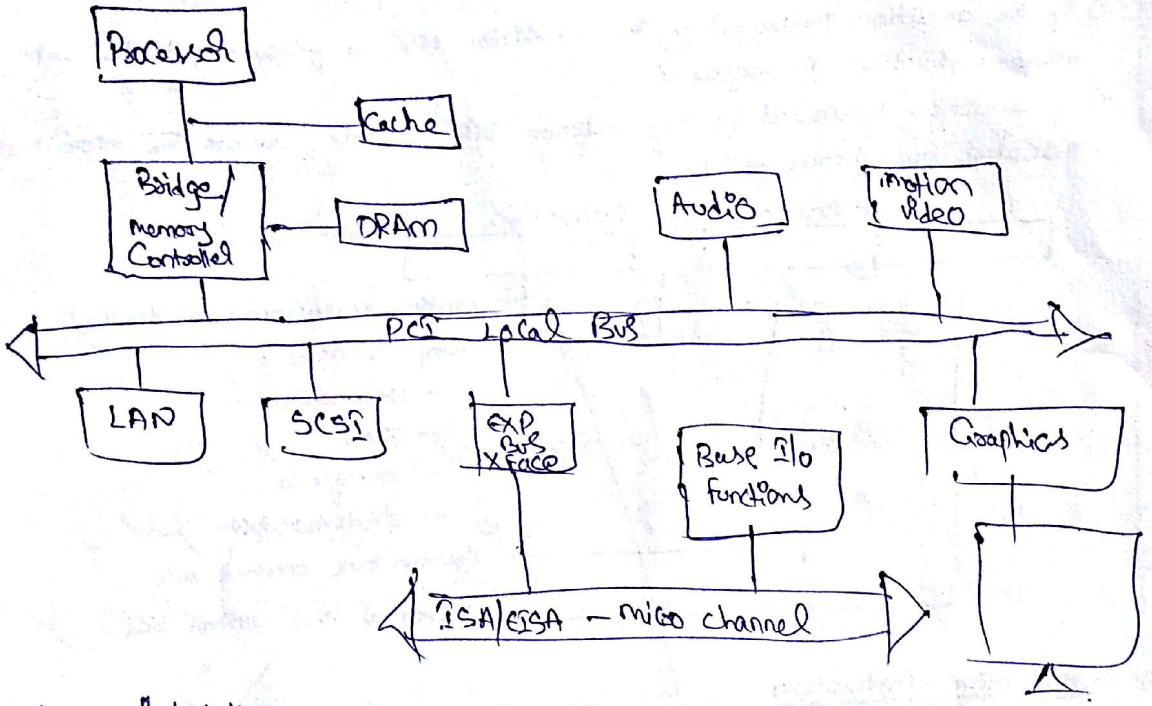
→ Parallel bus enables a host Computer (or) System to Communicate synchronously 32-bit (or) 64-bit with other devices.

Eg:- network interface Card (or) graphic Card.

→ When I/O devices in distributed Embedded System are networked all Can Communicate through a Common Parallel bus.

→ PCI Connects at high speed to other Subsystems having range of I/o devices at very short distances (< 25 cm)

General Block diagram:-



Technology - information:-

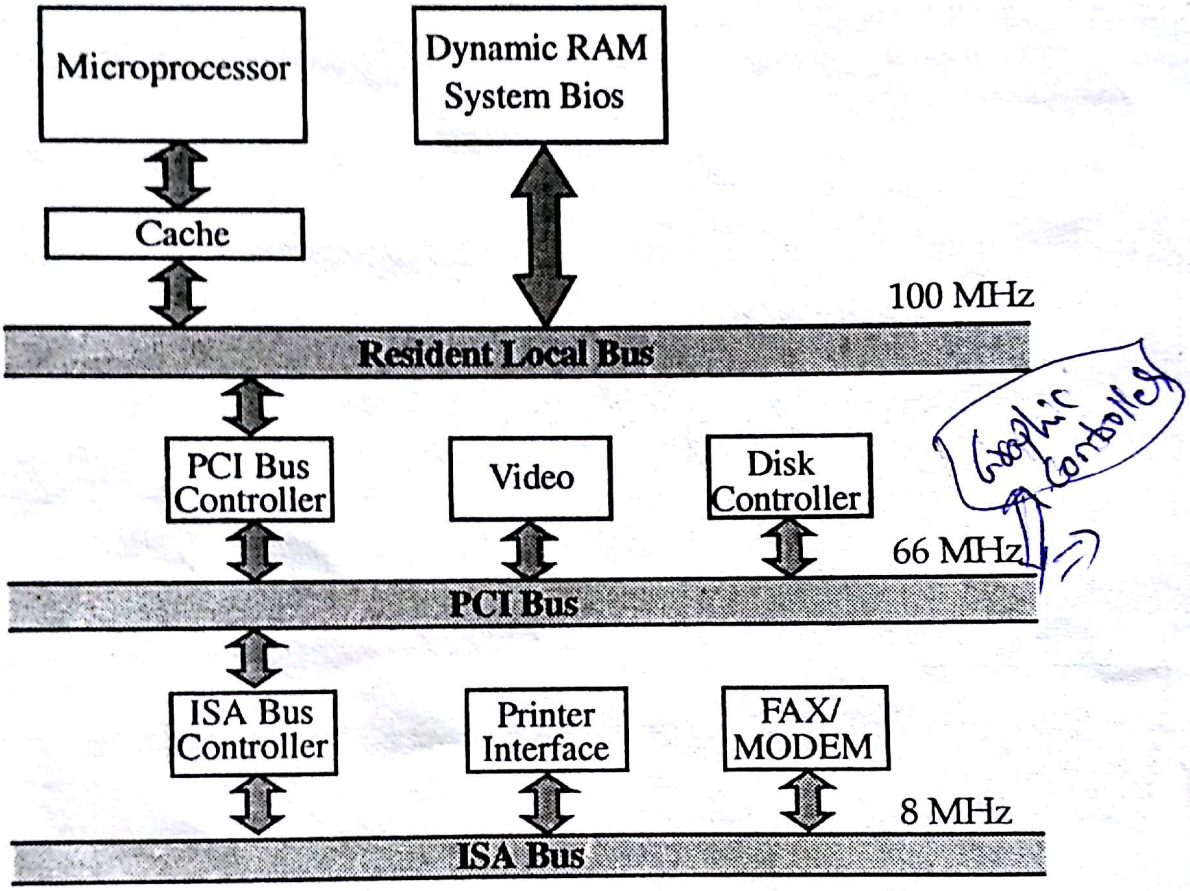
Conventional PCI:-

→ Intel PCI 1.0 Proposal by Intel in 1991
→ PCI Special interest Group (PCI SIG) as PCI 2.0 in 1993
→ version 2.1 approved in 1995
→ Recent version 2.3 approved in march 2002.

PCI-X version 1.0 approved on September 1999.
version 2.0 " " July 2002.

PCI Express:- formerly known as 3GIO
version 1.0 approved on July 2002.

PCI Bus System Structure



→ Conventional PCI :-

* → Plug and Play functionality
* → Standard PCI is 32-bit and operates at 33 MHz
  → Throughput 133 MB/sec
* PCI 2.1 ~~start~~ introduced.
  universal PCI boards supporting both 3.3V and 5V
  64-bit slots and 66 MHz capability
  32-bit throughput @ 66 MHz : 266 MB/sec
  64-bit " @ 66 MHz : 532 MB/sec
* PCI 2.3 system no longer support 5V-adapters.
  3.3V & universal PCI products are still fully supported.

→ PCI-X 1.0 :-
→ 64-bit slots with support for 3.3V & universal PCI
  → no support for 5V only boards.
→ fully backward compatible
  * Conventional 33/66 MHz PCI adapter can be used in PCI-X slots
  * PCI-X adapters can be used in conventional PCI slots.
→ Doubles two speed grades : 66 MHz and 133 MHz
  → The slowest board dictates the maximum speed on bus
  → Targeted high end data n/w and storage n/w applications

→ PCI-X 2.0 :- Based on PCI-X 1.0 (still fully backward compatible)
→ Introduces ECC (Error Correction Codes mechanism to improve reliability and data integrity.
→ Doubles additional speed grades
  PCI-X 266 : 266 MHz (2.13 GB/sec)
  PCI-X 533 : 533 MHz (4.26 GB/sec)
→ Bandwidth sufficient to support new breed of cutting edge technologies
  → 10 GB Ethernet / fibre channel
  → 4X / 12X Infini Band.

PCI-X speed limitations:

→ PCI-X supports point to point & multidrop loads

→ Highest speed grades are supported exclusively with point to point

PCI-X 133
PCI-X 266
PCI-X 533

Two PCI-X 133 loads operate at 100 MHz
Four loads operated at maximum of 66 MHz

Quick view of speed of Bus:-

Bus Type	Bus width	Bus Speed	MB/sec
ISA	16-bits	8 MHz	16 MBPsec
EISA	32-bits	"	32 MBPS
VL-Bus	32-bits	25 MHz	100 MBPS
VI-Bus	32-bits	— 33 MHz	132 MBPS
PCI	32-bits	— "	132 MBPS
PCI	64 "	— "	264 MBPS
PCI	64 "	→ 66 MHz	512 MBPS
PCI	64 "	→ 133 MHz	1 GBPS.
PCI			

The PCI Bus Pin-out :-

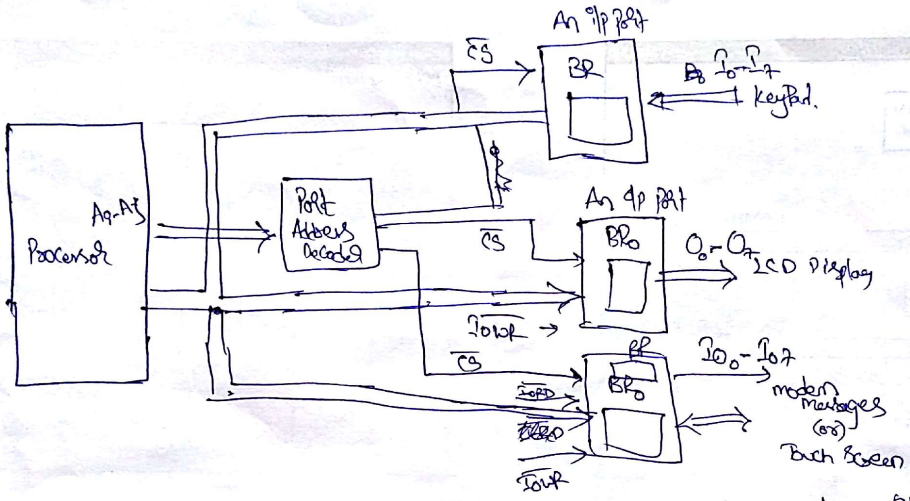
- PCI functions with 32 (or) 64 bits data bus and a full 32-bit address bus.
- Address & data buses, labeled AD_0 - AD_{63} are multiplexed to reduce size of Edge Connectors.
  - → 32-bit Card has Connections 1 through 62.
  - → 64 bit " " all 94 Connections.
  - → 64 bit Card can accomodate a 64-bit address if required at some future point.
- PCI most used interface with microprocessors.

The PCI Address / Data Connections :-

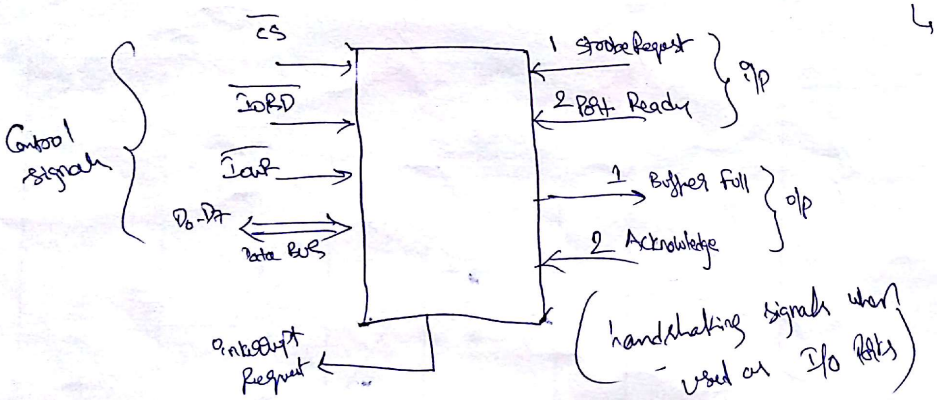
- The PCI Address appears on AD_0 - AD_{31} which are multiplexed with data
  - → Some systems have a 64-bit data bus using AD_{32} - AD_{63} for data transfer
  - → these pins can be used for extending the address to 64-bits.

Timing D

CLK
Frame
AD
C/BE
IRDY
TRDY
DEVSEL

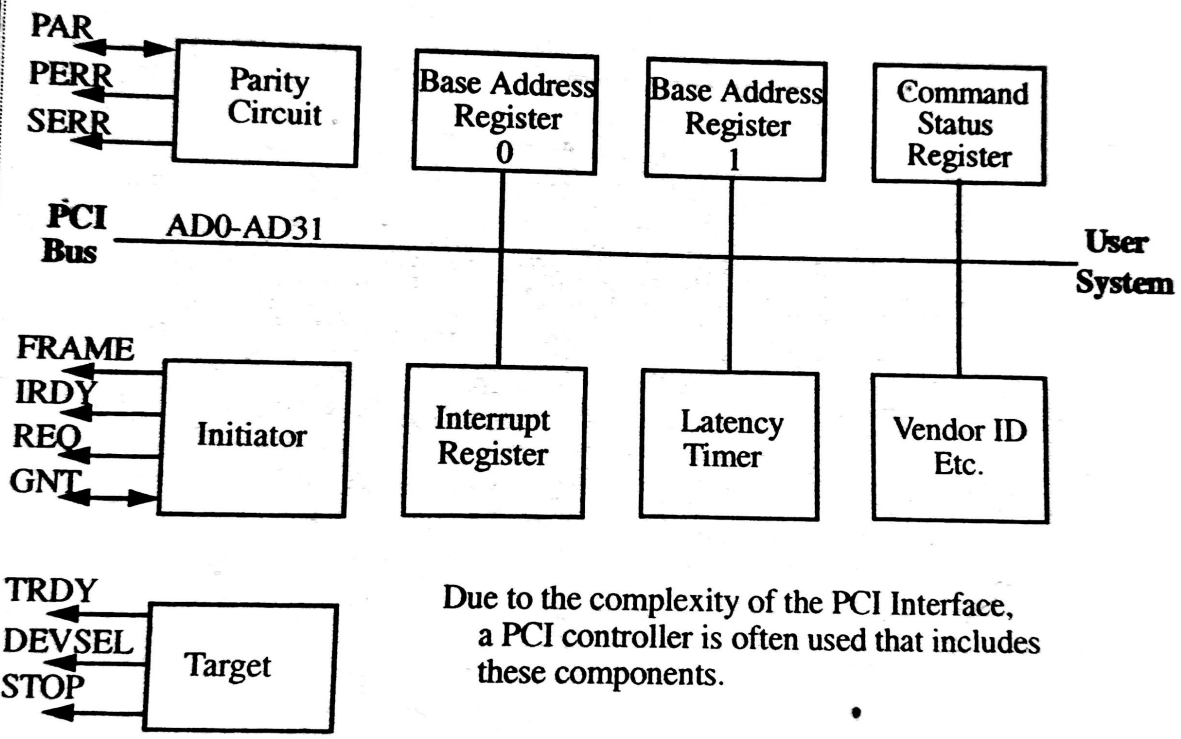


↳ Parallel i/p o/p port, and bi-directional port for Connecting the device.



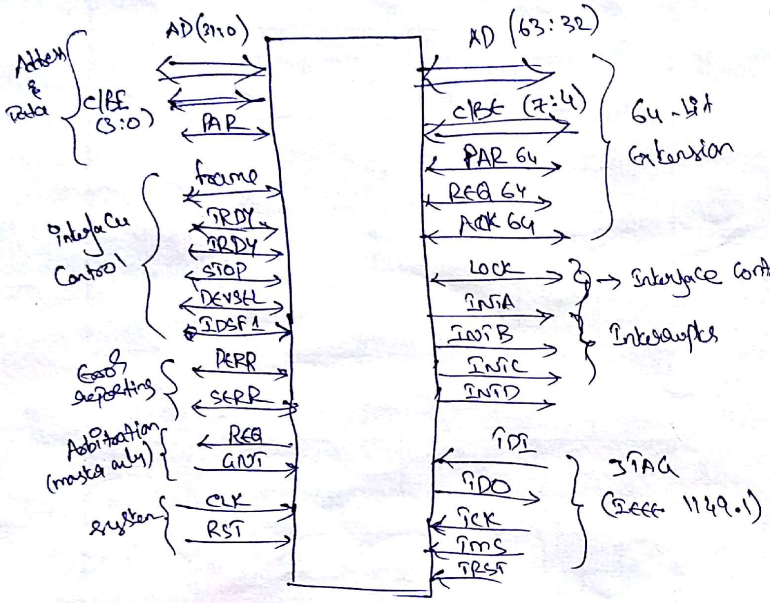
Strobe :- is a signal that is sent that validates data (or) other signals on adjacent parallel lines.

PCI Interface Block Diagram



Due to the complexity of the PCI Interface, a PCI controller is often used that includes these components.

PCI interface:-



CLK:- Clean signal derived from clock generator (33MHz & 66 MHz)

RST → Active low Asynchronous reset

PAR → Parity signal to ensure the parity across the AD bus and C/BE

AD - multiplexed address and data lines

C/BE - Command and Byte Enables

Frame - master indicating start/end of transfer.

IRDY - master (initiator) ready

TRDY - target ready.

DEVSEL - target device selected

REQ - Request for bus

GNT - Bus Grant

STOP[1:0] : target asserts to stop the transition in progress

IDSEL : used as chip select

LOCK[1:0] : Atomic semaphore coherently accessed target locked by initiator.

DEVSEL[1:0] :- Asserted by target when the target asserts has decoded its address (if by 6 clock not asserted = master abort)

PCI Configuration Registers:-

- Device ID
- Vendor ID
- Status / Command Reg
- Base Address [0,1,2,3,4,5]
- maximum latency
- Minimum Grant
- Subsystem ID, Subsystem vendor ID.

PCI Command Types [C/BE].

0000 →	INTR ack
0010 →	IO Read
0011 →	IO write
0110 →	memory Read
0111 →	memory write
1010 →	configuration Read
1011 →	" write

JTAG Boundary Scan:-

1) Test Access port
   - → Test clock
   - → Test data in
   - → Test data out
   - → Test mode Select
   - → Test Reset
2) IEEE Standard 1149.1 compliant

PCI Interrupts :-

→ Asynchronous Events
→ 4 - interrupts lines for multifunctional devices
→ Interrupt lines goes to the interrupt Controller to execute the ISR

PCI Bus Protocol - Transfer mechanism :-

→ Configuration read/write
→ IO Read/write
→ Burst
  → Basic form of data transfer
  → Includes one address phase
  → one (or) more data phase

Burst Transfer Mechanism :-

Assert REQ
GNT granted
wait for current transaction to end
Assert frame
Transfer data when both IRDY & TRDY are asserted
De-assert frame during last data phase

various Read transactions :-

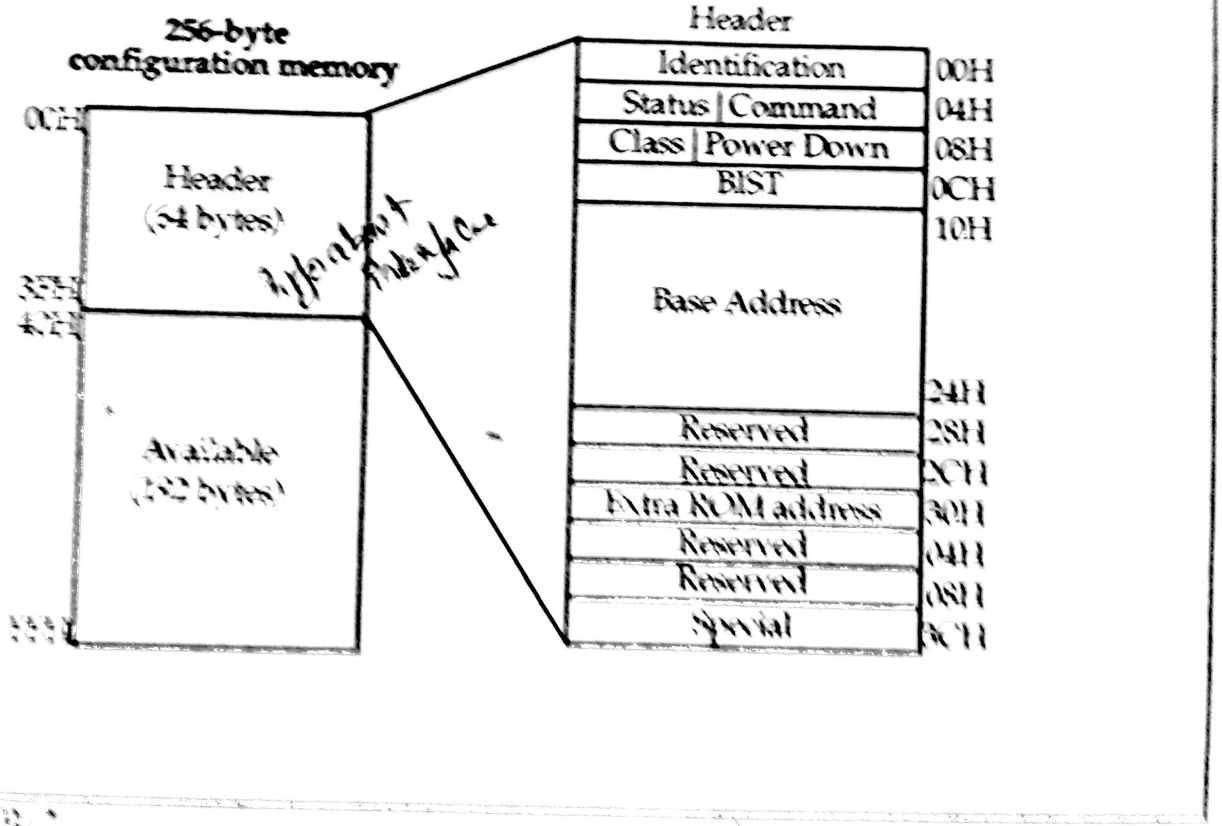
single cycle Read
Burst ~~Read~~ Data Read
Read with no wait states
Byte Enables can be changed for every data cycle
Data Cycle with no byte Enables.

PCI Bus Commands

The following commands can appear on the C/$\overline{BE}$ pins in cycle T1.

- **INTA Sequence**: Get the interrupt vector from the interrupt controller. The interrupt vector byte is returned during a read operation.
- **Special Cycle**: Used to transfer data to all PCI components, e.g. processor shutdown.
- **I/O Read Cycle:** Data are read from an I/O device at address AD0-AD15.
- **I/O Write Cycle:** Data are written to an I/O device.
- **Memory Read Cycle**: Data are read from memory device.
- **Memory Write Cycle:** Data are written to memory device.
- **Configuration Read:** Configuration information is read from PCI device
- **Configuration Write:** Configuration information is written to PCI device.
- **Memory Multiple Access:** Multiple data are read from memory device.
- **Dual Addressing Cycle:** Used for transferring data to a 64-bit PCI device which only contains a 32-bit data path.
- **Line Memory Access:** Used to read more than two 32-bit numbers.
- **Memory Write with Invalidation:** Same as line memory access, but used with write and bypasses write-back function of the cache.

PCI Bus Configuration Space



PCI Bus Configuration Space

The PCI interface contains 256-byte configuration memory that allows plug-and-play feature.

The header holds information about the PCI interface.

The header contains the *unit ID*, *vendor ID*, *class code* and manufacturer defined bits. The vendor ID and class ID are allocated by PCI SIG.

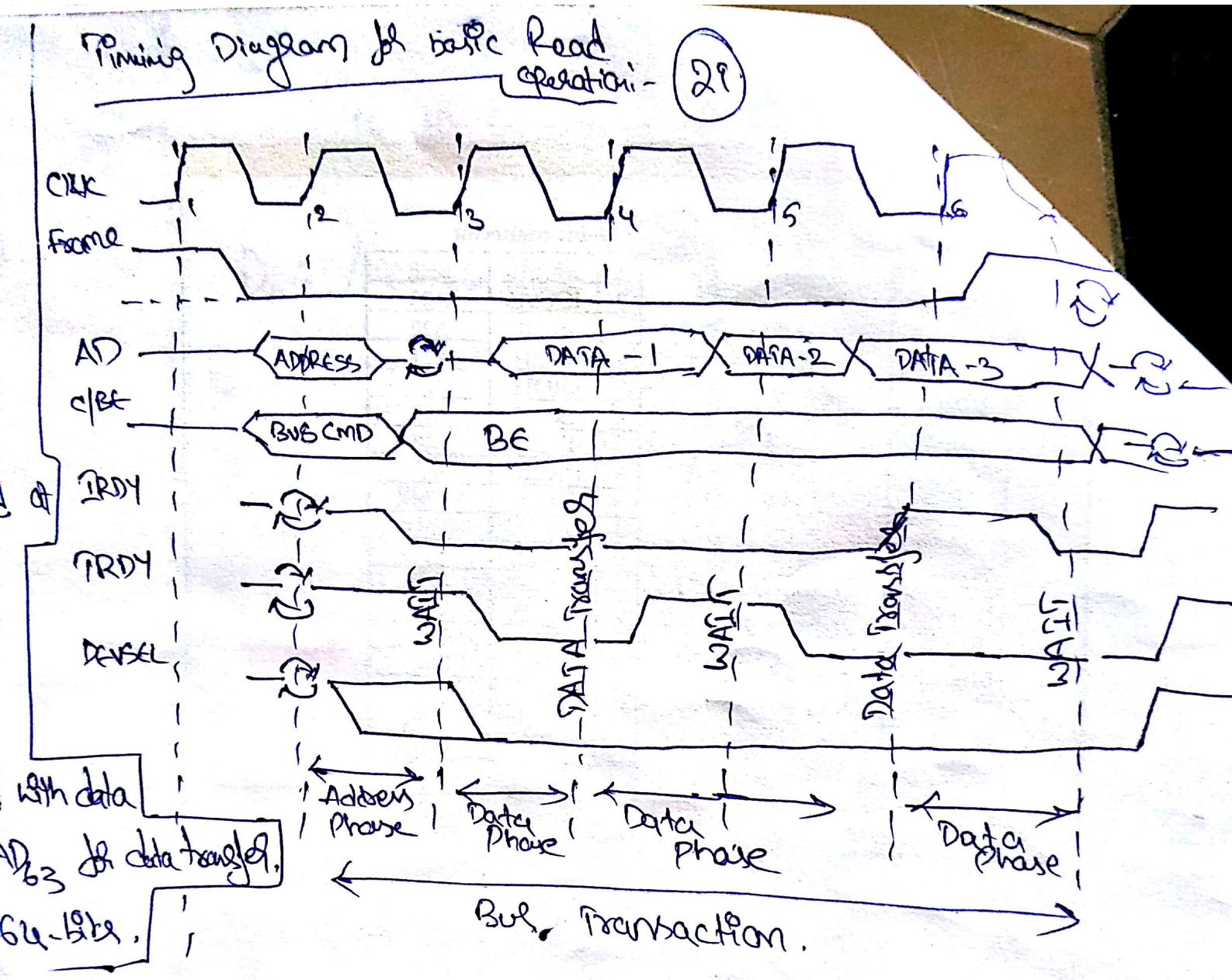
The base address space consists of a base address for the memory, a second for the I/O space and the third for the expansion ROM.

When a PCI bus is present, the system BIOS is extended to support it.
Access to this extended BIOS is through interrupt vector 1AH.
(See text for the currently available functions.)

Once the presence of the BIOS is established, the contents of the configuration memory can be read using other BIOS functions.

→ first 64 bytes contains information about PCI interface
→ the first 32-bit doubled word contain the unit ID code & vendor ID code.

Timing Diagram for basic Read operation:- (21)

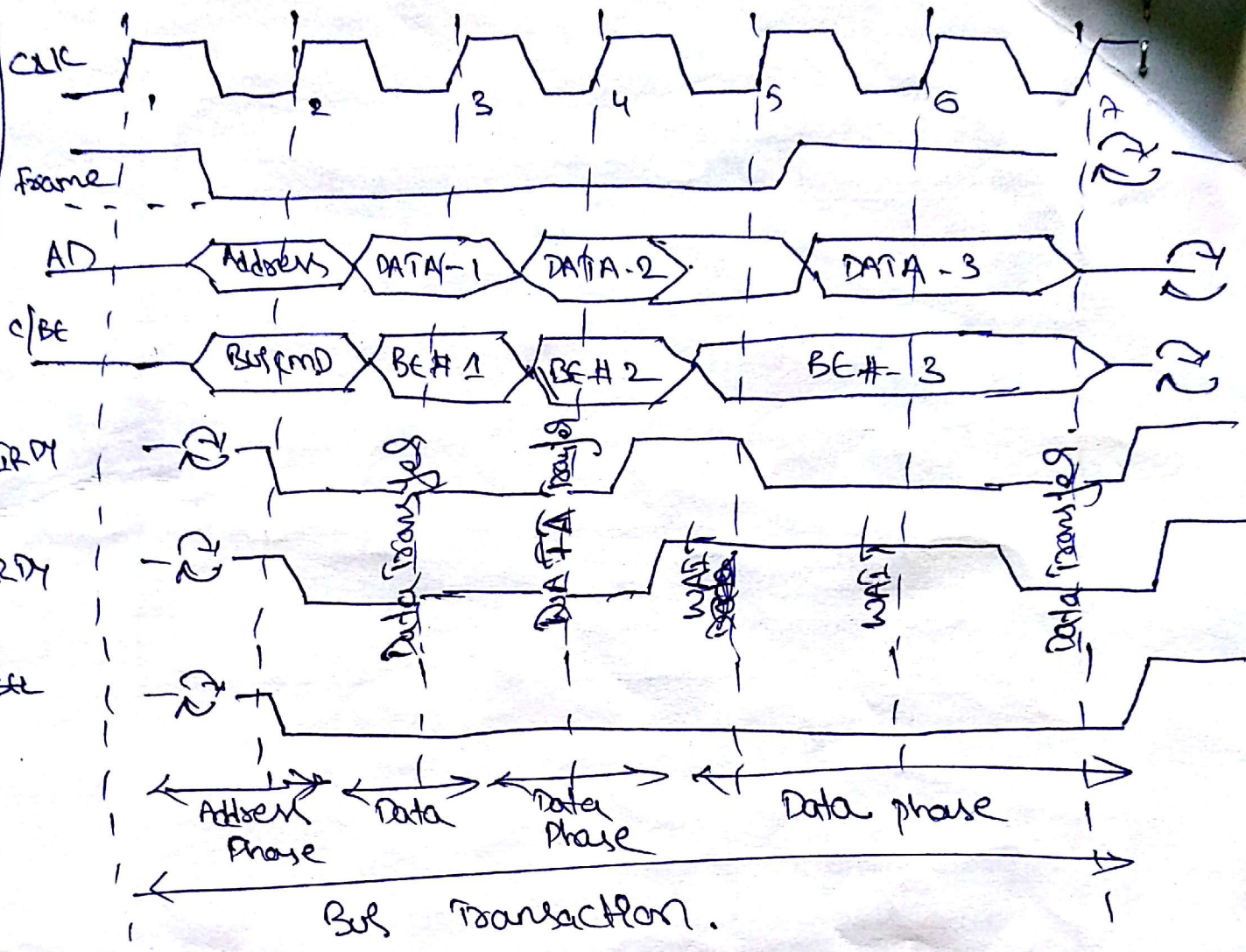


with data

AD_{63} for data transfer.

64-bits.

Basic write operation:-



Transaction Termination:-

Last data Phase Complete when

!Frame & IRDY (normal - master)

" & STOP (target termination)

" & Device Select Timer Expires (master abort)

DEVSEL : & STOP (Target abort).