

## 2.1 Core of Embedded Systems


- The core of the embedded system falls into any of the following categories:

1. General Purpose and Domain Specific Processors
2. Application Specific Integrated Circuits(ASICs)
3. Programmable Logic Devices (PDLs)
4. Commercial Off the shelf components(COTS)

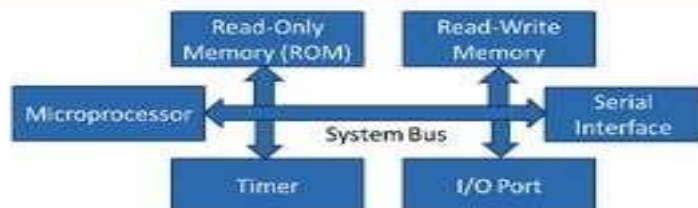


# Merits, Drawbacks and Application Areas of Microcontrollers and Microprocessors

- Microcontrollers are designed to perform **specific tasks**. However, Microprocessors are designed to perform **unspecific tasks** like developing software, games, website, photo editing, creating documents, etc.
- Depending on the input, some processing for microcontroller needs to be done and output is **defined**. However, the relationship between input and output for microprocessor is **not defined**.
- Since the applications of microcontroller are very specific, they need **small resources** like RAM, ROM, I/O ports etc. and hence can be embedded on **a single chip**. Microprocessors need **high amount of resources** like RAM, ROM, I/O ports etc.
- The clock speed of Microprocessor is **quite high** as compared to the microcontroller. Whereas the microcontrollers operate from a **few MHz** (from 30 to 50 MHz), today's microprocessor operate above **1 GHz** as they perform **complex tasks**.

- 
- **Microprocessor cannot be used stand alone.** They need other peripherals like RAM, ROM, buffer, I/O ports etc and hence a system designed around a microprocessor is **quite costly**.
  - **Application areas of microcontroller:** Mobile phones, CD/DVD players, Washing machines, Cameras, Security alarms, microwave oven, etc.
  - **Application areas of microprocessor:** Calculators, Accounting Systems, Games Machine, Complex Industrial Controllers, Data Acquisition Systems, Military applications, Communication systems, etc.

## Microprocessor



## Micro Controller



Microprocessor is heart of Computer system.

Micro Controller is a heart of embedded system.

It is just a processor. Memory and I/O components have to be connected externally

Micro controller has external processor along with internal memory and i/O components

Since memory and I/O has to be connected externally, the circuit becomes large.

Since memory and I/O are present internally, the circuit is small.

Cannot be used in compact systems and hence inefficient

Can be used in compact systems and hence it is an efficient technique

Cost of the entire system increases

Cost of the entire system is low

Due to external components, the entire power consumption is high. Hence it is not suitable to used with devices running on stored power like batteries.

Since external components are low, total power consumption is less and can be used with devices running on stored power like batteries.

Most of the microprocessors do not have power saving features.

Most of the micro controllers have power saving modes like idle mode and power saving mode. This helps to reduce power consumption even further.

Since memory and I/O components are all external, each instruction will need external operation, hence it is relatively slower.

Since components are internal, most of the operations are internal instruction, hence speed is fast.

Microprocessor have less number of registers, hence more operations are memory based.

Micro controller have more number of registers, hence the programs are easier to write.

Microprocessors are based on von Neumann model/architecture where program and data are stored in same memory module

Micro controllers are based on Harvard architecture where program memory and Data memory are separate

Mainly used in personal computers

Used mainly in washing machine, MP3 players

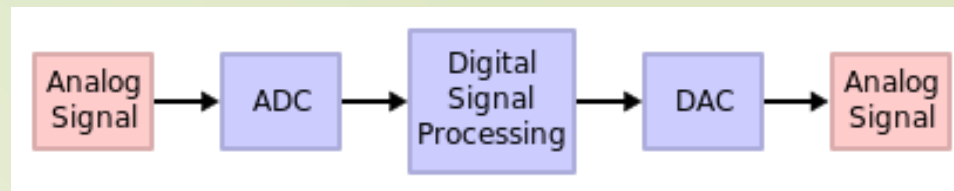




# Digital Signal Processors

- DSPs are **powerful special purpose 8/16/32 bit microprocessors** designed specifically to meet the **computational demands and power constraints** of today's embedded audio, video, and communications applications.
- Digital signal processors are **2 to 3 times faster** than the general purpose microprocessors in signal processing applications.
- A typical digital signal processor incorporates the following key units:
  - i. **Program Memory** : Memory **for storing the program** required by DSP to process the data.
  - ii. **Data Memory** : Working memory **for storing temporary variables/information and data/signal** to be processed.
  - iii. **Computational Engine** : **Performs the signal/math processing** , accessing the program from the Program Memory and the data from the Data Memory.
  - iv. **I/O Unit** : Acts as **an interface between the outside world and DSP**.

- **Application areas** : Audio video signal processing, telecommunication and multimedia applications.
- DSP employs **a large amount of real-time calculations, Sum of products (SOP) calculation, convolution, fast Fourier transform (FFT), discrete Fourier transform (DFT), etc.** are some of the operations performed by digital signal processors.



# RISC vs CISC Processors/Controllers

RISC	CISC
Lesser number of instructions	Greater number of Instructions
Instruction pipelining and increased execution speed	Generally no instruction pipelining feature
Orthogonal instruction set	Non-orthogonal instruction set
Operations are performed <b>on registers only</b> , the only memory operations are load and store.	Operations are performed <b>on registers or memory</b> depending on the instruction.
A large number of registers are available.	Limited number of general purpose registers.
Programmer <b>needs to write more code to execute a task</b> since the instructions are simpler ones.	Instructions are like macros in <b>C language</b> . A programmer can achieve the desired functionality with a single instruction which in turn provides the effect of using <b>more simpler single instructions</b> in RISC.

## Cont'd

RISC	CISC
Single, fixed length instructions	Variable length instructions
Less silicon usage and pin count	More silicon usage since more additional decoder logic is required to implement the complex instruction decoding.
With Harvard Architecture	Can be Harvard or Von-Neumann Architecture



VS





# Big-Endian vs. Little-Endian Processors/Controllers

- **Endianness** specifies the **order** in which **a sequence of bytes** are stored in computer memory.
- **Little-endian** is an order in which the “little end”/ the **lower-order byte of the data (least significant value in the sequence)** is stored in memory at the lowest address. (The little end comes first.)
- For example, a 4 byte long integer **Byte3, Byte2, Byte1, Byte0** will be stored in the memory as shown below:

Base Address+0

Byte 0

Byte 0

0x20000 (Base Address)

Base Address+1

Byte 1

Byte 1

0x20001 (Base Address+1)

Base Address+2

Byte 2

Byte 2

0x20002 (Base Address+2)

Base Address+3



Byte 3

Byte 3

0x20003 (Base Address+3)

Example : **90AB12CD** (Hexadecimal)

Address	Value
1000	CD
1001	12
1002	AB
1003	90

- 
- **Big-endian** is an order in which the “big end” / **the higher-order byte of the data (most significant value in sequence)** is stored in memory at the lowest address. (The big end comes first.)
  - For example, a 4 byte long integer **Byte3, Byte2, Byte1, Byte0** will be stored in the memory as shown below:
- 

Base Address+0

Byte 3

Byte 3

0x20000 (Base Address)

Base Address+1

Byte 2

Byte 2

0x20001 (Base Address+1)

Base Address+2

Byte 1

Byte 1

0x20002 (Base Address+2)

Base Address+3

Byte 0

Byte 0

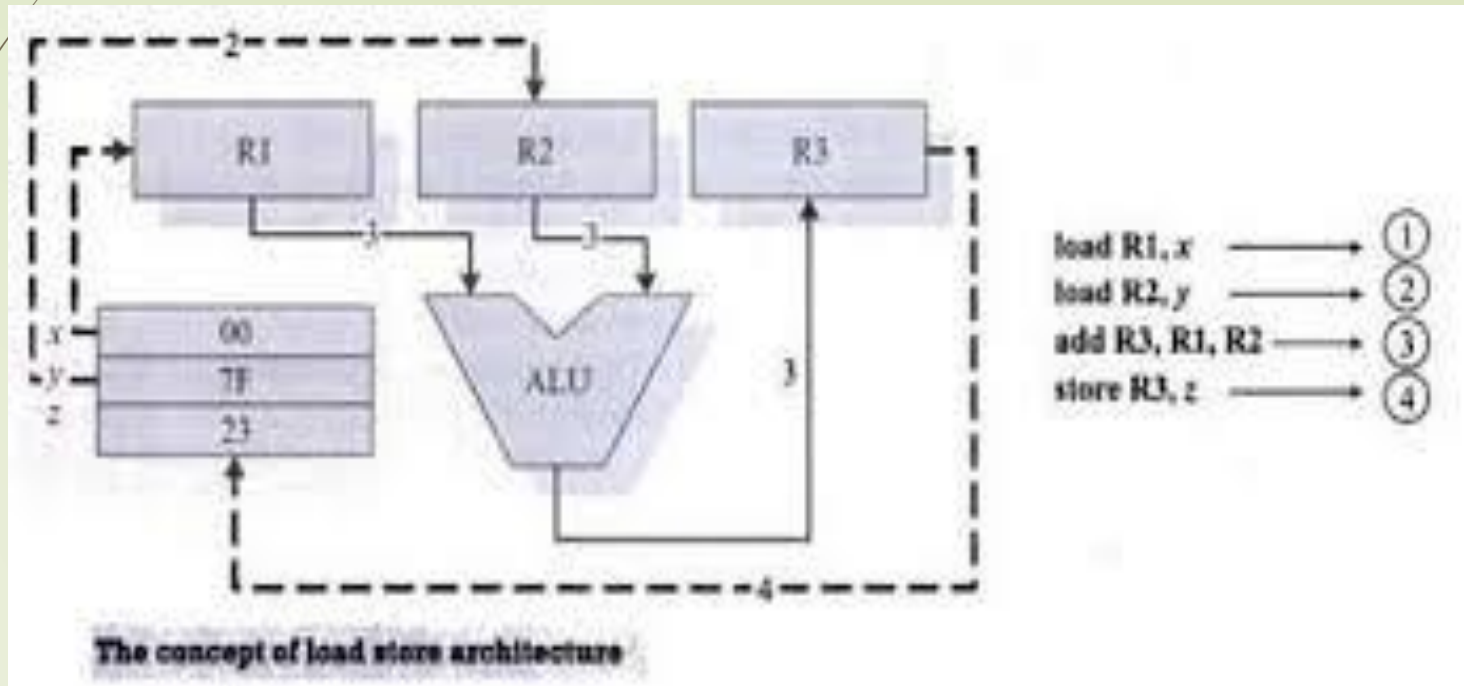
0x20003 (Base Address+3)

Example : **90AB12CD** (Hexadecimal)

Address	Value
1000	90
1001	AB
1002	12
1003	CD

# Load Store Operation and instruction pipelining

- RISC processor instruction set is **orthogonal**.
- **Memory access** related **operations** are performed by special instructions **load** and **store**.
- If operand is specified as memory location, content of it is loaded to register using **load instruction**.
- The **instruction store** stores data from a specified register to a specified memory location.



# Application Specific Integrated Circuits (ASICs)

- **ASIC** is microchip designed to **perform a specific or unique application**.
- It integrates several functions into a single chip and there by **reduces the system development cost**.
- Most of the **ASICs are proprietary products**.
- As a single chip, **ASICs consumes a very small area** in total system and thereby helps in design of smaller systems with high functionalities.
- ASICs based systems are profitable only for larger volume commercial productions.


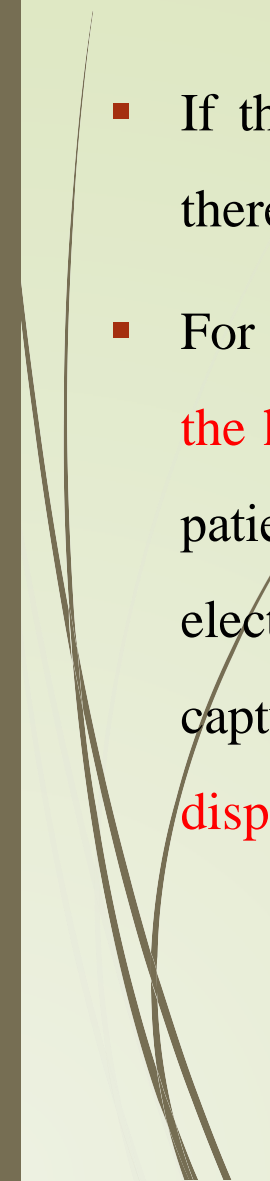


# Programmable Logic Devices(PLD)

- **Logic devices** provide specific functions, including device-to-device interfacing, data communication, signal processing, data display, time and control operations and almost every other operation a system must perform.
- **Logic Devices are classified as fixed and programmable.**
  - Circuits in a **fixed logic device** are permanent, they perform function or set of functions – once manufactured, they **cannot be changed**.
  - **PLDs can be reconfigured** to perform any number of function at any time.
- With PLDs , designers use inexpensive software tools to quickly develop, simulate and test their designs.
- Another benefit of PLDs is that during the design phase customer can change the circuitry as often as they want until the design operates to their satisfaction. That is because **PLDs are based on re-writable memory technology** - to change the design, the device is simply reprogrammed.
- Applications – different types of registers and counters, code converters, parity checkers, checksums and error correction and error detection.

# Sensors and Actuators

- A **sensor** is a transducer device that **converts energy from one form to another** for any measurement or control purpose.
  - The changes in system environment or variables **are detected** by the sensors **connected to the input port** of the embedded system.
- **Actuator** is a form of transducer device (mechanical or electrical) which **converts signals to corresponding physical action (motion)**. Actuator acts as an output device.
  - If the embedded system is designed for any controlling purpose, the system **will produce some changes** in the controlling variable **to bring the controlled variable to the desired value**.
  - It is achieved through an actuator connected to the output port of the embedded system.

- 
- If the embedded system is **designed for monitoring purpose** only, then there is **no need for including an actuator** in the system.
  - For example, take the case of an **ECG machine**. It is **designed to monitor the heart beat status of a patient** and it cannot impose a control over the patient's heart beat and its order. The **sensors used** here are the different electrode sets connected to the body of the patient. The variations are captured and presented to the user (may be a doctor) through **a visual display or some printed chart**.
- 

# I/O Subsystem

- **The I/O subsystem** of the embedded system facilitates the interaction of the embedded system with the external world.
- The interaction happens through **the sensors and actuators connected to the input and output ports** respectively of the embedded system.
- The **sensors** may not be directly interfaced to the input ports, instead they may be interfaced through signal conditioning and translating systems **like ADC, optocouplers, etc.**



## Cont'd

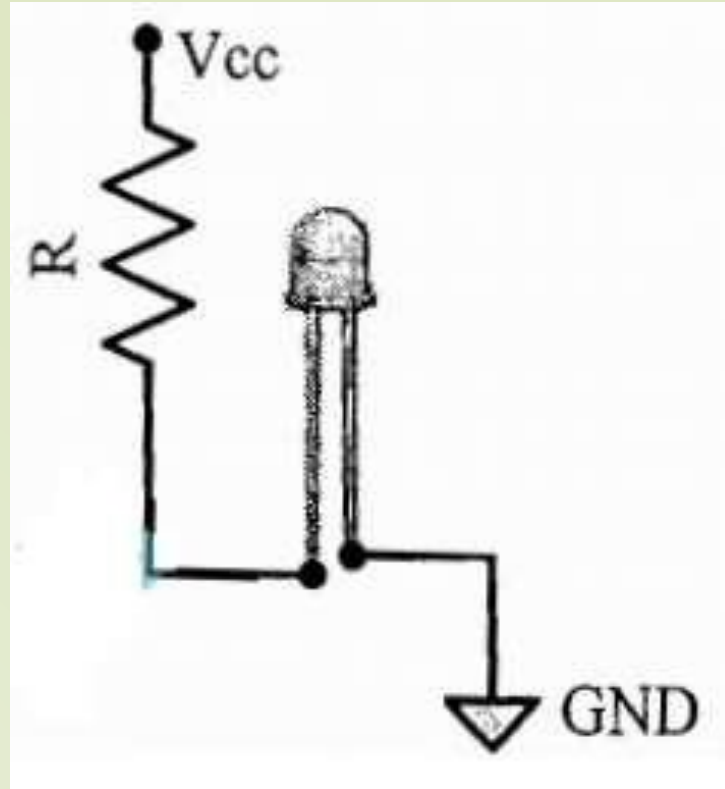
- Light Emitting Diode (LED)
- 7-Segment LED Display
- Optocoupler
- Stepper Motor
- Relay
- Piezo Buzzer
- Push Button Switch
- Keyboard
- Programmable Peripheral Interface (PPI)

# Light Emitting Diode (LED)

- **LED** is an important **output device for visual indication** in any embedded system. LED **can be used as an indicator** for the status of various signals or situations. Typical examples are **indicating the presence of power conditions** like 'Device ON', 'Battery low' or 'Charging of battery' for a battery operated handheld embedded devices.
- LED is a **p-n junction diode** and it contains **an anode and a cathode**. For proper functioning of the LED, the anode of it should be connected to **+ve terminal of the supply voltage** and cathode to the **-ve terminal** of the supply voltage. The current flowing through the LED **must be limited** to a value below the **maximum current** that it can conduct. A **resistor** is used in **series between the power supply and the LED** to limit the current through the LED.

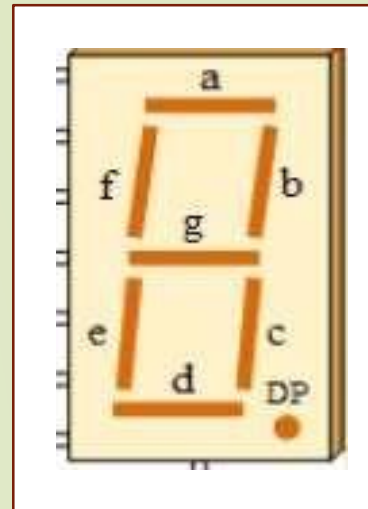






# 7- Segment LED Display

- The **7-segment LED display** is an output device for displaying alphanumeric characters. It contains **8 light-emitting diode (LED) segments** arranged in a special form. Out of the 8 LED segments, **7** are used for **displaying alphanumeric characters** and **1** is used for representing 'decimal point' in decimal number display.
- The **LED segments** are named **A to G** and the **decimal point** LED segment is named as **DP**.



## Cont'd

- The 7-segment LED displays are available in **two different configurations**, namely; **Common Anode** and **Common Cathode**. In the common anode configuration, the anodes of the 8 segments are connected commonly whereas in the common cathode configuration, the 8 LED segments share a common cathode line.

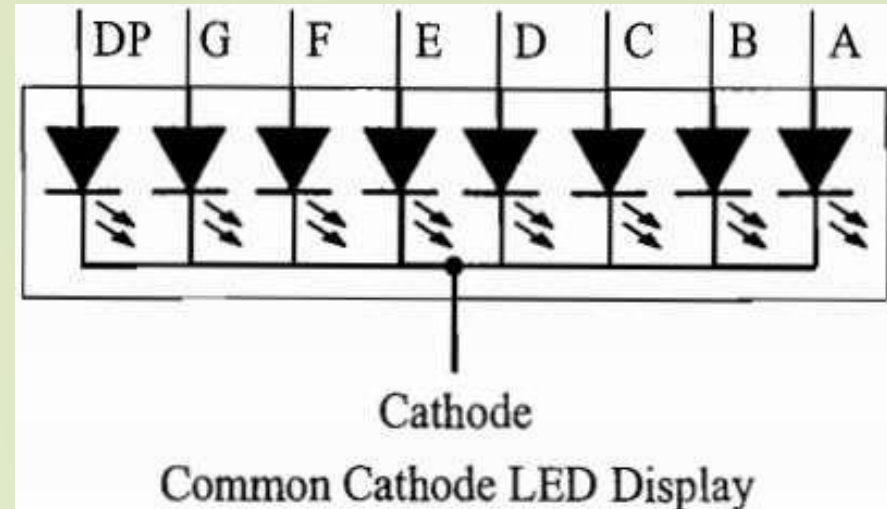
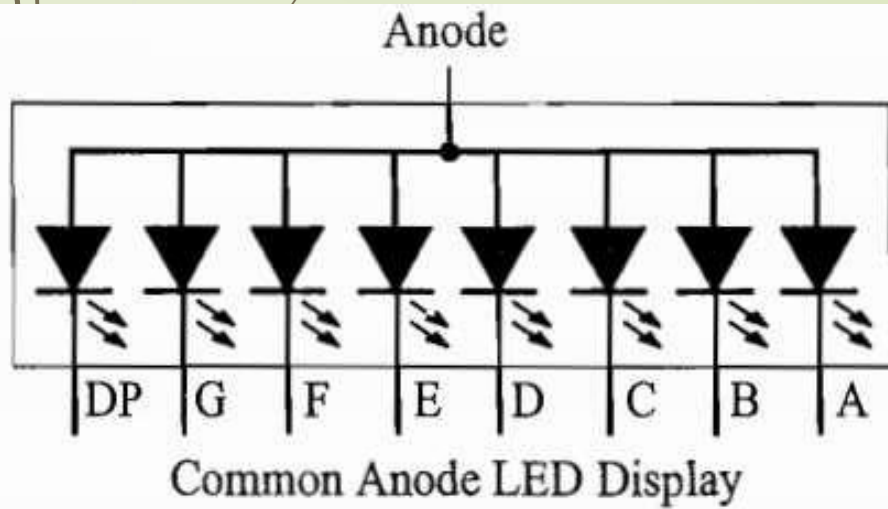
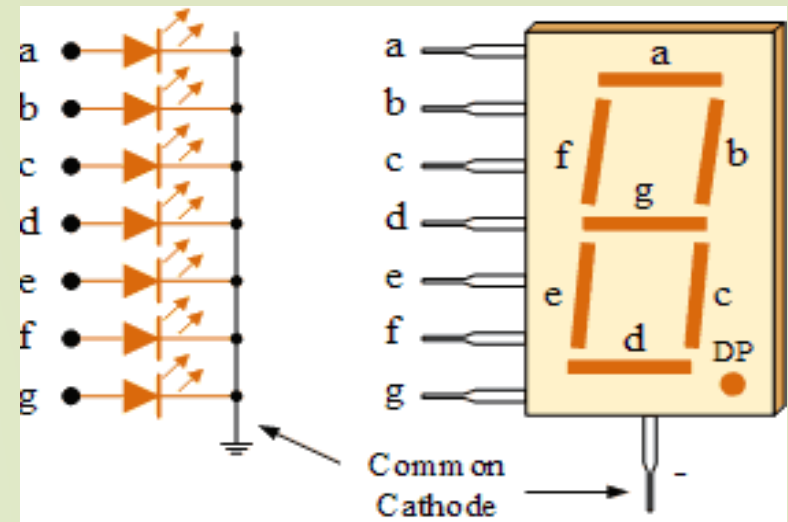
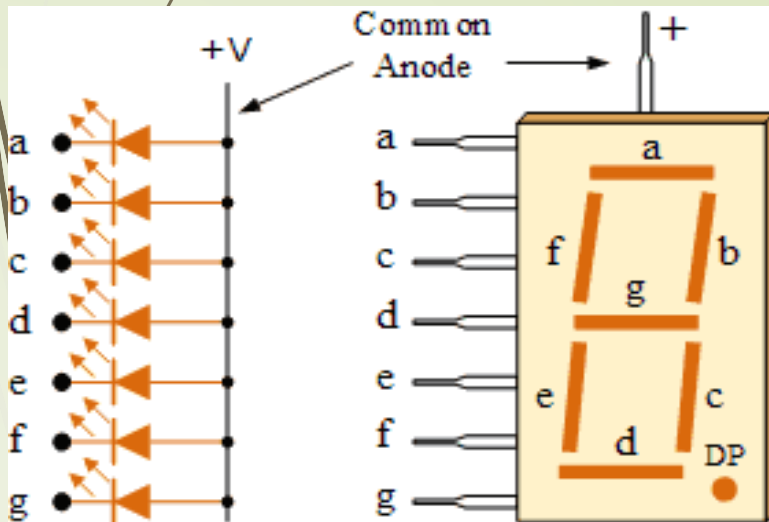
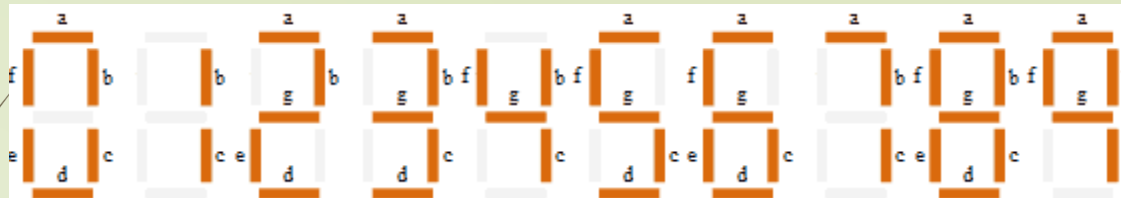


Figure Common anode and cathode configurations of a 7-segment LED Display

# Cont'd

- 7-segment LED display is a popular choice for low cost embedded applications like, Public telephone call monitoring devices, point of sale terminals, etc.





# Optocoupler

- Optocoupler is a solid state device to isolate two parts of a circuit.
- Optocoupler combines an LED and a photo-transistor in a single housing (package). Figure illustrates the functioning of an optocoupler device.
- Figure illustrates the functioning of an optocoupler device.

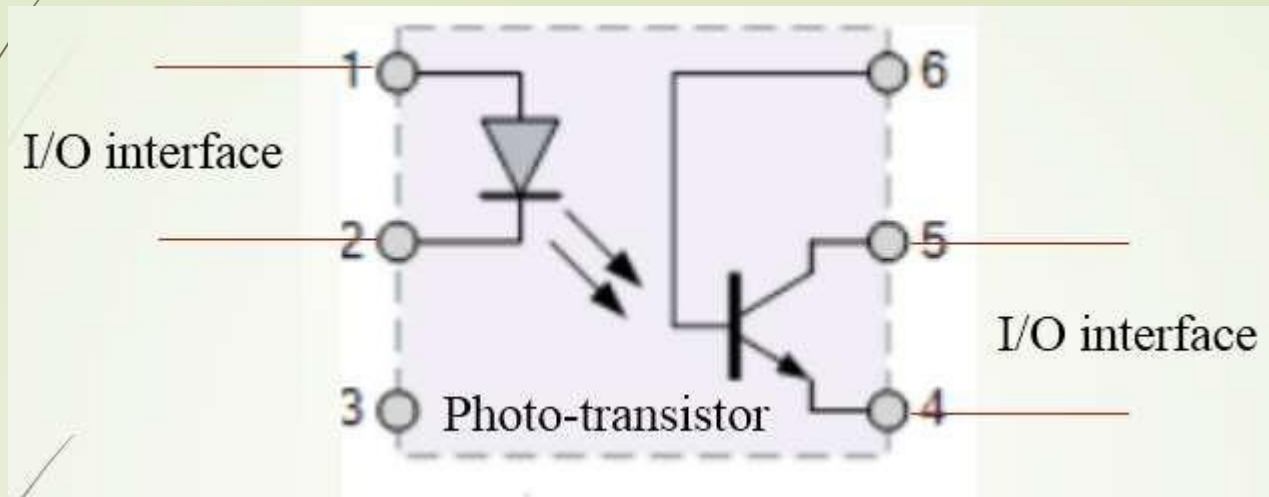


Figure. An optocoupler device

## Cont'd

- In electronic circuits, an optocoupler is used for suppressing interference in data communication, circuit isolation, high voltage separation, simultaneous separation and signal intensification, etc.
- Optocouplers can be used in either input circuits or in output circuits.
- Figure illustrates the usage of optocoupler in input circuit and output circuit of an embedded system with a microcontroller as the system core.

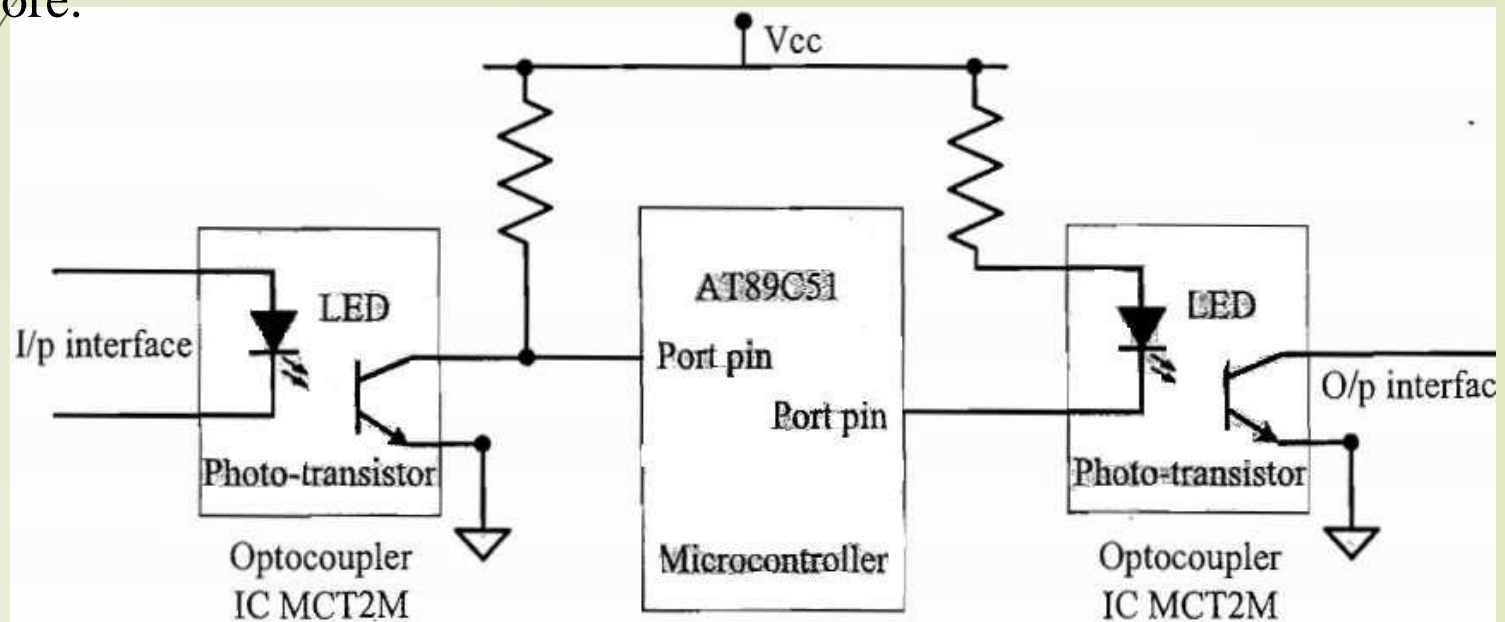


Figure. Optocoupler in Input and Output Circuit

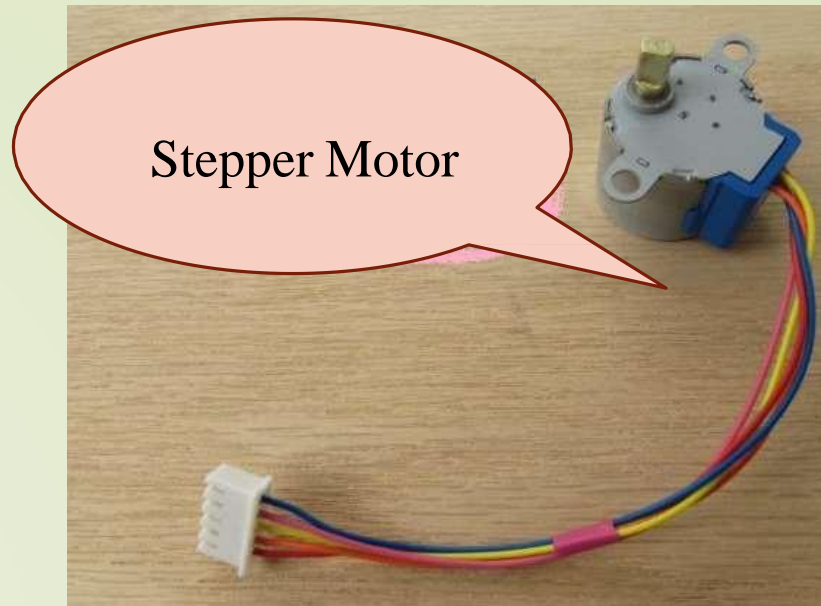


# Stepper Motor

- A stepper motor is an **electro-mechanical device** which generates discrete displacement (motion) in response to dc electrical signals.
- It **differs from the normal dc motor** in its operation.
- The dc motor produces **continuous rotation** on applying dc voltage whereas a stepper motor produces **discrete rotation** in response to the dc voltage applied to it.
- Stepper motors are widely used in industrial embedded applications, **consumer electronic products** and **robotics control systems**.
- The paper feed mechanism of a **printer/fax** makes use of stepper motors for its functioning.

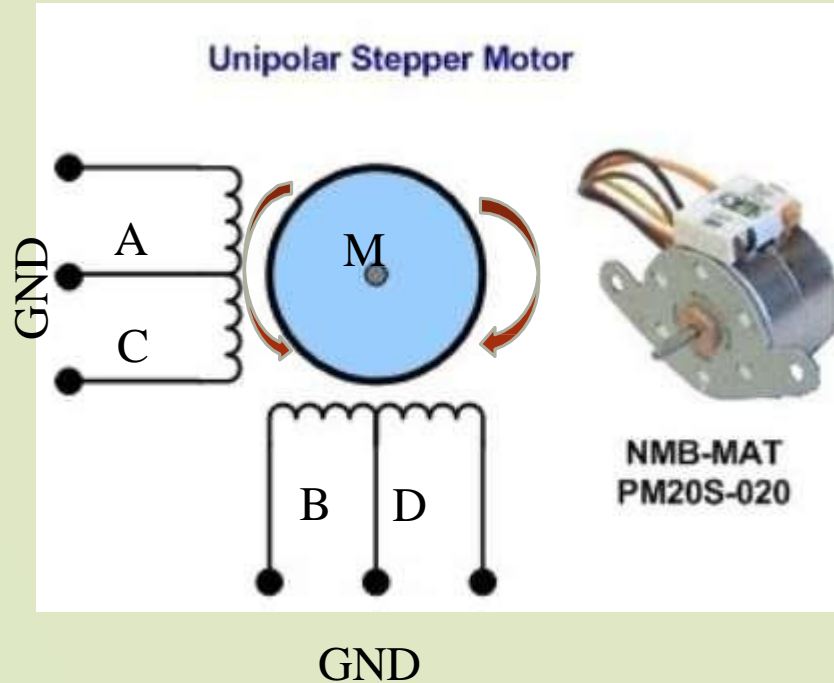
## Cont'd

- Based on the coil winding arrangements, a **two-phase stepper motor** is classified into two. They are:
  - Unipolar
  - Bipolar



# Unipolar

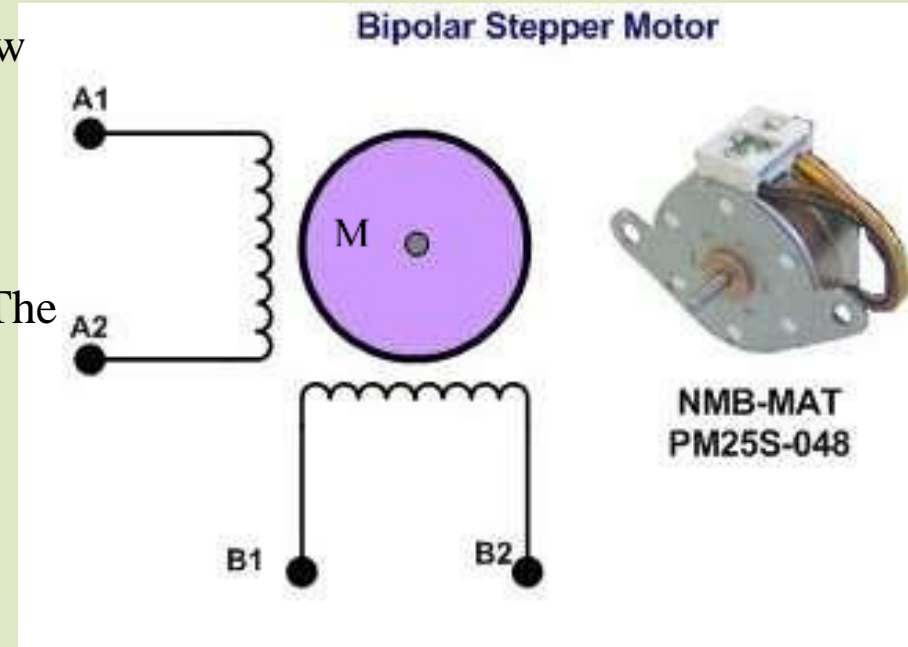
- A unipolar stepper motor contains **two windings per phase**.
- The direction of rotation (clockwise or anticlockwise) of a stepper motor is **controlled by changing the direction of current flow**.
- **Current in one direction** flows through one coil and in **the opposite direction** flows through the other coil.
- It is **easy to shift the direction of rotation** by just switching the terminals to which the coils are connected.
- Figure illustrates the working of a two-phase unipolar stepper motor.



2-Phase unipolar stepper motor

# Bipolar

- A bipolar stepper motor contains **single winding per phase**.
- For **reversing the motor rotation** the current flow through the windings is reversed dynamically.
- It requires **complex circuitry for current flow reversal**.
- There is **one disadvantage** of unipolar motors. The torque generated by them is **quite less**. This is because the current is flowing **only through the half the winding**. Hence they are used in **low torque applications**.
- On the other hand, bipolar stepper motors are a **little complex to wire** as we have to use a **current reversing H bridge driver IC like an L293D**. But the advantage is that the **current will flow through the full coil**. The resulting torque generated by the motor is **larger** as compared to a uni-polar motor.



## Cont'd

- The stepping of stepper motor can be implemented in different ways by changing the sequence of activation of the stator windings. The **different stepping modes supported by stepper motor** are explained below.
- **Full Step:** In the step mode both the phases are energized **simultaneously**. The coils A, B, C and D are energized in the following order:

Step	Coil A	Coil B	Coil C	Coil D
1	H	H	L	L
2	L	H	H	L
3	L	L	H	H
4	H	L	L	H

- It should be noted that out of the two windings, only one winding of a phase is energized at a time.

## Cont'd

- **Wave Step:** In the wave step mode only one phase is energized at a time and each coils of the phase is energies alternatively. The coils A, B, C and D are energized in the following order:

Step	Coil A	Coil B	Coil C	Coil D
1	H	L	L	L
2	L	H	L	L
3	L	L	H	L
4	L	L	L	H



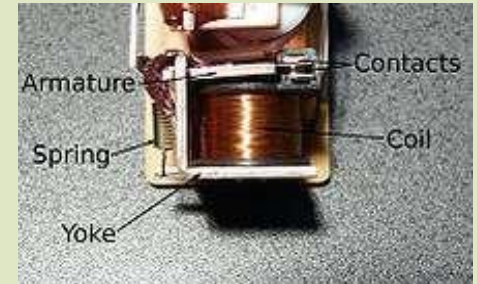
## Cont'd

- **Half Step** : It uses the combination of wave and full step. It has the highest torque and stability. The coil energizing sequence for half step is given below.

Step	Coil A	Coil B	Coil C	Coil D
1	H	L	L	L
2	H	H	L	L
3	L	H	L	L
4	L	H	H	L
5	L	L	H	L
6	L	L	H	H
7	L	L	L	H
8	H	L	L	H



# Relay



- Relay is an **electro-mechanical device**. In embedded application, the 'Relay' unit acts as **dynamic path selectors** for signals and power.
- The 'Relay' unit contains **a relay coil** made up of **insulated wire** on a metal core and **a metal armature** with one or more contacts.
- 'Relay' works on **electromagnetic principle**. When a **voltage is applied to the relay coil**, **current flows through the coil**, which in turn generates **a magnetic field**.
- The magnetic field attracts **the armature core** and moves **the contact point**. The **movement of the contact point** changes the power/signal flow path.
- 'Relays' are available in different configurations. Figure given below illustrates the widely used relay configurations for embedded applications.

# Cont'd

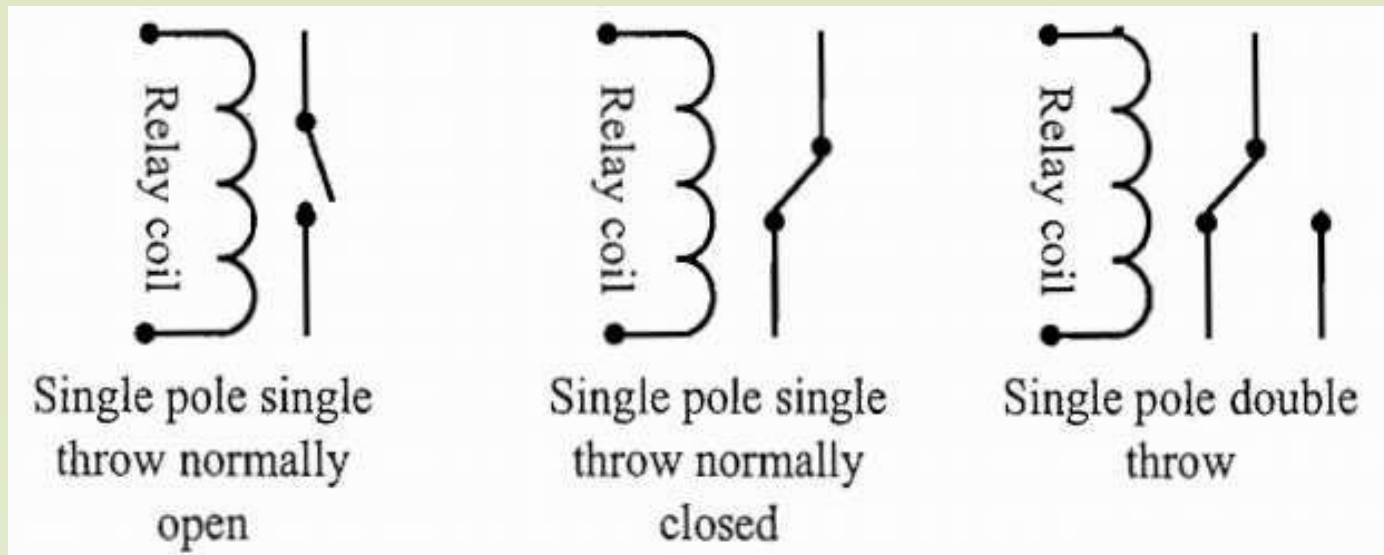


Figure Relay Configurations

- The **Single Pole Single Throw** configuration has **only one path** for information flow. The path is either open or closed in normal condition. For normally **Open** Single Pole Single Throw relay, the circuit is normally open and it **becomes closed when the relay is energized**. For normally **closed** Single Pole Single Throw configuration, the circuit is normally closed and it **becomes open when the relay is energized**. For **Single Pole Double Throw Relay**, there are **two paths** for information flow and they are selected by energizing or de-energizing the relay.

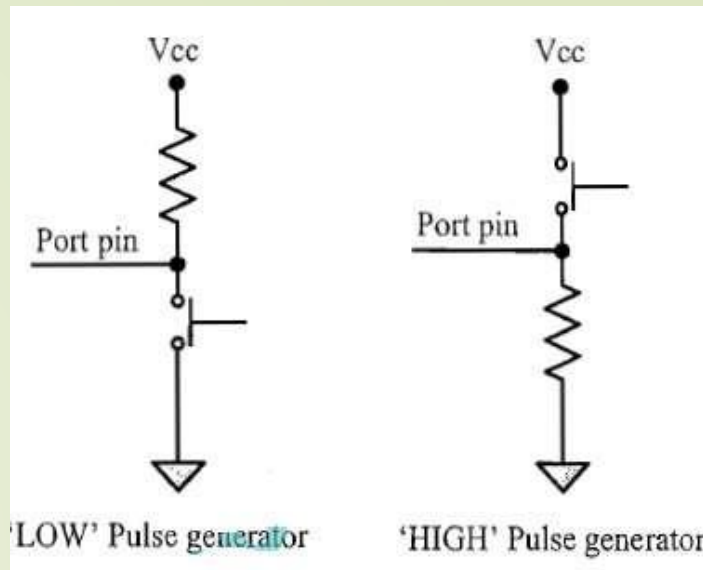
# Piezo Buzzer

- **Piezo buzzer** is a piezoelectric device **for generating audio indications** in embedded application.
- A piezoelectric buzzer contains a piezoelectric diaphragm **which produces audible sound** in response to the voltage applied to it.
- Piezoelectric buzzers are available in **two types**: ‘Self-driving’ and ‘External driving’.
- **The ‘Self-driving’ circuit** contains all the necessary components **to generate sound at a predefined tone**. It will generate **a tone** on applying the voltage.
- **External driving piezo buzzers** supports the **generation of different tones**. The **tone can be varied** by applying a variable pulse train to the piezoelectric buzzer.



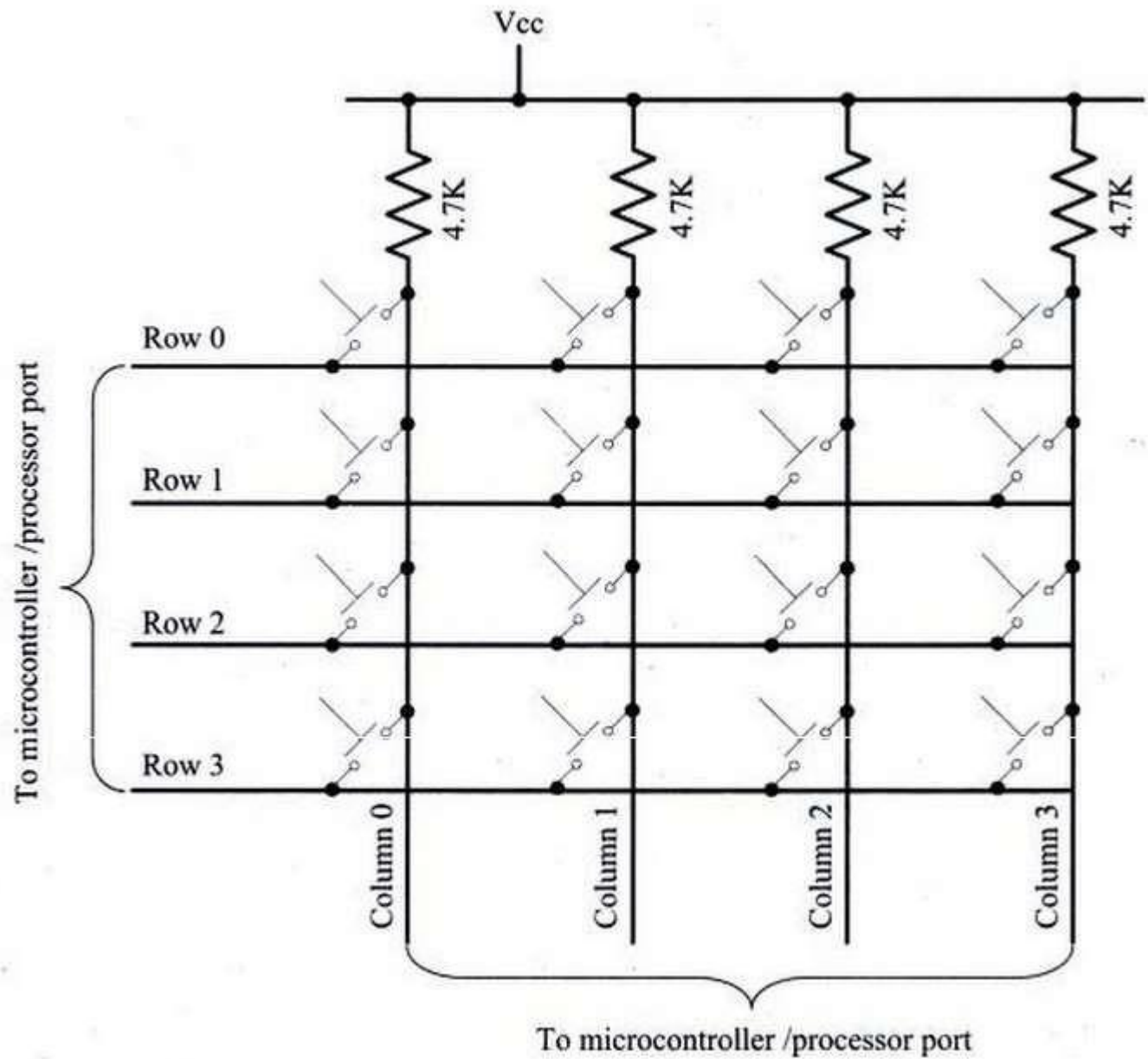
# Push Button Switch

- It is an **input device**. Push button switch comes in **two configurations**, namely 'Push to Make' and 'Push to Break'.
- **In the 'Push to Make' configuration**, the switch is **normally in the open state** and it **makes a circuit contact** when it is pushed or pressed.
- **In the 'Push to Break' configuration**, the switch is **normally in the closed state** and it **breaks the circuit contact** when it is pushed or pressed.
- In the embedded application push button is generally used **as reset** and **start switch**.



# Keyboard

- Keyboard is an **input device** for user interfacing. **If** the number of keys required is **very limited**, **push button switches** can be used and they can be directly interfaced to the port pins for reading.
- **Matrix keyboard** is an optimum solution for handling large key requirements. It greatly reduces the number of interface connections.
- For example, **for interfacing 16 keys**, in the direct interfacing technique **16 port pins** are required, whereas the matrix keyboard **only 8 lines are required**. The **16 keys** are arranged in a **4 column\*4 row matrix**.



Matrix Keyboard Interfacing

# Programmable Peripheral Interface (PPI)

- PPI devices are used for **extending the I/O capabilities of processor/controller**.
- Most of the processor/controllers provide very limited number of I/O and data ports and it may require more number of I/O ports than one supported by controller/processor.
- 8255A** is a popular PPI device for the controller/processor.
- 8255A supports 24 I/O pins and these I/O pins can be grouped as either three 8 bit parallel ports (Port A, Port B, Port C) or two 8 bit parallel ports (Port A and Port B) with Port C.



Similar to a 4-byte  
RAM

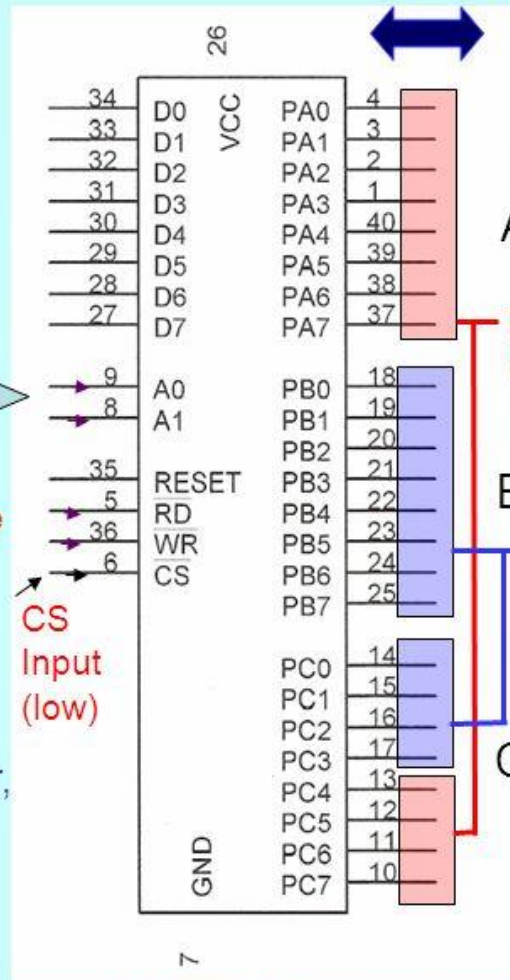
↔  
Data Bus

2-bit Address I/P  
(select port or  
Command register  
for Read or Write)

Read/Write  
Control

On the PC: Two 82C55s  
-One 82C55 occupies  
4 I/O ports 60H-63H:  
Handling Keyboard, timer,  
speaker, etc.

-One 82C55 occupies  
4 I/O ports 378H-37BH  
Parallel printer port



82C55 DIP Version

3 programmable  
8-bit I/O ports: A, B, C

2 Groups  
12-bit groups A, B

Port A + Upper half of C = Group A  
(12 bits)

Port B + lower half of C = Group B  
(12 bits)

RESET initializes the PPI to operate  
in mode 0 & all 3 ports as inputs  
at power up.

With all ports as **input ports**,  
this avoids damage to the device at  
Power up

# Communication Interface

- For an embedded product, the **communication interface** can be viewed **in two different perspectives**; namely; **Device/board level communication interface (Onboard Communication Interface)** and **Product level communication interface (External Communication Interface)**.
- **Embedded product** is a combination of different types of components (chips/devices) arranged on a printed circuit board (PCB). **Serial interfaces** like **I2C, SPI, UART, 1-Wire, etc** and **parallel bus interface** are examples of **‘Onboard Communication Interface’**.

# Onboard Communication Interfaces

- **Onboard Communication Interface** refers to the **different communication channels/buses** for interconnecting the various integrated circuits and other peripherals within the embedded system.

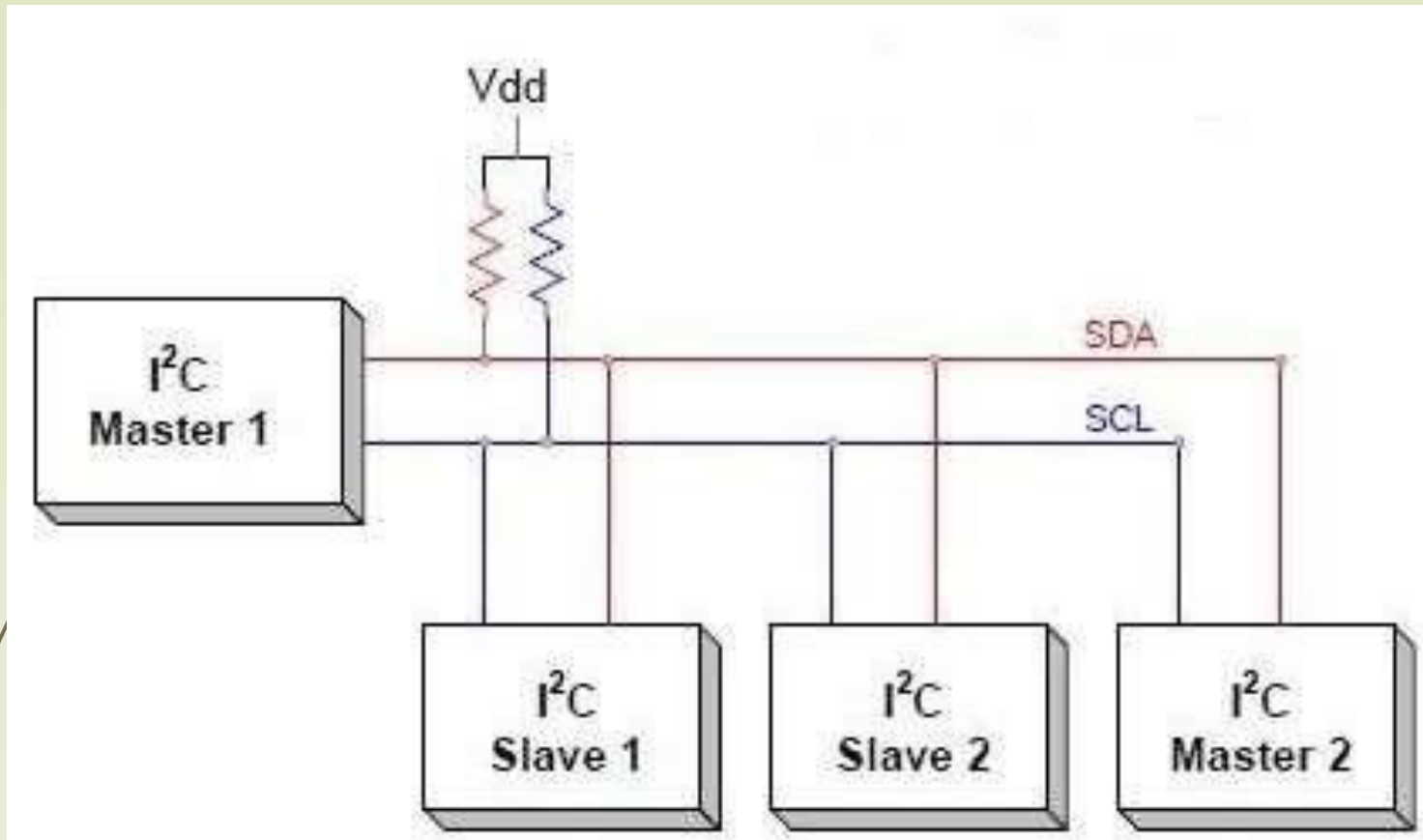
The **various interfaces** for onboard communication are as follows:

- i. Inter Integrated Circuit (I2C) Bus
- ii. Serial Peripheral Interface (SPI) Bus
- iii. Universal Asynchronous Receiver Transmitter (UART)
- iv. 1-Wire Interface
- v. Parallel Interface

# Inter Integrated Circuit (I2C) Bus

- The Inter Integrated Circuit Bus is a **synchronous bi-directional half duplex two wire serial interface bus**. The I2C bus comprise of **two bus lines**, namely; Serial Clock-SCL and Serial Data-SDA.
- **SCL line** is responsible **for generating synchronization clock pulses** and **SDA** is responsible **for transmitting the serial data across devices**.
- Devices connected to the I2C bus can act as either ‘Master’ device or ‘Slave’ device.
- The **‘Master’** device is responsible **for controlling** the communication by **initiating/terminating data transfer**, sending data and generating necessary synchronization clock pulses.
- **‘Slave’** devices **wait for the commands from the master and respond upon receiving the commands**. ‘Master’ and ‘Slave’ devices can act as either transmitter or receiver. **I2C supports multi masters** on the same bus.

I<sup>2</sup>C supports multi masters on the same bus



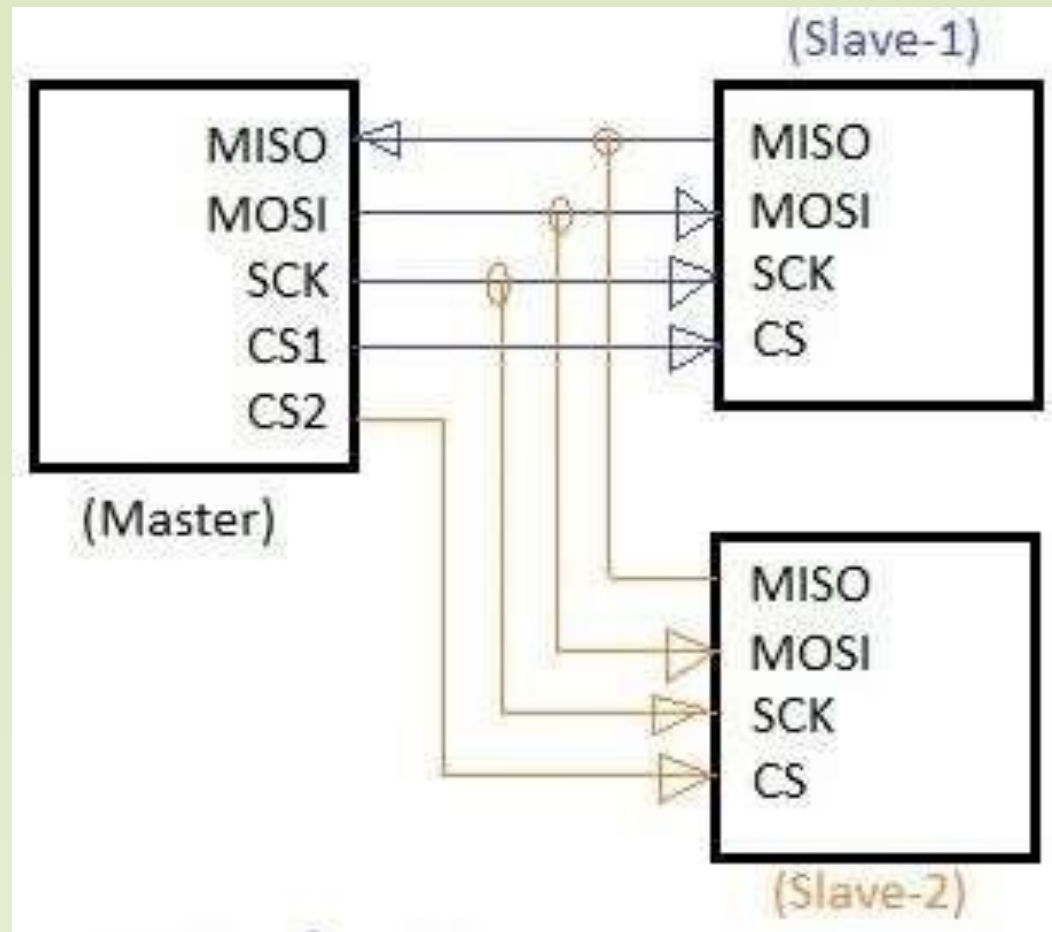
Sequence of operations for communicating with an I2C slave device is:

- The master device pulls the clock line (SCL) of the bus to High.
- The master device pulls the data line (SDA) of the bus to Low (start condition).
- The master device sends the address of the slave device to which it want to communicate, over SDA line.
- The master device sends the read or write bit(1- read, 0- write)
- The master device waits for the acknowledgement bit from the slave device whose address is sent on the bus along with read/write operation.
- The slave device responds to the master device by sending acknowledgement bit .
- Upon receiving the acknowledgement bit , master device sends the 8 bit data to slave device over SDA line(if request to write).Slave device sends the 8 bit data to master device over SDA line(if request to read).
- Master device waits for the transfer to complete.
- The master device terminates the transfer by pulling SDA line to High.

# Serial Peripheral Interface (SPI) Bus


- The Serial Peripheral Interface Bus (SPI) is a **synchronous bi-directional full duplex four-wire** serial interface bus.
- SPI is a **single master multi-slave** system.
- SPI requires **four signal lines** for communication.
  - MOSI – carrying data from master to slave
  - MISO – carrying data from slave to master
  - Serial Clock(SCLK) - carrying clock signals
  - Slave Select (SS) - slave device select.
- When compared to I2C, **SPI bus** is **most suitable** for applications requiring transfer of data in ‘streams’.





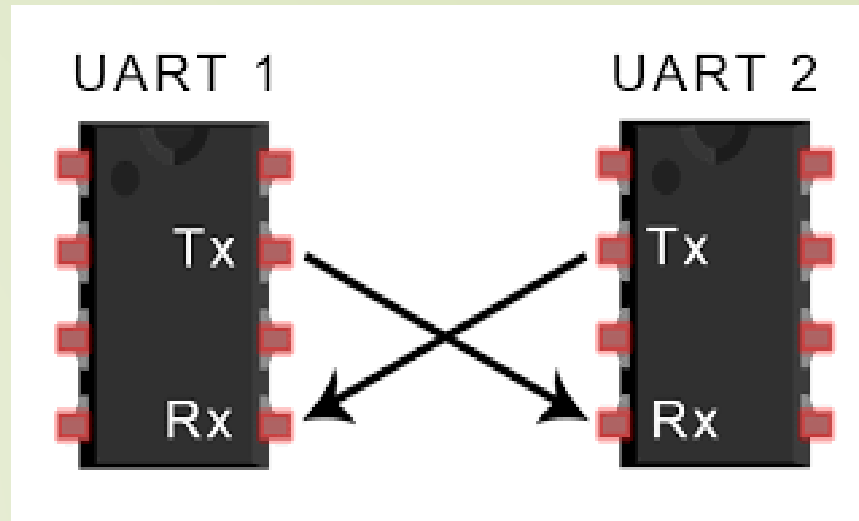
SPI bus interfacing



- 
- SPI works on principle of shift register. The master slave contains a special shift register for data to transmit or receive.
  - During transmission from master to slave, data in master's shift register is shifted out to MOSI pin and it enters the shift register of the slave device through MOSI pin of slave device.
  - During transmission from slave to master, data in slave's shift register is shifted out to MISO pin and it enters the shift register of the master device through MISO pin of master device.

# Universal Asynchronous Receiver Transmitter(UART)

- Asynchronous form of serial data transmission.
- UART based serial data transmission does not require a clock signal to synchronize the transmitting end and receiving end for transmission.
- Transmission- first received bit as the LSB and last received data bit as MSB.
- For proper communication, the transmit line of the sending device should be connected to the receive line of the receiving device.



# 1-Wire Interface

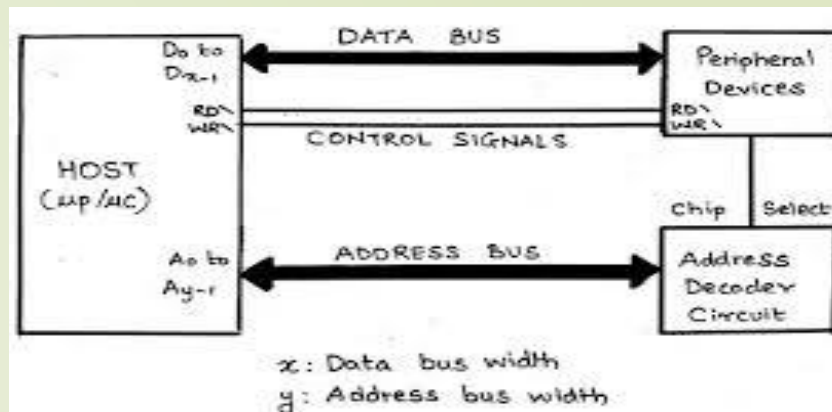
- It is an synchronous half duplex communication protocol.
- It makes use of only single signal line(DQ) for communication and follows master-slave communication.
- It supports single master and one or more slave device on the bus.
- Every 1-wire device contains a globally unique 64 bit identification number stored within it.

The sequence of operation for communication with 1-wire slave device is listed below-

- Master device sends a “reset” pulse on the 1-wire bus.
- Slave device present on the bus respond with “presence “ pulse.
- Master device sends command(followed by 64 bit address on the device).
- The master device sends read/write command to read/write in register of the slave device.
- The master initiates a read data/write data from the device or to the device.

# Parallel Interface

- On board parallel interface is normally used for communicating with peripheral devices which are memory mapped to the host of the system.
- The communication through the parallel bus is controlled by the control signal interface between the device and the host.
- The “control signal” for communication includes read/write signal and device select signal. The direction of data transfer can be controlled through the control signal lines for read and write.
- Address decoder circuit is used for generating the chip select signal for the device.
- Parallel data communication offers the highest speed for the data transfer.



# External Communication Interfaces

- The **External Communication Interface** refers to the different communication channels/buses used by the embedded system **to communicate with the external world**. The **various interfaces** for external communication are as follows
  - i. RS-232 C & RS-485
  - ii. Universal Serial Bus (USB)
  - iii. IEEE 1394 (Firewire)
  - iv. Infrared (IrDA)
  - v. Bluetooth (BT)
  - vi. Wi-Fi
  - vii. ZigBee
  - viii. General Packet Radio Service (GPRS)

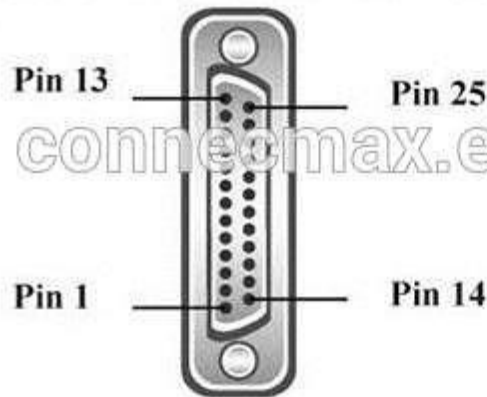
# RS-232 C & RS-485

- **RS-232C** is a legacy, full duplex, wired, **asynchronous** serial **communication interface**. RS-232 supports two different types of connectors, namely; **DB-9**; 9-Pin connector and **DB-25**: 25-Pin connector. RS-232 supports **only point-to-point communication** and not suitable for **multi-drop communication**.
- **RS-485** is the **enhanced version of RS-422** and it supports **multi-drop communication** with up to 32 transmitting devices (drivers) and 32 receiving devices on the bus.

# RS232 25 Pin

Pin 2	TXD
Pin 3	RXD
Pin 4	RTS
Pin 5	CTS
Pin 6	DSR
Pin 7	GND
Pin 8	DCD
Pin 20	DTR
Pin 22	RI

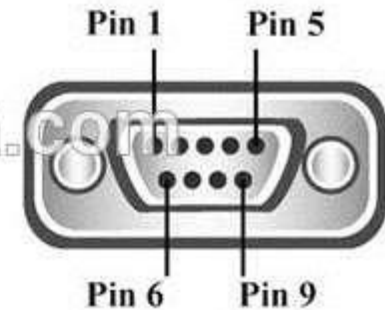
RS232 Pinout (25 Pin Male)



# RS232

Pin 1	DCD
Pin 2	RXD
Pin 3	TXD
Pin 4	DTR
Pin 5	GND
Pin 6	DSR
Pin 7	RTS
Pin 8	CTS
Pin 9	RI

RS232 Pinout (9 Pin Male)

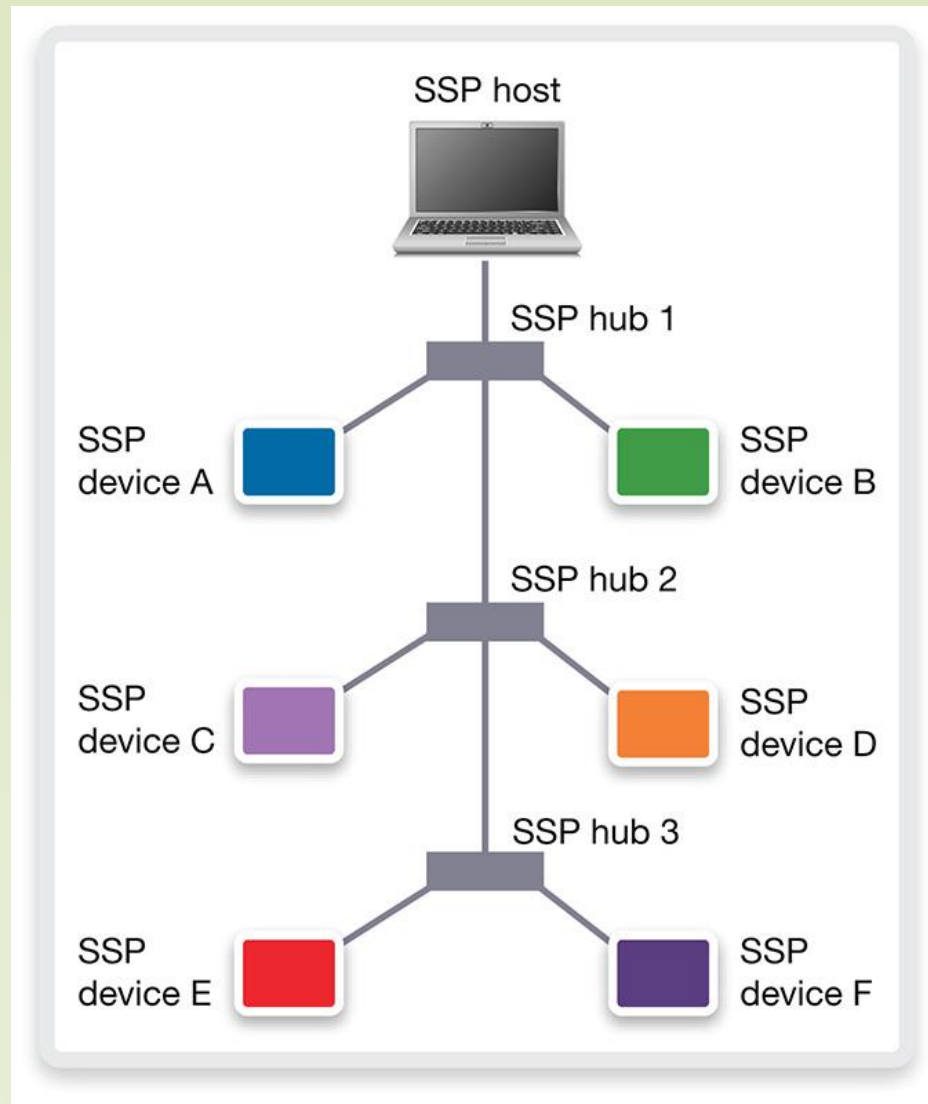


DB-9 and DB-25 RS232 Connector Interface

# Universal Serial Bus (USB)

- **Universal Serial Bus (USB)** is a **wired high speed serial bus** for data communication. The USB host can support **connections up to 127**.
- USB transmit data in packet format.
- Follows star topology.
- USB communication is a host initiated one. USB host contains a host controller which is responsible for data communication, including establishing connectivity with USB slave devices.
- Physical connection between USB peripheral device and master device is established with USB cable.
- USB cable supports communication distance of up to 5 meters.
- USB device contains a PID and VID(essential for loading the drivers corresponding to USB device for communication).





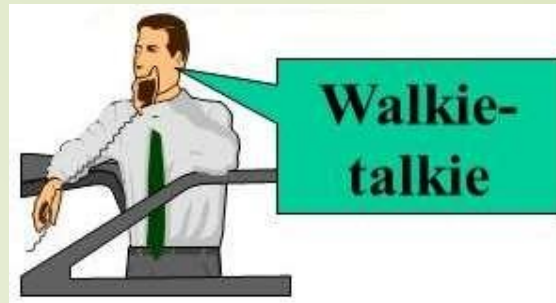
# IEEE 1394 (Firewire)

- **IEEE 1394 (Firewire)** is a **wired, isochronous high speed serial communication bus**. It is also known as **High Performance Serial Bus (HPSB)**.
- **1394** is a **popular communication interface** for connecting embedded devices like Digital Camera, Camcorder, Scanners to desktop computers **for data transfer and storage**.
- **Unlike USB interface**, IEEE 1394 **doesn't require a host** for communicating between devices. For example, you **can directly connect a scanner with a printer** for printing. The **data rate** supported by 1394 is **far higher than** the one supported by **USB 2.0 interface**. The 1394 hardware implementation is much **costlier than USB implementation**.



# Infrared Data Association (IrDA)

- **Infrared (IrDA)** is a serial, **half duplex**, line of sight based wireless **technology** for data communication between devices. It is in use from the olden days of communication and you may be **very familiar** with it. The **remote control** of your **TV, VCD player, etc** works on Infrared data communication principle.



# Bluetooth (BT)

**Bluetooth** is a **low cost, low power, short range wireless technology** for **data and voice communication**. Bluetooth supports point-to-point (device to device) and point-to-multipoint (device to multiple device broadcasting) wireless communication.

- A Bluetooth device can function as **either master or slave**. When a network is formed with **one Bluetooth device as master** and **more than one device as slaves**, it is called a **Piconet**. A **Piconet** supports a **maximum of seven slave devices**.

Bluetooth is the favorite choice for **short range data communication** in handheld embedded devices. Bluetooth technology is very popular **among cell phone users** as they are the easiest communication channel for **transferring ringtones, music files, pictures, media files, etc** between neighboring Bluetooth enabled phones.

- It supports a data rate of **up to 1 Mbps** and a range of approximately **30 feet** for data communication.





# Wi-Fi

- **Wi-Fi or Wireless Fidelity** is the popular wireless communication technique for networked communication of devices. Wi-Fi is intended for network communication and it supports Internet Protocol (IP) based communication. It is essential to have device identities in a multipoint communication to address specific devices for data communication.
- **Wi-Fi based communications** require an intermediate agent called Wi-Fi router/Wireless access point to manage the communications. Wi-Fi supports data rates ranging from 1 Mbps to 150 Mbps and offers a range of 100 to 300 feet.


# ZigBee

- **ZigBee** is a **low power, low cost, wireless network communication protocol** based on the IEEE 802.15.4-2006 standard.
- ZigBee is targeted for low power, **low data rate and secure applications** for Wireless Personal Area Networking (WPAN).
- ZigBee operates worldwide at the unlicensed bands of Radio spectrum, mainly at **2.400 to 2.484 GHz, 902 to 928 MHz and 868.0 to 868.6 MHz**.
- ZigBee supports an operating **distance of up to 100 meters** and a **data rate of 20 to 250Kbps**.



# Embedded Firmware

- **Embedded firmware** refers to the **control algorithm** (Program instructions) and/or the **configuration settings** that an embedded system developer dumps into the code (program) memory of the embedded system. It is an **un-avoidable part** of an embedded system. There are **various methods** available for developing the embedded firmware. They are listed below:
  1. Write the program in **high level languages** like Embedded C/C++ using an Integrated Development Environment.
  2. Write the program in **Assembly language** using the instructions supported by your application's target processor/controller.

- 
- The instruction set for each family of processor/controller is different and the program written in either of the methods given above **should be converted** into a processor **understandable machine code before loading it into the program memory**.
  - The process of converting the program written in either a high level language or processor/controller specific Assembly code to **machine readable binary code** is called '**HEX File Creation**'.
  - If the program is written in Embedded C/C++ using an IDE, **the cross compiler included in the IDE** converts it into **corresponding processor/controller understandable 'HEX File'**.
  - If you are following the Assembly language based programming technique, you can use the utilities supplied by the processor/controller vendors to convert **the source code into 'HEX File'**.



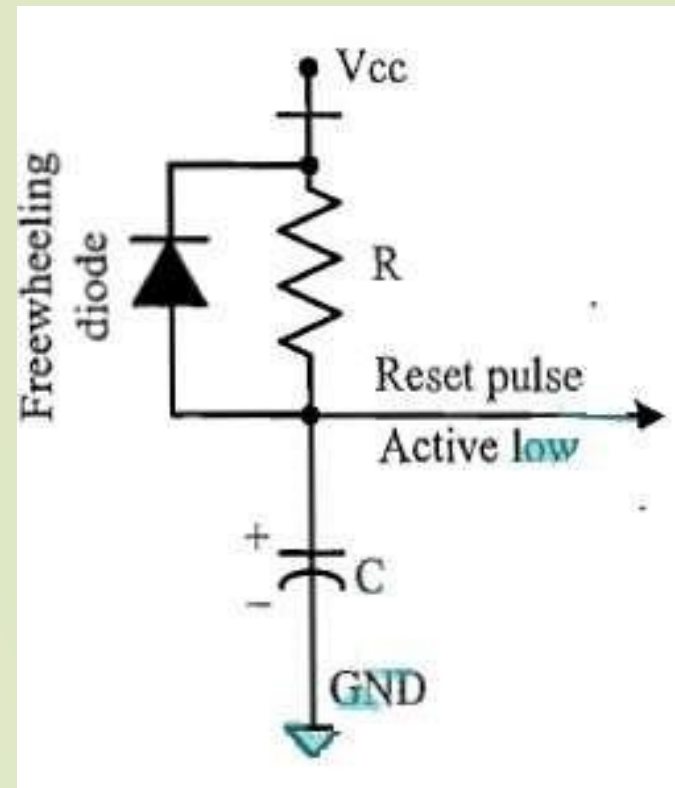
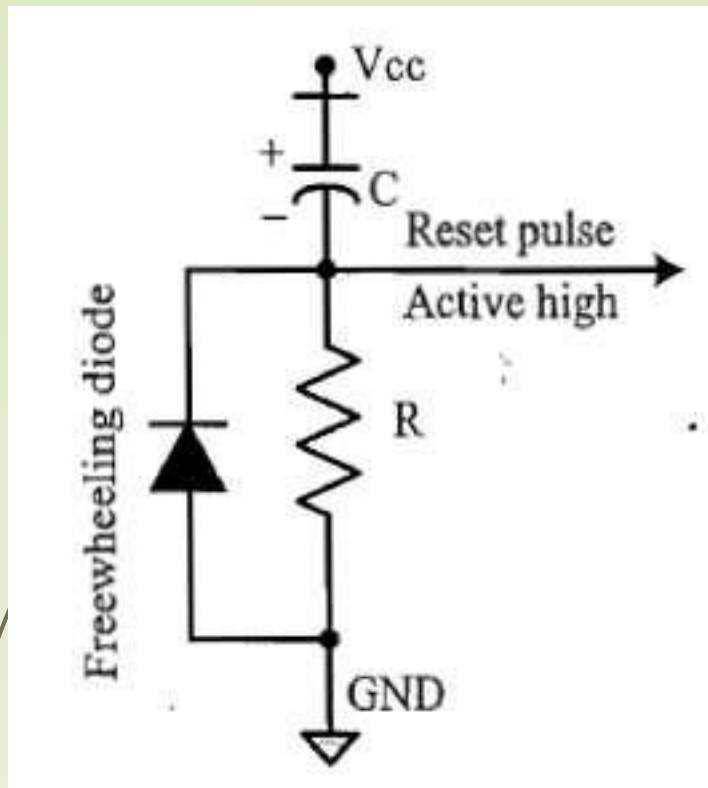
## Other System Components

- The other system components refer to the components/circuits/ICs which are necessary for the proper functioning of the embedded system.
- Reset Circuit, Brown-out Protection Circuit, Oscillator Unit, Real-Time Clock (RTC), Watchdog Timer are examples of circuits/ICs which are essential for the proper functioning of the processor/controllers.

# Reset Circuit

- The reset circuit is essential to ensure that the device is not operating at a voltage level where the device is not guaranteed to operate, during system power ON. The reset signal brings the internal registers and the different hardware systems of the processor/controller to a known state and starts the firmware execution from the reset vector.
- The reset signal can be either active high (The processor undergoes reset when the reset pin of the processor is at logic high) or active low (The processor undergoes reset when the reset pin of the processor is at logic low). Since the processor operation is synchronized to a clock signal, the reset pulse should be wide enough to give time for the clock oscillator to stabilize before the internal reset state starts.
- Some microprocessors/controllers contain built-in internal reset circuitry and they don't require external reset circuitry. Figure illustrates a resistor capacitor based passive reset circuit for active high and low configurations. The reset pulse width can be adjusted by changing the resistance value R and capacitance value C.

## RC-Based Reset Circuit



# Real-Time Clock (RTC)

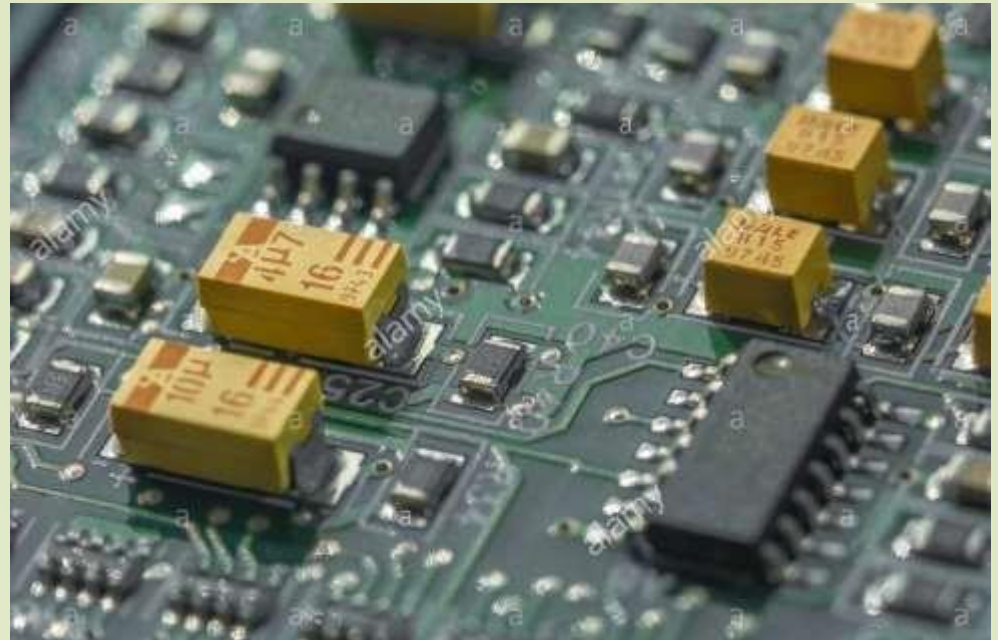
- **Real-Time Clock (RTC)** is a system component responsible for keeping track of time. RTC holds information like current time (in hours, minutes and seconds) in 12 hour/24 hour format, date, month, year, day of the week, etc. and supplies timing reference to the system. RTC is intended to function even in the absence of power. The RTC chip contains a microchip for holding the time and date related information and backup battery cell for functioning in the absence of power, in a single IC package. The RTC chip is interfaced to the processor or controller of the embedded system. For Operating System based embedded devices, a timing reference is essential for synchronizing the operations of the OS kernel.

# Watchdog Timer

- A watchdog is **to monitor the firmware execution** and **reset** the system processor/microcontroller when the program execution hangs up or **generates** an Interrupt in case **the execution time** for a task is exceeding the maximum allowed limit. **If the firmware execution doesn't complete due to malfunctioning**, within the time required by the watchdog to reach the maximum count, the counter will generate a reset pulse and this **will reset the processor** (if it is connected to the reset line of the processor).
- Most of the processors implement watchdog **as a built-in component** and provides status register to control the watchdog timer (like enabling and disabling watchdog functioning) and watchdog timer register for writing the count value. **If the processor/controller doesn't contain a built in watchdog timer**, the same can be implemented **using an external watchdog timer IC circuit**.

# PCB and Passive Components

- **Printed Circuit Board (PCB)** is the **backbone** of every embedded system. **After finalizing the components and the inter-connection** among them, a schematic design is created and according to the **schematic PCB is fabricated**. **PCS acts as a platform for mounting all the necessary components as per the design requirement**. Also it **acts as a platform for testing** your embedded firmware. You can also find some passive **electronic components** like resistor, capacitor, diodes, etc. on your board. They are the co-workers of various chips contained in your embedded hardware. They are very essential for the proper functioning of your embedded system.





# Brown-out Protection Circuit

- The brown-out protection circuit **prevents the processor/controller** from **unexpected program execution behavior** when **the supply voltage** to the processor/controller **falls** below a specified voltage.
- It is **essential for a battery powered devices** since there are greater chances for battery voltage to drop below the required threshold.
- A brown-out protection circuit holds the processor/controller in reset state , when the operating voltage falls below the threshold, until it rises above the threshold voltage.



# Oscillator

- It is analogous to heartbeat of a living being which synchronizes the execution of life whereas oscillator unit of embedded systems is responsible for generating the clock for the processor.
- The Oscillator unit **generates clock signals for synchronizing the operations of the processor.**
- Certain devices may not contain a built-in oscillator unit and require the clock pulses to be generated and supplied externally.
- **Quartz crystal oscillators** are available in the form of chip and can be used for generating clock pulses.