MODULE – 4

INTERFACING

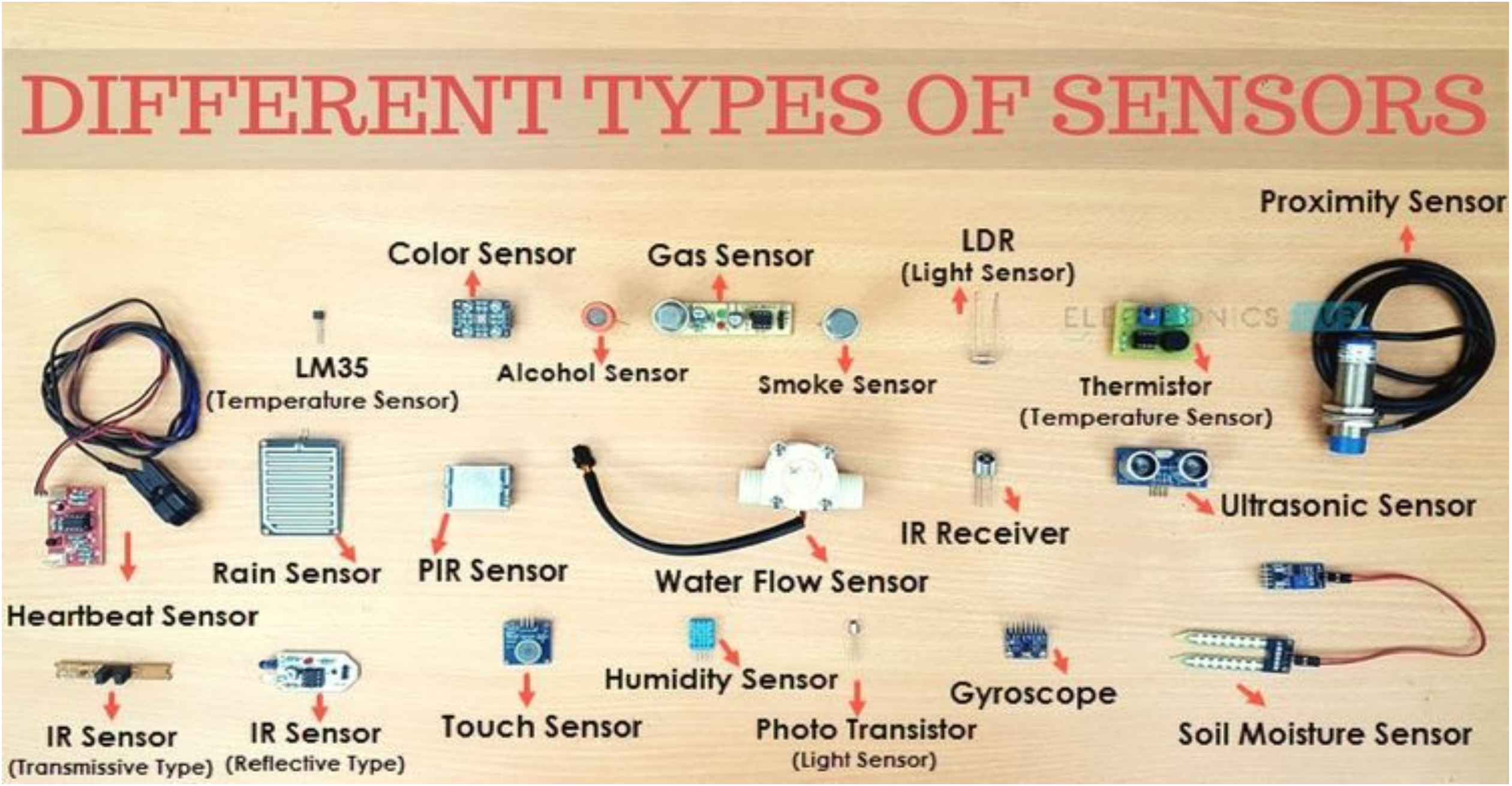
Sensors and Actuators

- An embedded system is in constant interaction with the real world and the controlling/monitoring functions executed by the embedded system is achieved in accordance with the changes happening to the real world.
- The changes in system environment or variables are detected by the sensors connected to the input port of the embedded system.
- 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.

Sensors and Actuators (continued)

- A sensor is a transducer device that converts energy from one form to another for any measurement or control purpose.
  - E.g.: Temperature sensor, magnetic hall effect sensor, humidity sensor, etc.
- An actuator is a form of transducer device (mechanical or electrical) which converts signals to corresponding physical action (motion).
  - Actuator acts as an output device.
  - E.g.: Stepper motor

DIFFERENT TYPES OF SENSORS



DC Motor

DC Gear Motor

RC Servo motor

Stepper motor

Manual Valve Actuators

Pneumatic Valve Actuators

BLDC Motor

Smart Servo motors

Harmonic drives

Linear electric actuator

Hydraulic Valve Actuators

Electric Valve Actuators

The 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.

Light Emitting Diode (LED)

- Light Emitting Diode (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.
  - E.g.: 'Device ON', 'Battery low' or 'Charging of battery' conditions
  - Light Emitting Diode is a p-n junction diode and it contains an anode and a cathode.
- For proper functioning of the LED, the anode is connected to +ve terminal of the supply voltage and cathode to the -ve terminal of supply voltage.
- The current flowing through the LED must be limited to a value below the maximum current that it can conduct.
  - A resister is used in series to limit the current through the LED.
- The ideal LED interfacing circuit is shown in the figure.

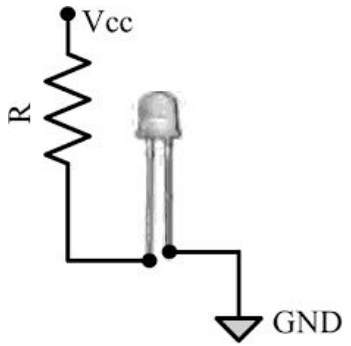
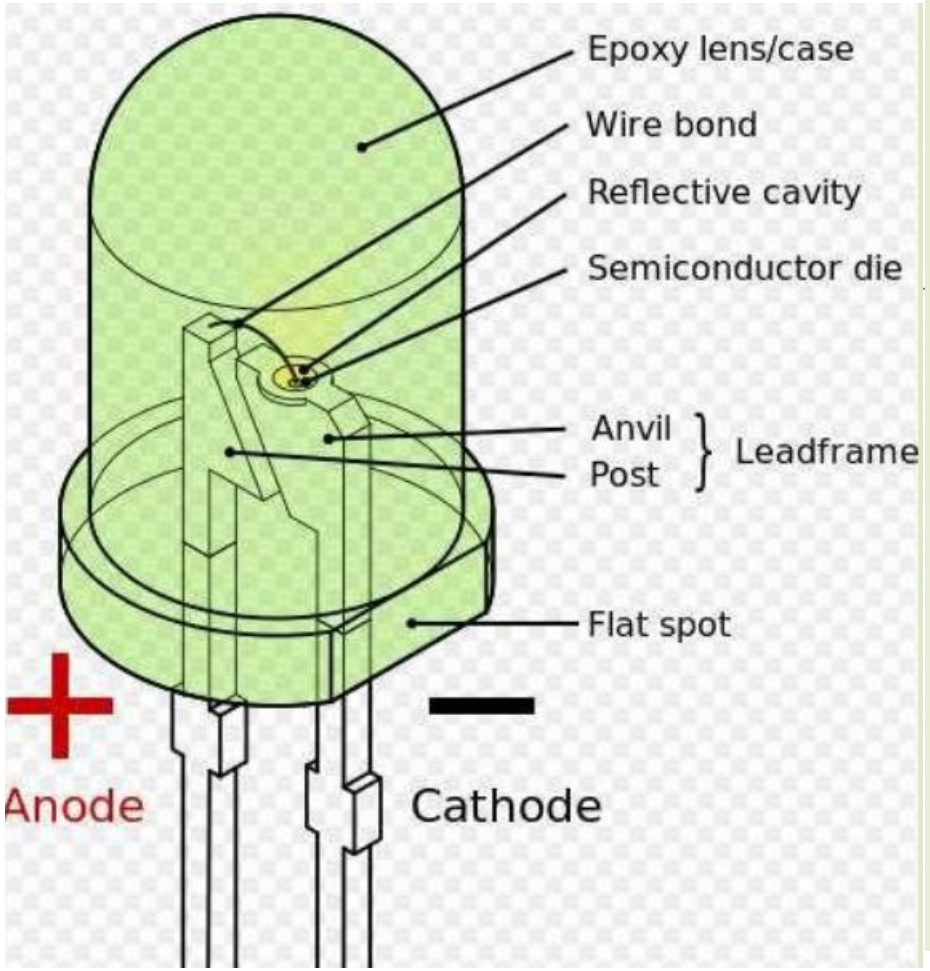
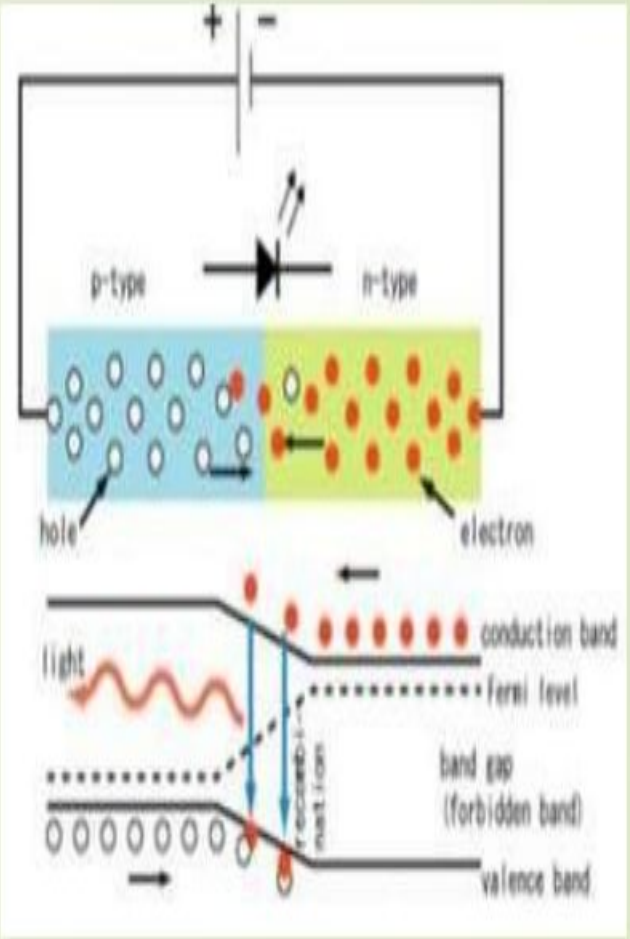


Fig: LED interfacing

Light Emitting Diode (LED) (continued)

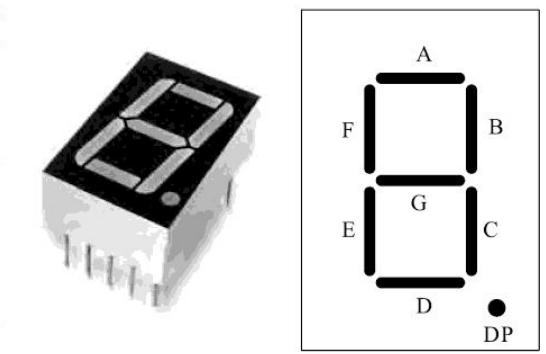
- LEDs can be interfaced to the port pin of a processor/controller in two ways:
  - In the first method, the anode is directly connected to the port pin and the port pin drives the LED.
    - The port pin 'sources' current to the LED when the port pin is at logic High (Logic ‘1’).
  - In the second method, the cathode of the LED is connected to the port pin of the processor/controller and the anode to the supply voltage through a current limiting resistor.
    - The LED is turned on when the port pin is at logic Low (Logic '0’).
    - Here the port pin 'sinks' current.


Epoxy lens/case
Wire bond
Reflective cavity
Semiconductor die
Anvil
Post
Leadframe
Flat spot
Anode
Cathode


+
−
p-type
n-type
hole
electron
conduction band
light
Fermi level
band gap
(forbidden band)
valence band

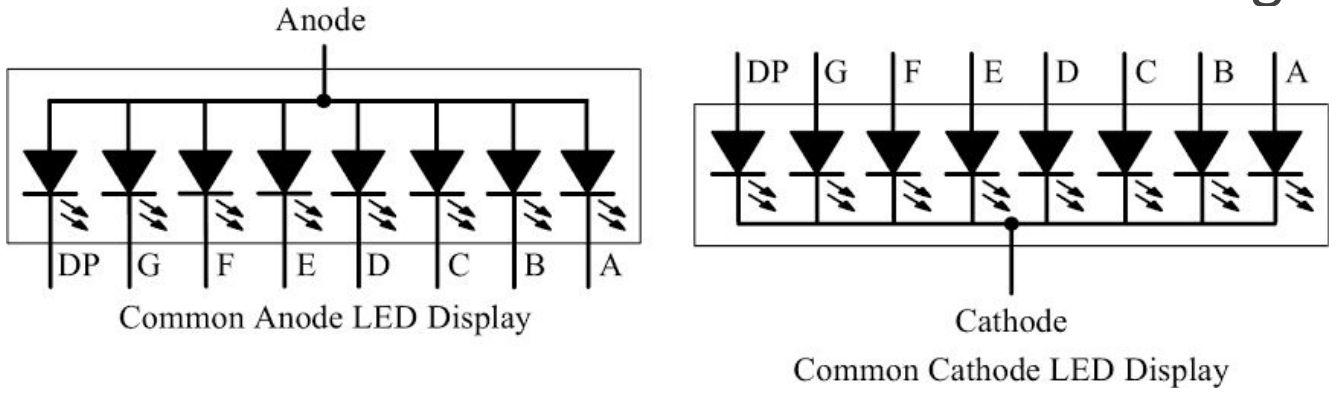
7-Segment LED Display

- This 7-segment LED display is an output device for displaying alpha numeric characters.
- It contains 7 LED segments arranged in a special form used for displaying alpha numeric characters and 1 LED 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.



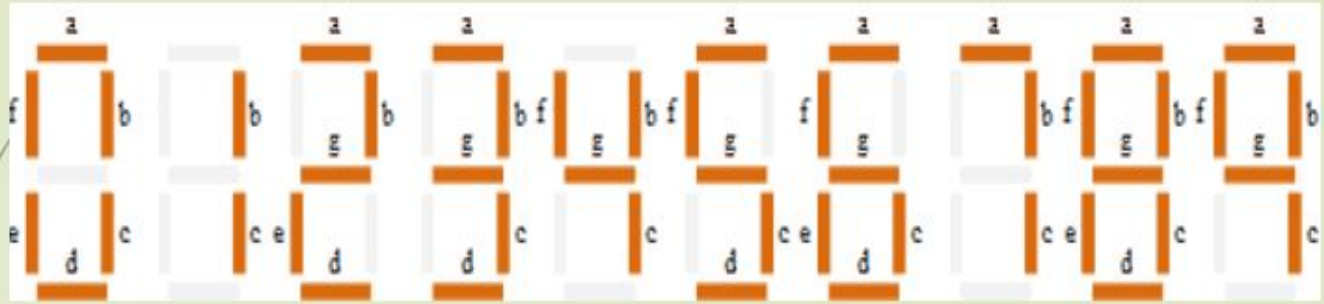
7-Segment LED Display (continued)

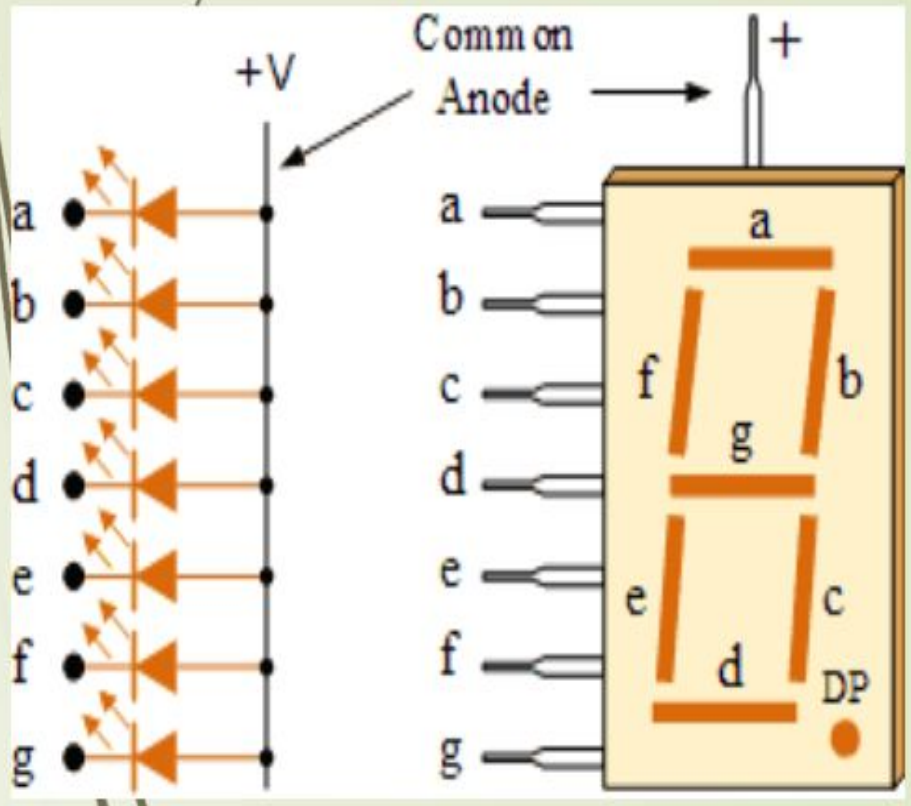
- The 7-segment LED displays are available in two different configurations, namely; Common Anode and Common Cathode.
- In the common anode configuration, the anodes of the 8 segments are connected commonly whereas in the common cathode configuration, the cathodes of 8 LED segments are connected commonly.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

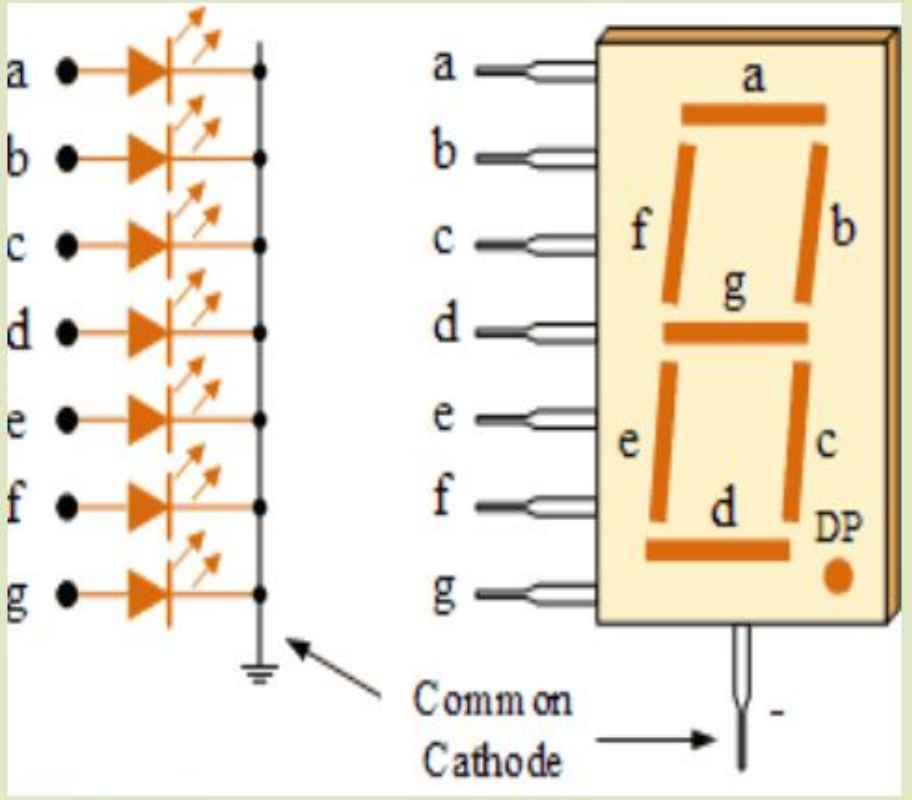


7-Segment LED Display (continued)

- Based on the configuration of the 7-segment LED unit, the LED segment's anode or cathode is connected to the port of the processor/controller in the order 'A' segment to the least significant port pin and DP segment to the most significant port pin.
- The current flow through each of the LED segments should be limited to the maximum value supported by the LED display unit.
  - The typical value is 20mA.
  - The current can be limited by connecting a current limiting resistor to the anode or cathode of each segment.
- 7-segment LED display is used in low cost embedded applications like Public telephone call monitoring devices, point of sale terminals, etc.


a
f
b
g
e
d
c


+V
Common
Anode
+
a
b
c
d
e
f
g
DP


a
b
c
d
e
f
g
DP
Common
Cathode
-

Stepper Motor

A stepper motor is an electro-mechanical device which generates discrete displacement (motion) in response to dc electricalsignals.

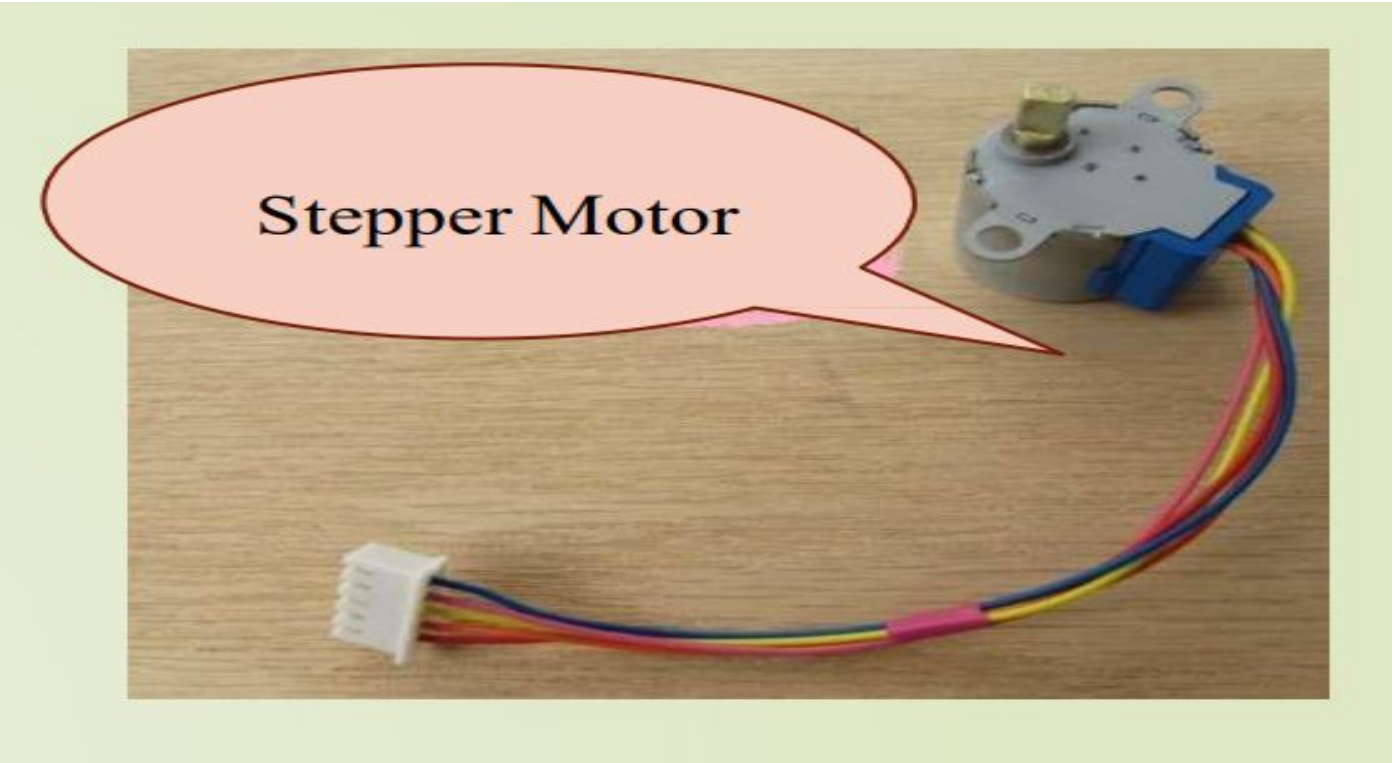
- It differs from the normal dc motor in its operation.
- The dc motor produces continuous rotation on applying dc voltage whereas a stepper motor produces discrete rotation in response to the dc voltage applied to it.
- Stepper motors are widely used in industrial embedded applications, consumer electronic 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

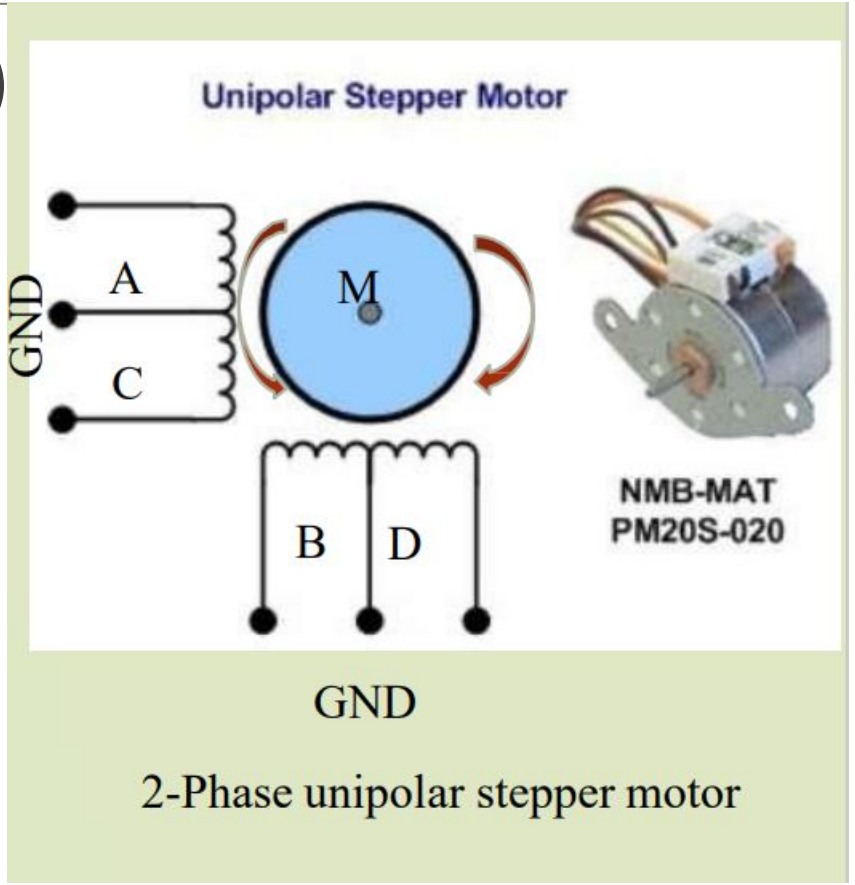


Unipolar

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.

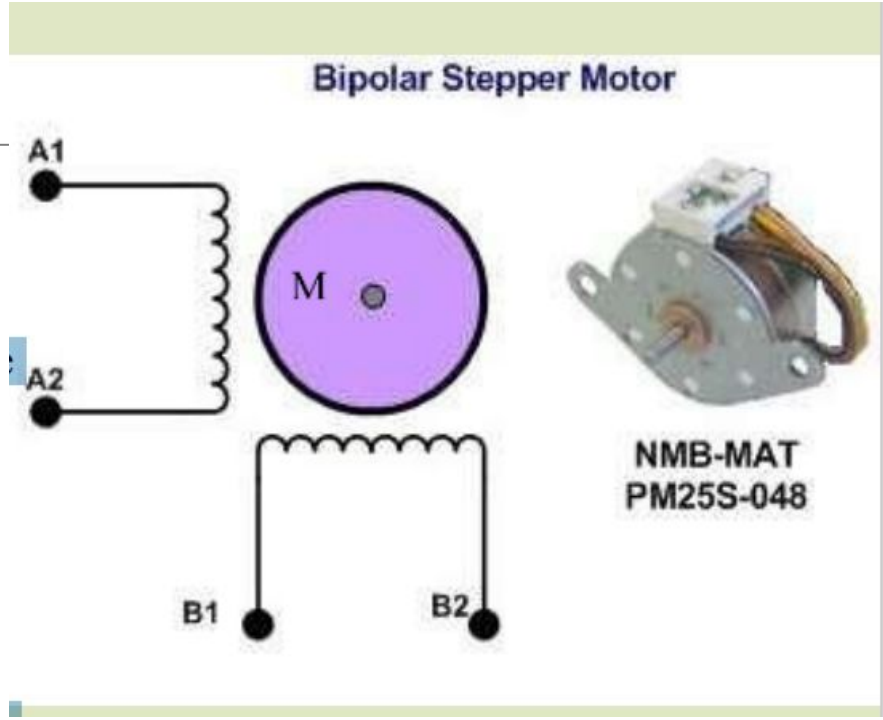
- Current in one direction flows through one coil and in the opposite direction flows through the other coil.
- It is easy to shift the direction of rotation by just switching the terminals to which the coils are connected.
- Figure illustrates the working of a two-phase unipolar stepper motor.



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

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.

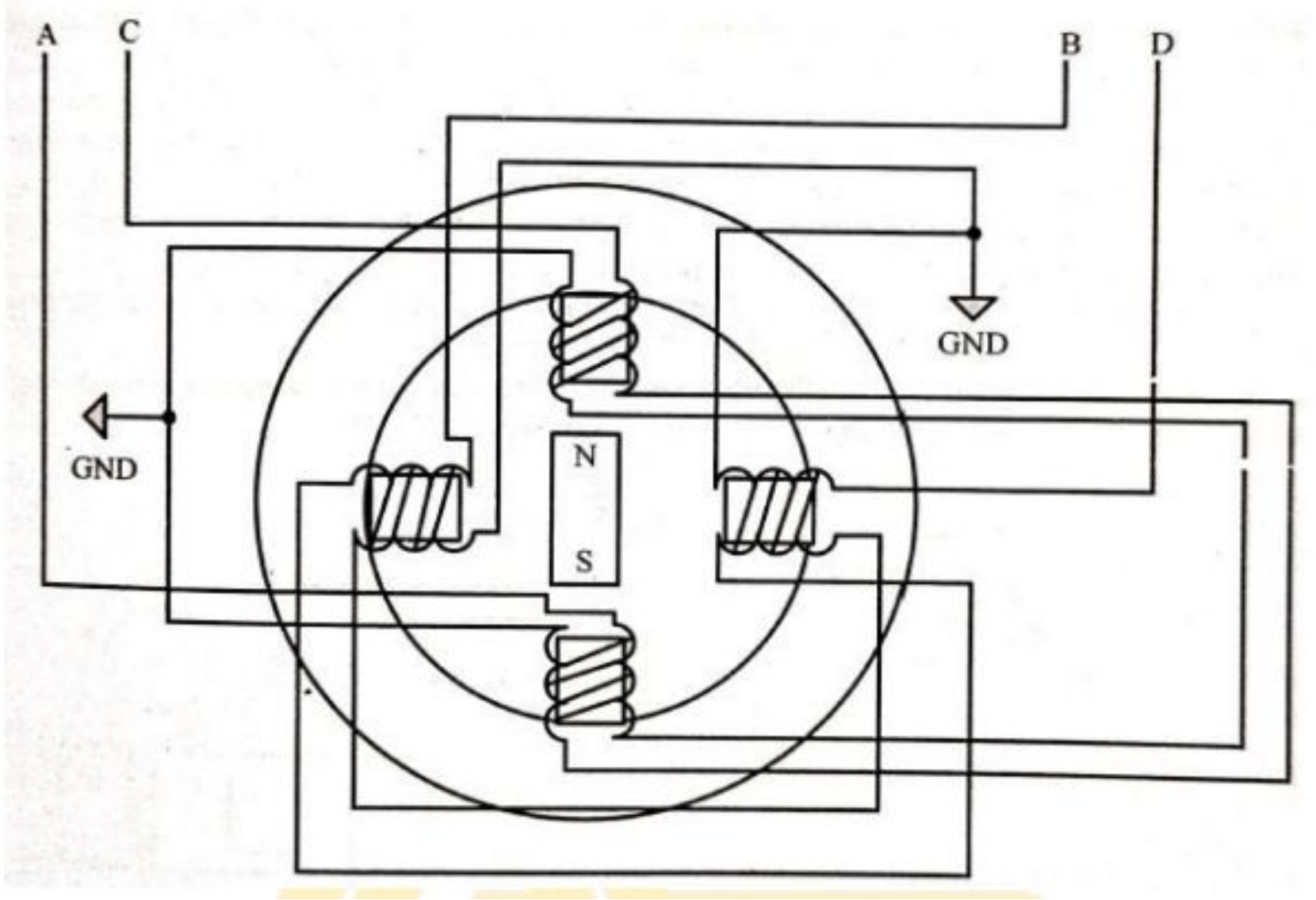
Step	Coil A	Coil B	Coil C	Coil D
1	H	L	L	L
2	H	H	L	L
3	L	H	L	L
4	L	H	H	L
5	L	L	H	L
6	L	L	H	H
7	L	L	L	H
8	H	L	L	H

The rotation of the stepper motor can be reversed by reversing the order in which the coil is energized.

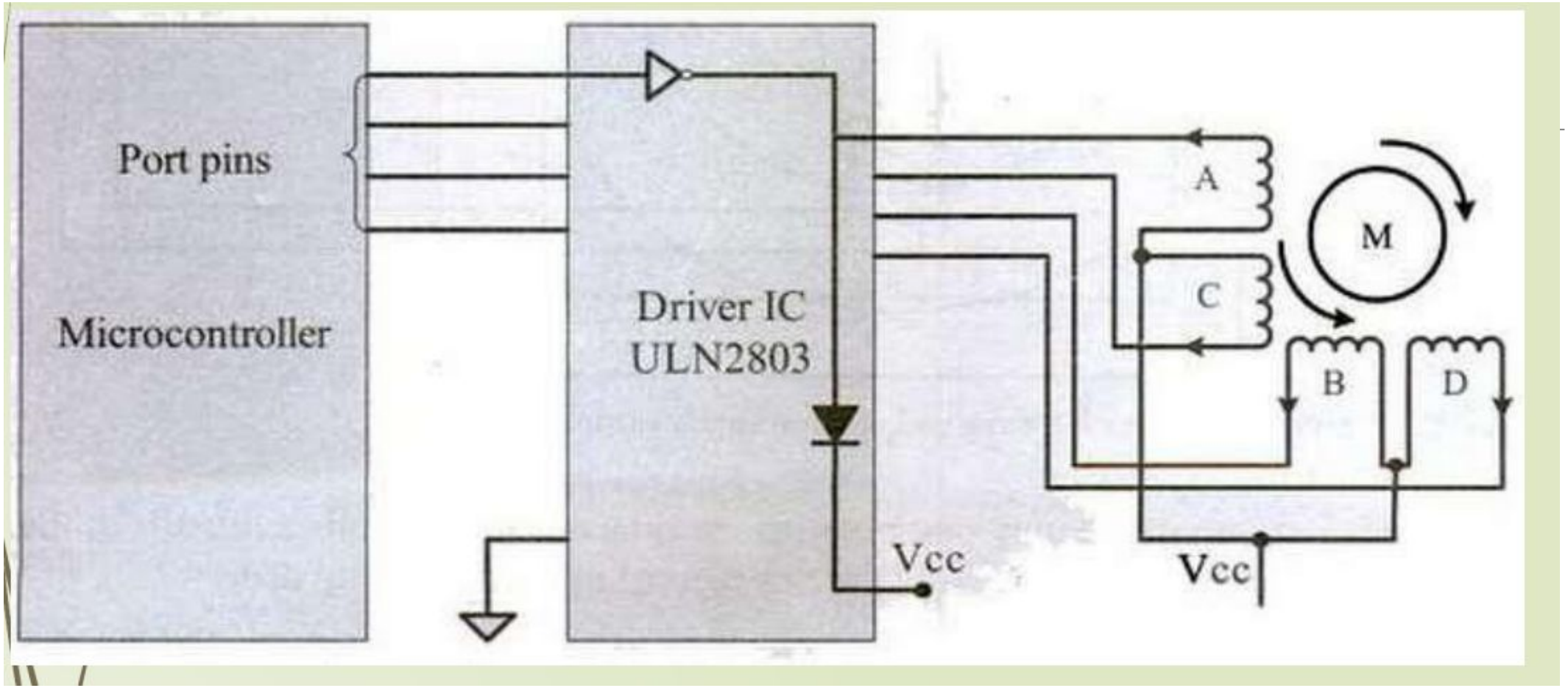
- Two-phase unipolar stepper motors are the popular choice for embedded applications.

- The current requirement for stepper motor is little high and hence the port pins of a microcontroller/processor may not be able to drive them directly.
- Also the supply voltage required to operate stepper motor varies normally in the range 5V to 24 V.
- Depending on the current and voltage requirements, special driving circuits are required to interface the stepper motor with microcontroller/processors.
- The following circuit diagram illustrates the interfacing of a stepper motor through a driver circuit connected to the port pins of a microcontroller/processor

The following Figure shows the stator winding details of Stepper motor:



Interfacing of stepper motor through driver circuit



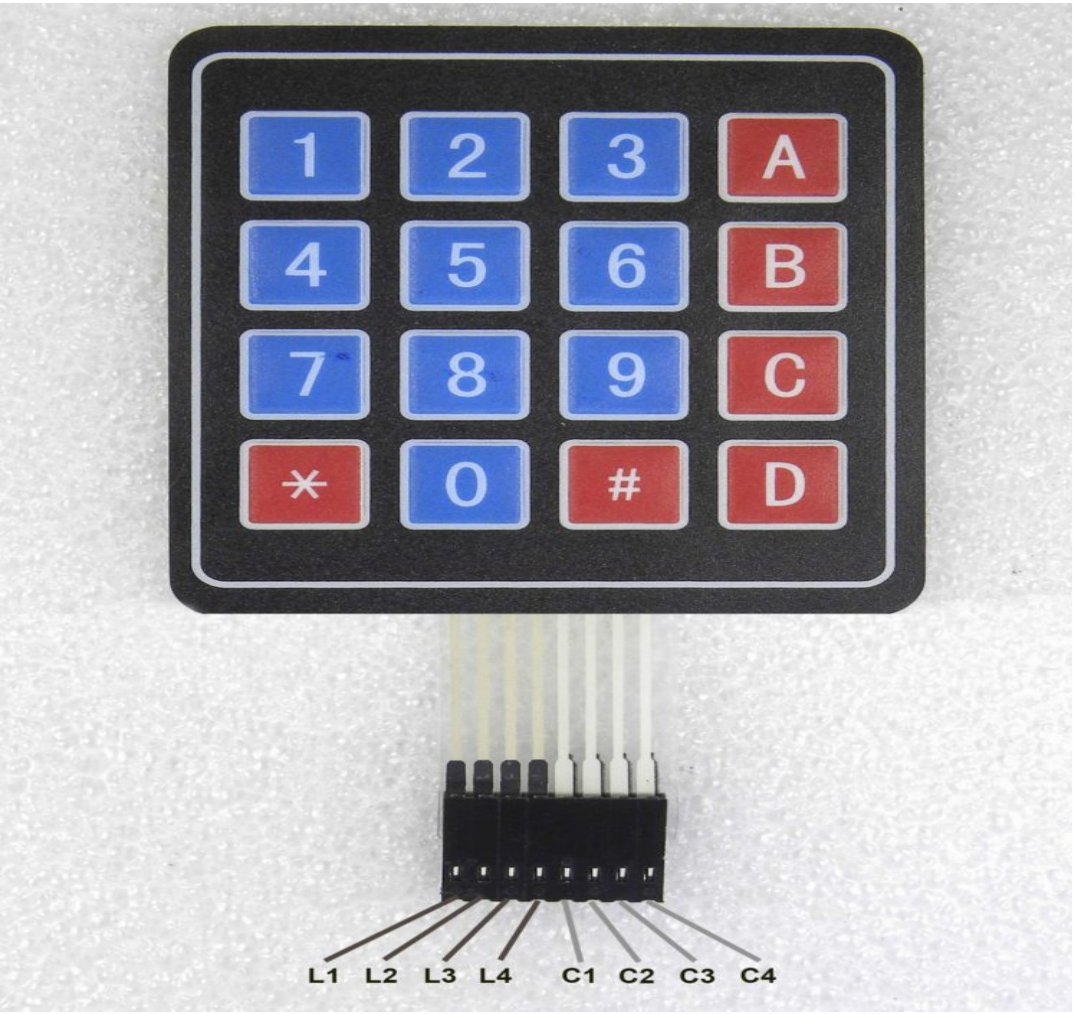
Push Button Switch

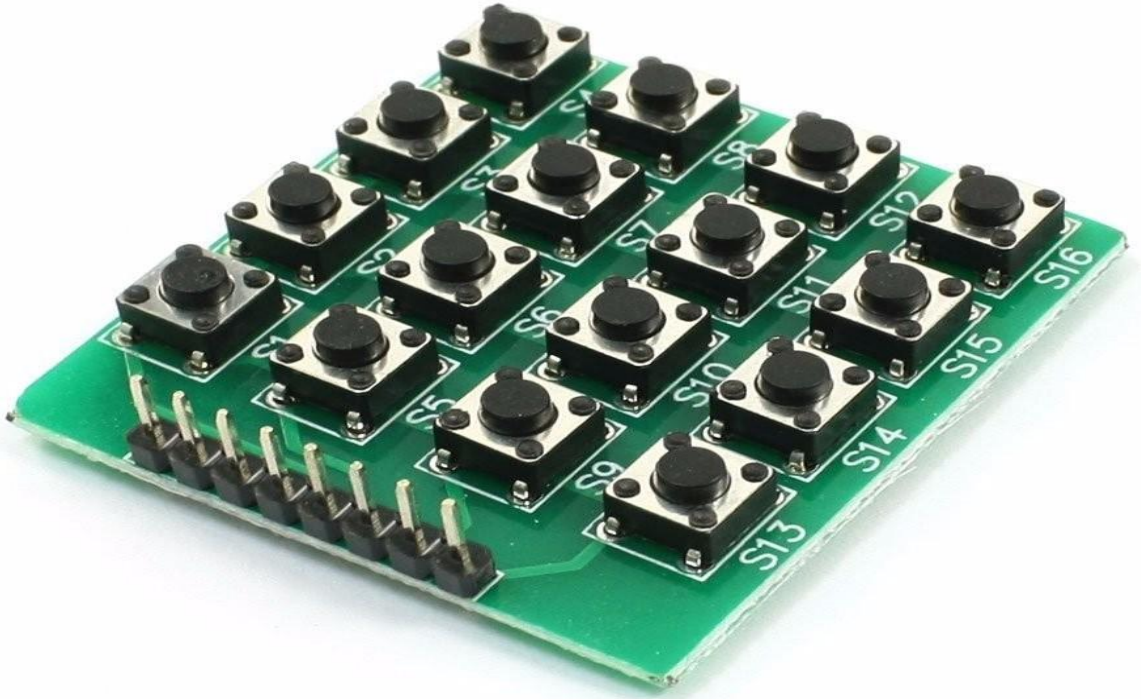
- It is an input device.
- Push button switch comes in two configurations, namely 'Push to Make' and 'Push to Break’.
- In the 'Push to Make' configuration, the switch is normally in the open state and it makes a circuit contact when it is pushed or pressed.
- In the 'Push to Break' configuration, the switch is normally in the closed state and it breaks the circuit contact when it is pushed or pressed.
- The push button stays in the 'closed' (For Push to Make type) or 'open'


EXTRA BASS
ADD

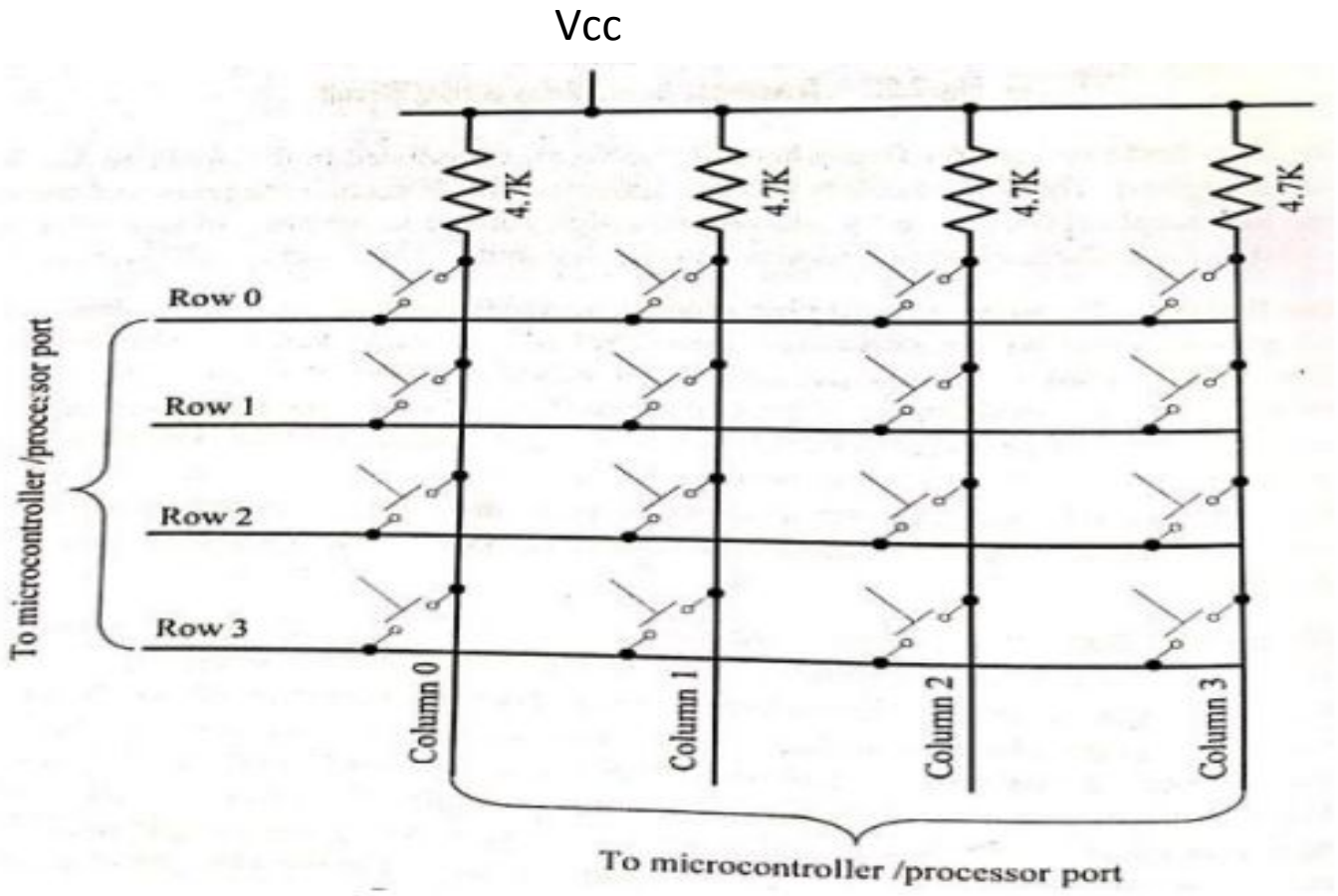
Keyboard

- Keyboard is an input device 'HIGH' Pulse generator 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.
- However, there may be situations demanding a large number of keys for user input (e.g. PDA device with alpha-numeric keypad for user data entry).
- In such situations it may not be possible to interface each keys to a port pin due to the limitation in the number of general purpose port pins available for the processor/ controller in use and moreover it is wastage of port pins.
- **Matrix keyboard** is an optimum solution for handling large key requirement. It greatly reduces the number of interface connections.


1
2
3
A
4
5
6
B
7
8
9
C
*
0
#
D
L1
L2
L3
L4
C1
C2
C3
C4


S1
S2
S3
S4
S5
S6
S7
S8
S9
S10
S11
S12
S13
S14
S15
S16

For example, for interfacing 16 keys, in the direct interfacing technique, 16 port pins are required, whereas in the matrix keyboard only 8 lines are required. The 16 keys are arranged in a 4 column x 4 Row matrix. The following Figure illustrates the connection o keys in a matrix keyboard.



Push Button Switch (continued)

- Push button is used for generating a momentary pulse.
- In embedded applications, push button is generally used as reset and start switch and pulse generator.
- The Push button is normally connected to the port pin
  of the host processor/controller.
- Depending on the way in which the push button interfaced to the controller, it can generate either a 'HIGH' pulse or a 'LOW' pulse.

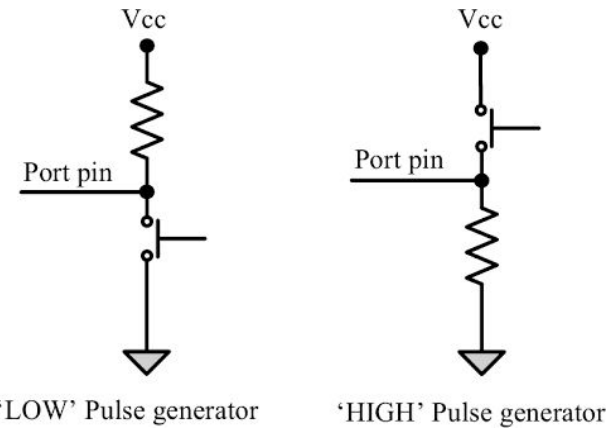


Fig: Push button switch configurations

Communication Interface

- Communication interface is essential for communicating with various subsystems of the embedded system and with the external world.
- For an embedded product, the communication interface can be viewed in two different perspectives:
  - Onboard Communication Interface (Device/board level communication interface)
    - E.g.: Serial interfaces like I2C, SPI, UART, 1-Wire, etc and parallel bus interface.
  - External Communication Interface (Product level communication interface)
    - E.g.: Wireless interfaces like Infrared (IR), Bluetooth (BT), Wireless LAN (Wi-Fi), Radio Frequency waves (RF), GPRS, etc. and wired interfaces like RS-232C/RS-422/RS-485, USB, Ethernet IEEE 1394 port, Parallel port, CF-II interface, SDIO, PCMCIA, etc.

Onboard Communication Interfaces

- An embedded system is a combination of different types of components (chips/devices) arranged on a printed circuit board (PCB).
- Onboard Communication Interface refers to the different communication channels/buses for interconnecting the various integrated circuits and other peripherals within the embedded system.
- E.g.: Serial interfaces like I2C, SPI, UART, 1-Wire, etc and parallel bus interface

Inter Integrated Circuit (I2C) Bus

- The Inter Integrated Circuit Bus (I2C Pronounced 'I square C') is a synchronous bi-directional half duplex two wire serial interface bus.
  - (Half duplex - one-directional communication at a given point of time)
- The concept of I2C bus was developed by Philips Semiconductors in the early 1980s.
- The original intention of I2C was to provide an easy way of connection between a microprocessor/microcontroller system and the peripheral chips in television sets.
- The I2C bus comprise of two bus lines:
  - Serial Clock (SCL line) – responsible for generating synchronisation clock pulses
  - Serial Data (SDA line) – responsible for transmitting the serial data across devices

Inter Integrated Circuit (I2C) Bus (continued)

- I2C bus is a shared bus system to which many number of I2C devices can be connected.(max 128 devices)
- Devices connected to the I2C bus can act as either 'Master' or 'Slave'.
  - The 'Master' device is responsible for controlling the communication by initiating/terminating data transfer, sending data and generating necessary synchronisation 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.
- Regardless whether a master is acting as transmitter or receiver, the synchronisation clock signal is generated by the 'Master' device only.
- I2C supports multi masters on the same bus.

Inter Integrated Circuit (I2C) Bus (continued)

- The following bus interface diagram illustrates the connection of master and slave devices on the I2C bus

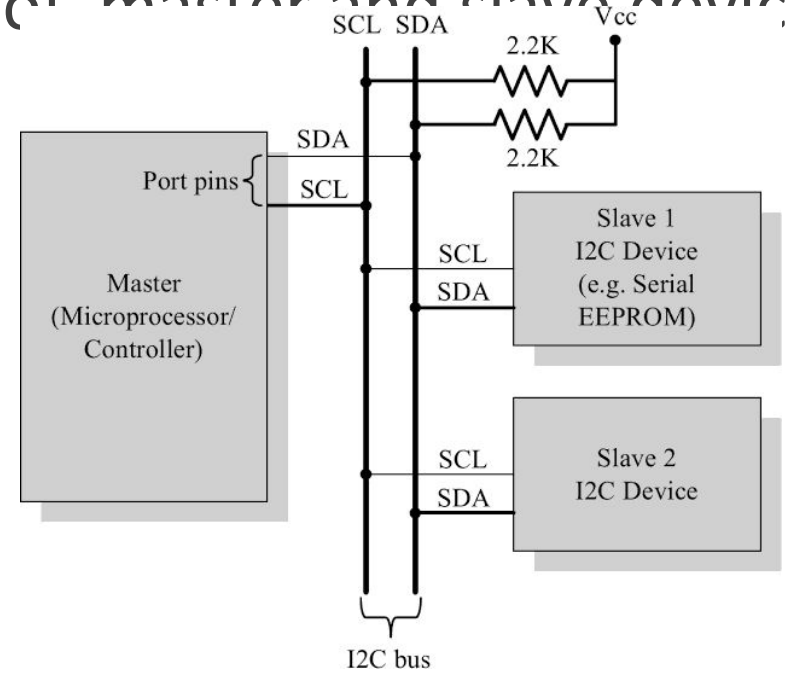
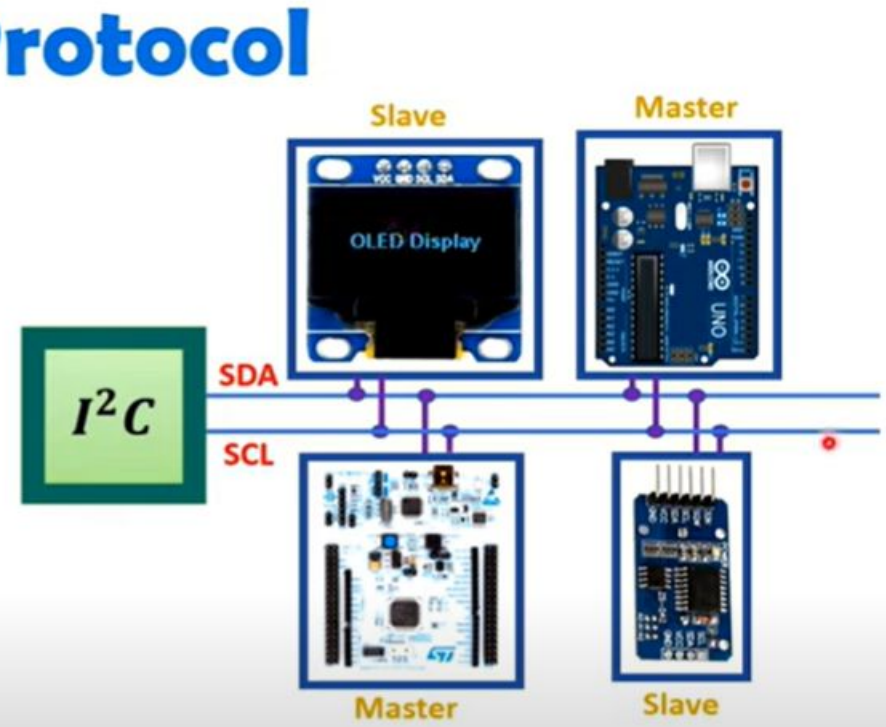


Fig: I2C Bus Interfacing



Inter Integrated Circuit (I2C) Bus (continued)

- The I2C bus interface is built around an input buffer and an open drain or collector transistor.
- When the bus is in the idle state, the open drain/collector transistor will be in the floating state and the output lines (SDA and SCL) switch to the 'High Impedance' state.
- For proper operation of the bus, both the bus lines should be pulled to the supply voltage (+5 V for TTL family and +3.3V for CMOS family devices) using pull-up resistors.
  - The typical value of resistors used in pull-up is 2.2K.
  - With pull-up resistors, the output lines of the bus in the idle state will be 'HIGH'
- The address of a I2C device is assigned by hardwiring the address lines of the device to the desired logic level.
  - Done at the time of designing the embedded hardware.

Inter Integrated Circuit (I2C) Bus (continued)

- The sequence of operations for communicating with an I2C slave device is listed below:
  1. The master device pulls the clock line (SCL) of the bus to 'HIGH'
  2. The master device pulls the data line (SDA) 'LOW', when the SCL line is at logic 'HIGH' (This is the 'Start' condition for data transfer)
  3. The master device sends the address (7 bit or 10 bit wide) of the 'slave' device to which it wants to communicate, over the SDA line.
     - Clock pulses are generated at the SCL line for synchronising the bit reception by the slave device.
     - The MSB of the data is always transmitted first.

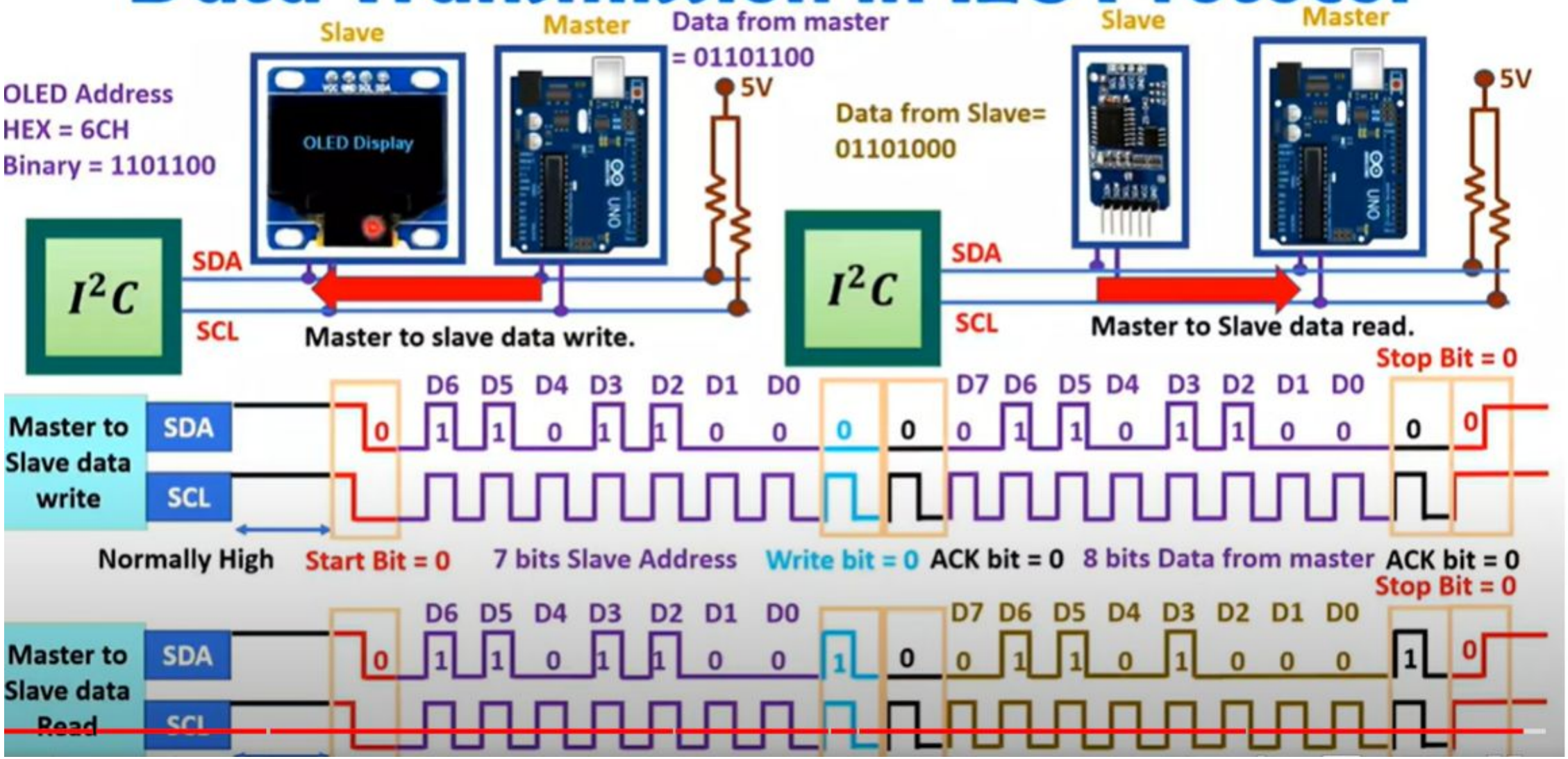
Inter Integrated Circuit (I2C) Bus (continued)

4. The master device sends the Read or Write bit (Bit value = 1 Read operation; Bit value = 0 Write operation) according to the requirement
5. The master device waits for the acknowledgement bit from the slave device whose address is sent on the bus along with the Read/ Write operation command.
   - Slave devices connected to the bus compares the address received with the address assigned to them
6. The slave device with the address requested by the master device responds by sending an acknowledge bit (Bit value 1) over the SDA line
7. Upon receiving the acknowledge bit, the Master device sends the 8 bit data to the slave device over SDA line, if the requested operation is 'Write to device'.
   - If the requested operation is 'Read from device', the slave device sends data to the master over the SDA line
8. The master device waits for the acknowledgement bit from the device upon byte transfer complete for a write operation and sends an acknowledge bit to the Slave device for a read operation
9. The master device terminates the transfer by pulling the SDA line 'HIGH' when the clock line SCL is at logic 'HIGH' (Indicating the 'STOP' condition)

Inter Integrated Circuit (I2C) Bus (continued)

- I2C bus supports three different data rates:
  - Standard mode (Data rate up to 100kbits/sec (100 kbps))
  - Fast mode (Data rate up to 400kbits/sec (400 kbps))
  - High speed mode (Data rate up to 3.4Mbits/sec (3.4 Mbps))

Data Transmission in I2C Protocol



Serial Peripheral Interface (SPI) Bus

- The Serial Peripheral Interface Bus (SPI) is a synchronous bi-directional full duplex four-wire serial interface bus.
- The concept of SPI was introduced by Motorola.
- SPI is a single master multi-slave system.
  - There can be more than one masters, but only one master device can be active at any given point of time.
- SPI requires four signal lines for communication. They are:
  - Master Out Slave In (MOSI) – Signal line carrying the data from master to slave device. It is also known as Slave Input/Slave Data In (SI/SDI)
  - Master In Slave Out (MISO) – Signal line carrying the data from slave to master device. It is also known as Slave Output (SO/SDO)
  - Serial Clock (SCLK) – Signal line carrying the clock signals
  - Slave Select (SS) – Signal line for slave device select. It is an active low signal

Serial Peripheral Interface (SPI) Bus (continued)

- The bus interface diagram shown in the figure illustrates the connection of master and slave devices on the SPI bus.

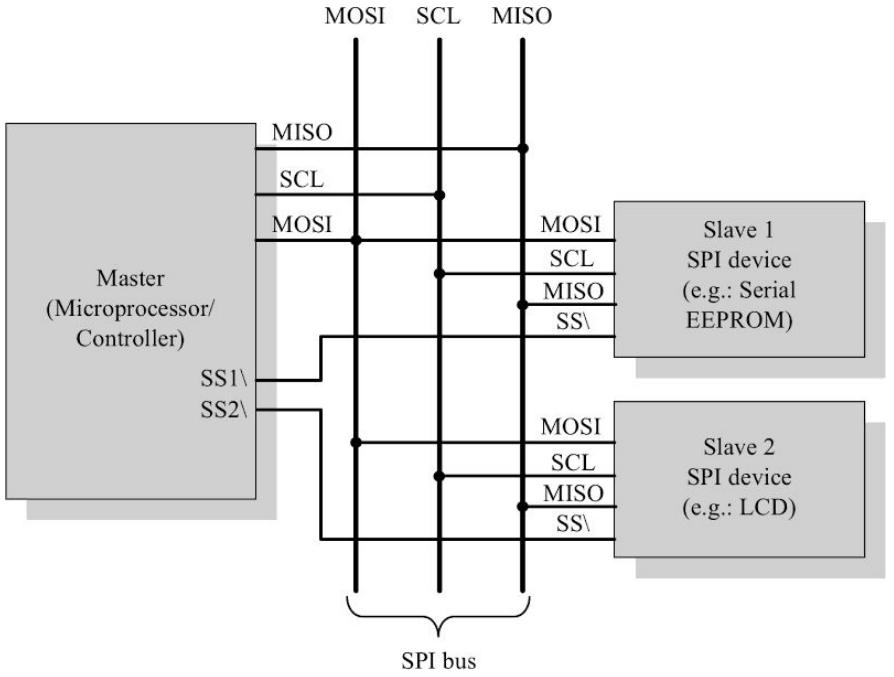


Fig: SPI Bus Interfacing

Serial Peripheral Interface (SPI) Bus (continued)

- The master device is responsible for generating the clock signal.
- It selects the required slave device by asserting the corresponding slave device's slave select signal 'LOW’.
- The data out line (MISO) of all the slave devices when not selected floats at high impedance state.
- The serial data transmission through SPI bus is fully configurable.
  - SPI devices contain a certain set of registers for holding these configurations.
  - The control register holds the various configuration parameters like master/slave selection for the device, baud rate selection for communication, clock signal control, etc.
  - The status register holds the status of various conditions for

Serial Peripheral Interface (SPI) Bus (continued)

- SPI works on the principle of 'Shift Register’.
- The master and slave devices contain a special shift register for the data to transmit
  or
  receive.
  - The size of the shift register is device dependent. Normally it is a multiple of 8.
- During transmission from the master to slave, the data in the master's shift register is shifted out to the MOSI pin and it enters the shift register of the slave device through the MOSI pin of the slave device.
- At the same time the shifted out data bit from the slave device's shift register enters the shift register of the master device through MISO pin.
- In summary, the shift registers of 'master' and 'slave' devices form a circular buffer.
- When compared to I2C, SPI bus is most suitable for applications requiring transfer of data in ' streams '.
- The only limitation is SPI doesn't support an acknowledgement mechanism.

Universal Asynchronous Receiver Transmitter (UART)

- Universal Asynchronous Receiver Transmitter (UART) based data transmission is an asynchronous form of serial data transmission.
- It doesn't require a clock signal to synchronise the transmitting end and receiving end for transmission.
- Instead it relies upon the pre-defined agreement between the transmitting device and receiving device.

Universal Asynchronous Receiver Transmitter (UART) (continued)

- The serial communication settings (Baudrate, number of bits per byte, parity, number of start bits and stop bit and flow control) for both transmitter and receiver should be set as identical.
- The start and stop of communication is indicated through inserting special bits in the data stream.
- While sending a byte of data, a start bit is added first and a stop bit is added at the end of the bit stream.
- The least significant bit of the data byte follows the 'start' bit.

Universal Asynchronous Receiver Transmitter (UART) (continued)

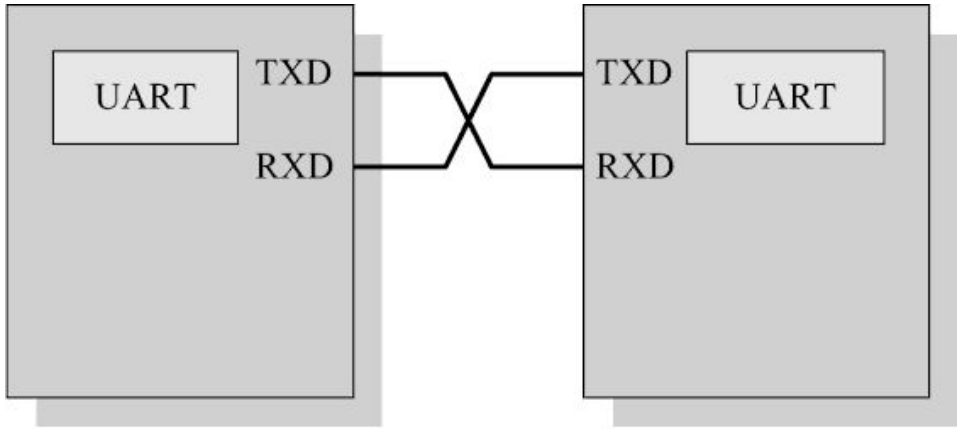
- The 'start' bit informs the receiver that a data byte is about to arrive.
- The receiver device starts polling its 'receive line' as per the baud rate settings.
  - If the baud rate is 'x' bits per second, the time slot available for one bit is 1/x seconds.
- The receiver unit polls the receiver line at exactly half of the time slot available for the bit.
- If parity is enabled for communication, the UART of the transmitting device adds a parity bit (bit value is 1 for odd number of 1s in the transmitted bit stream and 0 for even number of 1s).
- The UART of the receiving device calculates the parity of the bits received and compares it with the received parity bit for error checking.
- The UART of the receiving device discards the 'Start', 'Stop' and 'Parity' bit from the received bit stream and converts the received serial bit data to a word
  - In the case of 8 bits/byte, the byte is formed with the received 8 bits with the first received bit as the LSB and last received data bit as MSB.

Universal Asynchronous Receiver Transmitter (UART) (continued)

- For proper communication, the 'Transmit line' of the sending device should be connected to the 'Receive line' of the receiving device.
- In addition to the serial data transmission function, UART provides hardware handshaking signal support for controlling the serial data flow.
- UART chips are available from different semiconductor manufacturers.
  - National Semiconductor's 8250 UART chip is considered as the standard setting UART. It was used in the original IBM PC.
- Nowadays most of the microprocessors/controllers are available with integrated UART functionality and they provide built-in

Universal Asynchronous Receiver Transmitter (UART) (continued)

- Figure illustrates the UART interfacing.



TXD: Transmitter line
RXD: Receiver line

Fig: UART Interfacing

1-Wire Interface

- 1-wire interface is an asynchronous half-duplex communication protocol developed by Maxim Dallas Semiconductor.
- It is also known as Dallas 1-Wire protocol.
- It makes use of only a single signal line (wire) called DQ for communication and follows the master-slave communication model.
- One of the key feature of 1-wire bus is that it allows power to be sent along the signal wire as well.
- The slave devices incorporate internal capacitor (typically of the order of 800 pF) to power the device from the signal line.
- The 1-wire interface supports a single master and one or more slave devices on the bus.

1-Wire Interface (continued)

- The bus interface diagram shown in the figure illustrates the connection of master and slave devices on the 1-wire bus.

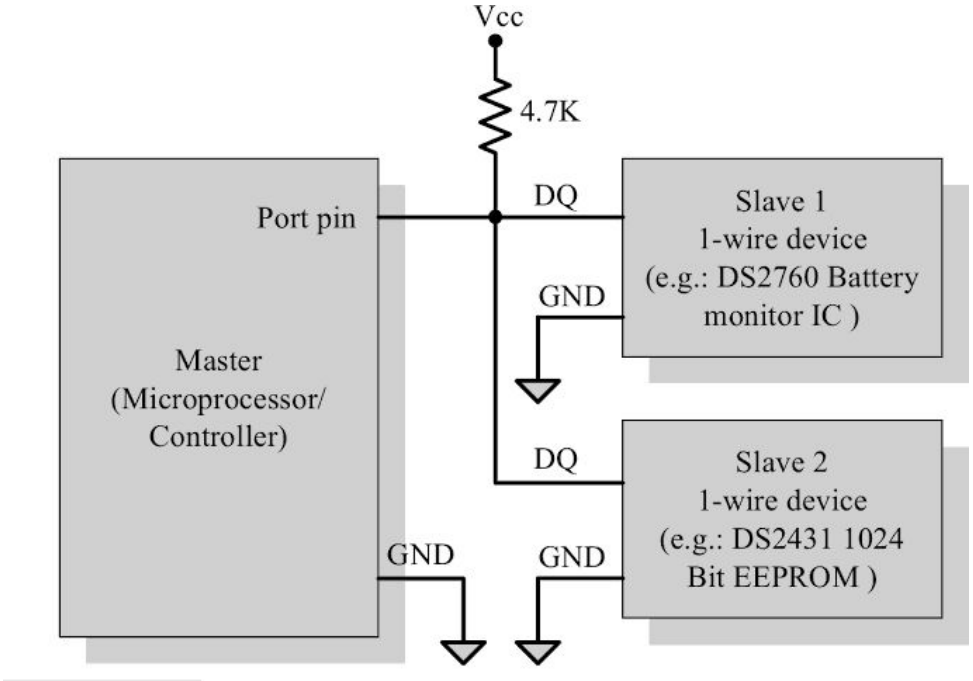


Fig: 1-Wire Interface Bus

1-Wire Interface (continued)

- Every 1-wire device contains a globally unique 64bit identification number stored within it.
- This unique identification number can be used for addressing individual devices present on the bus in case there are multiple slave devices connected to the 1-wire bus.
- The identifier has three parts: an 8 bit family code, a 48 bit serial number and an 8 bit CRC computed from the first 56 bits.

1-Wire Interface (continued)

- The sequence of operation for communicating with a 1-wire slave device is listed below:

1. The master device sends a 'Reset' pulse on the 1-wire bus.
2. The slave device(s) present on the bus respond with a 'Presence' pulse.
3. The master device sends a ROM command (Net Address Command followed
   by the 64 bit address of the device).
   - This addresses the slave device(s) to which it wants to initiate a communication.
4. The master device sends a read/write function command to read/write the internal memory or register of the slave device.
5. The master initiates a Read data/Write data from the device or to the device.

1-Wire Interface (continued)

- All communication over the 1 -wire bus is master initiated.
- The communication over the 1-wire bus is divided into timeslots of 60 microseconds.
- The 'Reset' pulse occupies 8 time slots. For starting a communication, the master asserts the reset pulse by pulling the 1-wire bus 'LOW' for at least 8 time slots (480 µs).
- If a 'slave' device is present on the bus and is ready for communication it should respond to the master with a 'Presence' pulse, within 60 µs of the release of the 'Reset' pulse by the master.
- The slave device(s) responds with a 'Presence' pulse by pulling the 1-wire bus 'LOW' for a minimum of 1 time slot (60 µs).

1-Wire Interface (continued)

- For writing a bit value of 1 on the 1-wire bus, the bus master pulls the bus for 1 to 15 µs and then releases the bus for the rest of the time slot.
  - A bit value of ‘0' is written on the bus by master pulling the bus for a minimum of 1 time slot (60 µs) and a maximum of 2 time slots (120 µs).
- To Read a bit from the slave device, the master pulls the bus 'LOW' for 1 to 15 µs.
  - If the slave wants to send a bit value ‘1' in response to the read request from
    the master, it simply releases the bus for the rest of the time slot.
  - If the slave wants to send a bit value '0', it pulls the bus 'LOW' for the rest

Parallel Interface

- The on board parallel interface is normally used for communicating with peripheral devices which are memory mapped to the host of the system.
- The host processor/controller of the embedded system contains a parallel bus and the device which supports parallel bus can directly connect to this bus system.
- The communication through the parallel bus is controlled by the control signal interface between the device and the host.
  - The Control Signals for communication includes Read/Write signal and device select signal.
- The device normally contains a device select line and the device becomes active only when this line is asserted by the host processor.
- The direction of data transfer (Host to Device or Device to Host) can be controlled through the control signal lines for 'Read' and 'Write'.
- Only the host processor has control over the 'Read' and 'Write' control signals.

Parallel Interface (continued)

- The device is normally memory mapped to the host processor and a range of address is assigned to it.
- An address decoder circuit is used for generating the chip select signal for the device.
- When the address selected by the processor is within the range assigned for the device, the decoder circuit activates the chip select line and thereby the device becomes active.
- The processor then can read or write from or to the device by asserting the corresponding control line (RD\ and WR\ respectively).

Parallel Interface (continued)

- The bus interface diagram shown in the figure illustrates the interfacing of devices through parallel interface.

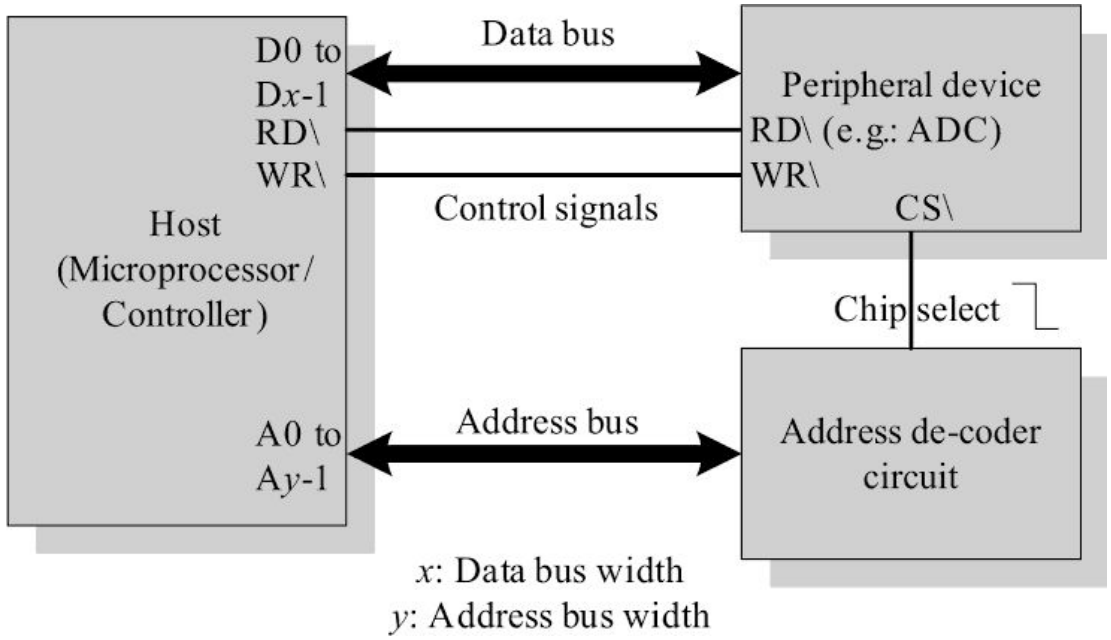


Fig: Parallel Interface Bus

Parallel Interface (continued)

- Parallel communication is host processor initiated.
- If a device wants to initiate the communication, it can inform the same to the processor through interrupts.
  - For this, the interrupt line of the device is connected to the interrupt line of the processor and the corresponding interrupt is enabled in the host processor.
- The width of the parallel interface is determined by the data bus width of the host processor.
  - It can be 4 bit, 8 bit, 16 bit, 32 bit or 64 bit etc.
  - The bus width supported by the device should be same as that of the host processor.
- Parallel data communication offers the highest speed for data transfer.

External Communication Interfaces

- External Communication Interface refers to the different communication channels/buses used by the embedded system to communicate with the external world.
- E.g.: RS-232 C & RS-485, Universal Serial Bus (USB), IEEE 1394 (Firewire), Infrared (IR), Bluetooth (BT), Wi-Fi, ZigBee, GPRS, etc.

RS-232 C & RS-485

- RS-232 (Recommended Standard number 232, revision C) is a legacy, full duplex, wired, asynchronous serial communication interface.
- The RS-232 interface was developed by the Electronics Industries Association (EIA) during the early 1960s.
- RS-232 extends the UART communication signals for external data communication.
- UART uses the standard TTL/CMOS logic (Logic ‘High’ corresponds to bit value 1 and Logic ‘Low’ corresponds to bit value 0) for bit transmission whereas RS-232 follows the EIA standard for bit transmission.
  - As per the EIA standard, a logic ‘0’ is represented with voltage between +3 and +25V and a logic ‘1’ is represented with voltage between -3 and

RS-232 C & RS-485 (continued)

- The RS232 interface defines various handshaking and control signals for communication apart from the 'Transmit' and 'Receive' signal lines for data communication.
- RS-232 supports two different types of connectors:
  - DB-9: 9-Pin connector
  - DB-25: 25-Pin connector.

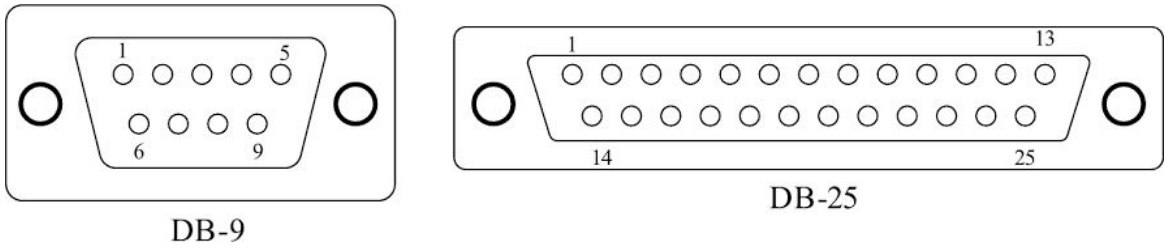


Fig: DB-9 and DB-25 RS-232 Connector Interface

RS-232 C & RS-485 (continued)

- The pin details for the two connectors are explained in the following table

Pin Name	Pin no: (For DB-9 Connector)	Pin no: (For DB-25 Connector)	Description
TXD	3	2	Transmit Pin for Transmitting Serial Data
RXD	2	3	Receive Pin for Receiving Serial Data
RTS	7	4	Request to send.
CTS	8	5	Clear To Send
DSR	6	6	Data Set Ready
GND	5	7	Signal Ground
DCD	1	8	Data Carrier Detect
DTR	4	20	Data Terminal Ready
RI	9	22	Ring Indicator
FG		1	Frame Ground
SDCD		12	Secondary DCD
SCTS		13	Secondary CTS
STXD		14	Secondary TXD
TC		15	Transmission Signal Element Timing
SRXD		16	Secondary RXD
RC		17	Receiver Signal Element Timing
SRTS		19	Secondary RTS
SQ		21	Signal Quality detector
NC		9	No Connection
NC		10	No Connection
NC		11	No Connection
NC		18	No Connection
NC		23	No Connection
NC		24	No Connection
NC		25	No Connection

Pin Name	Pin no: (For DB-9 Connector)	Pin no: (For DB-25 Connector)	Description
TXD	3	2	Transmit Pin for Transmitting Serial Data
RXD	2	3	Receive Pin for Receiving Serial Data
RTS	7	4	Request to send.
CTS	8	5	Clear To Send
DSR	6	6	Data Set Ready
GND	5	7	Signal Ground
DCD	1	8	Data Carrier Detect
DTR	4	20	Data Terminal Ready
RI	9	22	Ring Indicator
FG		1	Frame Ground
SDCD		12	Secondary DCD
SCTS		13	Secondary CTS
STXD		14	Secondary TXD
TC		15	Transmission Signal Element Timing
SRXD		16	Secondary RXD
RC		17	Receiver Signal Element Timing
SRTS		19	Secondary RTS
SQ		21	Signal Quality detector
NC		9	No Connection
NC		10	No Connection
NC		11	No Connection
NC		18	No Connection
NC		23	No Connection
NC		24	No Connection
NC		25	No Connection

RS-232 C & RS-485 (continued)

- RS-232 is a point-to-point communication interface and the devices involved in RS-232 communication are called 'Data Terminal Equipment (DTE)' and 'Data Communication Equipment (DCE)'.
- If no data flow control is required, only TXD and RXD signal lines and ground line (GND) are required for data transmission and reception.
  - The RXD pin of DCE should be connected to the TXD pin of DTE and vice versa for proper data transmission.
- If hardware data flow control is required for serial transmission, various control signal lines of the RS-232 connection are used appropriately.

RS-232 C & RS-485 (continued)

- The Request To Send (RTS) and Clear To Send (CTS) signals co-ordinate the communication between DTE and DCE.
  - Whenever the DTE has a data to send, it activates the RTS line and if the DCE is ready to accept the data, it activates the CTS line.
- The Data Terminal Ready (DTR) signal is activated by DTE when it is ready to accept data.
- The Data Set Ready (DSR) is activated by DCE when it is ready for establishing a communication link.
  - DTR should be in the activated state before the activation of DSR.
- The Data Carrier Detect (DCD) control signal is used by the DCE to indicate the DTE that a good signal is being received.
- Ring Indicator (RI) is a modem specific signal line for indicating an incoming call on the telephone line.

RS-232 C & RS-485 (continued)

- As per the EIA standard RS-232 C supports baudrates up to 20Kbps (Upper limit 19.2 Kbps)
  - The commonly used baudrates by devices are 300bps, 1200bps, 2400bps, 9600bps, 11.52Kbps and 19.2Kbps.
  - 9600 is the popular baudrate setting used for PC communication.
- The maximum operating distance supported by RS-232 is 50 feet at the highest supported baudrate.
- Embedded devices contain a UART for serial communication and they generate signal levels conforming to TTL/CMOS logic.
  - A level translator IC like MAX 232 from Maxim Dallas semiconductor is used for converting the signal lines from the UART to RS-232 signal lines for communication.

RS-232 C & RS-485 (continued)

- RS-232 supports only point-to-point communication and not suitable for multi- drop communication.
  - It uses single ended data transfer technique for signal transmission and thereby more susceptible to noise and it greatly reduces the operating distance.
- RS-422 is another serial interface standard from EIA for differential data communication.
  - It supports data rates up to 100Kbps and distance up to 400 ft.
- RS-422 supports multi-drop communication with one transmitter device and receiver devices up to 10.
- 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.
  - The communication between devices in the bus uses the 'addressing' mechanism to identify slave devices.

Universal Serial Bus (USB)

- Universal Serial Bus (USB) is a wired high speed serial bus for data communication.
- The first version of USB (USB 1.0) was released in 1995.
- The USB communication system follows a star topology with a USB host at the centre and one or more USB peripheral devices/USB hosts connected to it.
- A USB host can support connections up to 127, including slave peripheral devices and other USB hosts.

Universal Serial Bus (USB) (continued)

- Figure illustrates the star topology for USB device connection.

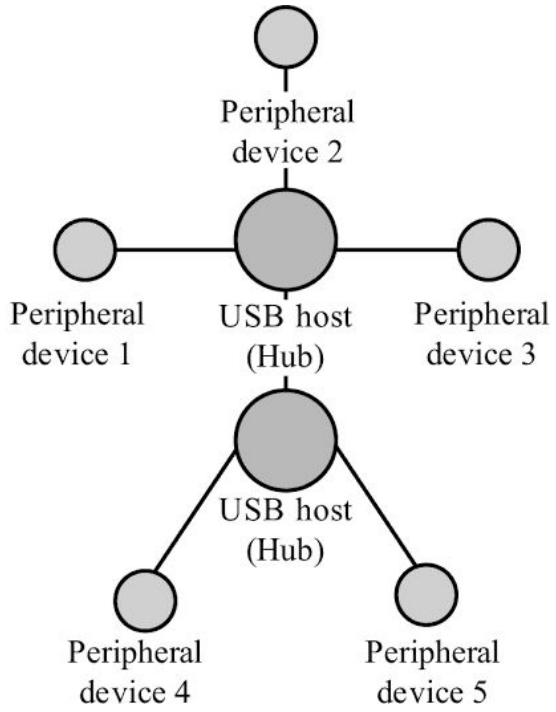


Fig: USB Device Connection topology

Universal Serial Bus (USB) (continued)

- USB transmits data in packet format.
- Each data packet has a standard format.
- The USB communication is a host initiated one.
- The USB host contains a host controller which is responsible for controlling the data communication, including establishing connectivity with USB slave devices, packetizing and formatting the data.
- There are different standards for implementing the USB Host Control interface:
  - Open Host Control Interface (OHCI)
  - Universal Host Control Interface (UHCI)

Universal Serial Bus (USB) (continued)

- The physical connection between a USB peripheral device and master device is established with a USB cable.
- The USB cable supports communication distance of up to 5 metres.
- The USB standard uses two different types of connector at the ends of the USB cable for connecting the USB peripheral device and host device.
- 'Type A' connector is used for upstream connection (connection with host) and Type B connector is used for downstream connection (connection with slave device).
- The USB connector present in desktop PCs or laptops are examples

Universal Serial Bus (USB) (continued)

- Both Type A and Type B connectors contain 4 pins for communication.
- The Pin details for the connectors are listed in the table given below.

Pin no:	Pin name	Description
1	V_{BUS}	Carries power (5V)
2	D–	Differential data carrier line
3	D+	Differential data carrier line
4	GND	Ground signal line

Universal Serial Bus (USB) (continued)

Type A Connector

Type B Connector

Type C Connector

Universal Serial Bus (USB) (continued)

- USB uses differential signals for data transmission.
  - It improves the noise immunity.
- USB interface has the ability to supply power to the connecting devices.
  - Two connection lines (Ground and Power) of the USB interface are dedicated for carrying power.
  - It can supply power up to 500 mA at 5 V.
  - It is sufficient to operate low power devices.
- Mini and Micro USB connectors are available for small form factor devices like portable media players.
- Each USB device contains a Product ID (PID) and a Vendor ID (VID).
  - Embedded into the USB chip by the USB device manufacturer.
  - The VID for a device is supplied by the USB standards forum.
  - PID and VID are essential for loading the drivers corresponding to a USB device for communication.

Universal Serial Bus (USB) (continued)

- USB supports four different types of data transfers:
- Control transfer : Used by USB system software to query, configure and issue commands to the USB device.
- Bulk transfer : Used for sending a block of data to a device.
  - Supports error checking and correction.
  - Transferring data to a printer is an example for bulk transfer.
- Isochronous data transfer : Used for real-time data communication.
  - Data is transmitted as streams in real-time.
  - Doesn't support error checking and re-transmission of data in case of any transmission loss.
  - All streaming devices like audio devices and medical equipment for data collection make use of the isochronous transfer.
- Interrupt transfer : Used for transferring small amount of data.
  - Interrupt transfer mechanism makes use of polling technique to see whether the USB device has any data to send.
  - The frequency of polling is determined by the USB device and it varies from 1 to 255 milliseconds.
  - Devices like Mouse and Keyboard, which transmits fewer amounts of data, uses Interrupt transfer.

Universal Serial Bus (USB) (continued)

- USB-IF is the standards body for defining and controlling the standards for USB communication.
- Presently USB supports four different data rates:
  - Low Speed (1.5Mbps) – USB 1.0
  - Full Speed (12Mbps) – USB 1.0
  - High Speed (480Mbps) – USB 2.0
  - Super Speed (4.8Gbps) – USB 3.0

IEEE 1394 (Firewire)

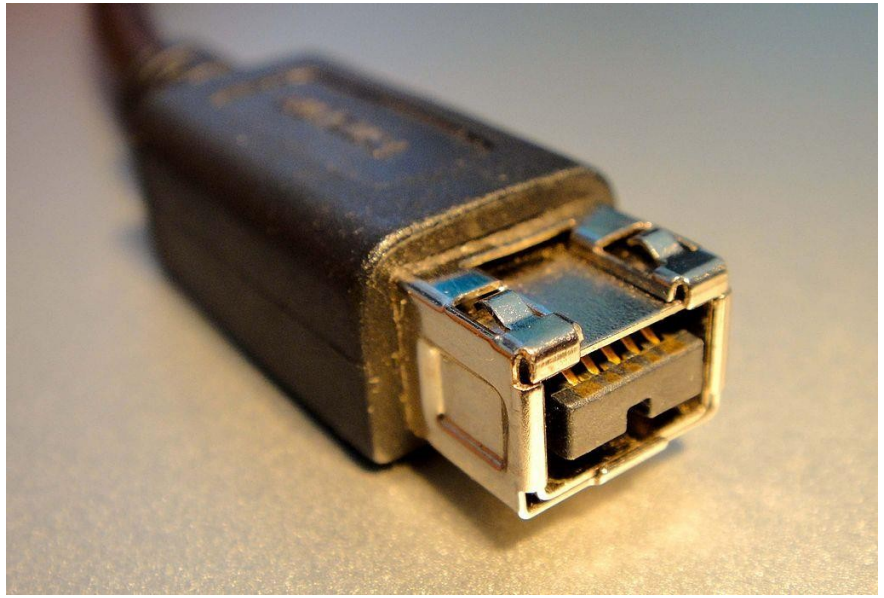
- IEEE 1394 is a wired, isochronous high speed serial communication bus.
- It is also known as High Performance Serial Bus (HPSB).
- The research on 1394 was started by Apple Inc. in 1985 and the standard for this was coined by IEEE.
- The implementation of 1394 is available from various players with different names:
  - Firewire is the implementation from Apple Inc
  - i.LINK is the implementation from Sony Corporation
  - Lynx is the implementation from Texas Instruments

IEEE 1394 (Firewire) (continued)

- 1394 supports peer-to-peer connection and point-to-multipoint communication allowing 63 devices to be connected on the bus in a tree topology.
- 1394 is a wired serial interface and it can support a cable length of up to 15 feet for interconnection.
- The 1394 standard supports a data rate of 400 to 3200 Mbits/second.
- The IEEE 1394 uses differential data transfer.
  - It increases the noise immunity.
- The interface cable supports 3 types of connectors, namely; 4-pin connector, 6-pin connector (alpha connector) and 9 pin connector (beta connector).
  - It can supply unregulated power in the range of 24 to 30V.
- The 6 and 9 pin connectors carry power also to support external devices.

IEEE 1394 (Firewire) (continued)

4-pin and 6-pin Connectors



9-pin Connector

IEEE 1394 (Firewire) (continued)

- The table given below illustrates the pin details for 4, 6 and 9 pin connectors.

Pin name	Pin no: (4 Pin Connector)	Pin no: (6 Pin Connector)	Pin no: (9 Pin Connector)	Description
Power		1	8	Unregulated DC supply. 24 to 30V
Signal Ground		2	6	Ground connection
TPB–	1	3	1	Differential Signal line for Signal line B
TPB+	2	4	2	Differential Signal line for Signal line B
TPA–	3	5	3	Differential Signal line for Signal line A
TPA+	4	6	4	Differential Signal line for Signal line A
TPA(S)			5	Shield for the differential signal line A. Normally grounded
TPB(S)			9	Shield for the differential signal line B. Normally grounded
NC			7	No connection

IEEE 1394 (Firewire) (continued)

- There are two differential data transfer lines A and B per connector.
- In a 1394 cable, normally the differential lines of A are connected to B (TPA+ to TPB+ and TPA– to TPB–) and vice versa.
- 1394 is a popular communication interface for connecting embedded devices like Digital Camera, Camcorder, Scanners to desktop computers for data transfer and storage.
- 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 USB2.0 interface.
- The 1394 hardware implementation is much costlier than USB implementation.

Infrared (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.
  - E.g.: The remote control of TV, VCD player, etc. works on Infrared.
- Infrared communication technique uses infrared waves of the electromagnetic spectrum for transmitting the data.
- It supports point-point and point-to-multipoint communication, provided all devices involved in the communication are within the line of sight.
- The typical communication range for IrDA lies in the range 10 cm to 1 m.
- The range can be increased by increasing the transmitting power of the IR device.

Infrared (IrDA) (continued)

- It supports data rates ranging from 9600bits/second to 16Mbps.
- Depending on the speed of data transmission IR is classified into:
  - Serial IR (SIR) – supports data rates ranging from 9600bps to 115.2kbps.
  - Medium IR (MIR) – supports data rates of 0.576Mbps and 1.152Mbps.
  - Fast IR (FIR) – supports data rates up to 4Mbps.
  - Very Fast IR (VFIR) – supports high data rates up to 16Mbps.
  - Ultra Fast IR (UFIR) – targeted to support a data rate up to 100Mbps.

Infrared (IrDA) (continued)

- IrDA communication involves a transmitter unit for transmitting the data over IR and a receiver for receiving the data.
- Infrared Light Emitting Diode (LED) is the IR source for transmitter and at the receiving end a photodiode acts as the receiver.
- Both transmitter and receiver unit will be present in each device supporting IrDA communication for bidirectional data transfer.
  - Such IR units are known as 'Transceiver'.
- Certain devices like a TV remote control always require unidirectional communication and so they contain either the transmitter or receiver unit.
  - The remote control unit contains the transmitter unit and TV contains

Infrared (IrDA) (continued)

- Infrared Data Association (IrDA) is the regulatory body responsible for defining and licensing the specifications for IR data communication.
- IR communication has two essential parts: a physical link part and a protocol part.
  - The physical link is responsible for the physical transmission of data between devices supporting IR communication
  - Protocol part is responsible for defining the rules of communication.
- The physical link works on the wireless principle making use of Infrared for communication.
- The IrDA specifications include the standard for both physical link and protocol layer.
- The IrDA control protocol contains implementations for Physical Layer (PHY), Media Access Control (MAC) and Logical Link Control (LLC).

Infrared (IrDA) (continued)

- IrDA is a popular interface for file exchange and data transfer in low cost devices.
- IrDA was the prominent communication channel in mobile phones before Bluetooth's existence.

Bluetooth (BT)

- Bluetooth is a low cost, low power, short range wireless technology for data and voice communication.
- Bluetooth was first proposed by Ericsson in 1994.
- Bluetooth operates at 2.4GHz of the Radio Frequency spectrum and uses the Frequency Hopping Spread Spectrum (FHSS) technique for communication.
- It supports a data rate of up to 1Mbps and a range of approximately 30 feet for data communication.

Bluetooth (BT) (continued)

- Bluetooth communication has two essential parts – a physical link part and a protocol part.
  - The physical link is responsible for the physical transmission of data between devices supporting Bluetooth communication
  - The protocol part is responsible for defining the rules of communication.
- The physical link works on the wireless principle making use of RF waves for communication.
- Bluetooth enabled devices essentially contain a Bluetooth wireless radio for the transmission and reception of data.

Bluetooth (BT) (continued)

- The rules governing the Bluetooth communication is implemented in the 'Bluetooth protocol stack'.
  - The Bluetooth communication IC holds the stack.
- Each Bluetooth device will have a 48 bit unique identification number.
- Bluetooth communication follows packet based data transfer.
- Bluetooth supports point-to-point (device to device) and point-to-multipoint (device to multiple device broadcasting) wireless communication.
- The point-to-point communication follows the master-slave relationship.
- 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 (BT) (continued)

- 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 neighbouring Bluetooth enabled phones.
- The Bluetooth standard specifies the minimum requirements that a Bluetooth device
  must support for a specific usage scenario.
- The Generic Access Profile (GAP) defines the requirements for detecting a Bluetooth device and establishing a connection with it.
  - All other specific usage profiles are based on GAP.
  - Serial Port Profile (SPP) for serial data communication, File Transfer Profile (FTP) for file transfer between devices, Human Interface Device (HID) for supporting human interface devices like keyboard and mouse are examples for Bluetooth profiles.
- The specifications for Bluetooth communication is defined and licensed by the standards body 'Bluetooth Special Interest Group (SIG)'.

Wi-F

- Wi-Fi or Wireless Fidelity is the popular wireless communication technique for networked communication of devices.
- Wi-Fi follows the IEEE 802.11 standard.
- 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.
- In an IP based communication each device is identified by an IP address, which
  is unique to each device on the network.

Wi-Fi (continued)

- Wi-Fi based communications require an intermediate agent called Wi-Fi router/Wireless Access point to manage the communications.
- The Wi-Fi router is responsible for restricting the access to a network, assigning IP address to devices on the network, routing data packets to the intended devices on the network.
- Wi-Fi enabled devices contain a wireless adaptor for transmitting and receiving data in the form of radio signals through an antenna.
- The hardware part of it is known as Wi-Fi Radio.
- Wi-Fi operates at 2.4 GHz or 5 GHz of radio spectrum and they co-exist with other ISM band devices like Bluetooth.

Wi-Fi (continued)

- Figure illustrates the typical interfacing of devices in a Wi-Fi network.

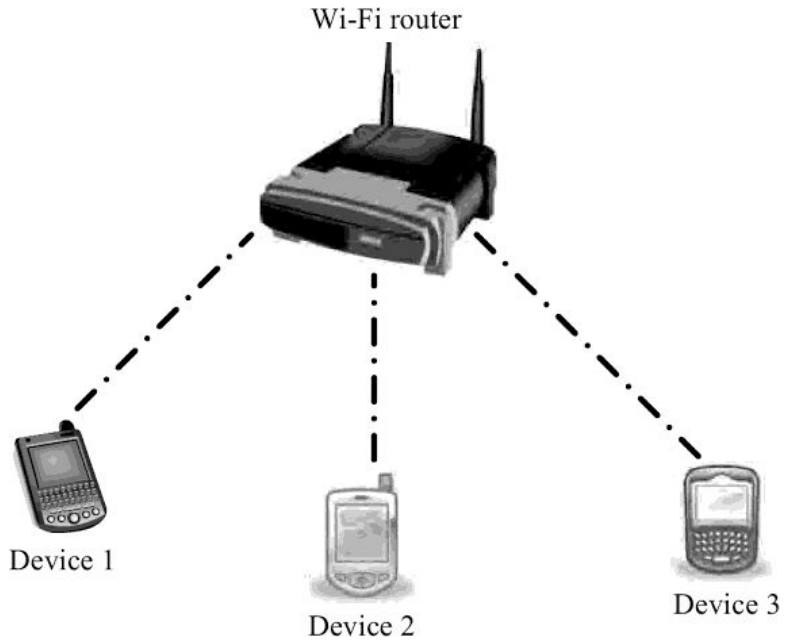


Fig: Wi-Fi Network

Wi-Fi (continued)

- For communicating with devices over a Wi-Fi network, the device when its Wi-Fi radio is turned ON, searches the available Wi-Fi network in its vicinity and lists out the Service Set Identifier (SSID) of the available networks.
- If the network is security enabled, a password may be required to connect to a particular SSID.
- Wi-Fi employs different security mechanisms like Wired Equivalency Privacy (WEP), Wireless Protected Access (WPA), etc. for securing the data communication.
- Wi-Fi supports data rates ranging from 1 Mbps to 1.73 Gbps depending on the standards (802.11a/b/g/n) and access/modulation method.
- Depending on the type of antenna and usage location (indoor/outdoor), Wi-Fi offers a range of 100 to 300 feet.

ZigBe

- 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).
- The ZigBee specifications support a robust mesh network containing multiple nodes.
- This networking strategy makes the network reliable by permitting messages to travel through a number of different paths to get from one node to another.
- 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 metres and a data rate

ZigBee (continued)

- In the ZigBee terminology, each ZigBee device falls under any one of the following ZigBee device category:
- ZigBee Coordinator (ZC)/Network Coordinator
  - The ZigBee coordinator acts as the root of the ZigBee network.
  - The ZC is responsible for initiating the ZigBee network and it has the capability to store information about the network.
- ZigBee Router (ZR)/Full function Device (FFD)
  - Responsible for passing information from device to another device or to another ZR.
- ZigBee End Device (ZED)/Reduced Function Device (RFD):
  - End device containing ZigBee functionality for data communication.
  - It can talk only with a ZR or ZC and doesn't have the capability to act as a mediator for transferring data from one device to another.

ZigBee (continued)

- The diagram shown in figure gives an overview of ZC, ZED and ZR in a ZigBee network.

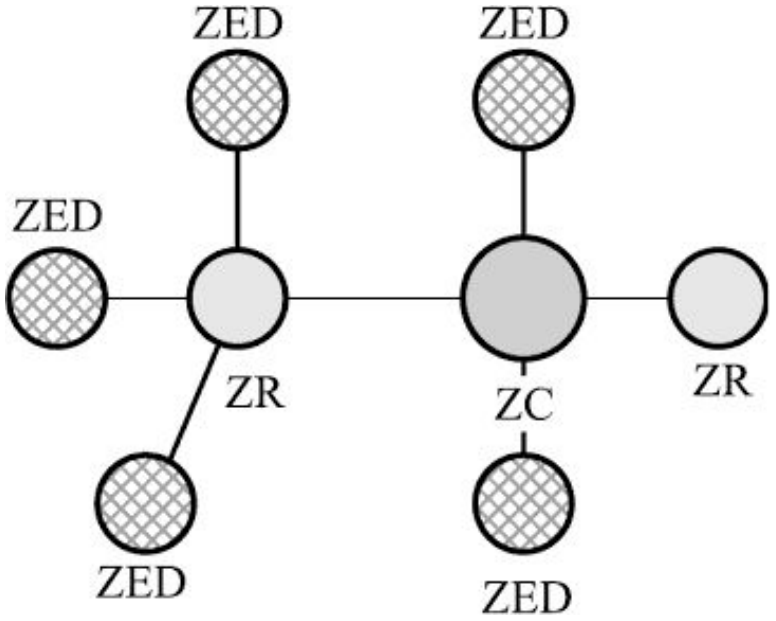


Fig: A ZigBee network model

ZigBee (continued)

- ZigBee is primarily targeting application areas like home & industrial automation, energy management, home control/security, medical/patient tracking, logistics & asset tracking and sensor networks & active RFID.
- Automatic Meter Reading (AMR), smoke detectors, wireless telemetry, HVAC control, heating control, lighting controls, environmental controls, etc. are examples for applications which can make use of the ZigBee technology.
- The specifications for ZigBee is developed and managed by the ZigBee Alliance, a non-profit consortium of leading semiconductor manufacturers, technology providers, OEMs and end-users worldwide.

General Packet Radio Service (GPRS)

- General Packet Radio Service (GPRS) is a communication technique for transferring data over a mobile communication network like GSM.
- Data is sent as packets in GPRS communication.
- The transmitting device splits the data into several related packets.
- At the receiving end the data is re-constructed by combining the received data packets.
- GPRS supports a theoretical maximum transfer rate of 171.2 kbps.
- In GPRS communication, the radio channel is concurrently shared between several users instead of dedicating a radio channel to a cell phone user.

General Packet Radio Service (GPRS) (continued)

- The GPRS communication divides the channel into 8 timeslots and transmits data over
  the available channel.
- GPRS supports Internet Protocol (IP), Point to Point Protocol (PPP) and X.25 protocols for communication.
- GPRS is mainly used by mobile enabled embedded devices for data communication.
- The device should support the necessary GPRS hardware like GPRS modem and GPRS radio.
- To accomplish GPRS based communication, the carrier network also should have support for GPRS communication.
- GPRS is an old technology and it is being replaced by new generation data communication techniques like EDGE, High Speed Downlink Packet Access (HSDPA), Long Term Evolution (LTE), etc. which offers higher bandwidths for

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:

1. Write the program in high level languages like Embedded C/C++ using an Integrated Development Environment (IDE).
   - The IDE will contain an editor, compiler, linker, debugger, simulator, etc. IDES are different for different family of processors/controllers.
   - For example, Keil µVision 4 IDE is used for all family members of 8051 microcontroller, since it contains the generic 8051 compiler C51.
2. Write the program in Assembly language using the instructions supported by your application's target processor/controller.

Embedded Firmware (continued)

- The program written in high level language or assembly code 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'.
- The methods used for 'HEX File Creation' is different depending on the programming techniques used.
  - 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 Assembly language based programming technique is used, the utilities supplied by the processor/controller vendors can be used to convert the source code into 'HEX File'.
  - Also third party tools are available, which may be of free of cost, for this conversion.

Embedded Firmware (continued)

- For a beginner in the embedded software field, it is strongly recommended to use the high level language based development technique.
  - Writing codes in a high level language is easy
  - The code written in high level language is highly portable
    - The same code can be used to run on different processor/controller with little or less modification.
    - The only thing you need to do is re-compile the program with the required processor's IDE, after replacing the include files for that particular processor.
  - The programs written in high level languages are not developer dependent.
    - Any skilled programmer can trace out the functionalities of the program by just having a look at the program.
    - It will be much easier if the source code contains necessary comments and documentation lines.
  - It is very easy to debug and the overall system development time will be reduced to a greater extent.

Embedded Firmware (continued)

- The embedded software development process in assembly language is tedious and time consuming.
- The developer needs to know about all the instruction sets of the processor/controller or at least he should carry an instruction set reference manual with him.
- A programmer using assembly language technique writes the program according to his view and taste.
- Often he may be writing a method or functionality which can be achieved through a single instruction as an experienced person's point of view, by two or three instructions in his own style.
- So the program will be highly dependent on the developer.
- It is very difficult for a second person to understand the code written in Assembly even if it is well documented.

Embedded Firmware (continued)

- Two types of control algorithm design exist in embedded firmware development:
  - The first type of control algorithm development is known as the infinite loop or 'super loop' based approach, where the control flow runs from top to bottom and then jumps back to the top of the program in a conventional procedure.
    - It is similar to the *while (1) { };* based technique in C.
  - The second method deals with splitting the functions to be executed into tasks and running these tasks using a scheduler which is part of a General Purpose or Real Time Embedded Operating System (GPOS/RTOS).

Other System Components

- The other system components refer to the components/circuits/ICs which are necessary for the proper functioning of the embedded system.
- Some of these circuits may be essential for the proper functioning of the processor/controller and firmware execution.
- E.g.: Watchdog timer, Reset IC (or passive circuit), brown-out protection IC (or passive circuit), etc.
- Some of the controllers or SoCs integrate these components within a single IC and doesn't require such components externally connected to the chip for proper functioning.

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
  - Normally from vector address 0x0000 for conventional processors/controllers.
- The reset signal can be either active high or active low.
- Since the processor operation is synchronised to a clock signal, the reset pulse should be wide enough to give time for the clock oscillator to stabilise before the internal reset state starts.

Reset Circuit (continued)

- The reset signal to the processor can be applied at power ON through an external passive reset circuit comprising a Capacitor and Resistor or through a standard Reset IC like MAX810 from Maxim Dallas.
- Select the reset IC based on the type of reset signal and logic level (CMOS/TTL) supported by the processor/controller in use.
- Some microprocessors/controllers contain built-in internal reset circuitry and they don't require external reset circuitry.

Reset Circuit (continued)

- 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.

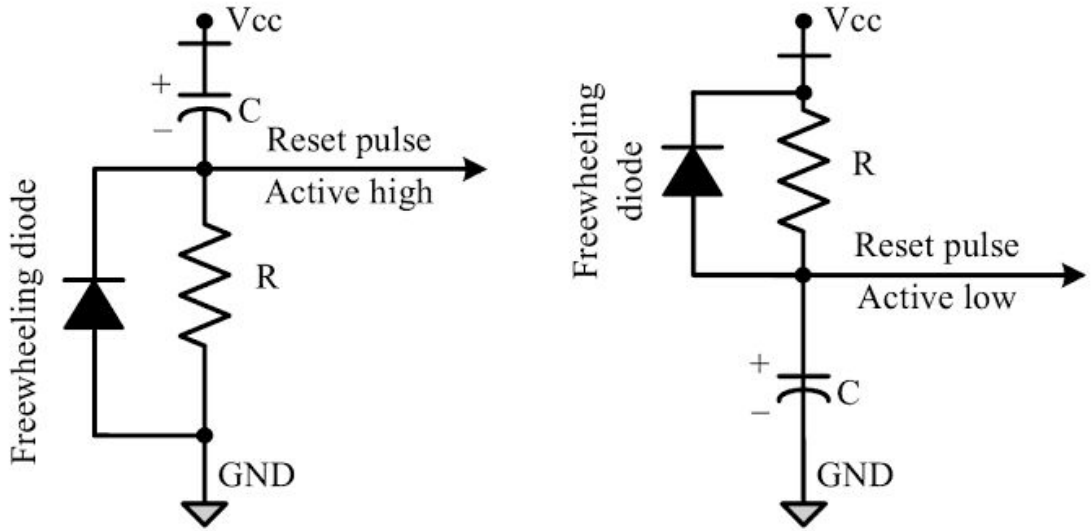


Fig: RC based reset circuit

Brown-out Protection Circuit

- Brown-out protection circuit prevents the processor/controller from unexpected program execution behaviour when the supply voltage to the processor/controller falls below a specified voltage.
- It is essential for battery powered devices since there are greater chances for the battery voltage to drop below the required threshold.
  - The processor behaviour may not be predictable if the supply voltage falls below the recommended operating voltage.
  - It may lead to situations like data corruption.

Brown-out Protection Circuit (continued)

- 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.
- Certain processors/controllers support built in brown-out protection circuit which monitors the supply voltage internally.
- If the processor/controller doesn't integrate a built-in brown-out protection circuit, the same can be implemented using external passive circuits or supervisor ICs.

Brown-out Protection Circuit (continued)

- Figure illustrates a brown-out circuit implementation using Zener diode and transistor for processor/controller with active low Reset logic.
- The Zener diode D_Z and transistor Q forms the heart of this circuit.
- The transistor conducts always when the supply voltage V_{CC} is greater than that of the sum of V_{BE} and V_Z (Zener voltage).
- The transistor stops conducting when the supply voltage falls below the sum of V_{BE} and V_Z .
- Select the Zener diode with required voltage for setting the low threshold value for V_{CC}.
- The values of R1, R2, and R3 can be selected based on the electrical characteristics of the transistor in use.
- Microprocessor Supervisor ICs like DS1232 from Maxim also provides Brown-out protection.

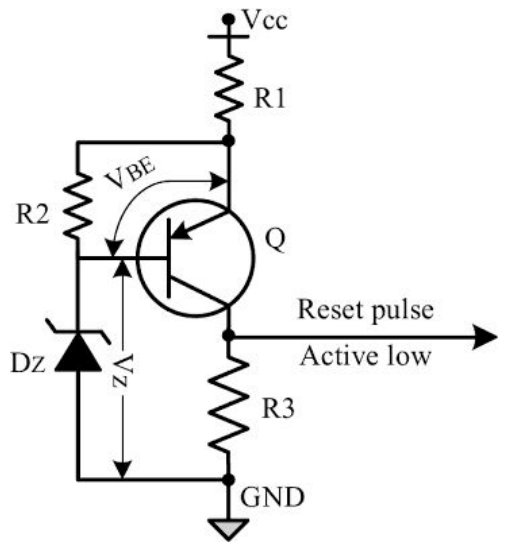


Fig: Brown-out protection circuit with Active low output

Oscillator Unit

- A microprocessor/microcontroller is a digital device made up of digital combinational and sequential circuits.
- The instruction execution of a microprocessor/controller occurs in sync with a clock signal.
- The oscillator unit of the embedded system is responsible for generating the precise clock for the processor.
  - Analogous to the heart in living beings which produces heart beats.
- Certain processors/controllers integrate a built-in oscillator unit and simply require an external ceramic resonator/quartz crystal for producing the necessary clock signals.

Oscillator Unit (continued)

- Quartz crystals and ceramic resonators are equivalent in operation, however they possess physical difference.
- A quartz crystal is normally mounted in a hermetically sealed metal case with two leads protruding out of the case.
- 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 chips and they can be used for generating the clock pulses in such cases.
- The speed of operation of a processor is primarily dependent on the clock frequency.
  - However we cannot increase the clock frequency blindly for increasing the speed of execution.
  - The logical circuits lying inside the processor always have an upper threshold value for the maximum clock at which the system can run, beyond which the system becomes

Oscillator Unit (continued)

- The total system power consumption is directly proportional to the clock frequency.
  - The power consumption increases with increase in clock frequency.
- The accuracy of program execution depends on the accuracy of the clock signal.
- The accuracy of the crystal oscillator or ceramic resonator is normally expressed in terms of +/-ppm (Parts per million).

Oscillator Unit (continued)

- Figure illustrates the usage of quartz crystal/ceramic resonator and external oscillator chip for clock generation.

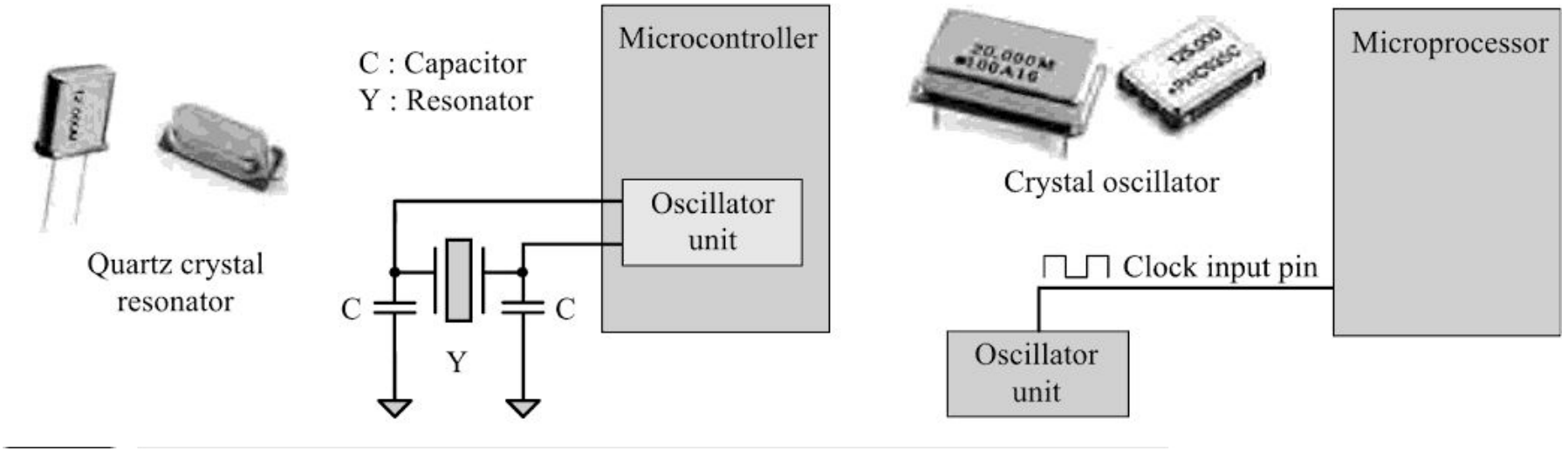


Fig: Oscillator circuitry using quartz crystal and quartz crystal oscillator

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.
- RTCs are available in the form of Integrated Circuits from different semiconductor manufacturers like Maxim/Dallas, ST Microelectronics etc.
- 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.

Real-Time Clock (RTC) (continued)

- For Operating System based embedded devices, a timing reference is essential for synchronising the operations of the OS kernel.
- The RTC can interrupt the OS kernel by asserting the interrupt line of the processor/controller to which the RTC interrupt line is connected.
- The OS kernel identifies the interrupt in terms of the Interrupt Request (IRQ) number generated by an interrupt controller.
- One IRQ can be assigned to the RTC interrupt and the kernel can perform necessary operations like system date time updation, managing software timers, etc. when an RTC timer tick interrupt occurs.
- The RTC can be configured to interrupt the processor at predefined intervals or to interrupt the processor when the RTC register reaches a specified value (used as alarm interrupt).

Watchdog Timer

- A watchdog timer, or simply a watchdog, is a hardware timer for monitoring the firmware execution and resetting the system processor/microcontroller when the program execution hangs up.
- Depending on the internal implementation, the watchdog timer increments or decrements a free running counter with each clock pulse and generates a reset signal to reset the processor if the count reaches zero for a down counting watchdog, or the highest count value for an up counting watchdog.

Watchdog Timer (continued)

- If the watchdog counter is in the enabled state, the firmware can write a zero (for up counting watchdog implementation) to it before starting the execution of a piece of code (which is susceptible to execution hang up) and the watchdog will start counting.
- 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 the firmware execution completes before the expiration of the watchdog timer you can reset the count by writing a 0 (for an up counting watchdog timer) to the watchdog timer register.

Watchdog Timer (continued)

- 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.
- The external watchdog timer uses hardware logic for enabling/disabling, resetting the watchdog count, etc. instead of the firmware based 'writing' to the status and watchdog timer register.
- The Microprocessor supervisor IC DS1232 integrates a hardware watchdog timer in it.
- In modern systems running on embedded operating systems, the watchdog can be implemented in such a way that when a watchdog timeout occurs, an interrupt is generated instead of resetting the processor.
- The interrupt handler for this handles the situation in an appropriate fashion.

Watchdog Timer (continued)

- Figure illustrates the implementation of an external watchdog timer based microprocessor supervisor circuit for a small scale embedded system.

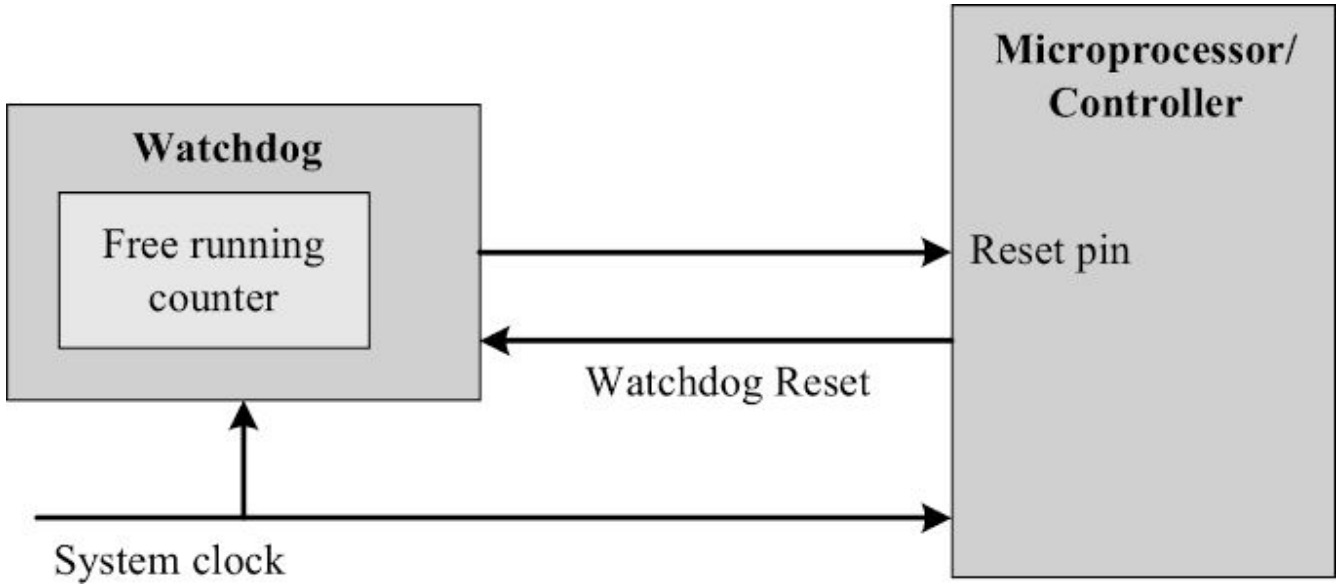


Fig: Watchdog timer for firmware execution supervision

PCB and Passive Components

- Printed Circuit Board (PCB) is the backbone of every embedded system.
- After finalising the components and the inter-connection among them, a schematic design is created and according to the schematic, the PCB is fabricated.
- PCB acts as a platform for mounting all the necessary components as per the design requirement.
- Also it acts as a platform for testing the embedded firmware.
- Apart from the subsystems mentioned already, passive electronic components like resistor, capacitor, diodes, etc. are also found on the board.
  - They are the co-workers of various chips contained in the embedded hardware.
  - They are very essential for the proper functioning of your embedded system.
  - For example for providing a regulated ripple-free supply voltage to the system, a regulator IC and spike suppressor filter capacitors are very essential.

Reference

1. Shibu K V, ***"Introduction to Embedded Systems"***, Tata McGraw Hill, 2009.
2. Raj Kamal, ***"Embedded Systems: Architecture and Programming",*** Tata McGraw Hill, 2008.