**MODULE 1: Introduction:** Embedded Systems and general purpose computer systems, history, classifications, applications and purpose of embedded systems. **Core of Embedded Systems :** Microprocessors and microcontrollers, RISC and CISC controllers, Big endian and Little endian processors, Application specific ICs, Programmable logic devices, COTS, sensors and actuators, communication interface, embedded firmware, other system components. PCB and passive components

## Embedded Systems

An Electronic/Electro mechanical system which is designed to perform a specific function and is acombination of both hardware and firmware (Software). E.g. Electronic Toys, Mobile Handsets, Washing Machines, Air Conditioners, Set Top Box, DVD Player etc.

## Embedded Systems vs. General Computing Systems

| No. | General Purpose Computing System | Embedded System (ES) |
|---|---|---|
| 1. | A system which is a combination of generic hardware and General Purpose Operating Systemfor executing a variety of applications. | A system which is a combination of special purpose hardware and embedded Operating System for executing a specific set of applications. |
| 2. | Contain a General Purpose Operating System (GPOS). | May or may not contain an operating system for functioning. |
| 3. | Applications are alterable (programmable) by user. (The end user can re-install the OS, and add or remove userapplications). | The firmware of the embedded system is pre-programmed and it is non-alterable by end-user. |
| 4. | Performance is the key deciding factor on the selection of the system. Always “Faster is Better”. | Application specific requirements (like performance, power requirements, memory etc) are the key deciding factors. |
| 5. | Less/not at all tailored towards reduced operating power requirements, options for different levels of power management. | Highly tailored to take advantage of the power saving modes supported by hardwareand Operating System. |
| 6. | Response requirements are not time critical. | For certain category of ES, the response time requirement is highly critical. |
| 7. | Need not be deterministic in execution behaviour. | Execution behaviour is deterministic for few ES like “Hard Real Time“ systems. |

## Classification of Embedded Systems

The criteria used in the classification of embedded systems are:

1. Based on generation.
2. Complexity and performance requirements.
3. Based on deterministic behaviour.
4. Based on triggering.

**1. Classification Based on Generation**

- **First Generation:** The early embedded systems were built around 8-bit microprocessors like 8085 and 280, and 4-bit microcontrollers. Simple in hardware circuits with firmware developed in Assembly code. Eg: Digital telephone keypads, stepper motor control units etc.
- **Second Generation:** Embedded systems built around 16-bit microprocessors and 8 or 16-bit microcontrollers. The instruction set became much more complex and powerful than the first generation processors/controllers. Eg: Data Acquisition Systems, SCADA systems etc.
- **Third Generation:** Embedded systems built around powerful 32-bit processors and 16-bit microcontrollers. Application and domain specific processors /controllers like Digital Signal Processors (DSP) and Application Specific Integrated Circuits (ASICs) came into the picture. The instruction set is more complex and powerful. Eg: Robotics, media, industrial process control, networking, etc.
- **Fourth Generation:** Embedded system built around System on Chips (SOC), reconfigurable processors and multicore processors. The fourth generation embedded systems are making use of high performance real time embedded operating systems for their functioning. Eg: Smart phone devices, mobile internet devices (MIDs), etc.

**2. Classification Based on Complexity and Performance**

- **Small-Scale Embedded Systems:** Small-scale ES are usually built around low performance and low cost 8 or 16 bit microprocessors / microcontrollers. These ES are suitable for simple applications and where performance is not time critical. It may or may not contain an operating system (OS) for functioning. Eg: Electronic toy.
- **Medium-Scale Embedded Systems:** Embedded systems built around medium performance, low cost 16 or 32 bit microprocessors/microcontrollers or DSPs. These embedded systems which are slightly complex in hardware and firmware (software)

requirements. It usually contains an Embedded OS for functioning.

- **Large-Scale Embedded Systems/Complex Systems:** ES built around 32 or 64 bit RISC processors/controllers or Reconfigurable System on Chip (RSC) or multi-core processors and programmable logic devices. These embedded systems involve highly complex hardware and firmware. They may contain multiple processors/controllers and co-units/hardware accelerators. Complex embedded systems usually contain a high performance Real Time Operating System (RTOS).

**3. Classification based on deterministic system behaviour**

It is applicable for Real Time systems. The application/task execution behaviour for an embedded system can be either deterministic or non-deterministic. Classified into:

- **Soft Real Time Systems:** Missing a deadline may not be critical and can be tolerated to a certain degree. Eg: ATM.
- **Hard Real Time Systems:** Missing any deadline may produce disastrous results (financial, human loss of life, etc.). Eg: ABS, Air bags etc.

**4. Classification based on triggering**

Embedded Systems which are 'Reactive' in nature can be classified based on the trigger. Reactive systems can be classified as **event triggered** or **time triggered**.

**Major Application Areas of Embedded Systems**

- **Consumer Electronics:** Camcorders, Cameras etc.
- **Household Appliances:** Television, Washing Machine, Fridge, Microwave Ovenetc.
- **Home Automation and Security Systems:** Air conditioners, CCTVs, Fire alarms etc.
- **Automotive Industry:** Anti-lock Breaking Systems (ABS), Engine Control, Automatic Navigation Systems etc.
- **Telecom:** Cellular Telephones, Telephone switches, Handset etc.
- **Computer Peripherals:** Printers, Scanners, Fax machines etc.
- **Computer Networking Systems:** Network Routers, Switches, Hubs, Firewalls etc.
- **Health Care:** Different Kinds of Scanners, EEG, ECG Machines etc.
- **Measurement & Instrumentation:** Digital multi meters, Digital CRO, Logic analyzers, PLCsystems etc.
- **Banking & Retail:** Automatic Teller Machines (ATM), Currency counters etc.
- **Card Readers:** Barcode, Smart Card Readers, Hand held Devices etc.
- **Cloud Computing and Internet of Things (IoT).**

## Purpose of Embedded Systems

Each Embedded Systems is designed to serve the purpose of any one or a combination of the following tasks:

**1. Data Collection/Storage/Representation**

Embedded system designed for the purpose of data collection performs acquisition of data from the external world. Data collection is usually done for storage, analysis, manipulation and transmission. Data can be analog or digital.

Embedded systems with analog data capturing techniques collect data directly in the form of analog signal whereas embedded systems with digital data collection mechanism converts the analog signal to the digital signal using analog to digital converters. If the data is digital it can be directly captured by digital embedded system.

A digital camera is a typical example of an embedded System with data collection/storage/representation of data. Images are captured and the captured image may be stored within the memory of the camera. The captured image can also be presented to the user through a graphic LCD unit.

**2. Data communication**

Embedded data communication systems are deployed in applications from complex satellite communication to simple home networking systems. The transmission of data is achieved either by a wire-line medium or by a wire-less medium. USB, TCP/ IP, PS2 are examples of wired communication and Bluetooth, ZigBee and Wi-Fi are examples for wireless communication.

Data can either be transmitted by analog means or by digital means. Network hubs, routers, switches are examples of dedicated data transmission embedded systems.

**3. Data (Signal) Processing**

Embedded systems with signal processing functionalities are employed in applications demanding signal processing like speech coding, audio video codec, transmission applications etc.

A digital hearing aid is a typical example of an embedded system employing data processing.

**4. Monitoring**

All embedded products coming under the medical domain are with monitoring functions. They are used for determining the state of some variables using input sensors.

Electro cardiogram (ECG) machine is intended to do the monitoring of the heartbeat of a patient but it cannot impose control over the heartbeat. Other examples with monitoring function are digital CRO, digital multi-meters, and logic analyzers.

**5. Control**

Embedded systems with control functionalities are used for imposing control over some variables according to the changes in input variables. A system with control functionality contains both sensors and actuators.

Sensors are connected to the input port for capturing the changes in environmental variable and the actuators connected to the output port are controlled according to the changes in the input variable. Air conditioner system used to control the room temperature to a specified limit is a typical example for control purpose.

**6. Application Specific User Interface**

These are the embedded systems that are designed for a specific application. Buttons, switches, keypad, lights, bells, display units etc. are application specific user interfaces.

Mobile phone is an example of application specific user interface. In mobile phone the user interface is provided through the keypad, system speaker, vibration alert etc.

## Elements of Embedded Systems

An embedded system is a combination of 3 things: Hardware, Software, Mechanical Components and it is supposed to do one specific task only. A typical embedded system contains a single chip controller which acts as the master brain of the system. Diagrammatically an embedded system can be represented as shown in figure 3.1.

Embedded systems are designed to regulate a physical variable or to manipulate the state of some devices by sending signals to the actuators or devices connected to the output system, in response to the input signal provided by the end users or sensors which are connected to the input ports. Hence the embedded systems can be viewed as a reactive system.

Keyboards, push button, switches, etc. are examples of input devices and LEDs, LCDs, Piezoelectric buzzers, etc. are examples for output devices for a typical ES.

Some embedded system can automatically sense input parameters from real world through sensors. Sensor information is passed to the processor after signal conditioning and digitization. The core of the system performs some predefined operations on input data with the help of embedded firmware and sends some actuating signals to the actuator.

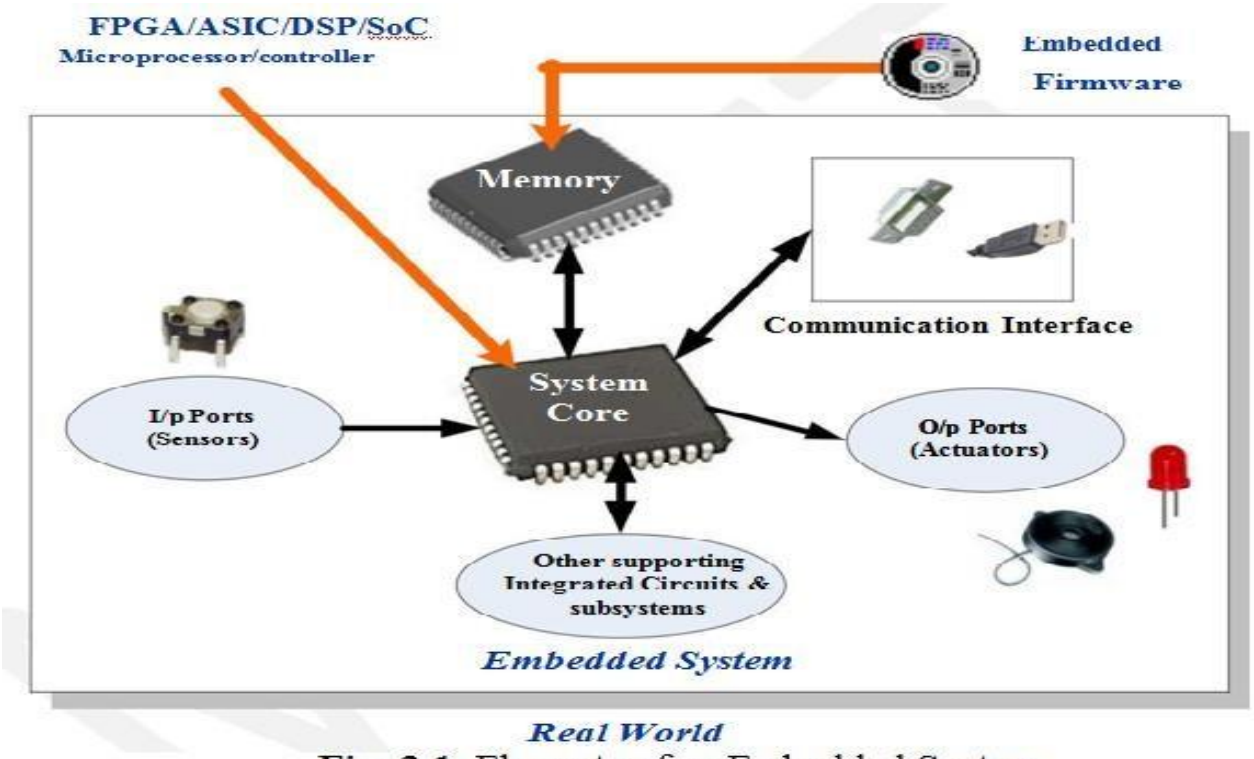



**Fig. 3.1:** Elements of an Embedded System

The memory of the system is responsible for holding the code. There are two types: Fixed memory (ROM) is used for storing code or program. The user cannot do any modifications in this type of memory. Common types used are OTP, PROM, EPROM, EEPROM & Flash memory. Temporary memory (RAM) is used for performing arithmetic operations or control algorithm executions. Common types used are SRAM, DRAM and NVRAM.

Memory for implementing the code may be present on the processor or may be implemented as a separate chip interfacing the processor. In a controller based embedded system, the controller may contain internal memory for storing code and such controllers are called Micro-controllers with on-chip ROM, eg. Atmel AT89C51.

## Core of the Embedded System

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

1. General Purpose and Domain Specific Processors:
   - Microprocessors
   - Microcontrollers
   - Digital Signal Processors
2. Programmable Logic Devices (PLDs)
3. Application Specific Integrated Circuits (ASICs)
4. Commercial off-the-shelf Components (COTS)

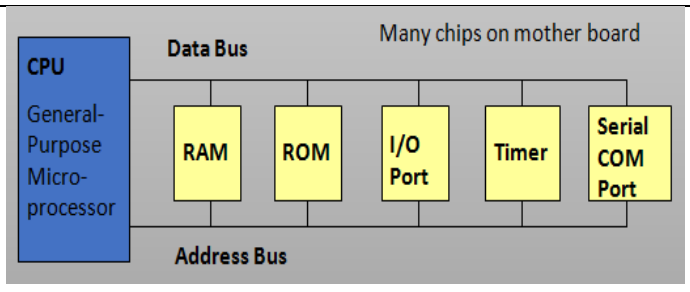
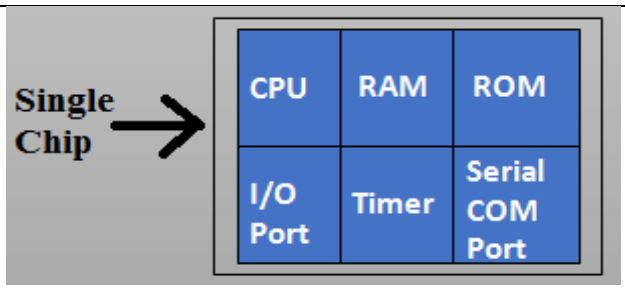
## 1. General Purpose and Domain Specific Processors

Almost 80% of the embedded systems are processor/ controller based. The processor may be microprocessor or a microcontroller or digital signal processor, depending on the domain and application.

**Microprocessor:** A silicon chip representing a Central Processing Unit(CPU), which is capable of performing arithmetic as well as logical operations according to a pre-defined set of instructions. In general, the CPU contains the Arithmetic and Logic Unit (ALU), control unit and working registers.

**Microcontroller:** A microcontroller is a highly integrated chip that contains a CPU, RAM, Special and General purpose Register Arrays, On Chip ROM/FLASH memory for program storage, Timer and Interrupt control units and dedicated I/O ports. Since a microcontroller contains all the necessary functional blocks for independent working, they found greater place in the embedded domain in place of microprocessors.

The differences between a Microprocessor and Microcontroller is given in below table:

| No. | Microprocessors | Microcontrollers |
|---|---|---|
| 1. | It is a dependent unit. It requires the combination of other chips like Timers, Program and data memory chips, Interrupt controllers etc. for functioning. | It is a self-contained unit & doesn't require external Interrupt Controller, Timer, and UART etc. for its functioning. |
| 2. | Most of the time general purpose in design and operation. | Mostly application oriented or domain specific. |
| 3. | Doesn't contain a built in I/O port. | Most of the processors contain multiple built-in I/O ports. |
| 4. | Targeted for high end market where performance is important. | Targeted for embedded market where performance is not so critical. |
| 5. | Limited power saving options. | Includes lot of power saving features |
| 6. |  |  |

**Digital Signal Processor (DSP)**

DSP's are powerful special purpose 8/16/32 bit microprocessor designed to meet the computational demands and power constraints of today's embedded audio, video and communication applications. DSP are 2 to 3 times faster than general purpose microprocessors in signal processing applications. This is because of the architectural difference between DSP and general purpose microprocessors. DSPs implement algorithms in hardware which speeds up the execution whereas general purpose processor implement the algorithm in software and the speed of execution depends primarily on the clock for the processors.

A typical digital signal processor incorporates the following four key units:

**1. Program Memory:** Memory for storing the program required by DSP to process the data.

**2. Data Memory:** Working memory for storing temporary variables and data/ signal to be processed.

**3. Computational Engine:** Performs the signal processing in accordance with the stored program memory. Computational Engine incorporates many specialized arithmetic units and each of them operates simultaneously to increase the execution speed.

**4. I/O Unit:** Input/Output unit acts as an interface between the outside world and DSP. It is responsible for capturing signals to be processed and delivering the processed signals.

Audio video signal processing, telecommunication and multimedia applications are typical examples where DSP is employed. Digital signal processing 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 V/s CISC Processors/Controllers**

The term RISC stands for Reduced Instruction Set Computing. All RISC processors/ controllers possess lesser number of instructions, typically in the range of 30 to 40.

CISC stands for Complex instruction Set Computing. The instruction set of CISC is complex and instructions are high in number.

From a programmers point of view RISC processors are comfortable, since he/she needs to learn only a few instructions, whereas CISC processor needs to learn more number of instructions and should understand the context of usage of each instruction.

The major differences between RISC and CICS controllers are given in below table:

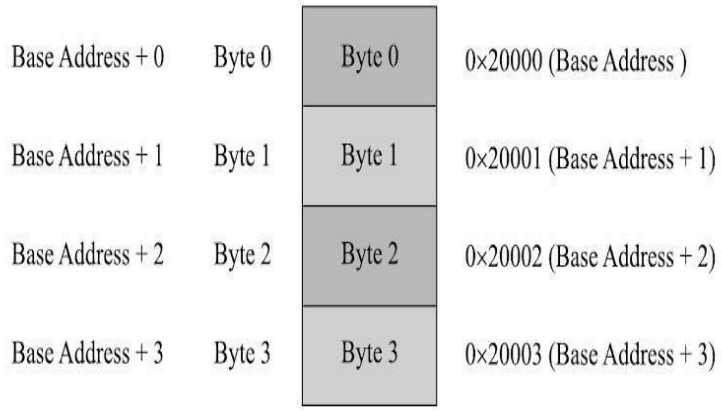
| No. | RISC Processors/Controllers | CISC Processors/Controllers |
|---|---|---|
| 1. | Lesser no. of instructions. | Greater no. of Instructions. |
| 2. | Instruction Pipelining and increased execution speed. | Generally no instruction pipelining feature. |
| 3. | Orthogonal Instruction Set. | Non Orthogonal Instruction Set. |
| 4. | Operations are performed on registers only, memory operations are load & store. | Operations are performed on registers or memory depending on the instruction |
| 5. | Large number of registers are available | Limited no. of general purpose registers |
| 6. | Programmer needs to write more code to execute a task since the instructions are simpler ones. | A programmer can achieve the desired functionality with a single instruction. |
| 7. | Single, Fixed length Instructions. | Variable length Instructions. |
| 8. | Less Silicon usage and pin count. | More silicon usage. |
| 9. | With Harvard Architecture | Harvard or Von-Neumann Architecture |

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

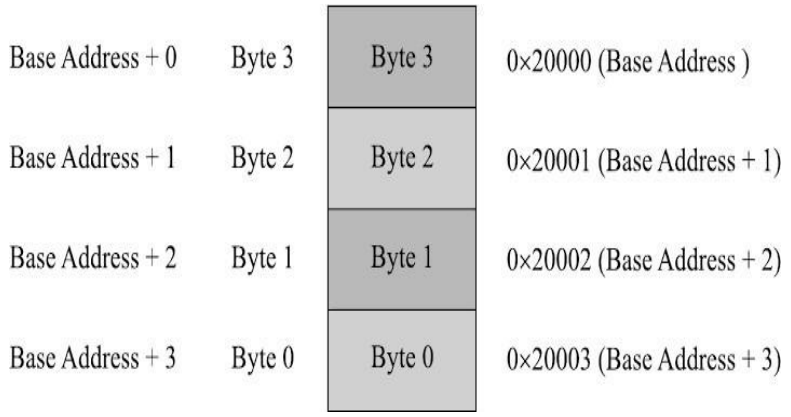
Endianness specifies the order in which the data is stored in the memory by processor operations in a multi byte system. Suppose the word length is two byte then data can be stored in memory in two different ways:

- Higher order of data byte at the higher memory and lower order of data byte at location just below the higher memory - Little-Endian. E.g.: Intel x86 Processors
- Lower order of data byte at the higher memory and higher order of data byte at location just below the higher memory - Big-Endian E.g.: Motorola 68000 Series Processors

Little-endian means the lower-order byte of the data is stored in memory at the lowest address, and the higher-order byte at the highest 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:



Big-endian means the higher-order byte of the data is stored in memory at the lowest address, and the lower-order byte at the highest 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:




## 2. Programmable Logic Devices (PLDs)

Logic devices are classified into two broad categories - Fixed and Programmable.

- The circuits in a fixed logic device are permanent, they perform one function or set of functions. Once manufactured, they cannot be changed.
- Programmable logic devices (PLDs) offer customers a wide range of logic capacity, features, speed, and voltage characteristics. These devices can be re-configured to perform any number of functions at any time.

Designers can use inexpensive software tools to quickly develop, simulate, and test their logic designs in PLD based design. The design can be quickly programmed into a device, and immediately tested in a live circuit. PLDs are based on re-writable memory technology and the device is reprogrammed to change the design.

Two major types of programmable logic devices are FPGA & CPLD.

- **Field Programmable Gate Arrays (FPGAs):** FPGAs offer the highest amount of logic density, the most features, and the highest performance. FPGAs are used in a wide variety of applications ranging from data processing and storage, to instrumentation, telecommunications, and digital signal processing
- **Complex Programmable Logic Devices (CPLDs):** CPLDs, by contrast, offer much smaller amounts of logic - up to about 10,000 gates. CPLDs offer very predictable timing characteristics and are therefore ideal for critical control applications

**Advantages of PLDs:**

1. PLDs offer customer much more flexibility during the design cycle.
2. PLDs do not require long lead times for prototypes or production parts because PLDs are already on a distributor's shelf and ready for shipment.
3. PLDs can be reprogrammed even after a piece of equipment is shipped to a customer.
4. PLDs allow customers to order just the number of parts they need.

### 3. Application Specific Integrated Circuit (ASIC)

ASIC is a microchip designed to perform a specific or unique application. It integrates several functions into a single chip and thereby reduces the system development cost. Most of the ASICs are proprietary products. ASIC consumes very small area in the total system and thereby helps the design of smaller systems with high capabilities/functionalities.

ASICs can be pre-fabricated for a special application or it can be custom fabricated by using the components from a re-usable "building block" library of components for a particular customer application. Fabrication of ASICs requires a non-refundable initial investment (Non-Recurring Engineering (NRE) charges) for the process technology & configuration expenses.

If NRE is born by a third party and the ASIC is made openly available in the market, the ASIC is referred as Application Specific Standard Product (ASSP).

### 4. Commercial off the Shelf Component (COTS)

Commercial off the Shelf product is one which is used 'as-is'. COTS components itself may be developed around a general purpose or domain specific processor or a ASICs or a PLDs. Typical examples of COTS hardware unit are remote controlled toy car control units.

Advantage of using COTS is that they are readily available in the market, are cheap and a developer can cut down his/her development time to a great extent.

Major drawback of using COTS components in embedded design is that the manufacturer of the COTS component may withdraw the product or discontinue the production of the COTS at any time if rapid change in technology occurs.

### Sensors and Actuators

A sensor is a special kind of transducer that is used to generate an input signal to a measurement, instrumentation or control system. The signal produced by a sensor is an electrical analogy of a physical quantity, such as distance, velocity, acceleration, temperature, pressure, light level, etc. Sensor acts as an input device.

Actuator is a form of transducer device (mechanical or electrical) which converts signals to corresponding physical action (motion). Actuator acts as an output device.

### The I/O Subsystem

The I/O subsystem of embedded system facilitates the interaction of embedded system with the external world. Various I/O devices used in embedded system applications are:

## Light emitted Diode (LED)

LED is an output device for visual indication in any embedded system. Used as an indicator for the status of various signals or situations.

LED is a p-n junction diode and it contains an anode & 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 supply voltage. A resister is used in series between the power supply and the resistor to limit the current through the LED. The ideal LED interfacing circuit is shown.

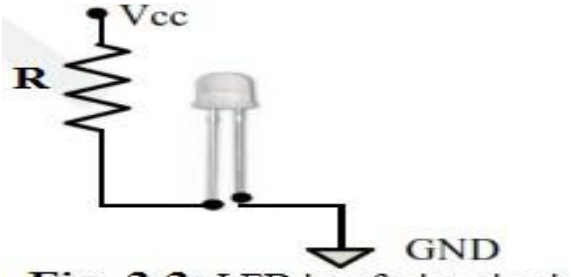



**Fig. 3.3:** LED interfacing circuit

There are two ways to interface an LED to a microprocessor/microcontroller:

1. The Anode of LED is connected to the port pin and cathode to Ground: In this approach the port pin sources the current to the LED when it is at logic High (i.e. 1).2. The Cathode of LED is connected to the port pin and Anode to Vcc: In this approach the port pin sources the current to the LED when it is at logic Low (i.e. 0).

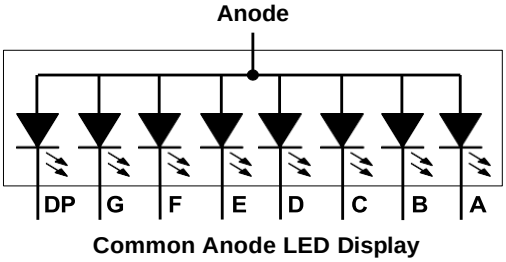
## 7-Segment LED Display

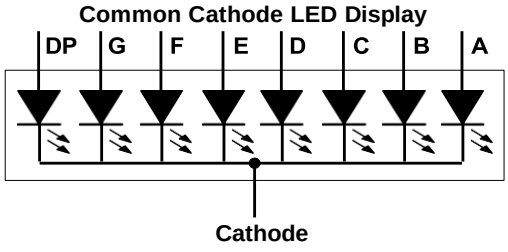
The 7 – segment LED display is an output device for displaying alpha numeric characters & contains 8 LED segments arranged in a special form. Out of the 8 LED segments, 7 are used for displaying alpha numeric characters. LED segments are named A to G and the decimal point LED segment is named as DP.

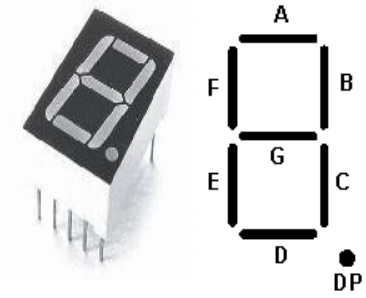
The 7 – segment LED displays are available in two different configurations:

- **Common anode configuration:** Anodes of 8 segments are connected commonly.
- **Common cathode configuration:** 8 LED segments share a common cathode line.

Figure 3.4 shows the common anode & cathode configurations of a 7 segment LED. The current flow through each of the LED segments should be limited to the maximum value supported by the LED display unit by connecting a current limiting resistor.



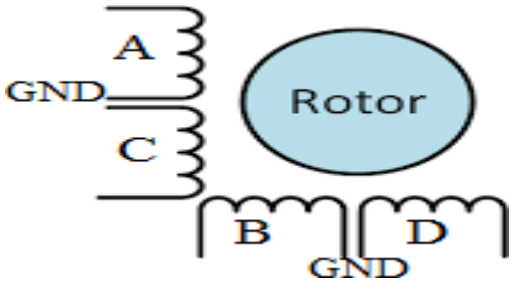



## Stepper Motor

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. Eg: Paper feed mechanism of printer/fax makes use of stepper motor for its functioning.

Based on the coil winding arrangements, a two phase stepper motor is classified into:

- **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 shows a two 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.

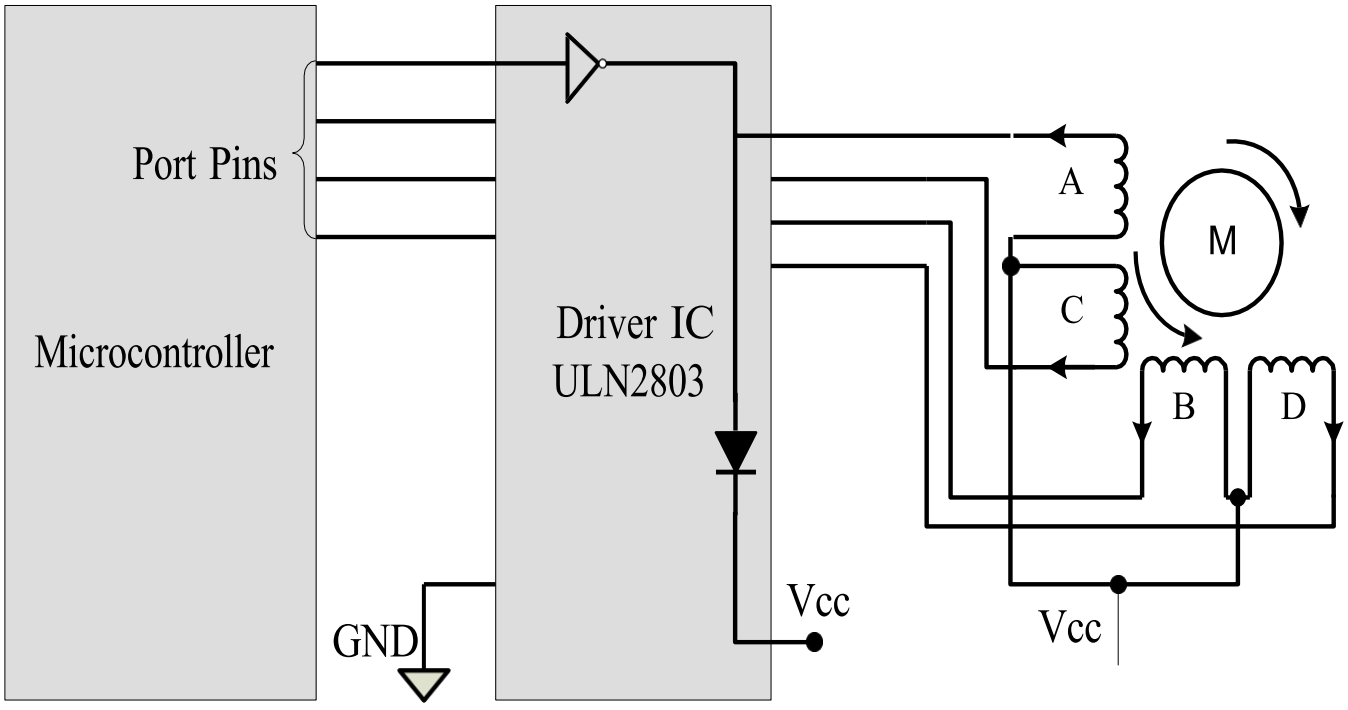
The stepping of stepper motor can be implemented in different ways by changing the sequence of activation of the stator windings. The different stepping modes supported by stepper motor are explained below:

- **Full Step:** In the full step mode both the phases are energized simultaneously. The coils A, B, C and D are energized in the order, as shown in the Table.
- **Wave Step:** In the wave step mode, only one phase is energized at a time and each coils of the phase is energized alternatively. The A, B, C and D are energized in the order, as shown in the Table.
- **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 in the Table below.

| | Full Step | | | | Wave Step | | | | | Half Step | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Step** | **Coil A** | **Coil B** | **Coil C** | **Coil D** | **Coil A** | **Coil B** | **Coil C** | **Coil D** | **Step** | **Coil A** | **Coil B** | **Coil C** | **Coil D** |
| **1** | **H** | **H** | **L** | **L** | **H** | **L** | **L** | **L** | **1** | **H** | **L** | **L** | **L** |
| | | | | | | | | | **2** | **H** | **H** | **L** | **L** |
| **2** | **L** | **H** | **H** | **L** | **L** | **H** | **L** | **L** | **3** | **L** | **H** | **L** | **L** |
| | | | | | | | | | **4** | **L** | **H** | **H** | **L** |
| **3** | **L** | **L** | **H** | **H** | **L** | **L** | **H** | **L** | **5** | **L** | **L** | **H** | **L** |
| | | | | | | | | | **6** | **L** | **L** | **H** | **H** |
| **4** | **H** | **L** | **L** | **H** | **L** | **L** | **L** | **H** | **7** | **L** | **L** | **L** | **H** |
| | | | | | | | | | **8** | **H** | **L** | **L** | **H** |

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 the motor directly. Special driving circuits are required to interface the stepper motor with microcontrollers. ULN2803 is an octal peripheral driver array available for driving a 5V stepper motor.

The following circuit diagram illustrates the interfacing of a stepper motor through a driver circuit connected to the port pins of a microcontroller/ processor.



## 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 shows of an optocoupler device.

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.

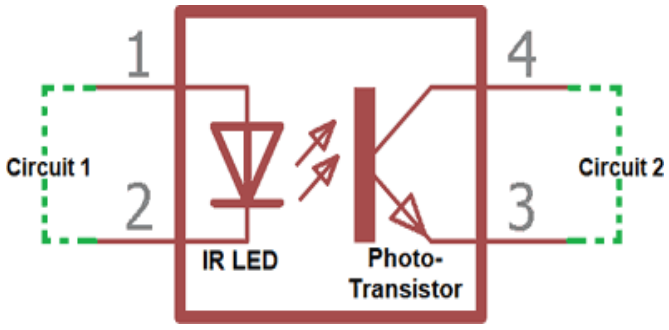
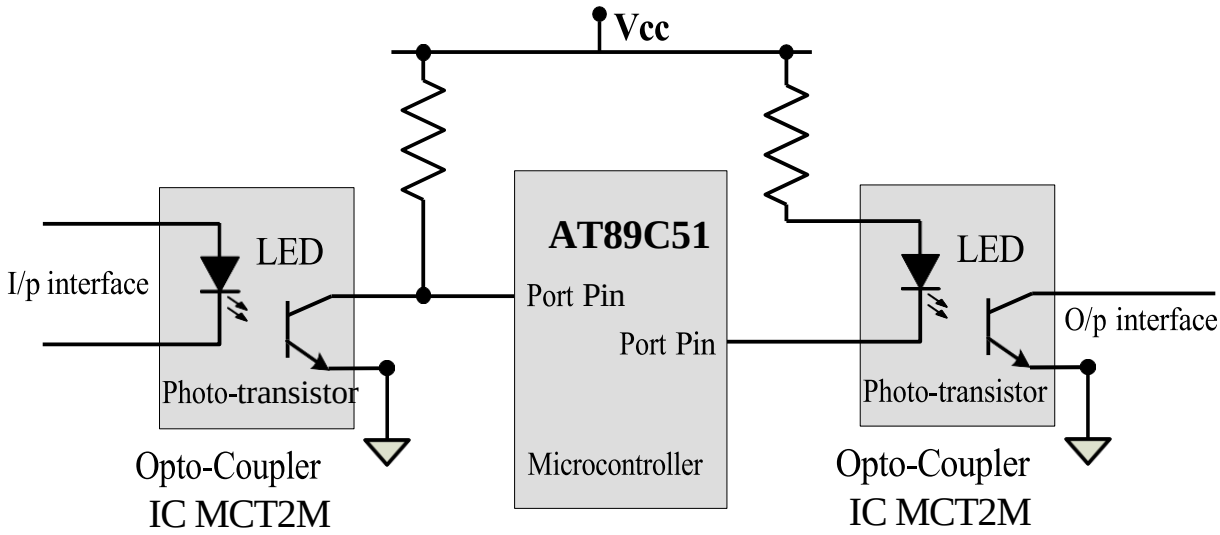



Figure illustrates the usage of optocoupler in input circuit and output circuit of an embedded system with a microcontroller as the system core.

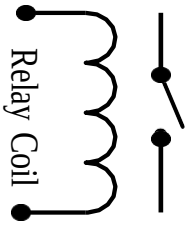



## Relay

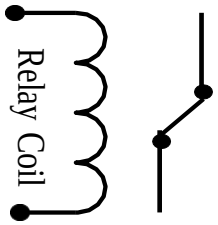
A relay is an electro mechanical device which acts as dynamic path selectors for signals and power. The 'Relay' unit contains a relay coil made up of insulated wire on a metal core and an armature with one or more contacts.

'Relay' works on electromagnet 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.

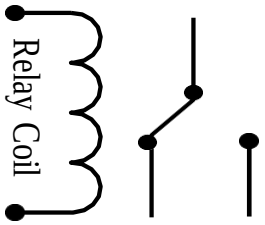
Figure below shows widely used relay configurations for embedded applications.




**Single Pole Single Throw Normally Open**




**Single Pole Single Throw Normally Closed**

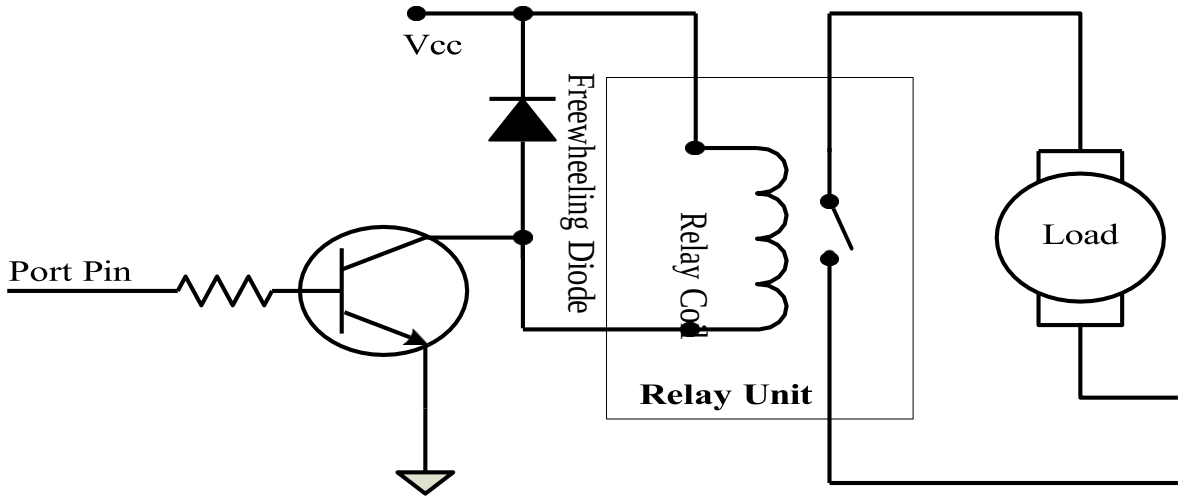



**Single Pole Double Throw**

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. The Relay is normally controlled using a relay driver circuit connected to the port pin of the processor/ controller. A transistor is used for building the relay driver circuit. The following Figure illustrates the same.




## Piezo Buzzer

It is a piezoelectric device for generating audio indications in embedded applications. A Piezo 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'

- Self-driving circuits 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

Push Button switch is an input device. This switch comes in two configurations, namely: “Push to Make” and “Push to Break”. The switch is normally in the open state and it makes a circuit contact when it is pushed or pressed in the “Push to Make” configuration. In the “Push to Break” configuration, the switch is normally in the closed state & 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“ (For Push to Break type) state as long as it is kept in the pushed state and it breaks/makes the circuit connection when it is released. Push button is used for generating a momentary pulse. In embedded application push button is used as reset, start switch & pulse generator.

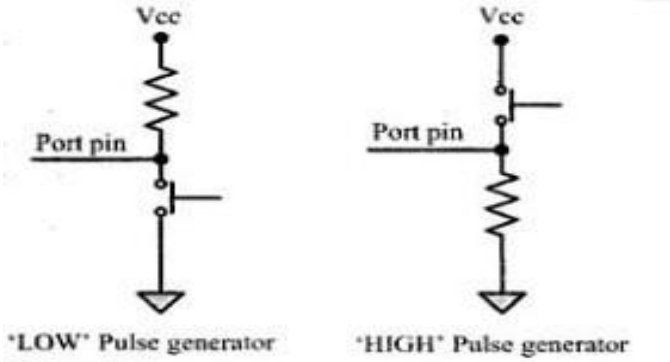


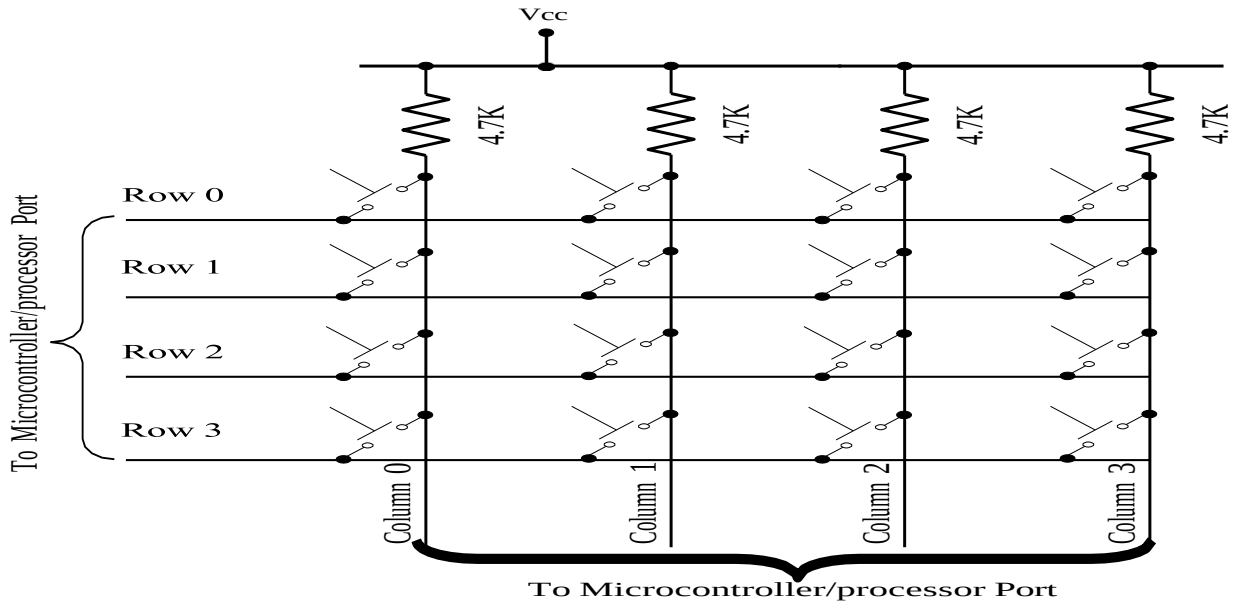

**Fig. 3.7:** Push Button Switch Configuration

Depending on the way in which the push button interfaced to the controller, it can generate either a 'HIGH' pulse or a 'LOW' pulse. Figure Illustrates how the push button can be used for generating 'LOW' and 'HIGH' pulses.

## Keyboard

Keyboard is an input device for user interface. If the number of keys required is very less, push button switches can be used. If a large number of keys are required, then Matrix keyboard is used. Figure illustrates the connection of keys in a matrix keyboard.

In a matrix keyboard, the keys are arranged in matrix fashion. For detecting a key press, each row of the matrix is pulled low & the columns are read. After reading the status of each columns corresponding to a row, the row is pulled high and the next row is pulled low and the status of the columns are read. This process is repeated until the scanning for all rows are completed. When a row is pulled low and if a key connected to the row is pressed, reading the column to which the key is connected will give logic 0. Pull-up resistors are connected to the column lines to limit the current that flows to the row line on a key press.




## Communication Interface

Communication interface is essential for communicating with various subsystems of the embedded system and with the external world. The communication interface can be viewed in two different perspectives; namely;

- **Device/board level communication interface (Onboard Communication Interface):** The communication channel which interconnects the various components within an embedded product is referred as Device/board level communication interface.
  Eg: Serial interfaces like I2C, SPI, UART, 1-Wire etc. and Parallel bus interface
- **Product level communication interface (External Communication Interface):** The Product level communication interface is responsible for data transfer between the embedded system and other devices or modules. The external communication interface can be either wired media or wireless media and it can be a serial or parallel interface.
  Eg: Infrared (IR), Bluetooth (BT), Wireless LAN (Wi-Fi), Radio Frequency waves (RF), GPRS etc. (wireless) and RS-232, USB, Parallel port etc. (wired).

## Onboard Communication Interfaces:

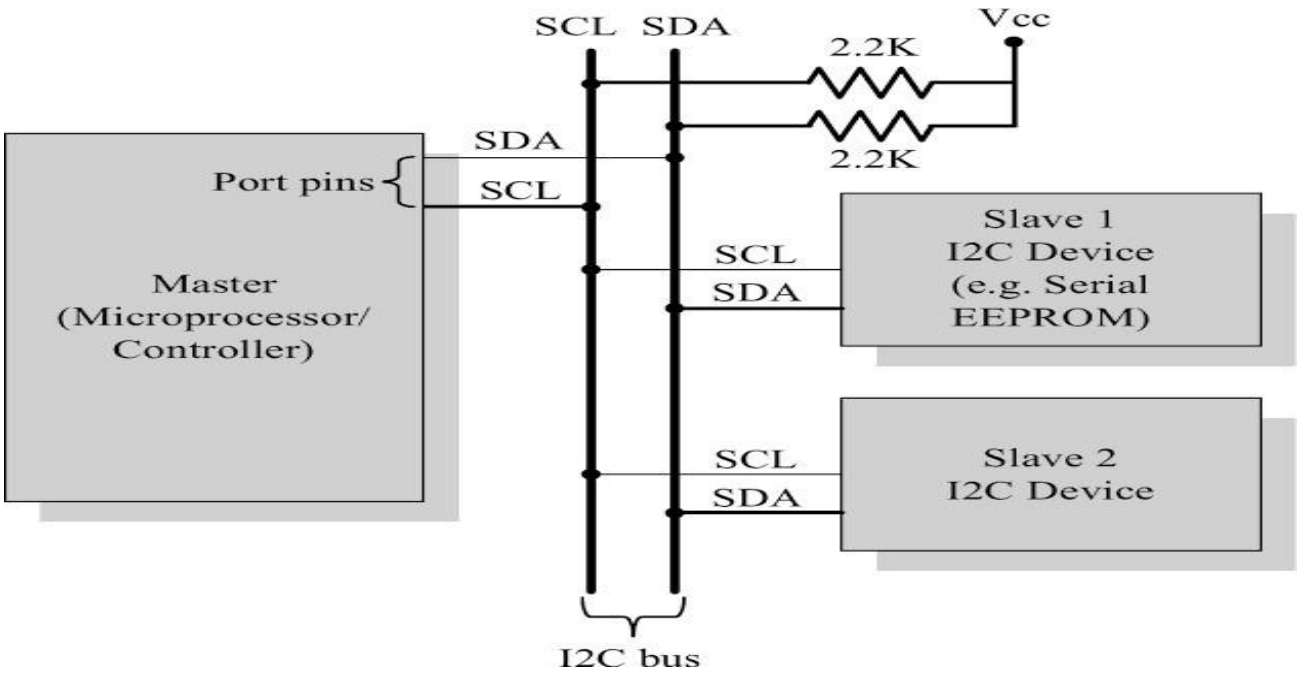
### 1. Inter Integrated Circuit (I2C) Bus

Inter Integrated Circuit Bus (I2C) is a synchronous bidirectional half duplex (one-directional communication at a given point of time) two wire serial interface bus. The I2C bus comprise of two bus lines:

- Serial Clock (SCL line): Responsible for generating synchronization clock pulses.
- Serial Data (SDA Line): Responsible for transmitting the serial data across devices.

I2C bus is a shared bus system to which many number of I2C devices can be connected. Devices connected to I2C bus can act as either 'Master' device or 'Slave' device. '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. Synchronization clock signal is generated by the 'Master' device only.

Figure below shows bus interface diagram, which illustrates the connection of master and slave devices on the I2C bus.




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 of the 'slave' device to which it wants to communicate, over the SDA line. Clock pulses are generated at the SCL line for synchronizing the bit reception by the slave device.
4. The master device sends the Read or Write bit 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.
6. The slave device with the address requested by the master device responds by sending an acknowledge bit over the SDA line

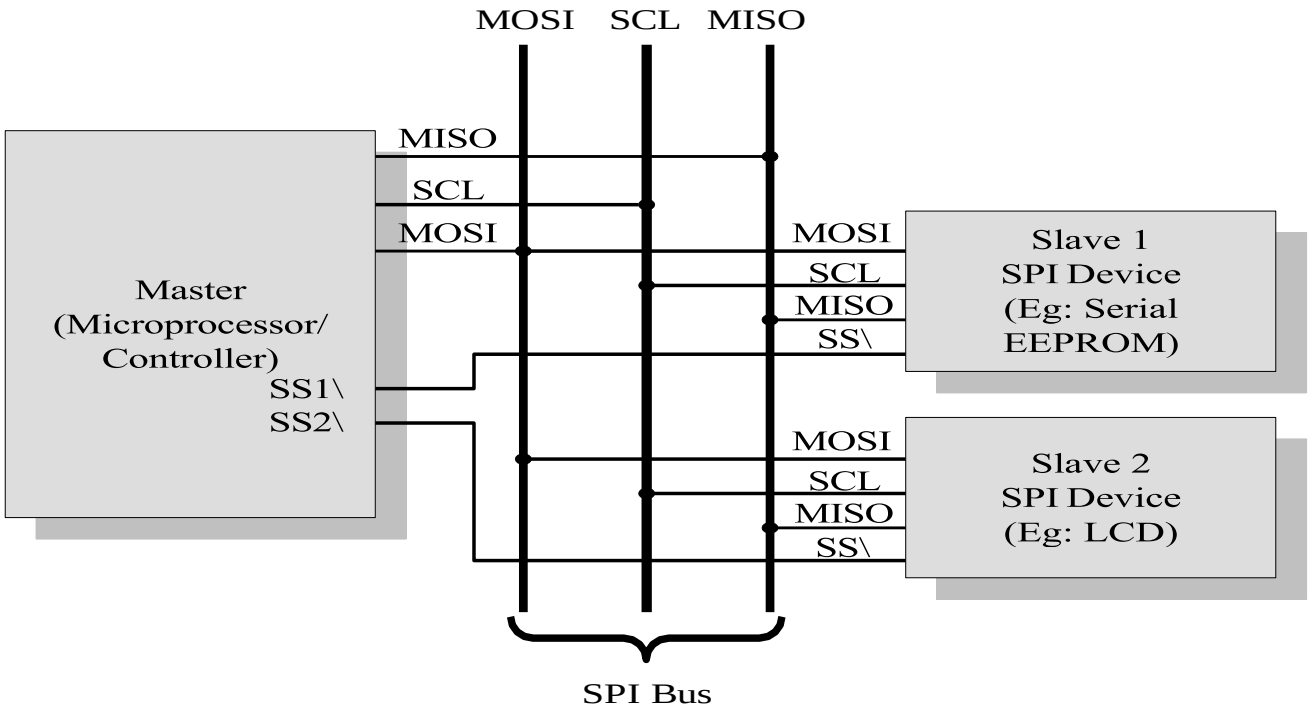
7. Upon receiving the acknowledge bit, the master device sends the 8-bit data to the slave device over SDA line, if the requested operation is 'Write to device'. If the requested operation is 'Read from device', the slave device sends data to the master over the SDA line.
8. The master device waits for 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).

## 2. Serial Peripheral Interface (SPI) Bus

SPI is asynchronous bi-directional full duplex four-wire serial interface bus. It is a single master multi-slave system. 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 / Slave Data Out (SO/ SDO).
- **Serial Clock (SCL):** Signal line carrying the clock signals.
- **Slave Select (SS):** Signal line for slave device select. It is an active low signal.

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



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'.

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. 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.

### 3. Universal Asynchronous Receiver Transmitter (UART)

UART based data transmission is an asynchronous form of serial data transmission. It doesn't require a clock signal to synchronize the transmitting end and receiving end for transmission. Instead it relies upon the pre-defined agreement between the transmitting device and receiving device. Serial communication settings for both transmitter & receiver should be set as identical. Start & stop of communication is indicated through inserting special bits in data stream

The 'start' bit informs the receiver that a data byte is about to arrive. The receiver device starts polling it's 'receive line'. 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.

For proper communication, the Transmit line (TX) of the sending device should be connected to the Receive line (RX) of the receiving device as shown below.

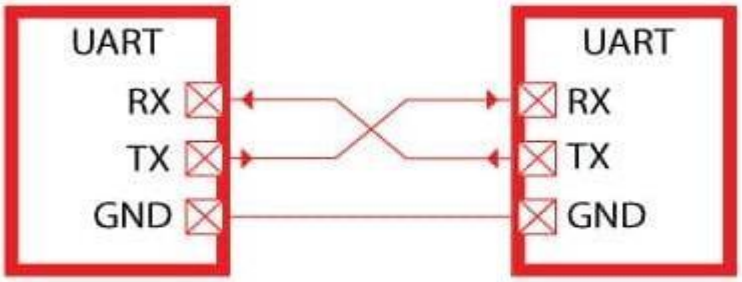
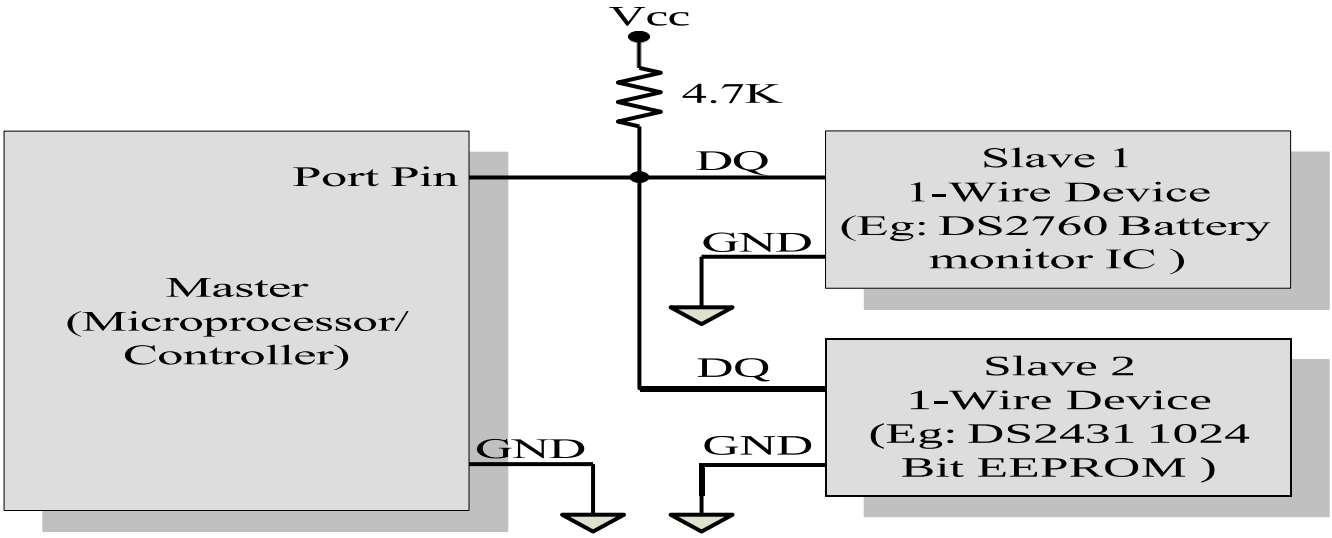



**Fig. 3.9:** UART Interfacing

### 4. 1-Wire Interface

1-wire interface is an asynchronous half-duplex communication 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 1-wire slave devices incorporate internal capacitor 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. The bus interface diagram is shown below.




Every 1-wire device contains a globally unique 64-bit identification number stored within it. 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.

The sequence of operation for communicating with a 1-wire slave device are:

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. 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.

All communication over the 1-wire bus is master initiated.

## 5. Parallel Interface

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 are read or write signals and device select signals. The device becomes active by selecting host processor.

Direction of data transfer is controlled through control signal lines for 'read' and 'write'. An address decoder circuit is used for generating chip select signal for the device. When the address selected is in the range, chip select line is activated by decoder circuit. If device wants to start communication, it can inform the same to processor through interrupts. The width of the parallel interface is determined by the data bus width of the host processor (4bit, 8bit, 16bit, 32bit or 64bit).Parallel data communication offers highest speed for data transfer. Figure below illustrates the interfacing of devices through parallel interface.

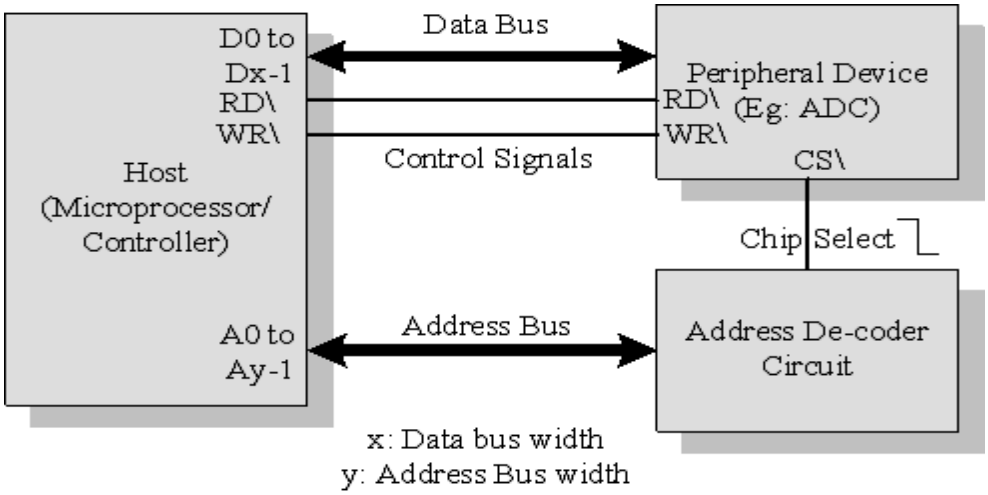


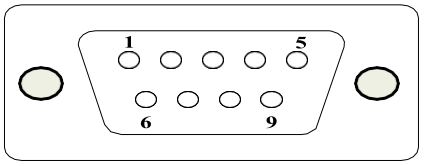

**Fig. 3.10:** Parallel Interface Bus

## External Communication Interfaces

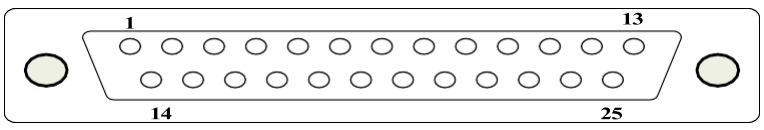
### 1. RS-232 C

RS-232 C (Recommended Standard number 232, revision C) is a legacy, full duplex, wired, asynchronous serial communication interface developed by the Electronics Industries Association (EIA). It was used for connections to modems, printers, uninterruptible power supplies, and other peripheral devices.

RS-232 supports only point-to-point communication. It supports data rates up to 20Kbps with a maximum operating distance support of 50 feet. The RS-232 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 and DB-25: 25-Pin connector.



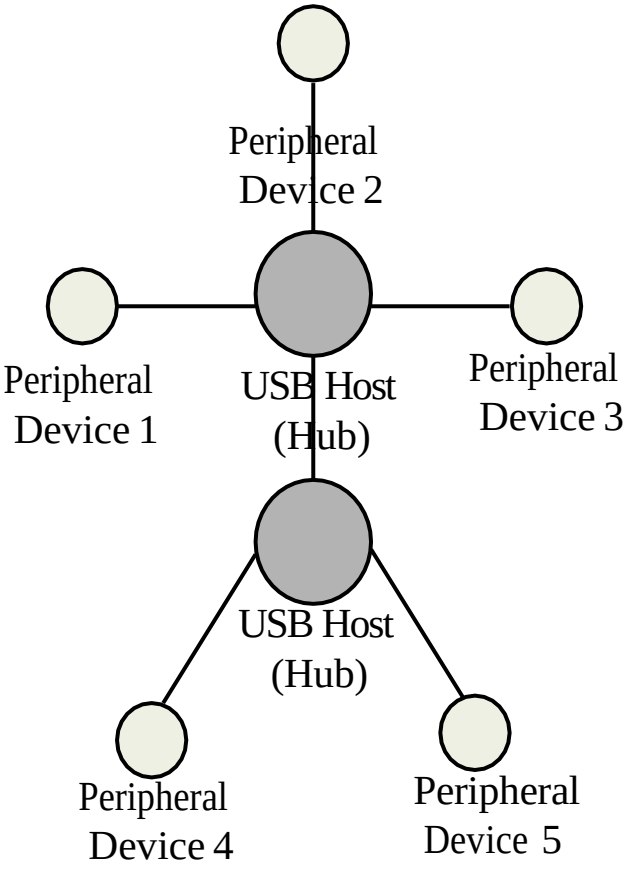

**DB-9**




**DB-25**

## 2. Universal Serial Bus (USB)

USB is a wired high speed serial bus for data communication. 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. Physical connection between a USB peripheral device and master device is established with a USB cable. USB cable supports communication distance of up to 5 meters. A USB host can support connections up to 127 devices. USB transmits data in packet format. Each data packet has a standard format. Figure illustrates the star topology for USB device connection.




USB host contains a host controller responsible for controlling the data communication. The USB standard uses two different types of connectors namely:

- Type A' connector: used for upstream connection (connection with host).
- 'Type B' connector: used for downstream connection (connection with slave device)

Both Type A and Type B connectors contain 4 pins for communication. (Table)

| 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 |

Each USB device contains a Product ID (PID) and a Vendor ID (VID). The PID and VID are embedded into the USB chip by the USB device manufacturer. The VID for a device is supplied by the USB standards forum.

USB supports four different types of data transfers:

- **Control transfer** is used by USB system software to query, configure and issue commands to the USB device

- **Bulk transfer** is used for sending a block of data to a device. Bulk transfer supports error checking and correction. Transferring data to a printer is an example.
- **Isochronous data transfer** is used for real time data communication. In Isochronous transfer, data is transmitted as streams in real time. All streaming devices like audio, video make use of isochronous transfer.
- **Interrupt transfer** is 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. Devices like Mouse, Keyboard make use of interrupt transfer.

USB supports four data transfers: Low Speed (USB 1.0: 1.5 Mbps), Full Speed (USB 1.1: 12 Mbps), High Speed (USB 2.0: 480 Mbps) & Super Speed (USB 3.0: 4.8 Gbps).

### 3. IEEE 1394 (Firewire)

IEEE 1394 is a wired, isochronous high speed serial communication bus. It is also known as High Performance Serial Bus (HPSB). IEEE 1394 supports peer-to-peer connection and point-to-multipoint communication allowing 63 devices to be connected on the bus in a tree topology. It is a wired serial interface and 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 which 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). IEEE 1394 is a popular communication interface for connecting embedded devices like Digital Camera, Camcorder, Scanners to desktop computers for data transfer and storage.

### 4. Infrared (IrDA)

IrDA serial, half duplex, line of sight based wireless technology for data communication between devices. Infrared communication technique makes use of Infrared waves of the electromagnetic spectrum for transmitting the data. IrDA 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 10cm to 1 m.IR supports data rates ranging from 9600bits/second to 16MbpsSIR supports transmission rates ranging from 9600bps to 115.2kbps.

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 used as the IR source for transmitter and at the receiving end a photodiode is used as the receiver.

## 5. Bluetooth

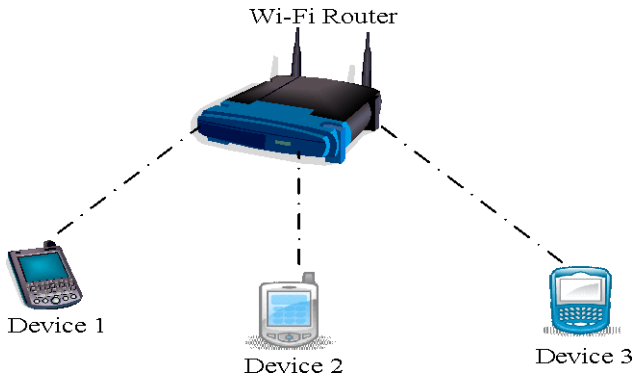
BT is a low cost, low power, short range wireless technology for data & voice communication. Bluetooth operates at 2.4GHz of the Radio Frequency spectrum and uses the Frequency Hopping Spread Spectrum (FHSS) technique for communication. Bluetooth supports a data rate of up to 1Mbps to 24Mbps and a range of approximately 30 to 100 feet for data communication. 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 and protocol part is responsible for defining the rules of communication.

Bluetooth enabled devices contain a Bluetooth wireless radio for the transmission and reception of data. 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. A Bluetooth device can function as either master or slave. A network formed with one Bluetooth device as master and more than one device as slaves is known as Piconet.

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.

## 6. Wireless Fidelity (Wi-Fi)

Wi-Fi is a 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. 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. Wi-Fi operates at 2.4GHZ or 5GHZ of radio spectrum.



For communicating, 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 supports data rates ranging from 1Mbps to 150Mbps depending on the standards (802.11a/b/g/n). Depending on the type of antenna and usage location (indoor/outdoor), Wi-Fi offers a range of 100 to 300 feet.

## 7. ZigBee

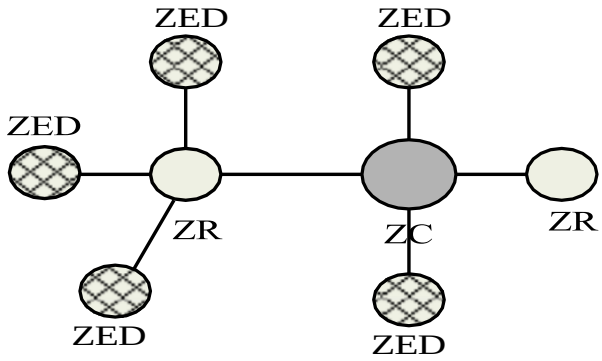
ZigBee is a low power, low cost, wireless network communication protocol based on the IEEE 802.15.4-2006 standard. ZigBee is targeted for low power, low data rate and secure applications for Wireless Personal Area Networking (WPAN). The ZigBee specifications support a robust mesh network containing multiple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

In the ZigBee terminology, each ZigBee device falls under any one of the following ZigBee device category:

**ZigBee Coordinator (ZC) / Network Coordinator:** It 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 is primarily targeting application areas like Home & Industrial Automation, Energy Management, Home control/security, and sensor networks & active RFID.

Automatic Meter Reading (AMR), smoke and detectors, heating control, lighting controls, environmental controls etc. are examples for applications which can make use of the ZigBee technology.

## 8. General Packet Radio Service (GPRS)

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 maximum transfer rate of 171.2kbps. In GPRS communication, the radio channel is concurrently shared between several users instead of dedicating a radio channel to a cell phone user. 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. GPRS is an old technology and it is being replaced by new generation data communication techniques (EDGE).

## Embedded Firmware

Embedded firmware refers to the control algorithm (Programs/Instructions) and/or the configuration settings that an embedded system developer dumps into the code (Program) memory of the embedded system. Various methods available for developing the embedded firmware are:

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.
   - IDE's are different for different family of processors/controllers.
2. Write the program in Assembly language using the instructions supported by your application's target processor/controller.

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'. (Machine Language)

For beginners, High Level Language (HLL) is recommended, because:

1. Writing codes in a high level language is easy.
2. **HLLs are not developer dependent:** Any skilled programmer can trace out the

functionalities of the program by just having a look at the program.

3. **HLL is highly portable:** You can use the same code to run on different processor/ controller with little or less modification.

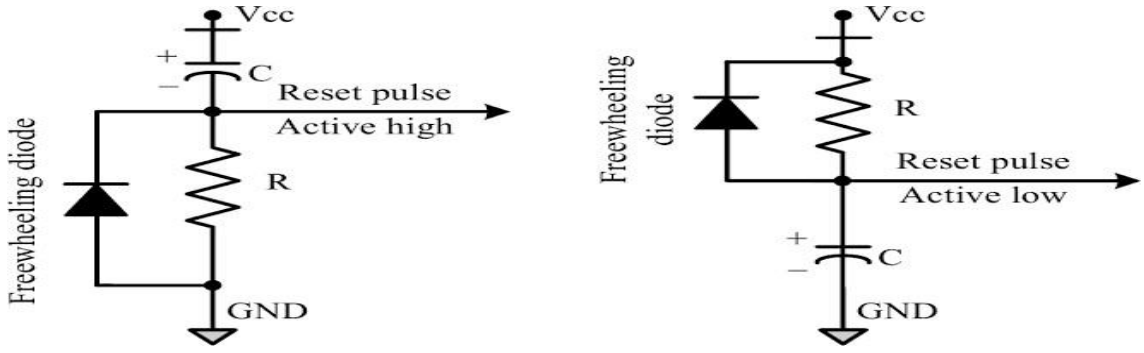
## Other System Components

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.

### 1. 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 (Processor undergoes reset when the reset pin of the processor is at logic high) or active low (Processor undergoes reset when the reset pin of the processor is at logic low). 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.




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.

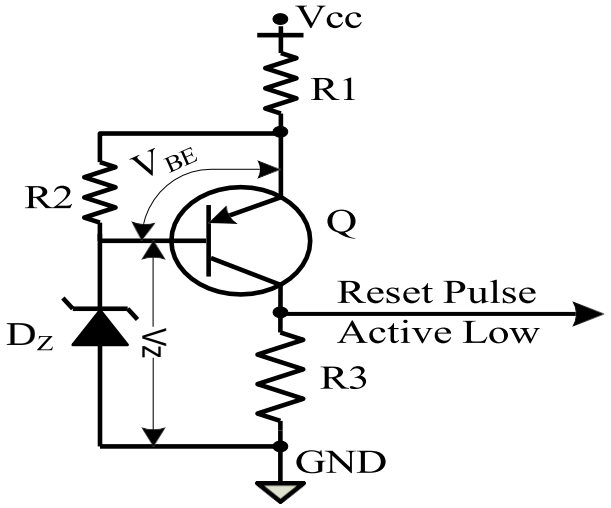
### 2. Brown-out Protection Circuit

*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 battery powered devices since there are greater chances for the battery voltage to drop below the required threshold. A brown-out protection circuit holds the processor/ controller in reset state, when operating voltage falls below the threshold, until it rises above the threshold voltage.

Figure illustrates a brown-out circuit implementation using Zener diode and transistor for processor/ controller with active low Reset logic. The Zener diode, Dz, and transistor, Q, forms the heart of this circuit. The transistor conducts always when the supply voltage Vcc is greater than that of the sum of $V_{BE}$ and Vz (Zener voltage). The transistor stops conducting when the supply voltage falls below the sum of $V_{BE}$ and Vz.

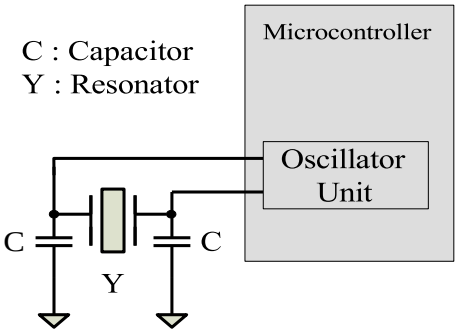
Select the Zener diode with required voltage for setting the low threshold value for Vcc. The values of Rl, R2, and R3 can be selected based on the electrical characteristics of the transistor in use.




### 3. Oscillator Unit

A microprocessor/ microcontroller is a digital device and its instruction execution occurs in synchronization with a clock signal. The oscillator unit of the embedded system is responsible for generating the precise clock for the processor. Certain processors/ controllers integrate a built-in oscillator unit and simply require an external ceramic resonator/ quartz crystal for producing the necessary clock signals. 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.

Quartz Crystal Resonator




Crystal Oscillator

The speed of operation of a processor is primarily dependent on the clock frequency. 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.

### 4. 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 hours/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. 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. 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.

### 5. Watchdog Timer

A watchdog timer, is a hardware timer for monitoring the firmware execution and resetting the processor/microcontroller when the program execution hangs up. A 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. 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.

Most of the processors implement watchdog as a built-in component and provides status register to control the watchdog timer 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.

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.

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