UNIT-1
INTRODUCTION TO EMBEDDED SYSTEMS

1.1. EMBEDDED SYSTEMS Vs GENERAL COMPUTING SYSTEMS

- An embedded system is an electronic/electro-mechanical system designed to perform a specific function and is combination of both hardware and firmware (software).
  The program instructions written for embedded systems are referred to as firmware, and are stored in Read-Only-Memory or Flash memory.
- Every embedded system is unique, and the hardware as well as software is highly specialized to the application domain. Embedded systems are designed to do some specific tasks, rather than be a general purpose computer for multiple tasks.

General Purpose Computing system	Embedded system
It is combination of generic hardware and a general purpose OS for executing a variety of applications.	It is combination of special purpose hardware and software for executing specific set of applications
It contains General purpose operating system	It may (or) may not contain an operating system for functioning.
Applications are alterable by the user (Possible to add/remove user applications)	The firmware of the Embedded system is pre-programmed and non-alterable by the user.
Performance is the key deciding factor in the selection of the system. Always 'faster is better'.	Application specific Requirements (Performance, power requirements, memory usage, size, design, cost, etc) are the key deciding factors.
Less options for different levels of power management	More options to take the advantage of the power saving modes supported by the hardware and software.
Response requirements are not time-critical.	For certain category of embedded systems like mission critical systems, the response time requirement is highly critical.
Need not be deterministic in execution behaviour.	Execution behaviour is deterministic for certain types of Embedded systems like 'Hard Real Time' systems.

1.2. HISTORY OF EMBEDDED SYSTEMS:

In the olden days embedded systems were built around the vacuum tubes and transistor technologies. Embedded algorithms were developed in low level languages.

The first recognised modern embedded system was **Apollo Guidance Computer (AGC):**

- **developed by MIT** instrumentation laboratory (1960 – 1966).
- The AGC provided electronic interfaces for guidance, navigation, and control of the spacecraft. They ran the inertial guidance systems of both the command module (CM) and Lunar Excursion Module (LEM).
- Original design was based on 4K words of fixed memory (ROM) and 256 words of erasable memory (RAM). The final configuration was **36K** words of fixed memory and **2K** words of erasable memory.
- The clock frequency of the microchip used in AGC was 1.024 MHz and it was derived from 2.048 MHz crystal oscillator.
- AGC consisted of approximately 11 instructions and 16 bit word logic.
- Used around 5000 ICs (3-input NOR gates, RTL logic) supplied by Fairchild Semiconductor.
- The user interface unit of AGC is known as DSKY (display/keyboard). It looked like a calculator type keypad with an array of numerals, which is used for inputting the commands to the module.

The first mass-produced embedded system was **Autonetics D-17 guidance computer**

- Developed for the Minuteman-I missile in 1961
- Built using discrete transistor logic and a hard disk for memory
- The first Integrate Circuit was produced in September 1958 but computers using them didn't begin to appear until 1963. Some of their early uses were in embedded systems, notably used by NASA for the Apollo Guidance Computer and US military in the Minuteman-II intercontinental ballistic missile.

First generation	→ Refer 1.3
Second generation	→ Refer 1.3
Third generation	→ Refer 1.3

1.3. CLASSIFICATION OF EMBEDDED SYSTEMS

Embedded systems can be classified into different types, based on different criteria:

(a) Classification based on generation
(b) Classification based on complexity and performance
(c) Based on functional requirements
(d) Classification based on deterministic behaviour
(e) Classification based on Triggering

(a) Classification based on generation

1. First generation (1G):

- ✓ Built around 8-bit microprocessors like Intel 8085, Zilog Z80.
- ✓ Simple in hardware circuit with firmware developed in Assembly code.

Examples: Digital telephone keypads, stepper motor control units

2. Second generation (2G):

- ✓ Built around 16-bit microprocessors and 8-bit/16-bit microcontrollers
- ✓ Ex: Intel 8086 processor, 8051 microcontroller
- ✓ Instruction set for processors/controllers are powerful than first generation

Examples: Data Acquisition system (DAS),
Supervisory Control and Data Acquisition systems (SCADA)

3. Third generation (3G):

- ✓ Built around 32-bit microprocessors and 16-bit/32-bit microcontrollers
- ✓ A new concept of application and domain specific processors / controllers like Digital Signal Processors (DSPs), Application Specific Integrated Circuits (ASICs) came into the picture.
- ✓ Ex: Pentium, Motorola 68K, MSP430, ARM, TMS320
- ✓ Instruction set of processors became more complex and powerful than 2G
- ✓ The concept of instruction pipelining is also evolved.
- ✓ Dedicated embedded RTOS entered into the embedded market

Examples: Robotics, Media, Industrial and process control, networking, etc.

4. Fourth generation:

- ✓ Built around 64-bit microprocessors 32-bit microcontrollers
- ✓ The advent of System on Chips (SoC), reconfigurable processors, multi-core processors increase the performance.
- ✓ High performance, tight integration, miniaturization, and very powerful.
- ✓ High performance RTOS

Examples: Smart Phones, Mobile Internet Devices, High speed network routers

(b) Classification based on Complexity and performance

1. Small-scale Embedded systems:

- ✓ Embedded systems which are simple in application needs and where the performance requirements are not time-critical.
- ✓ Built around low performance and low cost 8 bit or 16 bit µp/µc.

Example: an electronic toy, stepper motor control, washing machine

2. Medium-scale Embedded systems:

- ✓ These are slightly complex in hardware and firmware requirements.
- ✓ Built around medium performance 16 or 32-bit microcontrollers, DSPs, RISCs
- ✓ Usually contain embedded operating system (GPOS or RTOS) for functioning

Examples: Industrial applications, Video games, ATM machine
Monitoring and Control of Manufacturing Equipment

3. Large-scale / Complex / Sophisticated Embedded systems:

- ✓ Highly complex hardware and firmware.
- ✓ Built around 32 or 64 bit RISC µp/µc (or) PLDs (or) Reconfigurable SoC (or) multi-core processors, PLD, FPGA
- ✓ They may contain multiple processors/controllers and co-units/ hardware accelerators for offloading processing requirements from the main processor.
  - (Ex: Decoding/Encoding of media, Cryptographic function implementation are implemented using co-processors/ hardware accelerators)
- ✓ Contains high performance Real Time Operating System (RTOS)
- ✓ Performance requirements are not time-critical

Examples: Mission critical applications like Aircraft operating and control Systems, Missile tracking system, flight control systems, nuclear plant Safety system

(c) Classification based on functional requirements:

1. Stand-alone embedded systems

- ✓ As the name implies, stand-alone systems work in stand-alone mode.
- ✓ They take inputs, process them and produce outputs.
- ✓ The inputs can be electrical signals from transducers / commands from human being such as pressing of a button.
- ✓ The outputs can be electrical signals to drive another system, an LED display / LCD display/DC motors.

Examples: Digital Camera, Air Conditioner, Microwave oven, DVD player

2. Real time systems

- ✓ Real-Time Embedded System is strictly time specific which means these embedded systems provides output in a particular/defined time interval.

- ✓ Embedded systems in which some specific task has to be done in a specified period are called Real-Time systems. They have time constrains and they have to work against some deadlines. Meeting the deadlines is the most important requirement of Real-Time system. Hence the response time requirement is highly critical for Real time systems. A Real Time System should not miss any deadline. For example, in a nuclear plant safety system, missing a deadline may cause loss of life /damage to property.

- ✓ These type of embedded systems provide quick response in critical situations which gives most priority to time based task performance and generation of output. That's why real time embedded systems are used in defence sector, medical and health care sector, and some other industrial applications where output in the right time is given more importance.

Examples: Mission critical applications like Aircraft operating and control Systems, Missile tracking system, flight control systems, Nuclear plant Safety system

3. Networked Information appliances

- ✓ Embedded systems that are provided with network interfaces and accessed by networks such as Local Area Network (LAN) or Internet are called as Networked Information appliances.
- ✓ Such embedded systems are connected to a network, typically a network running TCP/IP (Transmission Control Protocol/ Internet Protocol) protocol suite. They run the complete TCP/IP protocol stack and can communicate with other nodes on the network.

Examples: A web camera which is connected to internet, ATM machine, Card swipe machine, A networked process control system

4. Mobile devices

- ✓ Mobile embedded systems are compact, easy to use and require fewer resources. These are portable embedded devices like mobile phones, digital cameras, mp3 players and personal digital assistants, etc.

- ✓ Mobile phones are capable of supporting high data rate services in addition to the voice services. Accessing internet services such as e-mail, World Wide Web and so on can be done while a person is on the move. They are capable of handling multimedia applications.

- ✓ The limitation of mobile devices – small size, lack of good user interfaces such as small keyboard and display, memory constrains, battery life time etc.

(d) Classification based on deterministic behaviour

The embedded system under this classification is based on the deterministic behaviour of an embedded system. Based on the execution behaviour, the embedded systems are classified into hard real-time systems and soft real-time systems

1. Hard Real-Time Embedded systems:

- ✓ A real-time system should strictly adhere to the timing constraints for a task.
- ✓ A hard real-time system must meet the timing deadlines without any delay.
- ✓ Missing the deadline would cause serious failure, including permanent data lose and irrecoverable damages to the system or user.
- ✓ Hard real-time systems emphasise the principle –

  "A late answer is always a wrong answer"

- ✓ The airbag control system and antilock braking system of vehicles are typical examples for hard real-time systems. When a vehicle is met with an accident, the airbag control system should operate immediately without any delay to safeguard the passenger. If there is any delay in the deployment of airbags, it will lead to the death of passengers in the vehicle.

2. Soft Real-Time Embedded systems:

- ✓ The real time embedded system which does not guarantee meeting deadlines, but offer the best effort to meet the deadline are called as 'Soft Real-Time embedded systems.
- ✓ Missing deadlines for tasks are acceptable for soft real-time systems, but the frequency of deadlines missing should be within the compliance limit of the Quality of Service (QoS).
- ✓ Soft real-time systems emphasise the principle –

  "A late answer is an acceptable answer", but it could have been done a bit faster.

- ✓ ATM is a typical example of a soft real-time system. While withdrawing money from ATM, if it takes a few seconds more than the normal operating time, it may not cause any serious problem.
- ✓ An Audio-Video play back system is another example for Soft-Real time system. No potential damage arises if sample comes late by fraction of a second, for playback.

(e) Classification based on Triggering

- ✓ The embedded systems are classified into two types based on the triggering of the systems: time-triggered and event-triggered.
- ✓ If a system is activated or triggered based on the pre-defined task or preset time, then such a system is said to be the time-triggered embedded system.
  Ex: Automatic Street light control system based on RTC input
- ✓ On the other side, if the system is triggered based on some activity like change in temperature or change in pressure, such system is said to be an event triggered embedded system
  Ex: Automatic Street light control system based on Light intensity

1.4. MAJOR APPLICATION AREAS & EXAMPLES OF E.S.

Every embedded system is unique, and the hardware as well as software is highly specialized to the application domain.
The different applications/Examples of Embedded systems are given below

1. *Consumer Electronics*: Camera, Camcorders (Video capture and recording)
2. *Household appliances*: Digital TVs, DVD players, Set top boxes, Washing machine, Refrigerator.
3. *Automotive industry*: Anti-lock breaking system (ABS), Engine control, automatic Navigation system, Motor control system
4. *Home automation & security systems*: Air conditioners, sprinklers, fire alarms, CC cameras, home security system
5. *Telecom*: Cellular phones, telephone switches, handset multimedia applications
6. *Computer peripherals*: Printers, scanners, fax machines
7. *Computer networking systems*: Network routers, switches, hubs, firewalls
8. *Healthcare*: EEG, ECG machines, Patient monitoring system
9. *Banking & Retail*: Automated teller machine (ATM), Currency counters
10. *Card Readers*: Barcode, smart card readers.
11. *Measurements & Instrumentation* : Logic Analyzers, Spectrum analyzers, PLC systems, Electronic data acquisition and supervisory control system, industrial process controller, digital meters
12. *Missiles and Satellites* : Defence, Aerospace, Communication, tracking system
13. *Robotics* : stepper motor controllers for a robotic system
14. *Entertainment systems* : video games, music system
15. *Signal & Image processing* : speech processing, pattern recognizer, video processing

1.5. PURPOSE OF EMBEDDED SYSTEMS

The embedded systems are used in various domains like Consumer electronics, home automation, telecommunications, computer networking system, automotive industry, healthcare, Instrumentation, retail and banking applications, etc. Within the domain itself, according to the application, they may have different functionalities.

Each embedded system is designed to serve the **purpose** of any one or a combination of the following tasks:

- ✓ Data Collection/Storage/Representation
- ✓ Data communication
- ✓ Data (signal) processing
- ✓ Monitoring
- ✓ Control
- ✓ Application specific user interface

Data Collection/Storage/Representation:

- Embedded systems 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.
- The data refers all kinds of information such as text, voice, image, video, electrical signals and any other measurable quantities.
- Embedded systems with analog data capturing techniques collect data directly in the form of analog signals. Embedded systems with digital data collection mechanism converts the analog signal to corresponding digital signal using analog to digital (A/D) converters and then collects the binary equivalent of the analog data.
- The collected data may be
  - stored directly in the system or
  - transmitted to some other systems or
  - processed by the system or
  - displayed for giving a meaningful representation.

Ex : A digital camera is a typical example of an embedded system with data collection/storage/representation of data.

Data Communication:

- Embedded data communication systems are deployed in applications ranging from complex satellite communication systems to simple home networking systems.
- The transmission is achieved either by a wire-line medium or by a wireless medium.
- The data collecting embedded terminal itself can incorporate data communication units like wireless modules (Bluetooth, ZigBee, Wi-Fi, EDGE, GPRS, etc.) or wire-line modules (RS-232C, USB, TCP/IP, PS2, etc.).
- Certain Embedded systems act as dedicated transmission units between sending and receiving terminals. (Network hubs and routers)

Ex : Wireless network router for data communication

Data (Signal) Processing:

- The data (voice, image, video, electrical signals and other measurable quantities) collected by embedded systems may be used for various kinds of data processing.
- Embedded systems with signal processing functionalities are employed in applications demanding signal processing like speech coding, synthesis, audio video codec, transmission applications, etc.

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

Monitoring:

- These systems are specifically designed for monitoring purpose.
- Almost all embedded products coming under the medical domain are with monitoring functions only.
- They are used for determining the state of some variables using input sensors.

Ex: ECG machine for monitoring the heartbeat of a patient
Digital oscilloscope, Digital multi-meters, Logic analysers

Control

- Embedded systems are basically designed to regulate a physical variable (or) to manipulate the state of some devices by sending some signals to the actuators or devices connected to the output ports of the system, in response to the input signals provided by the end users or sensors which are connected to the input ports.

- An Embedded system with control functionality contains both sensors and actuators.
- **Sensors** are used to sense/detect the changes in the input variables and convert into electrical signals for any measurements. The sensors are connected to the input port.
- **Actuators** are used to converts electrical signals into corresponding physical action.
- The actuators are connected at the output port.

Ex: Air conditioner system - contains temperature sensing element (sensor) which may be a thermistor and a handheld unit for setting up the desired temperature.
The air compressor unit acts as the actuator.

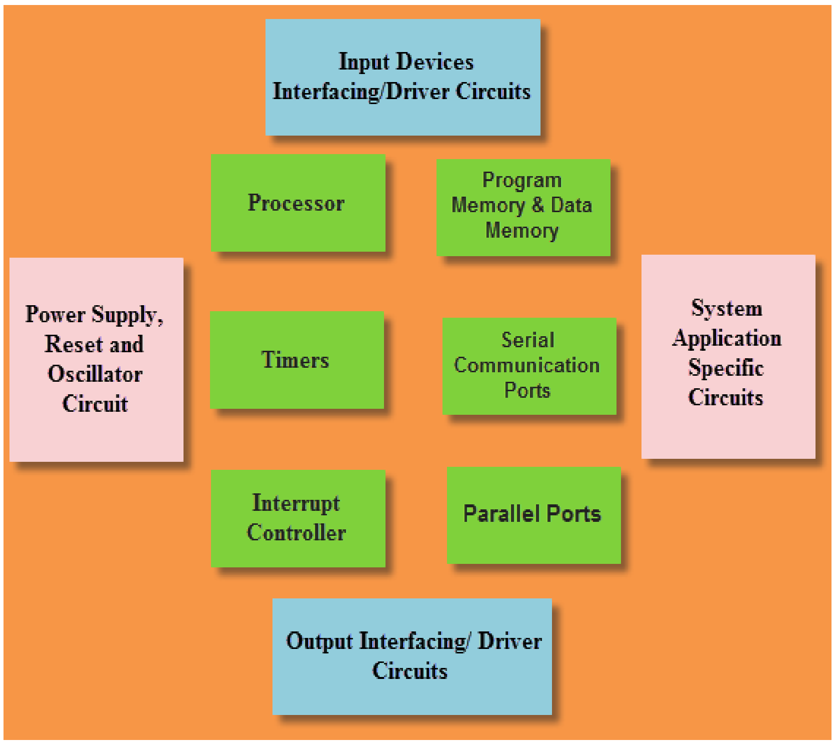
Application Specific User Interface

These are embedded systems with application-specific user interfaces like buttons, switches, keypad, lights, bells, display units, etc.
Mobile phone is an example for this.
In mobile phone the user interface is provided through the keypad, graphic LCD module, system speaker, vibration alert, etc.

1.6. EMBEDDED HARDWARE UNITS AND DEVICES

An embedded system is an electronic/electro-mechanical system designed to perform a specific function and is combination of both hardware and firmware (software). Every embedded system is unique, and the hardware as well as software is highly specialized to the application domain.

The basic hardware units of embedded system are shown in figure:


Input Devices Interfacing/Driver Circuits
Processor
Program Memory & Data Memory
Power Supply, Reset and Oscillator Circuit
Timers
Serial Communication Ports
System Application Specific Circuits
Interrupt Controller
Parallel Ports
Output Interfacing/ Driver Circuits

1.6.1. Power Source

- ✓ Most of the Embedded Systems have their own power.
- ✓ Various units in an Embedded system operate in one of the operating ranges of 5.0 V ±0.25V, 3.3V±0.3V, 2.0V ±0.2V, and 1.5V±0.2V.
- ✓ There is generally an inverse relationship between propagation delay in the logic gates and operational voltage. Therefore the 5V system processor and units used in most high performance systems.
- ✓ Certain systems do not have a power source of their own; they connect to external power supply or they are powered by host system.
- ✓ Low Voltage operations
  - In portable or handheld devices (when compared to 5V, a CMOS 2V circuit power dissipation reduces by one-sixth).
  - In smaller geometry systems, low voltage system processors and I/O circuits generate lesser heat.

1.6.2. Oscillator Circuit and Clocking Units

- ✓ The Oscillator unit is responsible for generating the precise clock for the processor, memory and all peripherals.
- ✓ Clock may be generated internally (or) obtained from a crystal or external source
- ✓ The Clock is required to keep the whole system synchronized
- ✓ The speed of operation of a processor primarily depends on the clock frequency.
- ✓ The total system power consumption is directly proportional to the clock frequency.

1.6.3. System Timers and Real-Time Clocks

- ✓ Timers are essential to almost any embedded application, used for
  - ▪ Generate fixed-period events
  - ▪ Periodic wakeup
  - ▪ Count edges / events
  - ▪ Measuring time intervals
  - ▪ Generate delays - Replacing delay loops with timer calls allows CPU to sleep, consuming less power
- ✓ A Timer circuit ticks and generates system interrupts periodically. The Interrupt Service Routine (ISR) then perform the required operation.

- ✓ The Real Time Clock (RTC) generates system interrupts periodically for schedulers, real time programs and for periodic saving of time and date in the system.
  The RTC module provides seconds, minutes, hours, day of week, day of month, month, and year in real-time clock with calendar function.

1.6.4. Reset Circuit, Power-up Reset and Watchdog-Timer Reset

- ✓ Reset means that the processor begins the processing of instructions from a starting address. The reset signal brings the internal registers and different hardware systems of the processor/controller to a known state and starts the firmware execution from the reset vector (normally from 0x0000)
- ✓ The reset can be activated by
  - an external reset circuit that activates on power-up,
  - switching on reset of the system or
  - detection of low voltage (< 4.5V for 5V system)

- ✓ Watchdog timer (WDT) is a timing device that resets that resets the system after a predefined timeout. In many embedded systems reset by a watchdog timer is very essential because it helps to bring back to initial state when a fault occurs.

- ✓ WDT is used to detect and recover from computer malfunctions. The WDT restarts the system on occurrence of a software problem (or) if a selected time interval expires. This is a safety feature, which resets the processor if the program becomes stuck in an infinite loop.

1.6.5. System core:

Embedded systems are domain and application specific and are built around a central core. The core of the embedded system falls into any of the following categories:

1. General purpose and Domain specific processors
   - Microprocessors
   - Microcontrollers
   - Digital Signal Processors
2. Application Specific Integrated Circuits (ASIC)
3. Programmable Logic Devices (CPLD and FPGA)
4. Commercial off-the-shelf components (COTS)

The System core /Central processing unit includes

- ✓ Arithmetic logic unit (ALU), which performs computation.
- ✓ Registers needed for the basic operation of the CPU, such as the program counter (PC), stack pointer (SP), and status register (SR).
- ✓ Further registers to hold temporary results
- ✓ Instruction decoder and other logic to control the CPU operations, to handle resets, interrupts, and so on.

1.6.6. Memory

The **memory** of the system is responsible for holding the control algorithm and other important configuration details. Memory for implementing the code may be present on the processor or may be implemented as a separate chip interfacing the processor. In a microcontroller based embedded system, the microcontroller contains internal memory for storing code. Various forms of memories used in embedded system are given below:

ROM (or) EPROM (or) Flash → Storing Codes for RTOS, Codes for each application Program, ISR and tasks, Codes for boot program, Addresses of various ISRs

RAM (Internal and external) → Storing the variables during program run and storing Stack Storing input output buffer

Memory stick → A flash memory stick is inserted in mobile computing system (or) digital Camera, which is used to store HD video, images, music, speech..etc

EEPROM (or) Flash → Storing Nonvolatile results of Processing

Cache → Storing copies of instructions and data in advance from external primary memory and storing results temporarily during processing.

1.6.7. I/O ports, I/O Buses and I/O interfaces

- ✓ The system gets inputs from physical devices through the input ports.
  Ex: A system gets inputs from the touch screen, keys in keypad, sensors and transducer circuits.

- ✓ The system has output ports through which it sends data to outside world.
  Ex: Output may be sent to an LED, LCD, Touch Screen display, Printer.
  A control system sends the outputs to alarms, actuators, or boilers.
  A robot will send out output for its various motors.

- ✓ A bus consists of a group of conducting lines to connect multiple devices, hardware units and systems for communication between any two of these. A bus may be a serial or parallel bus that transfers one or multiple data bits.

- ✓ An Embedded system connects to external physical devices and systems through parallel I/O ports.

- ✓ On-board communication interfaces → I2C, SPI, UART, 1-wire
  External communication interfaces → RS-232, USB, IEEE 1394
  Wireless → Infrared (IrDA), Bluetooth (BT), Wi-Fi, ZigBee

1.6.8. LCD, LED and Touchscreen Displays

- ✓ LCD screen shows a multiline display of characters or also show a small graph or icon. LCD needs little power to operate. LCD is a diode that absorbs or emit and 3-4 V and 50 Hz voltage pulses with currents less than 50 μA.
- ✓ LED is used to indicate the ON status of the system. The LED is a diode that emits light of different colours. Its emitting light is much brighter than LCD.
- ✓ A touch screen is an input as well as an output device, which can be used to enter a command and to give reply.

1.6.9. Keypad/Keyboard

- ✓ The keypad (or) keyboard is used for getting user inputs. The keypad has upto a maximum of 32 keys. A keyboard may have upto 104 keys or more.
- ✓ Mobile phones may have a T9 keypad has 16 keys and four up down right left menu keys.
- ✓ The system may need the necessary interfacing circuit and software to receive inputs directly from Keys and to send data to Display unit

1.6.10. Interrupt Handler

- ✓ The system provides an interrupt handling mechanism for executing the ISRs, in the case of interrupts from physical devices, systems, software instructions and software exceptions.
- ✓ The interrupt handler manages the interrupts which are generated by external devices connected to the microcontroller (or) Interrupts requested by most peripheral modules in the core of the MCU, such as Timer, ADC, UART.. etc.
  Ex: A Timing device sends a timeout interrupt when a preset time elapses.
- ✓ An interrupt may be hardware signal that indicates the occurrence of an event.
  A software interrupt may arise in an exceptional condition that may have developed while running a program.

1.6.11. DAC Using PWM and an ADC

- ✓ DAC is a circuit that converts digital 8 or 10 or 12 bits to the analog output.
- ✓ When all input bits to the DAC are 1's, then the analog output is the difference between the positive and negative reference pin voltages.
- ✓ When all input bits are 0's, then the analog output voltage equals to negative reference pin voltage. (usually 0 V).

- ✓ A pulse width modulator (PWM) with an integrator circuit is used for the DAC. The pulse width is made proportional to the analog input needed.

- ✓ ADC is a circuit that converts the analog input to digital 4,8,10 or 12 bits.
- ✓ The analog input is applied between the positive and negative pins and is converted with respect to reference voltage.
- ✓ When input is equal to difference between the positive and negative reference voltages, then all output bits equal 1.
- ✓ When input is equal to negative reference voltage (usually 0 V), then all output bits equal 0.
- ✓ The ADC in the system can be used in many applications such as data acquisition systems, digital cameras, analog control system.

1.6.12. Pulse Dialer, Modem and Transceiver

- ✓ In communication system, for user connectivity through the telephone line, the system provides the necessary interfacing and circuits.
- ✓ The system also provides software for pulse dialing through the telephone line, for modem connection for fax, for internet packets routing and for transmitting to a wireless cellular system.
- ✓ A transceiver is a circuit that can transmit as well as receive byte stream.

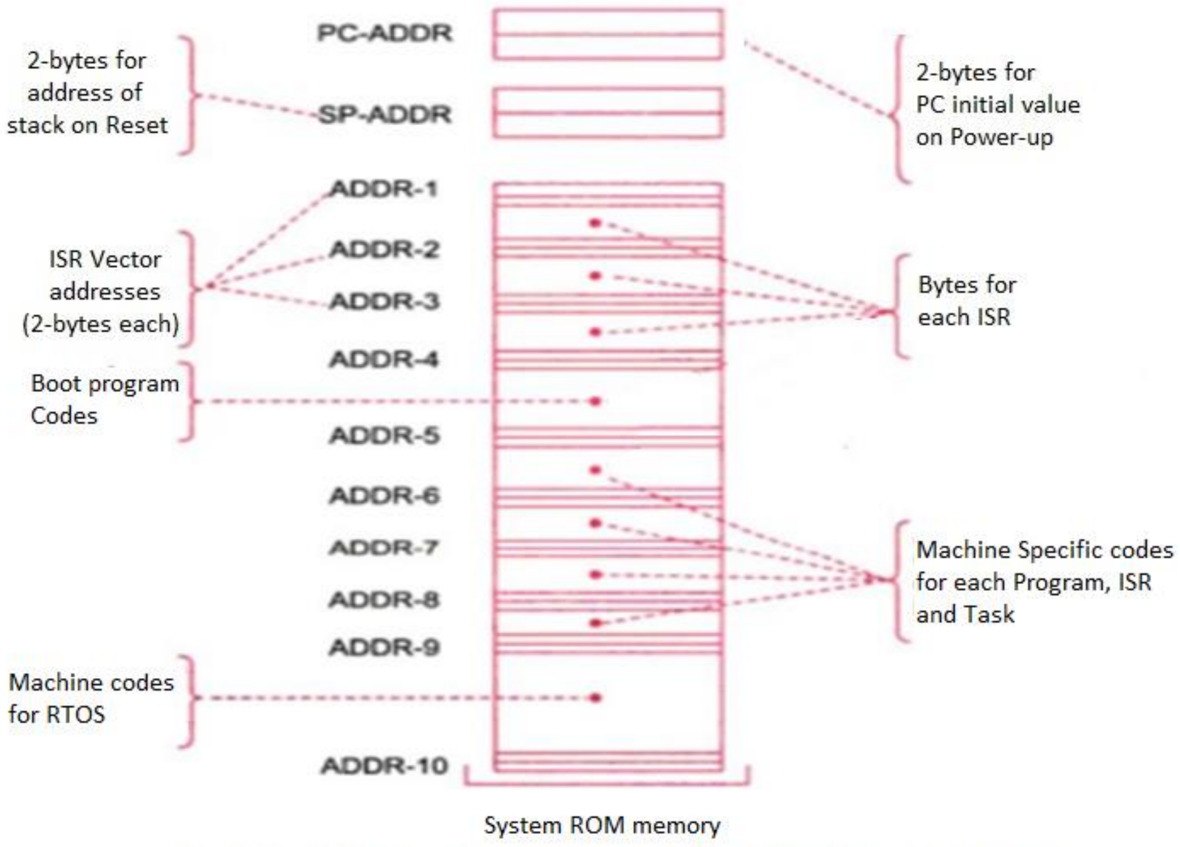
1.7. EMBEDDED SOFTWARE & PROGRAMMING LANGUAGES (OR) EMBEDDED FIRMWARE

The program instructions written for embedded systems are referred to as firmware, and are stored in Read-Only-Memory (ROM) or Flash memory.

Embedded firmware refers to the control algorithm (program instructions) and configuration settings the embedded system developer dumps into Program memory (ROM) of the embedded system.

1.7.1. Final Machine Implementable Software for a System:

- ✓ An Embedded system processor executes software that is specific to a given application of that system. The instruction codes and data in the final phase are placed in the ROM (or) Flash memory. This final stage software is also called the ROM image.
- ✓ A machine implementable software file is table having in each rows the address and bytes. The bytes are saved at each address of the system memory.
- ✓ The figure shows the ROM image in a system memory. The image consists of PC address pointers, Stack address pointers, boot up program, application programs, ISRs, RTOS, input data and vector addresses.



PC-ADDR
SP-ADDR
ADDR-1
ADDR-2
ADDR-3
ADDR-4
ADDR-5
ADDR-6
ADDR-7
ADDR-8
ADDR-9
ADDR-10
2-bytes for address of stack on Reset
ISR Vector addresses (2-bytes each)
Boot program Codes
Machine codes for RTOS
2-bytes for PC initial value on Power-up
Bytes for each ISR
Machine Specific codes for each Program, ISR and Task
System ROM memory

Fig: **System ROM embedding the software, RTOS, data and vector addresses**

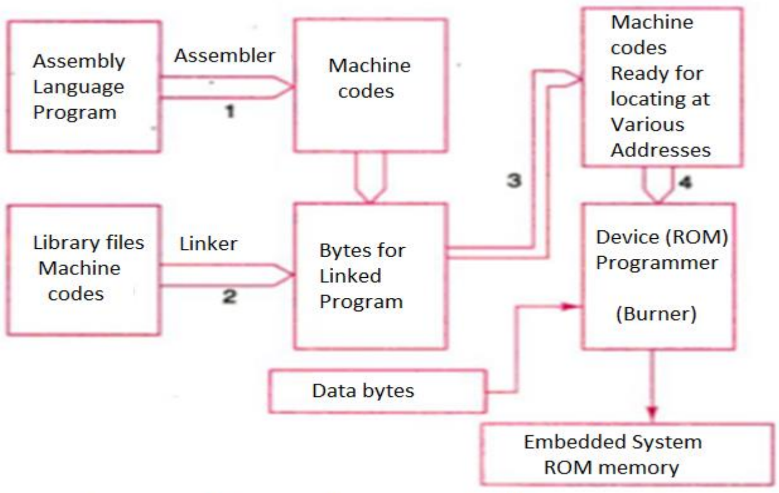
1.7.2. Coding (Programming languages)

(i) Coding of software in Machine codes

- ✓ To configure some specific physical device or subsystem, machine code based coding is used.
- ✓ Coding in machine implementable codes is time consuming.
- ✓ Programmer must understand the processor instructions set and memorize the instructions and their machine codes.

(ii) Software in Processor specific Assembly Language

- ✓ The assembly language uses 2 or 3 letter '***mnemonics***' to represent each low-level machine instruction (or) Op-code.
- ✓ To write an assembly language, the programmer must know the instruction set, architectural features and register set of the processor.
- ✓ The instruction set for each family of processor/controller is different.
- ✓ Full coding in assembly language may be done for small scale systems such as toys, automatic chocolate vending machines, robots and data acquisition system.


Assembly Language Program
Assembler
1
Machine codes
Machine codes Ready for locating at Various Addresses
3
4
Library files Machine codes
Linker
2
Bytes for Linked Program
Device (ROM) Programmer (Burner)
Data bytes
Embedded System ROM memory

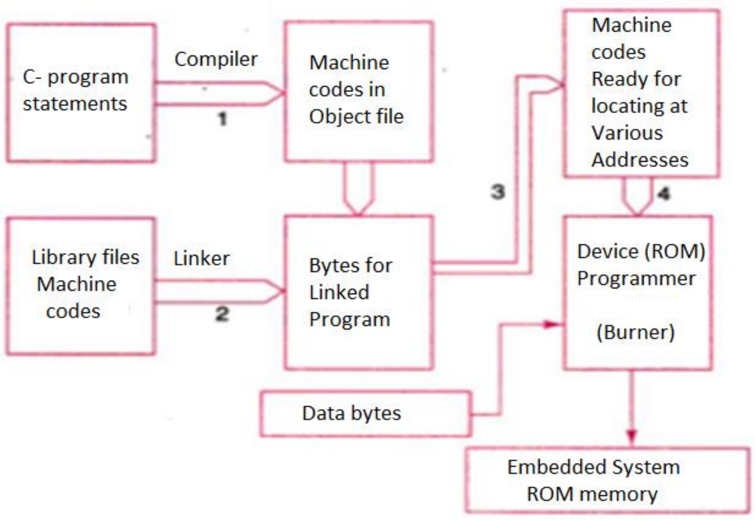
Process of converting ALP into machine codes and finally obtaining the ROM image

The figure (a) shows the process of converting an assembly language program into machine implementable software file and then finally obtaining a ROM image file.

1. An assembler is a software program that converts the assembly language instructions into machine codes.
2. A **linker** program combines the machine codes with the other library codes required.
3. In computer systems a **loader** is the part of an operating system that is responsible for loading programs and libraries.
4. A **Locator** is a tool that performs the conversion from re-locatable program to executable binary image that can be loaded into the target ROM.
5. Lastly, the device programmer takes the ROM image as input file and finally burns the image into the PROM or flash. The process of placing the codes in PROM is called burning.

(iii) Software in High level language:

- ✓ Since the coding in Assembly language is very time consuming in most cases, software is developed in a high level language - C/C++ /Visual C++ /Java/Python.
- ✓ Writing the program in High level language is easy and portable which means the same code can run on different processor with little modifications.
- ✓ The programs written in high level language are not developer dependent. Any skilled programmer can trace out the functionalities of the program.
- ✓ Overall system development time will be reduced to a greater extent.


C- program statements
Compiler
1
Machine codes in Object file
Machine codes Ready for locating at Various Addresses
3
4
Library files Machine codes
Linker
2
Bytes for Linked Program
Device (ROM) Programmer (Burner)
Data bytes
Embedded System ROM memory

Process of converting C-program into machine codes and finally obtaining the ROM image

The figure (b) shows the process of converting a C- program into machine implementable software file and then finally obtaining a ROM image file.

Compiler is a software program that converts a source code from high level language *(human readable language)* into low level language *(object code or machine code)* to create an executable program.

1.7.3. Software tools for designing embedded system:

A programming tool (or) software development tool is a program that software developers used to create, debug, maintain and support other programs and applications.

Editor :

- It is used for writing assembly mnemonics (or) high level programs like C/C++
- It allows entry, addition, deletion, insert, modification of programs.

Interpreter :

- It is used for expression by expression (line-by-line) translation to machine executable codes.

Compiler :

- Compiler is a software program that converts a source code from high level language *(human readable language)* into low level language *(object code or machine code)* to create an executable program.
- The compiler that runs on a computer platform and produces code for that same computer platform is called as Host compiler.
- A Compiler that runs on one computer platform and produces code for another computer platform is called as **Cross-compiler**.

Assembler :

- An assembler is a software program that converts the assembly language instructions into machine codes.
- An assembler that runs on host but produces binary codes appropriate for some other target is called **Cross-assembler**.

Linker:

- A linker is a program that takes one (or) more object files generated by compilers and combines them into a single executable file. Source code may be contained in more than one file. Hence, it must be combined.

Loader :

- In computer systems a **loader** is the part of an operating system that is responsible for loading programs and libraries.

Locator :

- A Locator is a tool that performs the conversion from re-locatable program to executable binary image that can be loaded into the target ROM.

Debugger :

- A **debugger** (or) **debugging tool** is a computer program that is used to test and debug other programs (the "target" program).

Simulator:

- It is a software tool used to simulate all functions of an embedded system including memory and peripherals. It also simulates the processes that will execute when the codes of a particular processor executes.

Integrated Development Environment (IDE)

- IDE is a fully integrated tool that consolidates basic software tools required for edit and test the embedded software. IDE consists of Editor, Compiler, Assembler, Cross assembler, Debugger, Simulator, Emulator, Logic analyzers and EPROM application codes burner.

1.7.4. Program Models for Software Designing

- The different models that are employed during the design process of the embedded software are as follows:
  - ✓ Sequential Program Model
  - ✓ Object oriented Program model
  - ✓ Control Data Flow graph Model (or) Synchronous Data Flow Graph Model.
  - ✓ Finite State Machine for Data path
  - ✓ Multithreaded Model for concurrent Processing.

1.7.5. Software for Device Drivers and Device Management in an OS :

- ✓ Am Embedded system is designed to perform multiple functions and has to control multiple physical and virtual devices.
- ✓ In an embedded systems, there may be no. of physical devices like Timers, Keyboards, Display, flash memory, parallel ports, and network cards.
- ✓ A program is also be developed using the concept of Virtual device. Examples of virtual device are :
  - A file (of records opened, read, written, closed and saved a stream of bytes)
  - A pile (for sending and receiving a stream of bytes from a source to destination)
  - A socket (for sending and receiving a stream of bytes b/w a client and server)
  - A RAM disk (for using the RAM in a way similar to files on the disk)

- ✓ The term virtual device follows from the analogy that just as a keyboard gives an input to the processor for a ***read*** and similarly a ***file*** also gives an input to the processor. The processor gives an output a printer for a ***write*** and similarly the processors writes an output to a file.

- ✓ A device for the purpose of control, handling, reading and writing actions consists of three components :

  (i) A control register – to control the device actions
  (ii) A status register – to show the device status (flags) to the device driver
  (iii) A device mechanism that controls the device actions

- ✓ A **device driver** is a software for opening (configuring), connecting or binding, reading, writing, closing and controlling actions of the device. It is written in high level language.
- ✓ A device driver controls three functions

  (i) initializing – by placing appropriate bits in Control register
  (ii) calling an ISR on interrupt and
  (iii) resetting the status flag after an interrupt service.
- ✓ A device driver accesses a parallel or serial port, keyboard, mice, disk, network, display, file, pipe and socket at specific addresses.

- ✓ A **device manager** software provides codes for detecting the presence of devices, for initializing and testing these devices that are present. It ensures that any device accesses to **one task only at any given instant**.

- ✓ The RTOS includes device drivers and device manager to control and facilitate the use of no. of physical and virtual devices in the system.

1.7.6. Software for concurrent Processing, Scheduling of Multiple Tasks and ISRs using an RTOS

- ✓ An embedded system program is most often designed using multiple processes (or) multitasks (or) multithreads.
- ✓ The multiple tasks are processed most often by the OS concurrently.
- ✓ Concurrent processing tasks can be interrupted for running ISRs, and a higher priority task pre-empts the running of low priority tasks.
- ✓ RTOS is used in most embedded systems which provides the OS functions for coding the system, provides IPC functions and controls the passing of messages and signals to a task.

1.8. SKILLS REQUIRED FOR EMBEDDED SYSTEM DESIGNER

An Embedded System designer has to develop a product using the available tools within the given specifications, cost and time frame.

Skills for Small Scale Embedded System Designer:

- ✓ Good knowledge of Microprocessor or Microcontroller to be used
- ✓ Computer architecture and organization
- ✓ Memories
- ✓ Memory allocation
- ✓ Interfacing memories
- ✓ Burning the executable machine codes in ROM
- ✓ Use of decoders and de-multiplexers
- ✓ Direct Memory Access
- ✓ Ports
- ✓ Device drivers
- ✓ Simple and Sophisticated Buses
- ✓ Timers
- ✓ Interrupt Service Mechanism
- ✓ C-Programming
- ✓ Memory optimization
- ✓ Selection of hardware and Microcontroller
- ✓ Use of ICE (In-Circuit Emulator), Cross Assembler and Testing equipment
- ✓ Debugging the software and Hardware bugs by Test vectors
- ✓ Basic Knowledge in other areas - Software engineering, data communication, Digital electronic design, control engineering, motors, actuators, sensors, measurements, analog electronic design and IC design and manufacturing

Skills for Medium Scale Embedded System Designer

- ✓ Knowledge of C /C++ /Java programming, RTOS programming and program modelling skills are must to design medium scale embedded-system.
- ✓ Tasks or threads and their scheduling by RTOS.
- ✓ Cooperative and pre-emptive scheduling
- ✓ Inter Process Communication functions
- ✓ Use of shared data ,programming critical sections and re-entrant functions
- ✓ Use of semaphores, mail boxes, queues, sockets and pipes
- ✓ Handling of interrupt latencies and meeting task deadlines.
- ✓ Use of various RTOS functions.
- ✓ Use of physical and virtual device drivers.
- ✓ Application Programming Interfaces (API)

Skills for Sophisticated Embedded System Designer

- ✓ A team is needed to co-design and solve the high level complexities of hardware and software design.
- ✓ Hardware engineers should have skills in hardware units and basic knowledge of C/C++ and JAVA,RTOS and other programming tool required.
- ✓ Software engineers should have basic knowledge in hardware and knowledge of C, RTOS and other programming tools.
- ✓ A final optimum design solution is then obtained by system integration.

1.9. PROCESSOR AND OS TRENDS IN EMBEDDED SYSTEM

The embedded industry has evolved a lot from the first mass produced embedded system Autonetics D-17 to the recently launched Apple iPhone, in terms of miniaturisation, performance, features, development cost and time to market.

Earlier (in 1970's)

- ✓ The embedded systems were based on 8-bit microprocessors/controllers.
- ✓ They used N-number of ICs for building a device.
- ✓ They require processor/controller, Reset circuit, Brown out circuit, ADC /DAC, Watchdog timer ICs ..etc separately for building a simple embedded system.
- ✓ Operating frequency in few MHz.

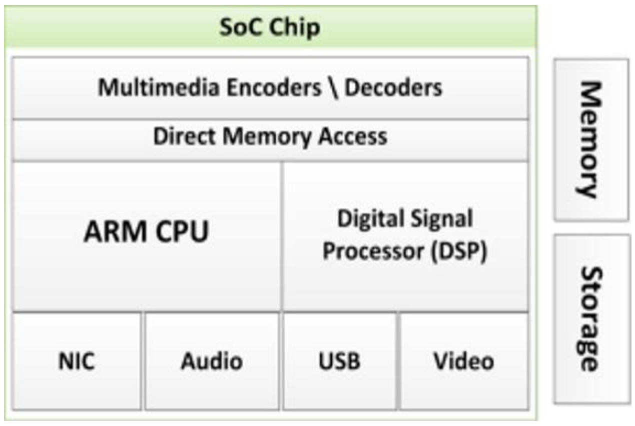
Now

- ✓ The embedded systems were based on 32-bit/64-bit processors/controllers/DSP
- ✓ ARM, Multicore, Reconfigurable processors
- ✓ With high degree of integration, processors and controllers are integrating multiple IC functionalities into a single chip (System of Chip)
- ✓ Tight integration, Miniaturisation, High performance, Cost saving
- ✓ Operating frequency in the range of GHz

Revolution in processor technology is a prime driving factor in the embedded development area. The embedded industry has witnessed the improvements over processor design from the 8-bit processor of the 1970's to today's System on Chip and multi-core processors.

Performance enhancements offered by general computing processors like 80x86, Pentium, Pentium pro, P-II, P-III, P-IV, Celeron, Core to duo processors, RISC processors, ARM processors increased drastically.

System on Chip (SoC)

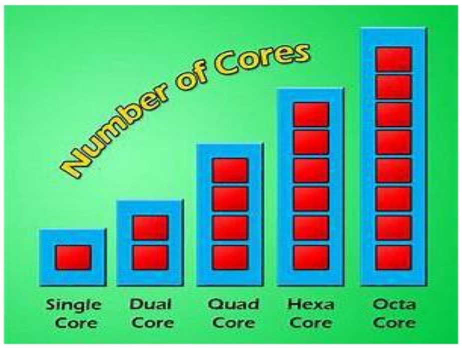


- ✓ As the name indicates, a System on Chip (SoC) makes a system on a single chip.
- ✓ The SoC technology places multiple function 'systems' on a single chip
  Nowadays SOCs are available for diverse application needs like Set Top Boxes, Portable media players, PDA..etc

- ✓ Ex: **iMX31 SoC** from free scale semiconductors is a typical example for SoC targeted for multimedia applications.
  It integrates the following on a single silicon wafer.
  - A powerful ARM 11 core,
  - USB OTG Interface,
  - Multimedia & human interface (Graphics Accelerator, MPEG4, Keypad)
  - Standard Interfaces (Timers, Watch Dog Timer, General Purpose I/O)
  - Image Processing Unit (Camera Interface, Image inversion/rotation) Display/TV control)

Multicore Processors/ Chip level Multi Processor (CMP):

- ✓ One way achieving increased performance is to increase the operating clock frequency, which increases the power consumption and cost.
- ✓ Multicore processors incorporate multiple processor cores on the same Chip and works on the same clock frequency supplied to the chip.
- ✓ Multicore processors implement multiprocessing (simultaneous execution)



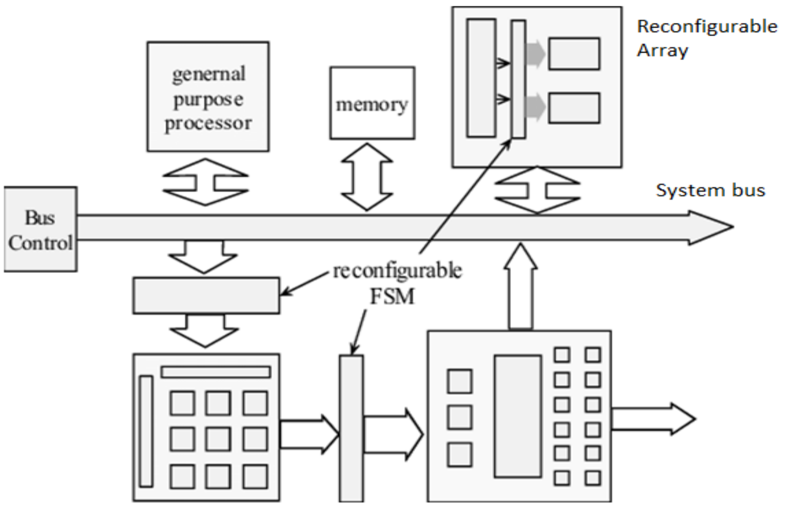
Dual core - 2 cores
Qual core - 4 cores
Hexa core - 6 cores
Octa core - 8 cores

Examples for multicore processors:

OCTEON™ CN3860 → Hexadeca core → secure network communication processor designed by Cavium networks – 16 MIPS processors cores – 1 GHz clock

Mediatek Helio G35 - Octa-core" processor → used in Android phones It has eight ARM Cortex A53 cores, with a max. frequency of up to 2.3GHz

Reconfigurable Processors



- ✓ Reconfigurable processor is a microprocessor/ controller with reconfigurable hardware features.
- ✓ They contain an array of Programming elements (PE) and RISC processor.
- ✓ Hardware features can be changed statically or dynamically.
- ✓ Reconfigurable Processor can entirely change their functionality to adapt to new requirements.
- ✓ Field Programmable System level IC (FPSLIC) from Atmel is an example for RSoC.
- ✓ RSoC use less silicon and consume low power.
- ✓ The system is configured from a library of pre- compiled IP cores stored in FLASH memory when required.

Embedded OS trends:

- Embedded OS industry is also undergoing revolutionary changes to take advantage of the potential offered by the trends in processor technologies.
- More options to select from a bunch of ***commercial*** and ***open source embedded OSs*** for building embedded devices.

- Microkernel architecture in place of the monolithic architecture
- This enables only essential services to be contained in the kernel and the rest are installed as services in the user space as is done in Mobile phone.

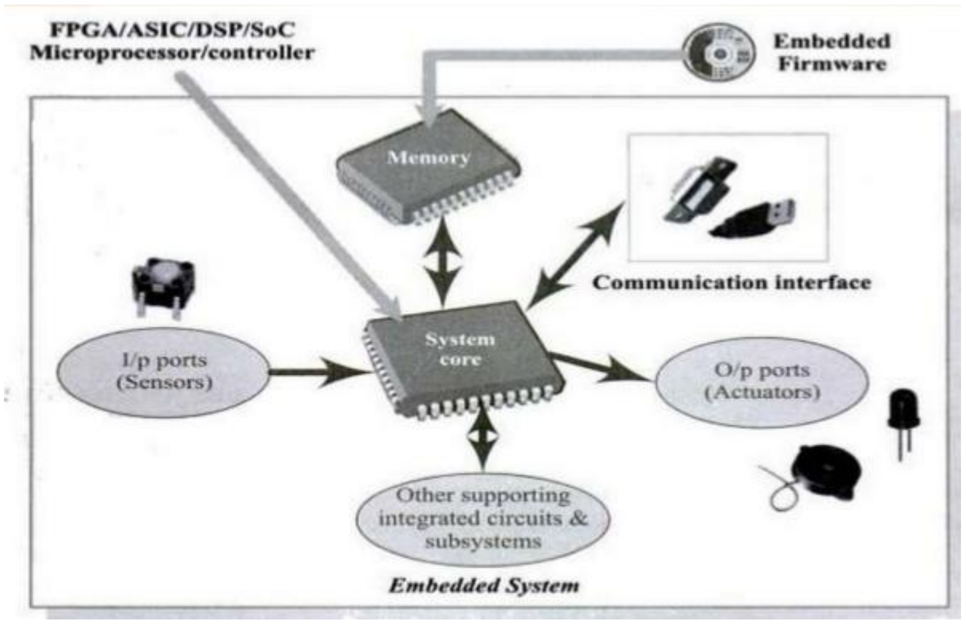
- Another noticeable trend adopting by OS suppliers is the 'Device oriented'
- Today OS suppliers are providing off-the-shelf OS customized for the device.
- The *Windows Mobile OS* from Microsoft is specially designed for mobile handsets

- Embedded Linux is Open source embedded OS
- The Ubuntu MID edition is specially designed for Mobile Internet Device (MID)

- VxWorks RTOS for multicore processors.
- The advancements in processor technology have caused a major change in the Embedded Operating System Industry.

UNIT-2
ELEMENTS OF EMBEDDED SYSTEMS

2.1. CORE OF THE EMBEDDED SYSTEM



The Embedded systems are domain and application specific and are built around a central core. The core of the embedded system falls into any one of the following categories:

1. General Purpose and Domain Specific Processors
   1.1 Microprocessors
   1.2 Microcontrollers
   1.3 Digital Signal Processors
2. Application Specific Integrated Circuits (ASICs)
3. Programmable Logic Devices (PLDs)
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.
- Most of the ES in industrial control and monitoring applications make use of the commonly available microprocessors and microcontrollers whereas domains which require signal processing such as speech coding, speech recognition..etc make use of special kind of digital signal processors supplied by manufacturers like Analog Devices, Texas Instruments,..etc

- A General Purpose Processor (GSP) is a processor designed for general computational tasks. The processor running inside laptop or desktop (Pentium 4/AMD Athlon, etc.) is a typical example for general purpose processor.
- Application Specific Instruction Set Processors (ASIPs) are processors with architecture and instruction set optimised to specific-domain/application requirements like network processing, automotive, telecom, media applications, digital signal processing, control applications, etc. (Ex: Automotive AVR, USB AVR from ATMEL, digital signal processors)

1.1 Microprocessors

- ✓ A microprocessor is a silicon chip representing a central processing unit (CPU), which is capable of performing arithmetic and logic operations according to a pre-defined set of instructions.
- ✓ In general, the CPU contains ALU, Control Unit and working registers.
- ✓ A microprocessor is a dependent unit and it requires the combination of other hardware like memory, timers, *interrupt* controller and I/O ports, etc. for proper functioning.
- ✓ Microprocessors are used in general purpose applications.

Ex: Intel 8085, Motorola 6800, Zilog Z80.

1.2 Microcontrollers

- ✓ A microcontroller is an integrated chip that contains CPU, special and general purpose registers, data memory (RAM), program memory (ROM/Flash), Timers, Interrupt control unit and dedicated I/O ports.
- ✓ Since a microcontroller contains all the necessary functional blocks for independent working, they found greater place in embedded domain.
- ✓ Apart from this, they are cost effective and are readily available in the market.

Ex: Intel 8051, AT89C51, PIC from Microchip, MSP430 from Texas Instruments

1.3 Digital Signal Processors

- ✓ DSPs are powerful special purpose 8/16/32 bit processors, designed specifically to meet the computational demands and power constraints of today's embedded audio, video, signal processing and communication applications.
- ✓ 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.
- ✓ DSP performs large amount of real-time calculations like FFT(Fast Fourier Transform), DFT(Discrete Fourier Transform), Convolution, SOP(Sum of Products) calculation etc.

- ✓ A typical digital signal processor incorporates the following key units:
  Program Memory : for storing the program required by DSP to process the data
  Data Memory : for storing temporary variables and data/signal to be processed.
  Computational Engine: Performs the signal processing in accordance with the stored program memory. It incorporates many specialised arithmetic units and each of them operates simultaneously to increase the execution speed. It also incorporates multiple hardware shifters for shifting operands and thereby saves execution time.
  I/O 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 DSPs are employed.

  Ex: Blackfin processor from Analog Devices, TMS320 from Texas Instruments

2. Application Specific Integrated Circuits (ASICs) Application Specific Standard Product (ASSP)

- ✓ Application Specific Integrated Circuit (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.
- ✓ ASIC consumes a very small area in the total system and thereby helps in 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.
  .
  Ex: ADE7760 Energy Metre ASIC developed by Analog Devices for Energy metering applications is a typical example for ASSP

3. Programmable Logic Devices

- ✓ Logic devices provide specific functions, including device-to-device interfacing, data communication, signal processing, data display, and timing and control applications.
- ✓ Logic devices can be classified into two broad categories – fixed and programmable As name implies, fixed logic devices are permanent, they perform one function or set of functions - once manufactured, they can't be changes. On the other hand, PLDs offer customers a wide range of logic capacity, features, speed, and voltage characteristics – and these devices can be reconfigured to perform any number of functions.

- ✓ The PLDs consists of a large no. of programmable gates on a VLSI chip
- ✓ With programmable logic devices like **CPLDs and FPGAs**, the designs can be quickly simulated, tested and programmed into devices and immediately tested in live circuits. During the design phase, the circuit can be changed by programming the device to get the desired outputs, because these devices are based on re-writable memory technology.
- ✓ These devices are used in Network routers, DSL modem, an automotive navigation system

  Ex: Xilinx Cool Runner, Xilinx Vertex, Spartan

4. Commercial Off-the-Shelf Components (COTS)

- ✓ A Commercial Off-the-Shelf(COTS) product is one which is used 'as-is
- ✓ COTS products are designed in such a way to provide easy integration and interoperability with existing system components.
- ✓ The COTS component itself may be developed around a general purpose or domain specific processor of ASIC or PLD.
- ✓ Typical examples of COTS hardware unit are remote controlled toy car control units including the RF circuitry part, high performance, high frequency microwave electronics (2-200 GHz), high bandwidth analog-to-digital converters.
- ✓ The major advantage of using COTS is that they are readily available in the market, are cheap and a developer can cut down development time to a great extent. This in turn reduces the time to market your embedded systems.

  Ex: TCP/IP plug-in module from WIZnet

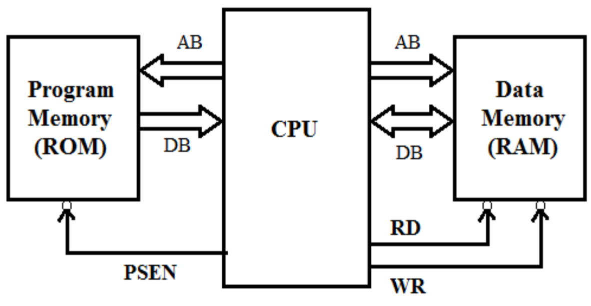
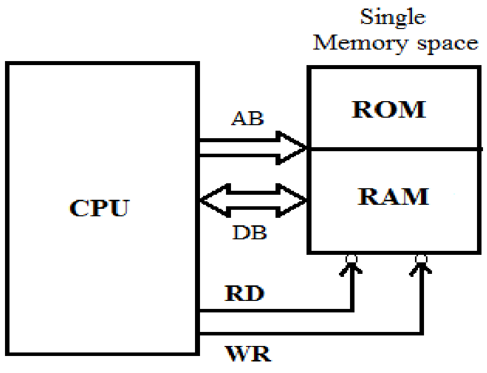
Microprocessors Vs Microcontrollers

	Microprocessor	Micro controller
1	A silicon chip representing a central processing unit (CPU), which is capable of performing arithmetic as well as logical operations according to a predefined set of instructions	A microcontroller is an integrated chip that contains CPU, special and general purpose registers, data memory (RAM), program memory (ROM/Flash), Timers, Interrupt control unit and dedicated I/O ports.
2	It is a dependent unit, which requires the combination of other chips like timers, program and data memory chips, interrupt controllers, etc. for functioning.	It is a self-contained Unit and it doesn't require external interrupt controller, timer, UART, etc for its functioning
3	General purpose in design and operation.	Application-oriented or domain-specific
4	Doesn't contain a built in I/O ports. The I/O port functionality needs to be implemented with the help of external programmable peripheral interface chips like 8255	Contain multiple built-in I/O ports which can be operated as a single 8 or 16 or 32bit port or as individual port pins.
5	Targeted for high end market where performance is important	Targeted for embedded market where performance, power, size, design and cost are the Key deciding factors
6	Limited power saving options compared to microcontrollers	Includes lot of power saving features

RISC Vs CISC

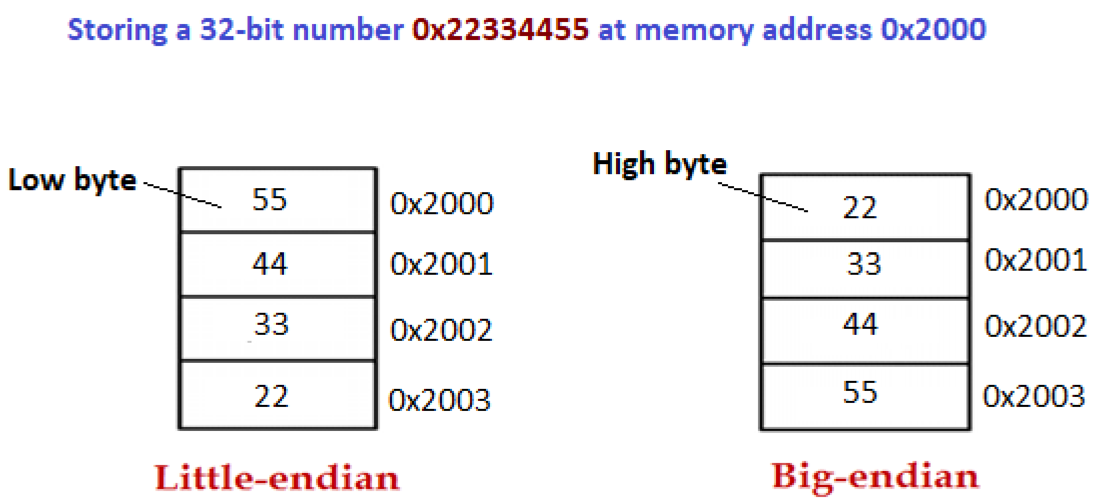
	RISC (Reduced Instruction Set Computer)	CISC (Complex Instruction Set Computer)
1	Supports lesser number of instructions.	Supports greater number of instructions.
2	Fixed length instructions	Variable length instructions
3	Instructions take fixed amounts of time for execution	Instructions take varying amounts of time for execution
4	Orthogonal instruction set	Non-orthogonal Instruction set
5	**Instruction pipelining works effectively and increases the execution speed.**	**Instruction pipelining concept is not effectively works as different sizes and different execution times of instructions.**
6	Less silicon usage and decoding logic is simple due to simple instructions	More silicon usage and decoding logic is complex due to complex instructions decoding
7	Design of compiler is easy due to lesser number of instructions	Design of compiler is complex due to larger number of different and complex instructions
8	A larger number of registers are available.	Limited no. of general purpose registers
9	**Large code sizes** (Programmer needs to write more code to execute a task since instructions are simpler ones)	**Small code sizes** (The programmer can achieve the desired functionality with a single instruction)

Harvard Architecture Vs Von-neumann Architecture

Harvard architecture	Von-Neumann architecture
 Program Memory (ROM) AB DB CPU AB DB Data Memory (RAM) RD WR PSEN Separate buses for instruction and data fetching.	 Single Memory space CPU AB DB ROM RAM RD WR Common bus for instruction and data fetching.
Separate control signals are used to access program memory and data memory.	Common control signals are used for accessing both program memory and data memory.
It allows simultaneous access to the program and data memories.	First fetches the instruction and then fetches the data. The two separate fetches slows down the processor's operation
Easier to pipeline, so high performance can be achieve.	Low performance as compared to Harvard architecture
No chances for accidental corruption of program memory, since data memory and program memory are physically separated.	Accidental corruption of program memory may occur, if data memory and program memory are stored physically in the same chip.
No memory alignment problems	Allows self-modifying codes
Comparatively high cost	Low cost

Little-endian Vs Big-endian processors:

Endianness specifies the order in which the data is stored in the memory by the processor operation in a multi byte system.



Little-endian operation: Low byte of data is stored at lower memory address and high byte of data is stored at higher memory address

Big-endian operation: High byte of data is stored at lower memory address and low byte of data is stored at higher memory address

2.2. MEMORY TYPES

- ✓ A Memory module is a physical device which is used to store programs and data on a temporary (or) permanent basis for use in digital electronics.
- ✓ An efficient memory increases the performance of embedded systems.
- ✓ Various forms of memories used in embedded system are given below:

ROM (or) EPROM (or) Flash → Storing Codes for RTOS, Codes for each application Program, ISR and tasks, Codes for boot program, etc.

RAM (Internal and external) → Storing the variables during program run and storing Stack Storing input output buffer data

Memory stick → A flash memory stick is inserted in mobile computing system (or) digital Camera, which is used to store HD video, images, music, speech..etc

EEPROM (or) Flash → Storing Nonvolatile results of Processing

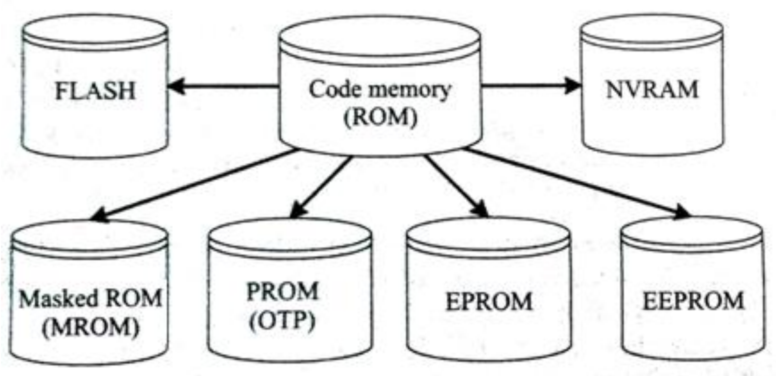
Cache → Storing copies of instructions and data in advance from external primary memory and storing results temporarily during processing.

- ✓ The different types of memory modules used in embedded systems are

  Program Memory /Read Only Memory: MROM, PROM, EPROM, EEPROM, Flash

  Data memory /Random Access Memory: SRAM, DRAM, NVRAM

(A) Program Memory / Read Only Memory (ROM)



- The Program memory /ROM are used to store the **firmware** in embedded systems.
- ROM can only be used to read from, but cannot be written upon.
- These memory devices are **NON-VOLATILE**, which retains its contents even if power is switched off.
- ROM is used for Storing Codes for RTOS, Codes for boot program, Codes for each application Program, constant data, ISR and tasks, etc.

Types of ROM :

(i) Masked ROM (MROM) :

- Masked ROM makes use of the hardwired technology for storing data.
- It is **factory programmed** by masking and metallization process at the time of production itself, according to the data provided by the end user.
- The advantage of this is low cost for high volume production.
- The limitation with MROM is the **inability to modify the device firmware** against firmware updates.

(ii) Programmable Read Only Memory (PROM):

- The Programmable ROM can be modified only once by the user and hence it is called as One Time Programmable (OTP) memory.
- The PROM is manufactured with series of fuses. The chip is programmed by the programmer wherein some fuses are burnt. The open fuses are read as logic "1", while the burned fuses are read as logic "0".
- The limitation of PROM → **Not useful for development purpose**, as the code is subject to continuous change during the development phase.

(iii) Erasable Programmable Read Only Memory (EPROM) :

- The Erasable Programmable ROM is one of the special types of memory modules that can be programmed any number of times to correct the errors. It can retain its contents until exposed to ultraviolet light.
- The ultraviolet light erases its contents and making it possible to re-program the memory. To write and erase the EPROM memory chip, we need a special device called EPROM programmer.
- Even though the EPROM chip is flexible in terms of re-programmability, it needs to be taken out of the circuit board and put in a UV eraser device for 20 to 30 min. So it is a tedious and time consuming process.

(iv) Electrical Erasable Programmable ROM (EEPROM) :

- As the name indicates, the information contained in EEPROM can be altered by using electrical signals at the register/byte level.
- The EEPROM data is stored and removed one byte of data at a time.
- These are in-circuit programmable, i.e., EEPROM does not need to be removed from the computer to be modified and does not require the additional equipment.
- It provides greater flexibility for system design. The only limitation is their capacity is limited when compared to standard ROM.
- These memory devices are used in computers and other electronic devices to store small amount of data that must be saved when the power supply is removed.

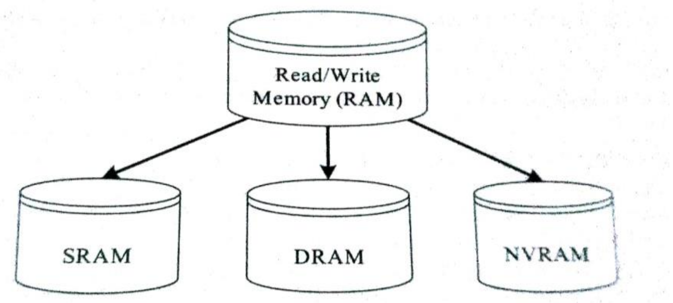
(v) Flash memory:

- FLASH memory is the latest and most popular ROM technology used in today's embedded designs. It is a variation of EEPROM technology. It combines the re-programmability of EEPROM and the high capacity of standard ROMs.
- Flash memory is organized as sectors/blocks/pages. The erasing of memory can be done at sector level/page level without affecting the other sectors (or) pages.
- The flash memory keeps its data even with no power at all.

(B) Data Memory / Random Access Memory (RAM)

- RAM is the data memory (or) working memory of the controller / processor.
- RAM is a Read-Write memory, i.e., the processor can read from it and write to it.
- These memory devices are VOLATILE, which hold their content till power is applied to them. When power is switched off, these memories lose their content.
- The RAM allows information to be stored and accessed quickly from any desired random location directly.

Types of RAM:



SRAM (Static RAM)

- Static RAM stores data in the form of Voltage and they are made up of flip-flops.
- SRAM cell is realized using 6- MOSFETs. (4-for building latch and 2-for control)
- The SRAM does not need to be refreshed periodically.
- Consumes less power
- The SRAM provides faster access to the data
- SRAM is complex due to the usage of a large number of transistors
- Low storage capacity (Less dense)
- It is more expensive than DRAM.

DRAM (Dynamic RAM)

- DRAM stores data in the form of charge
- DAM cell is realized using a MOSFET and a Capacitor
- The DRAM need to be refreshed periodically to maintain the charge in the capacitor
- Consumes more power
- Slow in operations due to refresh requirements
- DRAM is simple to design and implement
- High storage capacity
- It is a low cost device.

Non-Volatile RAM:

- Non-volatile RAM is a random access memory with battery backup.
- It contains static RAM based memory and a minute battery for providing supply to the memory in the absence of external power supply.
- The memory and battery are packed together in a simple package.
- NVRAM is used for non-volatile storage of results/operations/ flags etc.
- The life time of NVRAM is expected to be around 10 years.
- DS1744 from Maxim/Dallas is an example for 32KB NVRAM

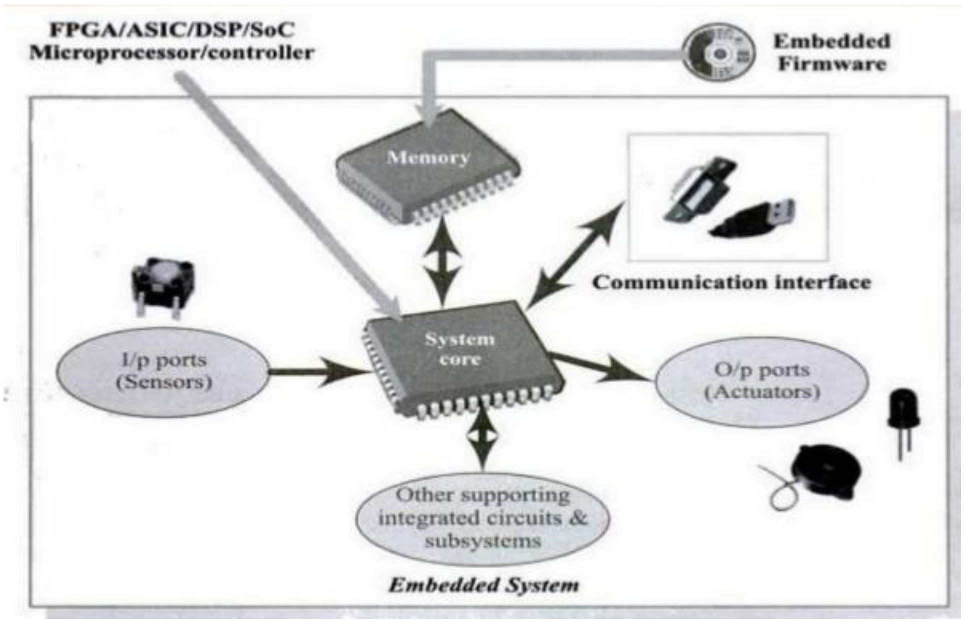
2.3. SENSORS AND ACTUATORS

Sensors are used to sense/detect the changes in the input variables and convert into electrical signals for any measurements. The sensors are connected to the input port.

Ex : Temperature Sensors, Position Sensors, Pressure Sensors
Force Sensors, Vibration Sensors, Piezo Sensors, Humidity Sensors

Actuators are used to converts electrical signals into corresponding physical action.
The actuators are connected at the output port

Ex: Electric motors, DC servo motors, Solenoids,
Hard drive stepper motors, Comb drives.



- Embedded systems are basically designed to regulate a physical variable (or) to manipulate the state of some devices by sending some signals to the actuators (or) devices connected to the output ports of the system.

- An embedded system is in constant interaction with the real world. The changes in variables are detected by the sensors, which are connected to the input port of the embedded system.

- If 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. Ex: Air conditioner

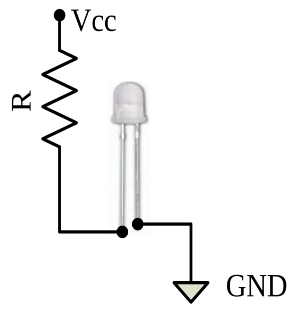
- If embedded system is designed for monitoring purpose only, then there is no need for including an actuator in the system. Ex: ECG machine which is used to monitor the heartbeat and its order. The variations are captured and presented to the doctor through visual display.

I/O subsystem

The I/O subsystem of the embedded system facilitates the interaction of the embedded system with the external world. 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.

Some of the sensors and actuators used in embedded systems and the I/O systems to facilitate the interaction of embedded systems with external world are given below:

(1) Light Emitting Diode



- ✓ Light Emitting Diode (LED) is an output device for visual indication in any embedded system.
- ✓ LED can be used as an indicator for the status of various signals or situations.
- ✓ Typical examples are indicating the presence of power conditions like 'Device ON', 'Battery low' or 'Charging of battery' for a battery operated handheld embedded devices.

- ✓ 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.
- ✓ LEDs can be interfaced to the port pin of the processor/controller in two ways.
  In the first method, Anode is connected to the port pin and Cathode is connected to Ground. In this connection, the port pin **sources** current to the LED when it is at Logic 1.
  In the second method, Cathode is connected to the port pin and Anode is connected to the supply voltage through current limiting resistor. Here the port pin **sinks** current. The second approach is mostly used method because the current is sourced by the power supply and port pin act as the **sink** for current.

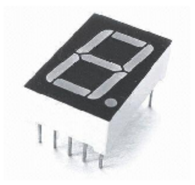
(2) 7-Segment LED Display

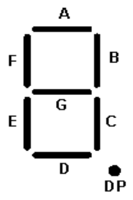
- ✓ The 7 – segment LED display is an output device for displaying numeric values.

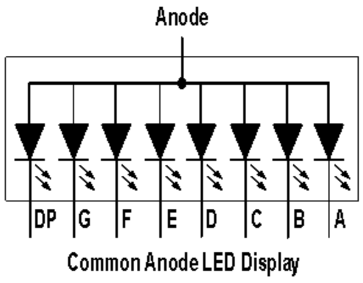
- ✓ It contains 8 light-emitting diode (LED) segments arranged in a special form. Out of the 8 LED segments, 7 are used for displaying alpha numeric characters and 1 is used for representing 'decimal point'. The LED segments are named A to G and the decimal point LED segment is named as DP.

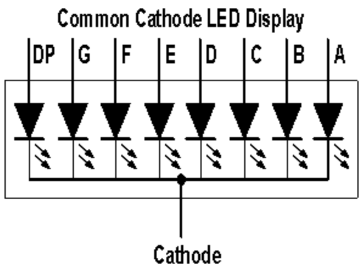
- ✓ The 7 – segment LED displays are available in two different configurations, namely; Common anode and Common cathode. In the Common anode configuration, the anodes of the 8 segments are connected commonly whereas in the Common cathode configuration, the 8 LED segments share a common cathode line.

- ✓ The current through each segment can be limited by connecting a current limiting resistor to the anode or cathode of each segment. he typical value for the current falls within the range of 20mA
- ✓ 7-segment LED display is a popular choice for low cost embedded systems like Public telephone monitoring, Temperature display units.



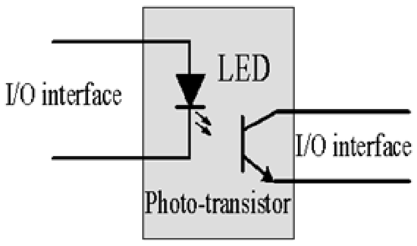




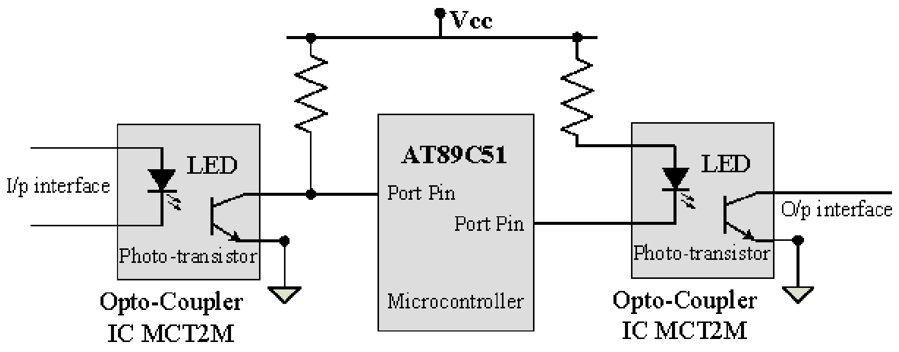


(3) Optocoupler

- ✓ 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)
- ✓ In electronic circuits, optocoupler is used for suppressing interference in data communication, circuit isolation, high voltage separation, etc.
- ✓ Optocouplers can be used in either input circuits or output circuits as shown in fig.



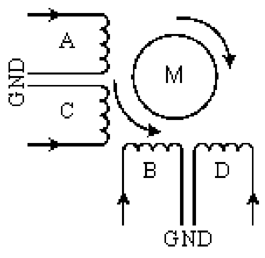
An optocoupler device



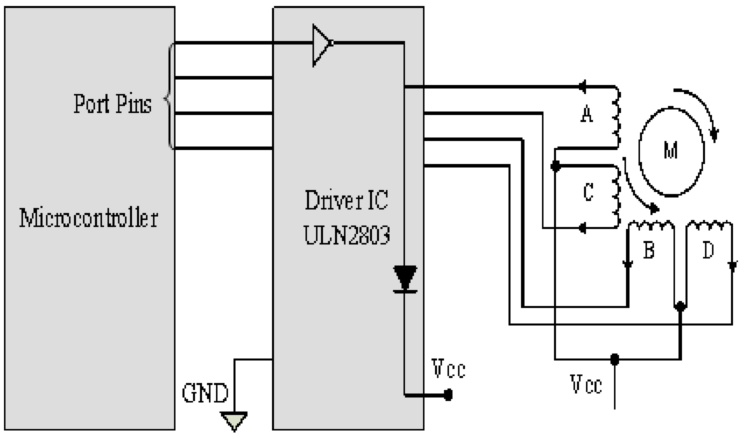
Optocoupler connected at input ant output ports

(4) Stepper Motor

- ✓ Stepper motor is an electro mechanical device which generates discrete displacement (motion) in response to dc electrical signals
- ✓ Stepper motors are widely used in industrial embedded applications, consumer electronic products and robotics control systems
- ✓ The paper feed mechanism of a printer/fax machine makes use of stepper motors for its functioning.



2-phase unipolar stepper motor



Interfacing of stepper motor through driver circuit

- ✓ Based on the coil winding arrangements, a two phase stepper motor is classified into two types – Unipolar and 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.

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.

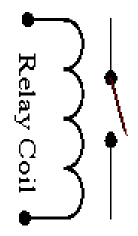
- ✓ Depending on the current and voltage requirements, special driving circuits are required to interface the stepper motor with microcontroller/processors. Stepper motor driving ICs like ULN2803 or simple transistor based driving circuit can be used for interfacing stepper motors with processor/controller.

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

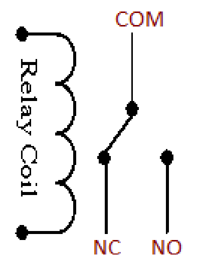
(5) Relay

- ✓ 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 a metal armature with one or more contacts.

- ✓ 'Relay' works on electromagnetic principle. When a voltage is applied to the relay coil, current flows through the coil, which in turn generates a magnetic field. The magnetic field attracts the armature core and moves the contact point. The movement of the contact point changes the power/signal flow path.

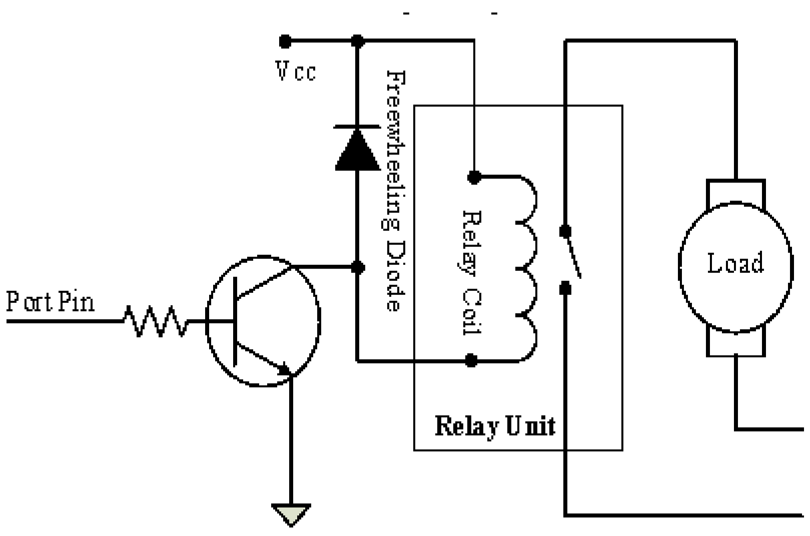


Single Pole Single Throw



Single Pole Double Throw

- ✓ 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.

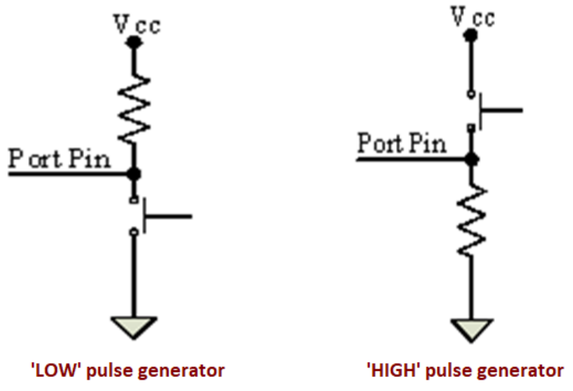


Transistor based Relay driving circuit

(6) Push button switch

- ✓ Push Button switch is an input device
- ✓ Push button switch comes in two configurations - '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' or 'open' state as long as it is kept in the pushed state and it breaks/makes the circuit connection when it is released.



Push button switch configurations

- ✓ Push button is used for generating a momentary pulse.
- ✓ In embedded systems, push button is used as Reset and Start switch pulse generator
- ✓ The push button is normally connected to the input port pin of the microcontroller. Depending on the way in which the push button is interfaced to microcontroller, it can generate either 'HIGH' pulse or a 'LOW' pulse.

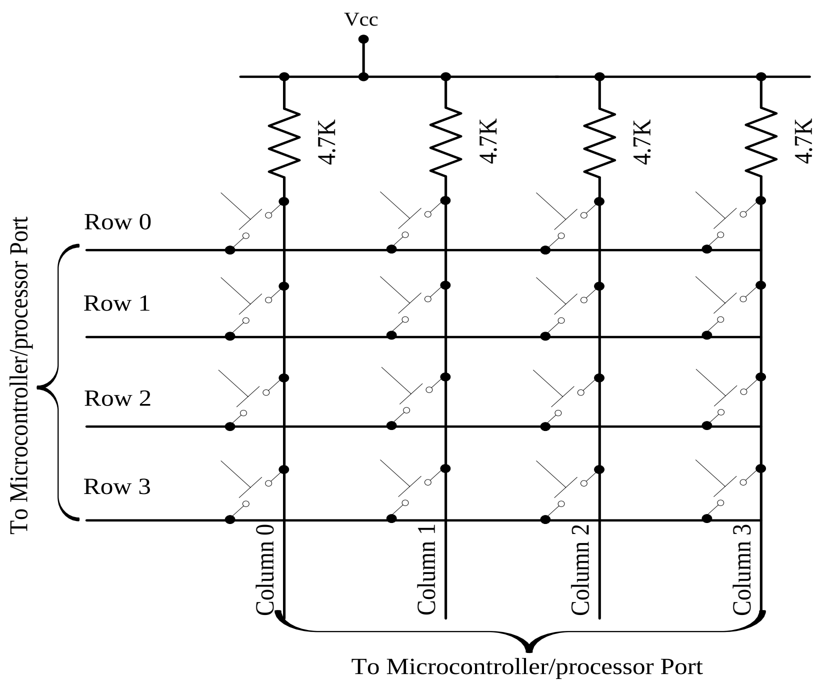
(7) Piezo Buzzer:

- ✓ Piezo buzzer is a piezoelectric device for generating audio indications in embedded application. It contains a piezoelectric diaphragm which produces audible sound in response to the voltage applied to it.
- ✓ Piezoelectric buzzers are available in two types. 'Self-driving' and 'External driving'. The 'Self-driving' circuit contains all the necessary components to generate sound at a predefined tone.
- ✓ 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.
- ✓ A Piezo buzzer can be directly interfaced to the port pin of the processor/controller. Depending on the driving current requirements, the Piezo buzzer can also be interfaced using a transistor based driver circuit as in the case of a 'Relay'.

(8) Keyboard

- ✓ Keyboard is an input device for user interfacing.
- ✓ If the number of keys required is very limited, push button switches can be used and they can be directly interfaced to the port pins for reading.
- ✓ Matrix keyboard is an optimum solution for handling large number of key requirements. It greatly reduces the number of interface connections. For example, for interfacing 16 keys, in the direct interfacing technique 16 port pins are required, where as in the matrix keyboard only 4 columns and 4 rows are required for interfacing 16 keys.

- ✓ In matrix keyboard, keys are arranged in matrix fashion, i.e., they are connected in a row and column style as shown in figure.
- ✓ The key press in matrix keyboard is identified with row-column scanning technique, in which each row of the matrix is pulled 'LOW' and columns are read. This process is repeated until the scanning for all rows are completed. The key press is identified by identifying the row and column numbers.



2.4. COMMUNICATION INTERFACES

- ✓ Communication interface is essential for communicating with various subsystems of the embedded system and with the external world.
- ✓ The need for communication interfaces are
    - to interact with another system for sharing data
    - to share data with other embedded systems which are connected to a network
    - to communicate with host system/PC/work station

✓ Examples: Reading certain memory ICs like EEPROM, SD cards
Accessing DACs and ADCs,
Reading hardware sensors
RTC (Real Time Clock) interfacing
Displays like LCDs
Communicating with multiple microcontroller

✓ Two types of communication Interfaces
Device/board level communication interface (Onboard Communication Interface)
Product level communication interface (External Communication Inter face)

Onboard Communication Interfaces:

The communication channels/buses which interconnects the various components within the embedded system is referred as "Onboard communication interface"

1. Inter Integrated circuit (I2C)
2. Serial Peripheral Interface (SPI)
3. Universal Asynchronous Receiver Transmitter (UART)
4. 1-Wire Interface
5. Parallel bus interface

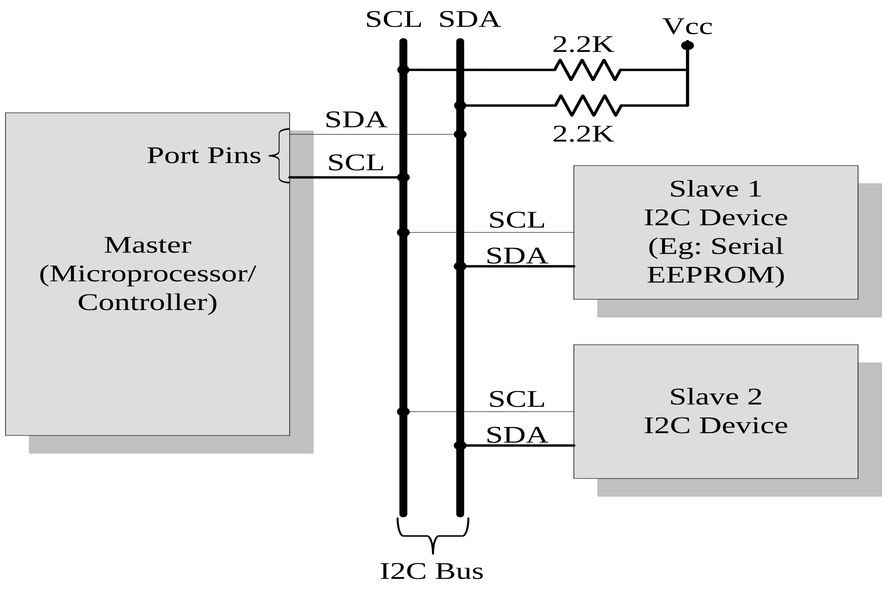
(1) Inter Integrated Circuit:

The concept of I2C bus was developed by 'Philips Semiconductors'

- ✓ It is a **Synchronous Half duplex** Serial communication bus
- ✓ **I^{2}C bus uses only two bidirectional lines**
  SDA - Serial data
  SCL - Serial clock
- ✓ It is often called the *two-wire interface***.**

- ✓ I2C bus is a shared bus system to which many number of I2C devices can be connected. Devices connected to the I2C bus can act as either 'Master' device or 'Slave' device. Transfers on the bus take place between a master and a slave.

- ✓ The 'Master' device is responsible for
  - Generating the Clock signal
  - Selection of salve device
  - Controlling the communication by initiating/terminating data
- ✓ 'Master' and 'Slave' devices can act as either transmitter (or) receiver
- ✓ Regardless whether a master is acting as transmitter (or) receiver, the clock signal is generated by the 'Master' device only
- ✓ Each slave has a unique address, which is usually 7 bits long.
- ✓ Slave' devices wait for the commands from the master and respond upon receiving the commands


SCL SDA
Vcc
2.2K
2.2K
SDA
SCL
Port Pins
Master (Microprocessor/ Controller)
SCL
SDA
Slave 1 I2C Device (Eg: Serial EEPROM)
SCL
SDA
Slave 2 I2C Device
I2C Bus

The master starts the transfer, provides the clock, addresses a particular slave, manages the transfer, and finally terminates it. The sequence of operations for communicating with an I2C slave device are listed below:

- The master device sends the 'Start' condition on SDA line
- The master sends slave address (7-bit or 10-bit) to select the slave device
- The master sends a Read or Write control signal (1-for Read, 0-for Write)
- The selected slave device responds to Master by sending ACK signal on SDA line
- For next clock pulses, data transfer takes place between Master and Slave devices
- The transmitter sends the ACK to the receiver for every byte transfer.
- Finally, the master terminates the data transfer by sending 'Stop' condition

The I2C bus supports three different data rates:

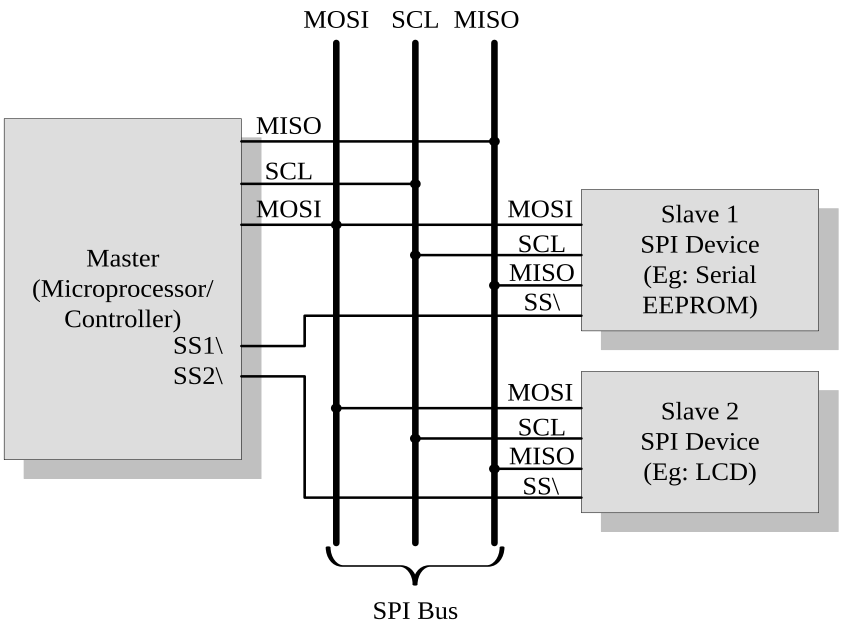
Standard mode →	up to 100 Kbits/Sec
Fast mode →	up to 400 Kbits/Sec
High speed mode →	up to 3.4 Mbits/Sec

(2) Serial Peripheral Interface (SPI):

The serial peripheral interface (SPI) was introduced by Motorola

- ✓ It is a **Synchronous Full duplex** Serial communication bus
- ✓ **SPI bus uses 4- signal lines for communication**

MOSI – Master Out Slave In	- carries data from Master to Slave
MISO – Master in slave out	- carries data from Slave to Master
SCLK – Serial Clock	- carries clock signal from Master to Slave
SS – *Slave Select signal*	- used to select the Slave device



MOSI
SCL
MISO
MISO
SCL
MOSI
Master (Microprocessor/ Controller)
SS1\
SS2\
MOSI
SCL
MISO
SS\
Slave 1 SPI Device (Eg: Serial EEPROM)
MOSI
SCL
MISO
SS\
Slave 2 SPI Device (Eg: LCD)
SPI Bus

- SPI is a single master multi-slave system. The master device is responsible for generating the Clock signal and selecting the slave device.
- SPI works on the principle of 'Shift Registers'. The master and slave devices contain a special shift register for the data transmit and receive.
- For every clock pulse, the master sends a data bit to the slave through MOSI pin and at the same time the salve will send a data bit to the master through MISO pin. In summary, the shift registers of master and slave form a circular buffer (loop).

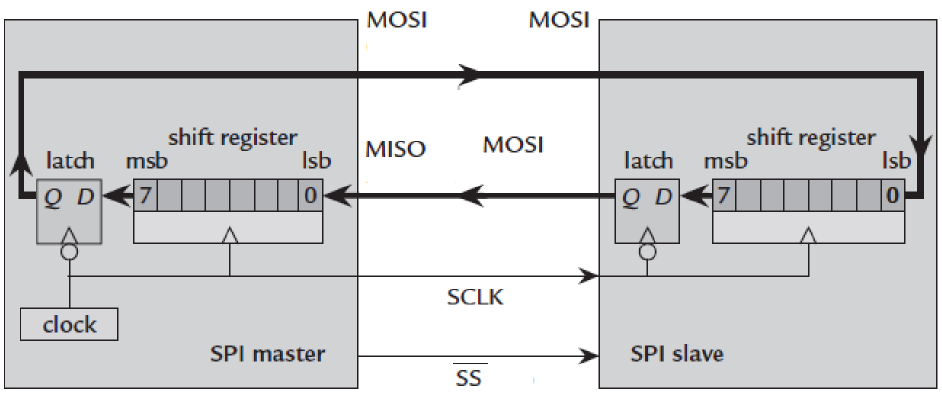


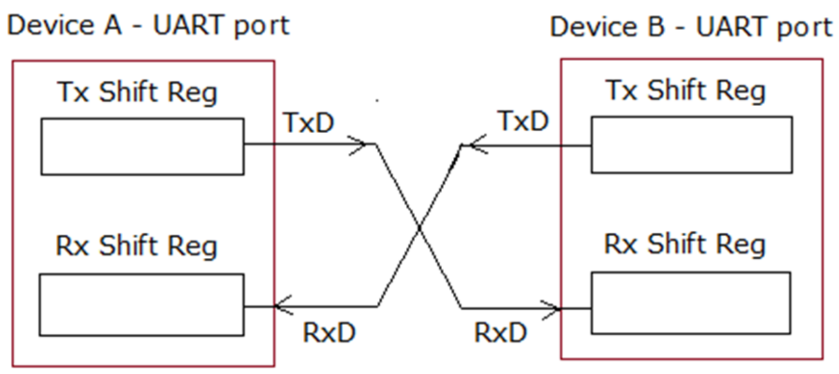
Figure : Serial peripheral interface between a master and a single slave

(3) Universal Asynchronous Receiver Transmitter (UART):

- ✓ **Asynchronous Full duplex** Serial communication protocol
- ✓ No CLK signal is transmitted along with data
- ✓ UART requires 2- signal lines for communication

  TxD - Transmit data

  RxD – Receive data



- ✓ UART based communication is an asynchronous form of serial data transmission, in which transmitter and receiver are operated with different clock signals. It relies upon the pre-defined agreement between the transmitter and receiver.
- ✓ The serial communication settings (Baud rate, character length, parity, no. of start bits and stop bits) for both transmitter and receiver should be set as identical.
- ✓ 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. The 'Start' bit informs the receiver that a data byte is about to arrive. If parity is enabled for communication, the UART of the transmitting device adds a parity bit. The UART of the receiving device calculates the parity of the bits received and compares it with the received parity bit for error checking.
- ✓ Detects 3- types of errors: Framing Error, Parity error and Overrun error

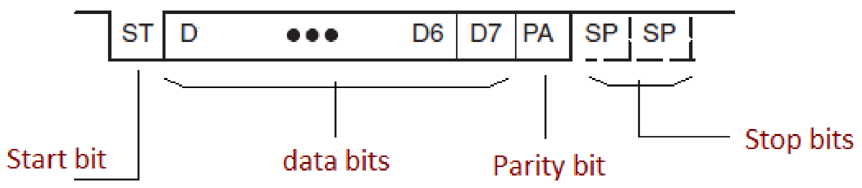
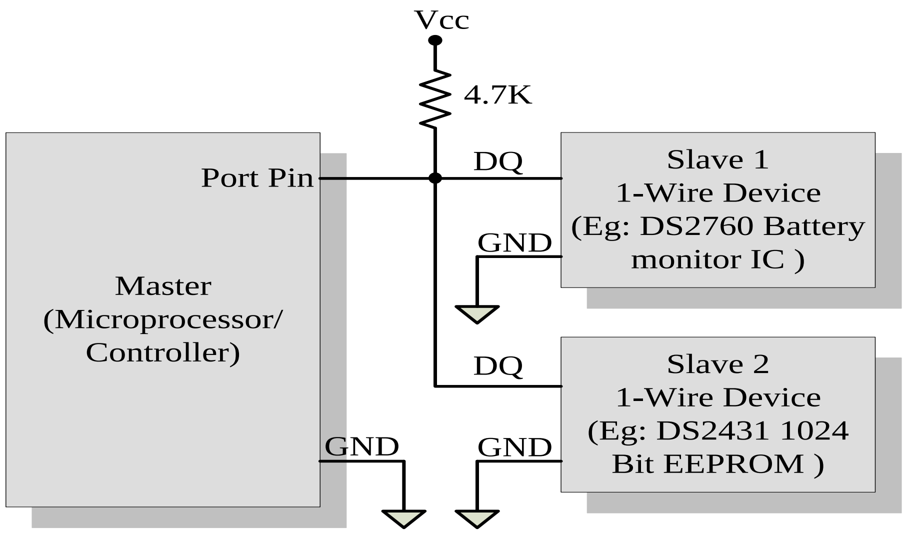


Fig: UART character format

The UART character format consists of a start bit, 7 or 8 data bits, parity bit (optional) and one or two stop bits, as shown in above figure.

(4) 1-Wire Interface (Dallas 1-Wire protocol)

- ✓ **Asynchronous Half duplex** serial communication protocol
- ✓ It makes use of only a signal line (DQ) for communication
- ✓ It allows power to be sent along with the signal wire as well.
- ✓ It follows master-slave communication model.
- ✓ It supports a single master and one (or) more slave devices on the bus



Every 1-wire device contains a globally unique 64-bit identification number stored within it. This unique identification number can be used for addressing individual devices present on the bus, if the multiple slave devices are connected to the 1-wire bus.

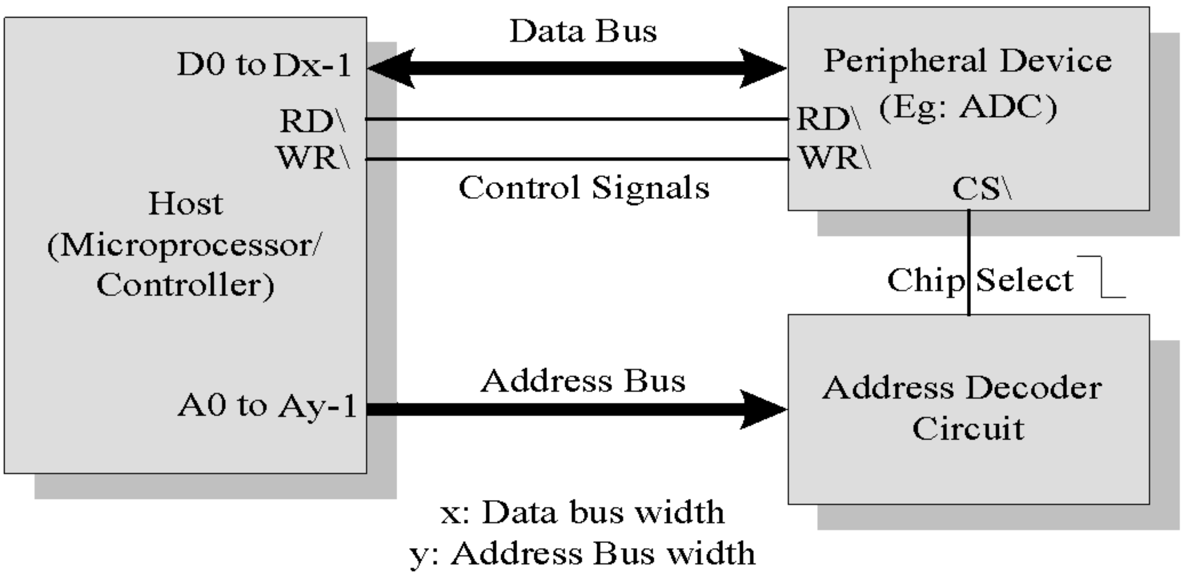
The sequence of operation for communicating with a 1-Wire slave device is:

1. Master device sends a 'Reset' pulse on the 1-Wire bus.
2. Slave device(s) present on the bus respond with a 'Presence' pulse.
3. Master device sends a ROM Command, which addresses the slave device to which it wants to initiate a communication
4. Master device sends a Read/Write function command
5. Master initiates data transfer from/to the slave

All communication over the 1-wire bus is master initiated. The communication over the 1-wirebus is divided into timeslots of 60 µs. For starting a communication, the master asserts the Reset pulse by pulling the 1-wire bus 'LOW' for at least 8-time slots (480µs). If a 'slave' device is present on the bus and is ready for communication, it should respond to the master within 60µs of the release of the 'Reset' pulse by the master. The slave device responds with a 'Presence' pulse by pulling the 1-wire bus 'LOW' for a minimum of 1 time slot (60µs).
For writing a bit value of 1 on the 1-wire bus, the master pulls the bus for 1 to 15µs 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 min. of 1 time slot (60µs) and a max. of 2 time slots (120µs). To Read a bit from the slave device, the master pulls the bus 'LOW' for 1 to 15µs. If the slave wants to send a bit value 'l' 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 of the time slot.

(5) Parallel bus Interface

- ✓ 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 host and the device. The 'Control Signals' for communication includes 'Read/Write' signal and device select signal.
- ✓ The device normally contains a device select line (Chip select) 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



- ✓ The parallel communication is initiated by the host processor.
- ✓ Strict timing characteristics are followed for parallel communication.
- ✓ If a device wants to initiate the communication, it can inform the same to the processor through interrupts. For this, interrupt line of the device is connected to the interrupt line of the processor and corresponding interrupt is enabled in the host system.
- ✓ 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.
- ✓ Parallel data communication offers the highest speed for data transfer.

External Communication Interfaces:

- ✓ The product level (or) external communication interface refers to the different communication channels/buses used by the embedded system to communicate with external world. It is responsible for data transfer between the embedded system and external devices or modules.
- ✓ The external communication interface can be wired or wireless and it can be serial or parallel interface.
    1. RS-232C and RS-485
    2. Universal Serial Bus (USB)
    3. Infrared (IrDA)
    4. Bluetooth (BT)
    5. Wi-Fi
    6. ZigBee

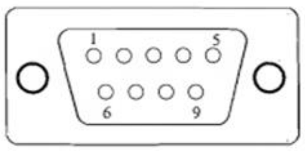
(1) RS-232C and RS-485

- ✓ RS-232 C interface is developed by Electronic Industry Association (EIA)
- ✓ It is full duplex, wired, asynchronous serial communication interface
- ✓ RS-232 extends the UART communication signals for external data communication.
- ✓ The RS-232 defines various handshaking and control signals for communication, apart from TxD and RxD signal lines for data communication.

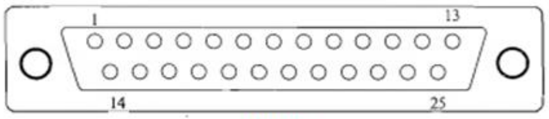
- ✓ RS-232 is a point-to-point communication interface and the devices involved in RS-232 communication are called as 'Data Terminal Equipment (DTE)' and 'Data Communication Equipment (DCE)'.

- ✓ RS-232 supports 2- different types of connectors namely DB-9 and DB-25.

Pin No. (DB9)	Pin Name	Description
1	DCD	Data Carrier Detect
2	RxD	Receive pin for receiving serial data
3	TxD	Transmit pin for transmitting serial data
4	DTR	Data Terminal Ready
5	GND	Signal Ground
6	DSR	Data Set Ready
7	RTS	Request to Send
8	CTS	Clear to Send
9	RI	Ring Indicator



DB-9



DB-25

If hardware data flow control is required for serial transmission, various control signal lines of the RS-232 connection are used appropriately. The control signals are implemented mainly for MODEM communication and some of them may not be relevant for other type of devices.

The Data Terminal Ready (DTR) signal is activated by DTE when it is ready to be connected. The Data Set Ready (DSR) is activated by DCE when it is ready for establishing a communication link.

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 Carrier Detect (DCD) control signal is used by the DCE to indicate the DTE that a connection has been established with a remote device.

Ring Indicator (RI) is a modem specific signal used by DCE to indicate the DTE when a ring signal (incoming call) is detected on the telephone line.

- ✓ UART uses the standard TTL/CMOS logic, in which

  Logic 1 state → 3.5 to 5 V
  Logic 0 state → 0 to 0.5 V

  whereas, RS-232 follows the EIA standard, in which

  Logic 1 state → - 3V to - 25 V
  Logic 0 state → +3V to +25 V

- ✓ Hence, a level translator IC like MAX 232 is used for converting the signal lines from UART to RS-232 signals lines for communication.

(2) Universal Serial Bus

- ✓ Universal Serial Bus (USB) is a high speed serial bus for data communication between host system and peripherals.

- ✓ The USB was developed to simplify and improve the interface between host and peripheral devices. Because of its wide variety of uses, including support for electrical power, the USB has replaced a wide range of interfaces like the parallel and serial port.

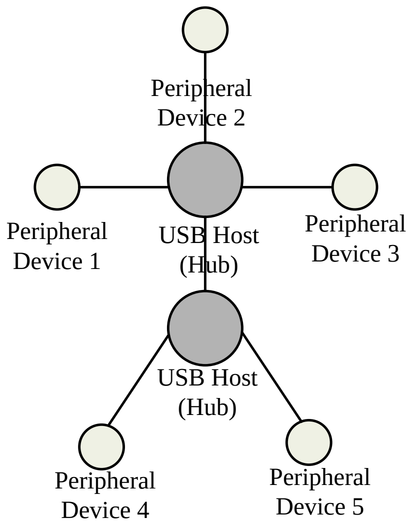
- ✓ It connects peripheral devices such as digital camera, mice, keyboard, printer, scanner, media device, modems, joysticks, external hard drive and flash memory cards etc.

Features

- True Plug-and-Play
- Self-configuring
- Hot swapping - plug and unplug without rebooting
- Small devices can be powered directly from the USB interface
- Ease of use
- Up to maximum 127 physical devices
- High speed
- Low cost cables and connectors

The 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 packet. There are different standards for implementing the USB Host Control interface; namely Open Host Control Interface (OHCI) and Universal Host Control Interface (UHCI).

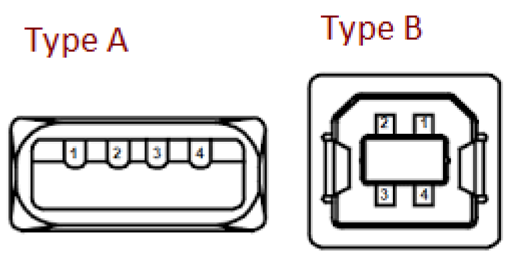
A USB host can support connections up to 127, including slave peripheral devices and other USB hosts. The USB communication system follows a star topology with a USB host at the center and one or more USB peripheral devices/USB hosts connected to it



Although there are several types of USB connectors for connecting the USB peripheral and host devices, the majority of USB cables are one of two types, Type A and Type B.

Type A connector is used for upstream connection (connection with host). It has a flat rectangle interface that inserts into a hub or USB host which transmits data and supplies power. The USB connectors present in desktop PCs / laptops are examples for Type A.

Type B connector is used for downstream connection (connection with slave device). It is square with slanted exterior corners. The type B connector also transmits data and supplies power. Some type B connectors are used only as a power connection.



Pin No.	Pin Name	Description
1	+Vcc	+Vcc
2	D -	Differential Data -
3	D+	Differential Data +
4	GND	Ground

Both Type A and Type B connectors contains 4-pins for communication.

- Two for power (+5v & GND) and
- Two for differential data signals (labelled as D+ and D- in pin out).

The Data+ and Data- signals are transmitted on a twisted pair with no termination needed. Half-duplex differential signalling is used to reduce the effects of electromagnetic noise on longer lines. D+ and D- operate together; they are not separate simplex connections

USB supports four different types of data transfers

Control transfer:

Used by USB system software to configure and issue commands to the USB device.

Isochronous transfer:

Used for real-time data communication (e.g., real time audio or video).

It doesn't support error checking and re-transmission of data, in case of transmission loss.

It guaranteed data rate (for fixed-bandwidth streaming data) but with possible data loss.

Bulk transfer:

Used to send block of data to a device (e.g., file transfers). It supports error checking and correction. It doesn't guarantee on bandwidth or latency

Interrupt transfer:

Used for transferring small amounts of data. It makes use of polling technique to see whether the USB device has and data to send. Devices like Mouse and Keyboard, which transfers small amounts of data, uses Interrupt transfer.

The USB supports different data rates – Low speed 1.5 Mbps (USB 1.0),

- - High speed 480 Mbps (USB 2.0)
- - Super speed 4.8Gbps (USB 3.0)

(3) Infrared (IrDA)

- ✓ Infrared (IrDA) is a serial, half duplex, line of sight based wireless technology for data communication between devices. The remote control of TV, DVD player, etc. works on Infrared data communication principle.

- ✓ IrDA supports point-point and point-to-multipoint communication, provided all devices involved in the communication are within the line of sight.

- ✓ 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.

- ✓ IR supports data rates ranging from 9600 bps to 16Mbps. Depending on the speed of data transmission IR is classified as follows:
  Serial IR (SIR) – supports transmission rates 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) - support high data rates up to 16Mbps
  Ultra-Fast IR (UFIR) - supports data rate up to 100Mbps.

- ✓ 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 the receiver unit).

- ✓ 'Infra-red Data Association' is the regulatory body responsible for defining and licensing the specifications for IR data communication. IrDA 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. The protocol part is responsible for defining the rules of communication. The IrDA control protocol contains implementations for Physical Layer (PHY), Media Access Control (MAC) and Logical Link Control (LLC). The Physical Layer defines the physical characteristics of communication like range, data rates, power, etc.

- ✓ 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. Even now most of the mobile phone devices support IrDA.

(4) Bluetooth (BT)

- Bluetooth is a low cost, low power, short range wireless technology for data and voice communication.
- 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.

- Like IrDA, Bluetooth communication also 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.

- 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. 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 is the favourite 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.

(5) Wi-Fi (Wireless Fidelity)

- Wi-Fi is the popular wireless communication technique for networked communication of devices. Wi-Fi follows the IEEE 802.11 standard.

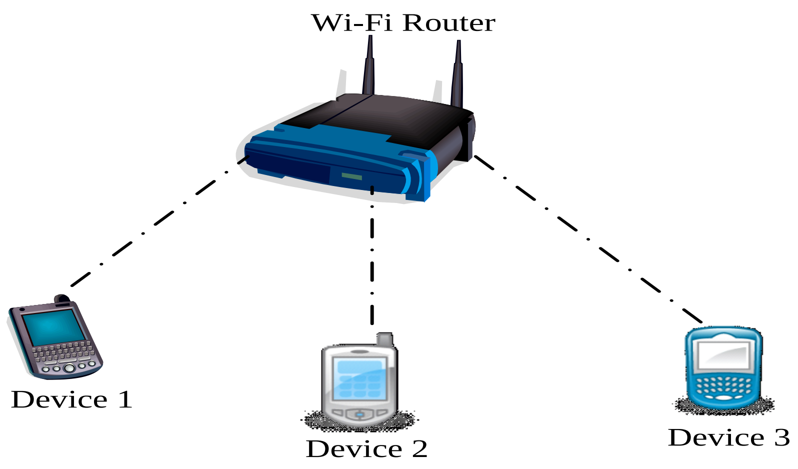
- Wi-Fi is intended for network communication and it supports Internet Protocol (IP) based communication. It is essential to have device identities in a multipoint communication to address specific devices for data communication. In an IP based communication each device is identified by an IP address, which is unique to each device on the network.

- Wi-Fi 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 called as Wi-Fi Radio. Wi-Fi operates at 2.4GHz or 5GHz of radio spectrum.

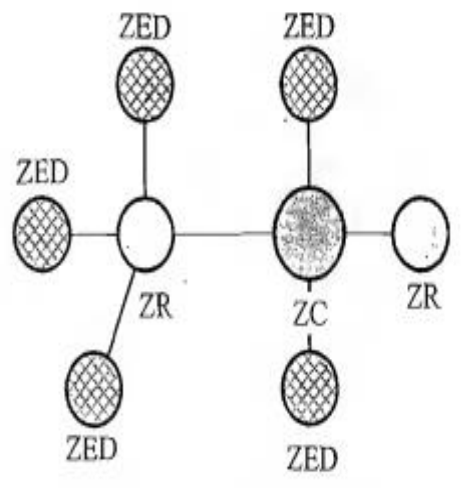
- For communicating with devices over a Wi-Fi network, the device when its Wi-Fi radio is turned ON, searches the available Wi-Fi network in its vicinity and lists out the Service Set Identifier (SSID) of the available networks. If the network is security enabled, a password may be required to connect 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 1Mbps to 150Mbps (Growing towards higher rates as technology progresses) 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. The IEEE 802.11ax Wi-Fi protocol supports up to 2.4 Gbps.



(6) 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, 902 to 928 MHz and 868.0 to 868.6 MHz.
- ZigBee Supports data rate of 20 to 250Kbps & operating distance of up to 100 metres.



In the ZigBee terminology, each ZigBee device falls under any one of the following 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 is primarly targeting application areas like home and industrial automation, energy management, home security, medical/patient tracking, logistics and asset tracking. Automatic meter reading, smoke detectors, wireless telemetry, heating control, lighting controls, environmental controls, etc are examples for applications of ZigBee technology.

2.6. 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.

The following are examples of circuits/ICs which are essential for the proper functioning of the processor/controllers.

1. Reset Circuit
2. Brown-out Protection
3. Oscillator Unit
4. Watchdog Timer (WDT)
5. Real Time Clock (RTC)
6. PCB and Passive components

Some of the controllers (or) SoCs integrate these components within a single IC. Depending on the system requirement, the embedded system may include other integrated circuits for performing specific functions, level translator ICs for interfacing circuits with different logic levels, etc.

(1) Reset Circuit

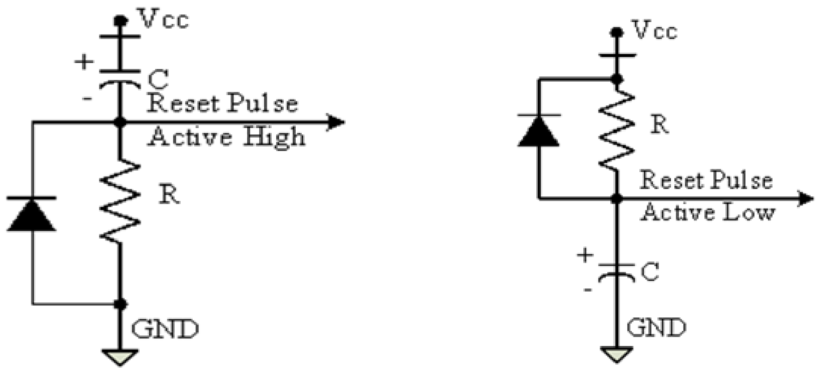


Fig: Resistor Capacitor based passive Reset circuits for Active High and Active Low configurations

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.

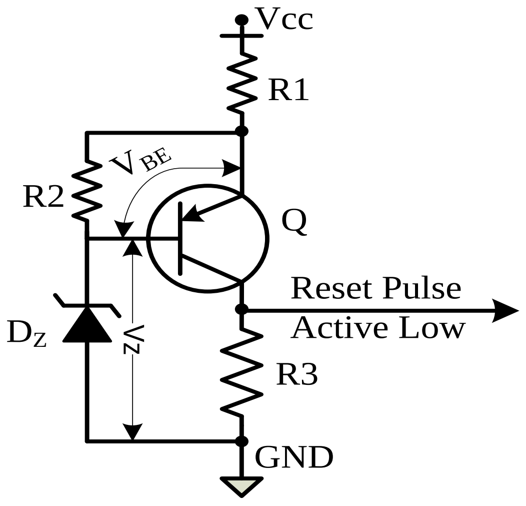
The reset signal brings the internal registers and the different hardware systems of the processor/controller to a known state and starts the firmware execution from the reset vector (Normally from vector address 0x0000 for conventional processors/controllers. The reset vector can be relocated to an address for processors supporting bootloader).

The reset signal can be either active High (The processor undergoes reset when the reset pin of the processor is at logic High) or active Low (The processor undergoes reset when the reset pin of the processor is at logic Low).

Since the processor operation is 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. 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, they don't require external circuitry.

(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 (threshold).

It is essential for battery powered devices since there are greater chances for the battery voltage to drop below the required threshold.

The processor behavior may not be predictable if the supply voltage falls below the recommended operating voltage. It may lead to situations like data corruption.

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.

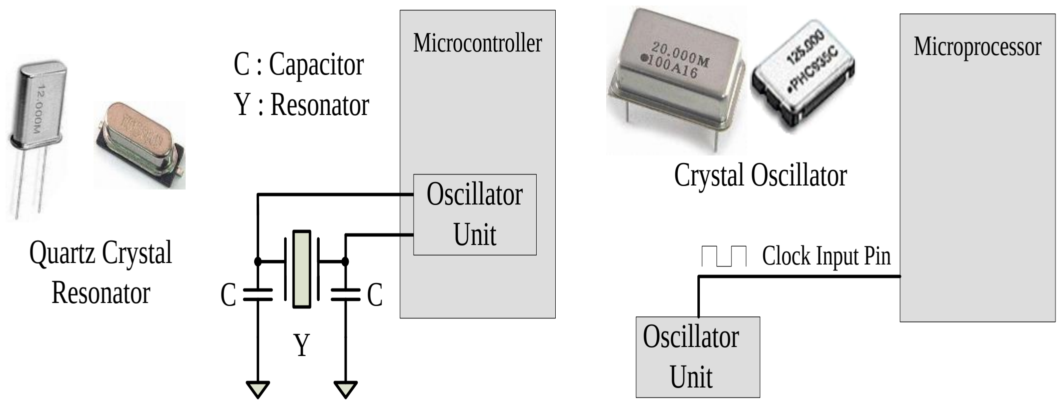
The above figure illustrates a brown-out circuit implementation using Zener diode and transistor with active Low reset logic. The transistor conducts always when supply voltage is greater than V_{CC}. The transistor stops conducting when the supply voltage falls below (V_{BE} +V_Z) and the output goes to LOW state. Zener diode with required voltage is selected for setting the threshold value.

(3) Oscillator Unit

- ✓ A microprocessor/microcontroller is a digital device made up of digital combinational and sequential circuits. The instruction execution of a processor/controller occurs in sync with a clock signal.

- ✓ The Oscillator unit is responsible for generating the precise clock for the processor/controller, memory and all peripherals.

- ✓ The Clock is required to keep the whole system synchronized
- ✓ Clock may be generated internally (or) obtained from a crystal or external source



- ✓ 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 processor/controller chips 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 primarily depends on the clock frequency.
- ✓ The total system power consumption is directly proportional to the clock frequency.
- ✓ The accuracy of program execution depends on the accuracy of clock signal.

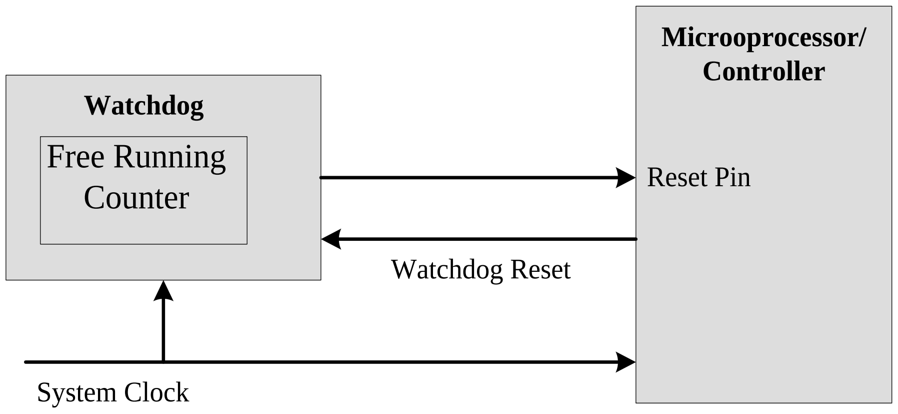
(4) Watchdog Timer

- A watchdog timer (WDT) is an electronic timer that is used to detect and recover from computer malfunctions. The WDT module restarts the system on occurrence of a software problem (or) if a selected time interval expires.

- During normal operation, the system regularly restarts the WDT to prevent it from elapsing, or "timing out". If the system fails to restart the WDT due to a hardware fault (or) program error, the timer will elapse and generate a timeout signal.

- The timeout signal is used to initiate corrective actions like placing the system in a safe state and restoring normal system operation.

- The main purpose of the watchdog timer is to protect the system against failure of the software, such as the program becoming trapped in an unintended, infinite loop. Left to itself, the watchdog counts up and resets the system when it reaches its limit.



- Watchdog Timer is a hardware timer for monitoring the firmware execution. 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.

- 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 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 the WDT can be stopped from action.

- 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, instead of firmware based 'writing' to the status and watchdog registers.

(5) Real Time Clock

- ✓ Real Time Clock (RTC) is a system component responsible for keeping track of time.
- ✓ RTC holds information like current time (hour, minutes and seconds) in 12Hr/24Hr format, date, month, year, day of the week 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 is interfaced to the processor or controller of the embedded system.
- ✓ 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.
- ✓ For Operating System based embedded devices, a timing reference is essential for synchronizing 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 Real Time Clock (RTC) generates system interrupts periodically for schedulers, real time programs and for periodic saving of time and date in the system.
- ✓ The RTC can be configured to interrupt the processor at pre-defined intervals (or) to interrupt the processor when the RTC register reaches a specified value (used as alarm interrupt)

(6) 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 hardware components as per the design requirement. Also it acts as a platform for testing the embedded firmware.
- Apart from the important subsystems/ICs, some passive electronic components like resistors, capacitors, diodes, etc on the board, which are required for various chips contained in embedded hardware. They are very essential for the proper functioning of embedded system.

2.7. CHARACTERISTCS OF EMBEDDED SYSTEM

An embedded system is an electronic/electro-mechanical system designed to perform a specific function and is combination of both hardware and firmware (software).

Unlike general purpose computing system, embedded systems possess certain specific characteristics and these are unique to each embedded system. Some of the important characteristics of embedded system are

(1). Application and Domain Specific

- Every embedded system is unique and the hardware as well as software is highly specialized to the application domain.
- They do a very specific task; they can't be used for any other purpose.
- **Both hardware & software in an embedded system are optimized for the specific task.**
- For ex., we can't replace the embedded control unit of microwave oven with air-conditioners embedded control unit, because the embedded control units of microwave oven and air-conditioner are specifically designed to perform certain specific tasks.

  Similarly, we can't replace an embedded control unit developed for a particular domain say telecom with another control unit designed to serve another domain like consumer electronics.

(2). Reactive and Real Time

- Embedded systems are in constant interaction with the real world through sensors and user-defined input devices. Any changes happening in the Real world (called as an Event) are captured by the sensors/input devices and the control algorithm reacts in a designed manner to bring the controlled output variables to the desired level.
- The event may be a periodic one (or) unpredicted one. If the event is unpredicted one, then such systems should be de designed in such a way that it should be scheduled to capture the events without missing them.
- Embedded systems produce changes in output in **response** to the changes in the input and hence they are generally referred **as Reactive systems.**
- Real Time System operation means the timing behaviour of the system should be deterministic, meaning that the system should respond to requests in a known amount of time. The real time systems have to perform the specific task in a specified period. They have time constrains and they have to work against some deadlines. **Meeting the deadlines is the most important requirement of Real-Time system**. Hence the response time requirement is highly critical.
- Real Time System should not miss any deadline for tasks or operations. For example, in a nuclear plant safety system, missing a deadline may cause loss of life/ damage to property.

  Examples of Real Time Systems – Missiles, flight control systems,
  Nuclear plane safety system
  Antilock Break System (ABS)

(3). Operates in harsh environments

- It is not necessary that all embedded systems should be deployed in controlled environment. Some of the embedded systems are to be operated in harsh environment conditions like high temperature zone, areas subject to vibrations and shock etc. Embedded Systems placed in such areas should be capable to withstand all these adverse operating conditions.
- The design should take care of the operating conditions of the area where the system is going to implement. For example, if the system needs to be deployed in a high temperature zone, then all components used in the system should be of high temperature grade. Proper shock absorption techniques should be provided to systems which are going to be operated in places subject to high shock. Power supply fluctuations, corrosion and component aging, etc are the other factors that need to be taken into consideration for embedded systems to work in harsh environments.

(4). Distributed

- The term distributed means that embedded systems may be a part of larger systems. Many numbers of such distributed embedded systems form a single large embedded control unit.
- A typical example for distributed embedded system is an Automatic Teller Machine (ATM). The ATM contains a card reader embedded unit, responsible for reading and validating the user's ATM card, transaction unit for performing transactions, a currency counter for dispatching/vending currency to the authorised person and a printer unit for printing the transaction details. Each of them are independent embedded units but they work together to perform overall function.
- Another typical example of distributed embedded system is the Supervisory Control and Data Acquisition System (SCADA) used in Control and instrumentation applications. It contains physically distributed individual embedded control units connected to a supervisory module.

(5). Small Size and Weight

- Product aesthetics (size, weight, shape and style) will be one of deciding factors to choose a product. For example, when we plan to buy a new mobile phone, we make a comparative study on the pros and corns of the products available in the market.
- It is more convenient to handle a compact device than a bulky product.
- Most of the application demands small sized and low weight products.

(6). Power concern

- Power management is another important factor that needs to be considered in designing embedded system. Embedded systems should be designed in such a way as to minimise the heat dissipation by the system.
- Most of the Embedded Systems operate through a battery. To reduce the battery drain and to avoid frequent recharging of the battery, the power consumption has to be very low. Hardware designers must address this issue – for example, select the design according to the low power components like low dropout regulators, and controllers with power saving modes, switch the processor operation mode to low-power mode when there is no operation to perform.

2.8. QUALITY ATTRIBUTES OF EMBEDDED SYSTEM

Quality attributes are the non-functional requirements that need to be addressed properly in any system design. If the quality attributes are more concrete and measurable, it will give a positive impact on the system development process and the end product.

Quality attributes are classified into two, namely

- Operational Quality Attributes and
- Non-Operational Quality Attributes

Operational Quality Attributes

The quality attributes that are related to the embedded system when it is in the operational mode or online mode.

The important operational quality attributes are listed below:

1. Response
2. Throughput
3. Reliability
4. Maintainability
5. Security
6. Safety

(1) Response

- Response is a measure of quickness of the system. It gives you an idea about how fast your system is tracking the changes in input variables.
- Most of the embedded systems demand fast response which should be almost Real Time. For example, an embedded system deployed in flight control application should respond in a Real Time manner. Any response delay in the system will create potential damages to the safety of the flight as well as passengers.

(2) Throughput

- Throughput deals with the efficiency of a system. It is defined as the rate of operation of a defined process over a period of time.
- For example, in the case of a Card reader, throughput means how many transactions the Reader can perform in a minute, or in an hour or in a day.
- Throughput is generally measured in terms of 'Benchmark'. A 'Benchmark' is a reference point by which something can be measured. Benchmark can be a set of performance criteria that a product is expected to meet, or a standard product that can be used for comparing other products of the same product line.

(3) Reliability

- Embedded Systems have to work for thousands of hours without break. This calls for very reliable hardware and software.
- Reliability is a measure of how much percentage we can rely upon the proper functioning of the system without failures
- Mean Time Between Failures (MTBF) and Mean Time To Repair (MTTR) are the terms used in defining system reliability.
- MTBF gives the frequency of failures in hours/weeks/months. MTTR specifies how long the system is allowed to be out of order following a failure.

(4) Maintainability

- Maintainability deals with support and maintenance to the end user or client in case of technical issues and product failures or on the basis of a routine system check-up.
- Reliability and Maintainability are considered as two complementary disciplines. A more reliable system means a system with less corrective maintainability requirements and vice versa. As the reliability of the system increases, the chances of failure and non-functioning also reduces, there by the need for maintainability is also reduced.
- Maintainability can be broadly classified into two categories, namely,
  'Scheduled or Periodic Maintenance (preventive maintenance)' and 'Maintenance to unexpected failures (corrective maintenance)'.
- A printer is a typical example for illustrating the two types of maintainability.
  An inkjet printer uses ink cartridges, which are consumable components and are to be replaced after each 'n' number of print outs to get quality prints. This is an example for 'Scheduled or Periodic maintenance.
  If the paper feeding part of the printer fails, the printer fails to print and it requires immediate repairs to rectify this problem. This is an example for 'Maintenance to unexpected failure. In both cases, the printer needs to be brought offline, and it will not be available to the user during this period. Hence, it is obvious that maintainability is simply an indication of the availability of the product for use.

(5) Security

- Confidentiality, Integrity and Availability are the three major measures of information security.
- Confidentiality deals with the protection of data and application from **unauthorised disclosure**. Integrity deals with the protection of data and application from **unauthorised modification**. Availability deals with protection of data and application from **unauthorized users**.
- A typical example for the 'Security' aspect in an embedded product is PDA (Personal Digital Assistant)

(6) Safety

- Safety deals with the possible damages that can happen to the operators, public and the environment due to the breakdown of an embedded system (or) due to the emission of radioactive (or) hazardous materials from the embedded products.
- The breakdown of an embedded system may occur due to a hardware failure or a firmware failure.

Non-Operational Quality Attributes

The quality attributes that needs to be addressed for the product 'not' on the basis of operational aspects are grouped under this category. The important quality attributes coming under non-operational are listed below.

1. Testability & Debug-ability
2. Evolvability
3. Portability
4. Time to prototype and market .
5. Per unit and total cost

(1) Testability and Debug-ability

- Testability deals with how easily one can test design, application and product. For an embedded product, testability is applicable to both the embedded hardware and firmware.
- Debug-ability is a means of debugging the product for figuring out the probable sources that create unexpected behaviour in the total system.
- Debug-ability has two aspects in the embedded system development context, namely, hardware level debugging and firmware level debugging.
- Hardware debugging is used for figuring out the issues created by hardware problems. Software debugging is employed to figure out the probable errors that appear in the firmware.

(2) Evolvability

- For an embedded system, the quality attribute 'Evolvability' refers to the ease with which the embedded product (including firmware and hardware) can be modified to take advantage of new firmware or hardware technologies.

(3) Portability

- Portability is a measure of 'system independence'. An embedded product is said to be portable if the product is capable of functioning in various environments, target processors/controllers and embedded operating systems. A standard embedded product should always be flexible and portable.

- In embedded products, the term 'porting' represents the migration of the embedded firmware written for one target processor to a different target processor. If the firmware is written in a high level language like 'C', it is very easy to port the firmware for the new processor by replacing some 'target processor-specific functions' and re-compiling the program for the new target processor to generates the new target processor-specific machine codes.
- If the firmware is written in Assembly Language for a particular family of processor, it will be very difficult to translate the assembly language instructions to the new target processor specific language and so the portability is poor.
- Another example is, applications developed using Microsoft technologies (e.g. Microsoft Visual C++ using Visual studio) is capable of running only on Microsoft platforms and will not function on other operating systems; whereas applications developed using 'Java' from Sun Microsystems works on any operating system that supports Java standards.

(4) Time to prototype and Market

- Time-to-market is the time elapsed between the conceptualisation of a product and the time at which the product is ready for selling (for commercial product) or use (for non-commercial products). The commercial embedded product market is highly competitive and time to market the product is a critical factor for the success.
- There may be multiple players in the embedded industry who develop products of the same category (like mobile phone, portable media players, etc.). If you come up with a new design and if it takes long time to develop and market it, the competitor product may take advantage of it with their product.
- Also, embedded technology is one where rapid technology change is happening. If you start your design by making use of a new technology and if it takes long time to develop and market the product, by the time you market the product, the technology might have outdated with a new technology.

(5) Per Unit Cost and Revenue

- Cost is a factor which is closely monitored by both end user and product manufacture. Cost is a highly sensitive factor for commercial products. Any failure to position the cost of a commercial product at a nominal rate, may lead to the failure of the product in the market.
- Proper market study and cost benefit analysis should be carried out before taking a decision on the per-unit cost of the embedded product.

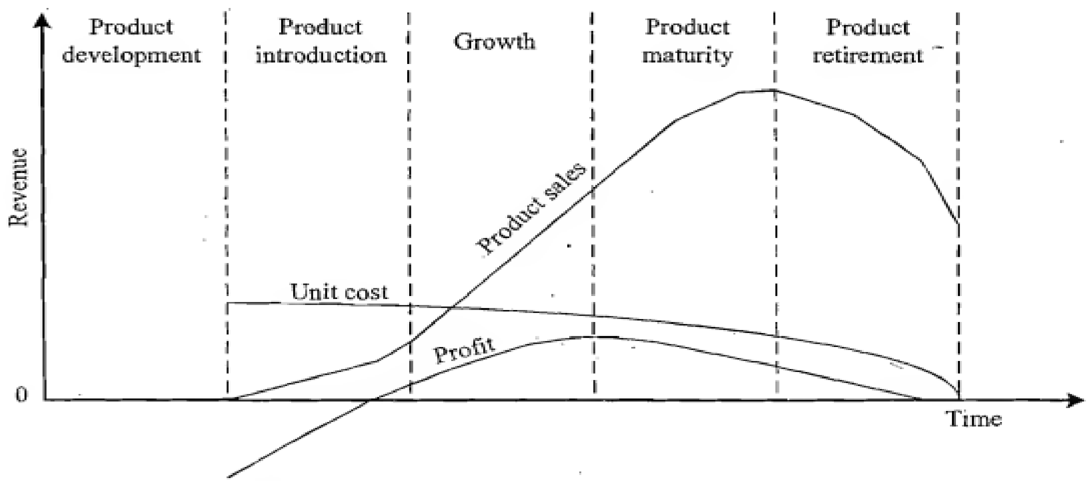


Figure: Product life cycle (PLC) curve

- From a designer/product development company perspective the ultimate aim of a product is to generate marginal profit. So the budget and total system cost should be properly balanced to provide a marginal profit.
- Every embedded product has a product life cycle which starts with the **design and development phase**. The product idea generation, prototyping, Roadmap definition, actual product design and development are the activities carried out during this phase. During the design and development phase there is only investment and no returns.
- Once the product is ready to sell, it is introduced to the market. This stage is known as the **Product Introduction stage**. During the initial period the sales and revenue will be low. There won't be much competition and the product sales and revenue increases with time. In the **growth phase**, the product grabs high market share. During the **maturity phase**, the growth and sales will be steady and the revenue reaches at its peak. The Product **Retirement/Decline phase** starts with the drop in sales volume, market share and revenue. The decline happens due to various reasons like competition from similar product with enhanced features or technology changes, etc. At some point of the decline stage, the manufacturer announces discontinuing of the product.
- For a commercial embedded product, the unit cost is peak during the introductory stage and revenue is peak during the maturity stage.
- **Product Life-cycle (PLC) curve is** the graphical representation of the unit cost, product sales and profit with respect to the various life cycle stages of the product starting from conception to disposal.