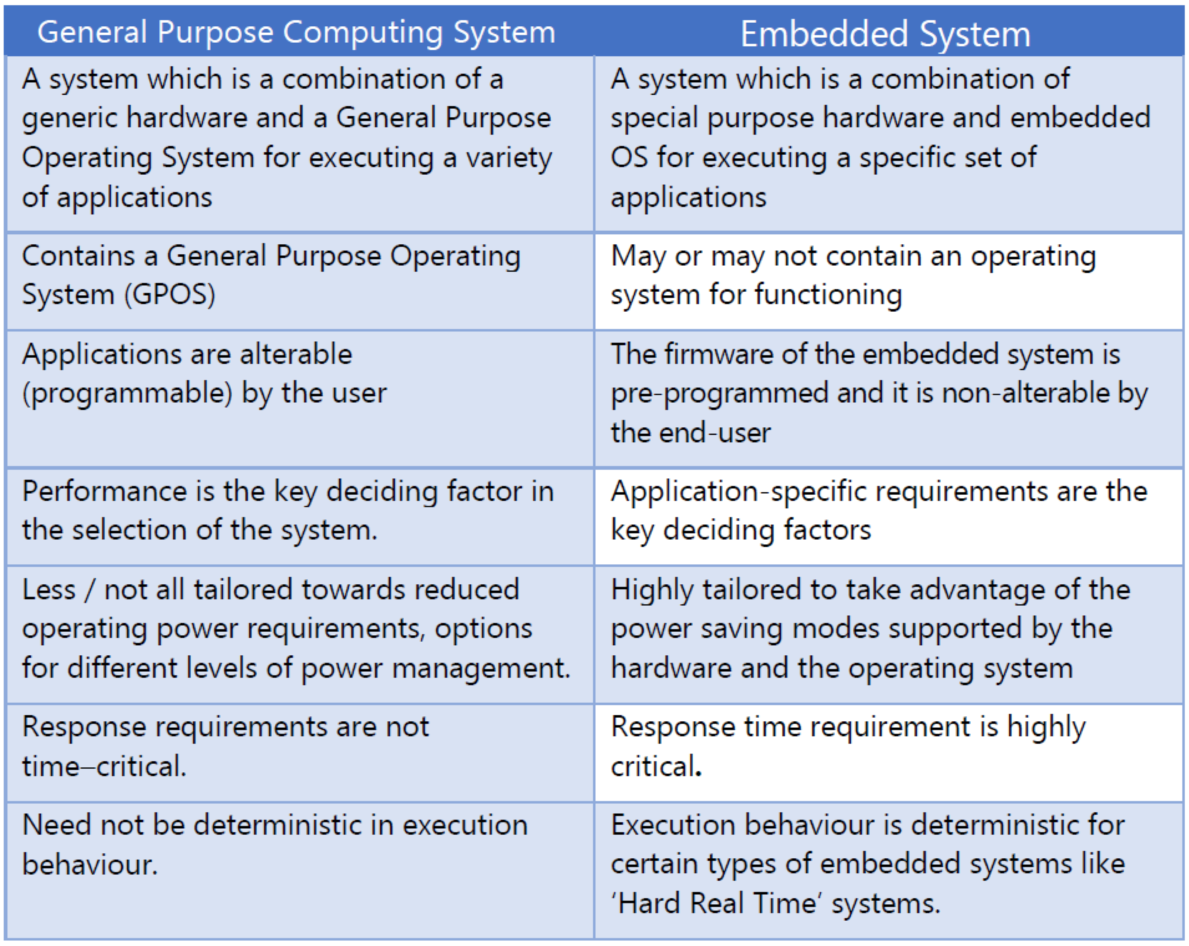
IES Unit 1

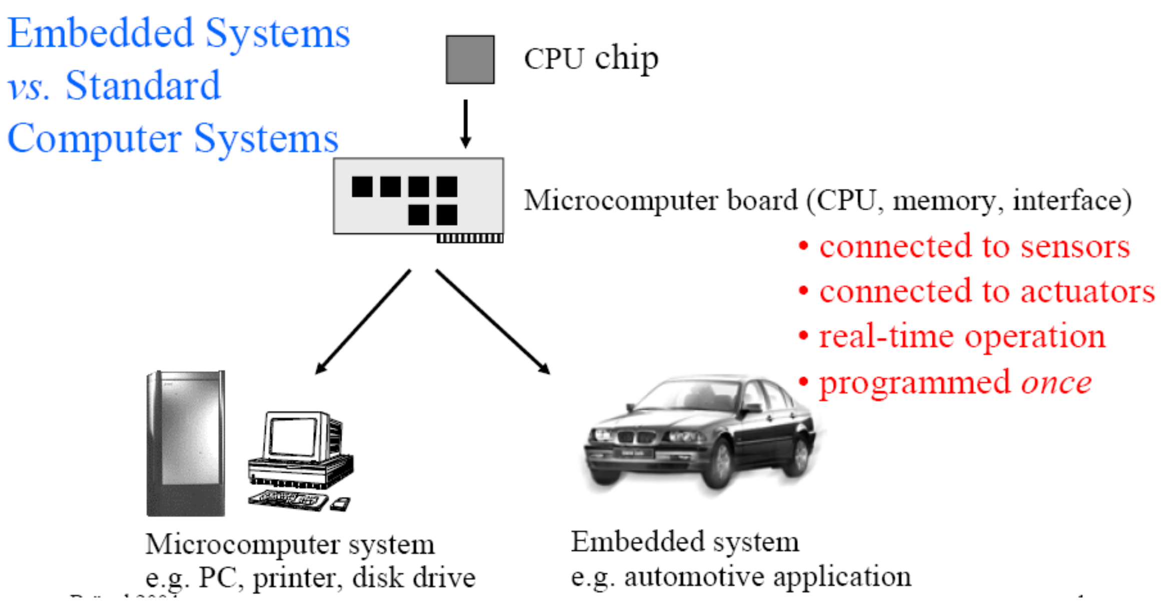
What is Embedded System?

- An embedded system is an electronic/electro-mechanical system designed to perform a specific function and is a combination of both hardware and firmware (software).
- Every embedded system is unique, and the hardware as well as the firmware is highly specialized to the application domain. Embedded systems are becoming an inevitable part of any product or equipment in all fields including household appliances, telecommunications, medical equipment, industrial control, consumer products, etc.

General Purpose Computing System VS Embedded System -



General Purpose Computing System	Embedded System
A system which is a combination of a generic hardware and a General Purpose Operating System for executing a variety of applications	A system which is a combination of special purpose hardware and embedded OS for executing a specific set of applications
Contains a General Purpose Operating System (GPOS)	May or may not contain an operating system for functioning
Applications are alterable (programmable) by the user	The firmware of the embedded system is pre-programmed and it is non-alterable by the end-user
Performance is the key deciding factor in the selection of the system.	Application-specific requirements are the key deciding factors
Less / not all tailored towards reduced operating power requirements, options for different levels of power management.	Highly tailored to take advantage of the power saving modes supported by the hardware and the operating system
Response requirements are not time–critical.	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.


Embedded Systems *vs.* Standard Computer Systems
CPU chip
Microcomputer board (CPU, memory, interface)
• connected to sensors
• connected to actuators
• real-time operation
• programmed *once*
Microcomputer system
e.g. PC, printer, disk drive
Embedded system
e.g. automotive application

Classifications for Embedded Systems -

- It is possible to have a multitude of classifications for embedded systems, based on different criteria.
- Some of the criteria used in the classification of embedded systems are -

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

Classification Based on Generation -

- This classification is based on the order in which the embedded processing systems evolved from the first version to where they are today.
- As per this criterion, embedded systems can be classified into -

1. **First Generation**

- Built around 8 bit microprocessors and 4bit microcontrollers
- Simple in hardware circuits with firmware developed in Assembly code
- Digital telephone keypads, stepper motor control units etc.

2. **Second Generation**

- Built around 16 bit microprocessors and 8 or 16 bit microcontrollers
- Instruction set were much more complex and powerful than the first generation
- Some of the second generation embedded systems contained embedded operating systems for their operation
- Data Acquisition Systems, SCADA systems, etc. are examples of second generation embedded systems.

3. **Third Generation**

- Use of powerful 32bit processors and 16bit microcontrollers
- Digital Signal Processors (DSP) and Application Specific Integrated Circuits (ASICs)

- Instruction set became more complex and powerful and the concept of instruction pipelining also evolved
- Dedicated embedded real time and general purpose operating systems
- Robotics, media, industrial process control, networking, etc.

4. **Fourth Generation**

- The advent of System on Chips (SoC), reconfigurable processors and multicore processors
- High performance, tight integration and miniaturization into the embedded device market.
- Use of high performance real time embedded operating systems
- Smart phone devices, mobile internet devices (MIDs), etc.

Classification Based on Complexity and Performance -

- This classification is based on the complexity and system performance requirements. According to this Classification, embedded systems can be grouped into:

1. **Small−Scale Embedded Systems**

- Performance requirements are not time critical
- Built around low performance and low cost 8 or 16 bit microprocessors / microcontrollers.
- may or may not contain an operating system for its functioning

2. **Medium−Scale Embedded Systems**

- Slightly complex in hardware and firmware (software)
- Built around medium performance, low cost 16 or 32 bit microprocessors / microcontrollers or digital signal processors.
- Contain an embedded operating system

3. **Large−Scale Embedded Systems / Complex Systems**

- Highly complex hardware and firmware requirements

- Built around high performance 32 or 64 bit RISC processors / controllers or Reconfigurable System on Chip (RSoC) or multi-core processors and programmable logic devices
- May contain multiple processors/controllers and co-units/hardware accelerators for offloading the processing requirements from the main processor of the system
- Decoding/ encoding of media, cryptographic function implementation, etc.
- Complex embedded systems usually contain a high performance Real Time Operating Systems (RTOS) for task scheduling, prioritization and management.

Major Application Areas of Embedded Systems -

1. **Consumer electronics**: Camcorders, cameras, etc.
2. **Household appliances:** Television, DVD players, washing machine, fridge, microwave oven, etc.
3. **Home automation and security systems**: Air conditioners, sprinklers, intruder detection alarms, closed circuit television cameras, etc.
4. **Automotive industry**: Anti-lock breaking systems (ABS), engine control, ignition systems, automatic navigation systems, etc.
5. **Telecom**: Cellular telephones, telephone switches, handset multimedia applications, etc.
6. **Computer peripherals**: Printers, scanners, fax machines, etc.
7. **Computer networking systems:** Network routers, switches, hubs, firewalls, etc.
8. **Healthcare**: Different kinds of scanners, EEG, ECG machines etc.
9. **Measurement & Instrumentation**: Digital multi meters, digital CROs, logic analyzers PLC systems, etc.
10. **Banking & Retail**: Automatic teller machines (ATM) and currency counters, point of sales (POS)
11. **Card Readers:** Barcode, smart card readers, hand held devices, etc.

Purpose of Embedded Systems -

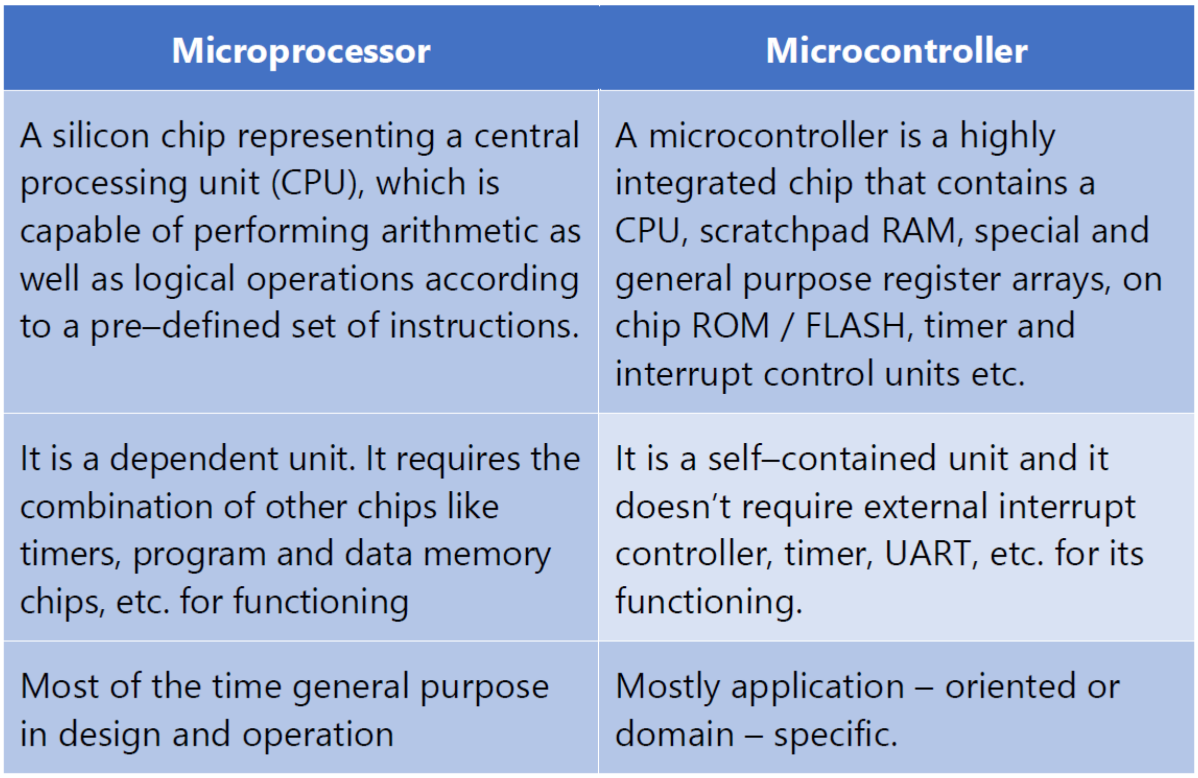
- As mentioned in the previous section, embedded systems are used in various domains like consumer electronics, home automation, telecommunications, automotive industry, healthcare, control & instrumentation, retail and banking applications, etc.
- Within the domain itself, according to the application usage context, they may have different functionalities.
- Each embedded system is designed to serve the purpose of any one or a combination of the following tasks -

1. Data collection / Storage / Representation
2. Data communication
3. Data (signal) processing
4. Monitoring
5. Control
6. Application specific user interface

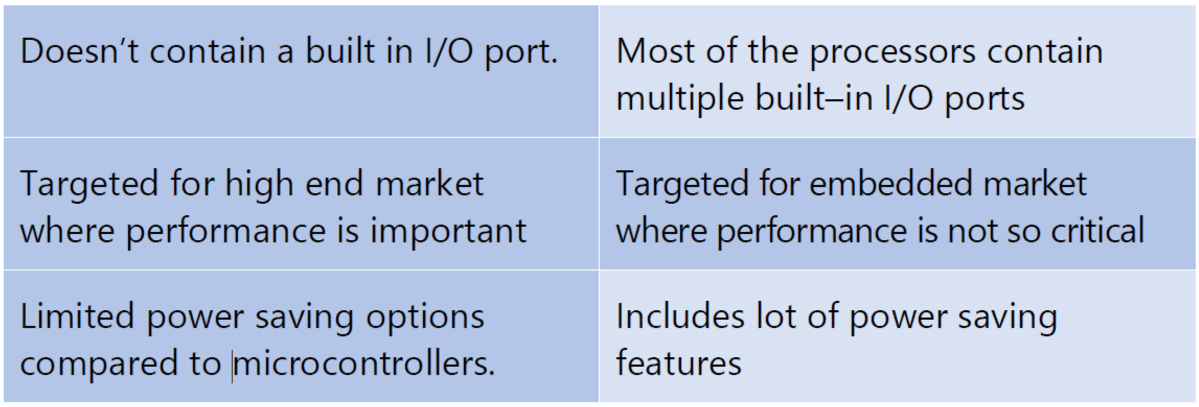
Core of the Embedded System -

- 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 -
- General Purpose and Domain Specific Processors
    1. Microprocessors
    2. Microcontrollers
    3. Digital Signal Processors
- Application Specific Integrated Circuits (ASICs)
- Programmable Logic Devices (PLDs)
- Commercial off-the-shelf Components (COTS)
- If you examine any embedded system you will find that it is built around any of the core units mentioned above.

Microprocessor vs. Microcontroller -



Microprocessor	Microcontroller
A silicon chip representing a central processing unit (CPU), which is capable of performing arithmetic as well as logical operations according to a pre–defined set of instructions.	A microcontroller is a highly integrated chip that contains a CPU, scratchpad RAM, special and general purpose register arrays, on chip ROM / FLASH, timer and interrupt control units etc.
It is a dependent unit. It requires the combination of other chips like timers, program and data memory chips, etc. for functioning	It is a self–contained unit and it doesn't require external interrupt controller, timer, UART, etc. for its functioning.
Most of the time general purpose in design and operation	Mostly application – oriented or domain – specific.



Doesn't contain a built in I/O port.	Most of the processors contain multiple built–in I/O ports
Targeted for high end market where performance is important	Targeted for embedded market where performance is not so critical
Limited power saving options compared to microcontrollers.	Includes lot of power saving features

RISC vs. CISC Processors/Controllers -

RISC	CISC
Lesser number of instructions	Greater number of Instructions
Instruction pipelining and increased execution speed	Generally no instruction pipelining feature
Orthogonal instruction set	Non-orthogonal instruction set
Operations are performed on registers only, the only memory operations are load and store	Operations are performed on registers or memory depending on the instruction
A large number of registers are available	Limited number of general purpose registers
Programmer needs to write more code to execute a task since the instructions are simpler ones	A programmer can achieve the desired functionality with a single instruction which in turn provides the effect of using more simpler single instructions in RISC
Single, fixed length instructions	Variable length instructions
With Harvard Architecture	Can be Harvard or Von-Neumann Architecture

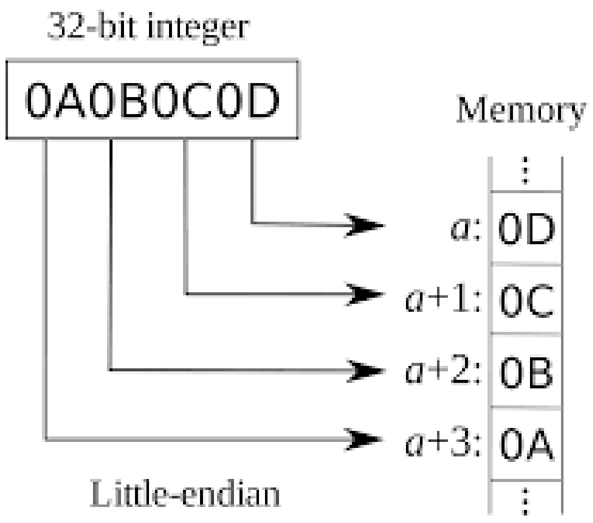
Big-Endian vs. Little-Endian Processors/Controllers -

- Endianness specifies the order in which the data is stored in the memory by processor operations in a multi byte system (Processors whose word size is greater than one byte).
- Suppose the word length is two byte then data can be stored in l memory in two different ways:

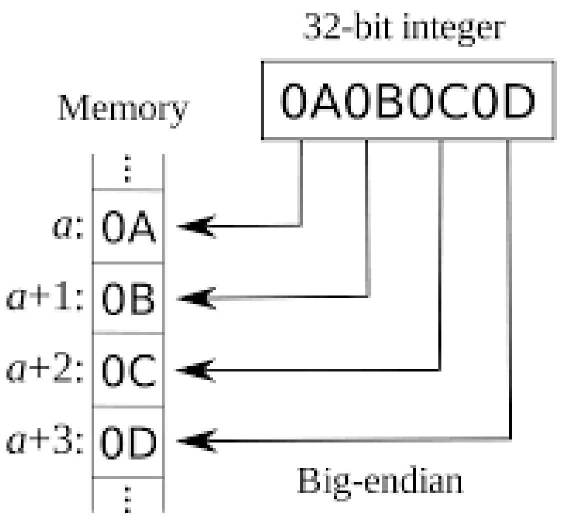
1. Higher order of data byte at the higher memory and lower order of data byte at location just below the higher memory.

2. Lower order of data byte at the higher memory and higher order of data byte at location just below the higher memory.

Little−endian (Fig. 10) means the lower−order byte of the data is stored in memory at the lowest address, and the higher−order byte at the highest address. (The little end comes first). For example, a 4 byte long integer Byte3 Byte2 Byte1 Byte0 will be stored in the memory as shown below -



Big−endian (Fig. 11) means the higher−order byte of the data is stored in memory at the lowest address, and the lower−order byte at the highest address. (The big end comes first.) For example, a 4 byte long integer Byte3 Byte2 Byte1 Byte0 will be stored in the memory as follows -



Application Specific Integrated Circuits (ASICs) -

- It integrates several functions into a single chip and there by reduces the system development cost helps in the design of smaller systems with high capabilities / functionalities
- ASIC based systems are profitable only for large volume commercial productions
- Fabrication of ASICs requires a non-refundable initial investment for the process technology and configuration expenses. This investment is known as Non-Recurring Engineering Charge (NRE) and it is a one time investment

Programmable Logic Devices -

- Logic devices provide specific functions, including device-to-device interfacing, data communication, signal processing, data display, timing and control operations, and almost every other function a system must perform. Logic devices can be classified into two broad categories − fixed and programmable. As the name indicates, the circuits in a fixed logic device are permanent, they perform one function or set of functions − once manufactured, they cannot be changed. On the other hand, Programmable Logic Devices (PLDs) offer customers a wide range of logic capacity, features, speed, and voltage characteristics-and these devices can be re-configured to perform any number of functions at any time.
- With programmable logic devices, designers use inexpensive software tools to quickly develop, simulate, and test their designs. Then, a design can be quickly programmed into a device, and immediately tested in a live circuit. There are no NRE costs and the final design is completed much faster than that of a custom, fixed logic device. Another key benefit of using PLDs is that during the design phase customers can change the circuitry as often as they want until the design operates to their satisfaction. That’s because PLDs are based on re-writable memory technology−to change the design, the device is simply reprogrammed

Commercial Off-the-Shelf Components (COTS) -

- Designed to provide easy integration and interoperability with existing system components
- They are readily available in the market, are cheap and a developer can cut down his/her development time to a great extent
- Reduces the time to market
- There are no operational and manufacturing standards
- Component manufactured by a vendor need not have hardware plug-in and firmware interface compatibility with other vendor
- Manufacturer of the COTS component may withdraw the product or discontinue the production of the COTS at any time

Sensors and Actuators -

- An embedded system is in constant interaction with the Real world and the controlling / monitoring functions executed by the embedded system is achieved in accordance with the changes happening to the Real world. The changes in system environment or variables are detected by the sensors connected to the input port of the embedded system.
- **Sensors**: A sensor is a transducer device that converts energy from one form to another for any measurement or control purpose.
- **Actuators**: Actuator is a form of transducer device (mechanical or electrical) which converts signals to corresponding physical action (motion). Actuator acts as an output device.

Communication Interface -

- Communication interface is essential for communicating with various subsystems of the embedded system and with the external world. For an embedded product, the communication interface can be viewed in two different perspectives; namely;
- Device/board level communication interface (Onboard Communication Interface)
- Product level communication interface (External Communication Interface).

- The communication channel which interconnects the various components within an embedded product is referred as device / board level communication interface (onboard communication interface). Serial interfaces like I2C, SPI, UART, 1−Wire, etc. and parallel bus interface are examples of ‘Onboard Communication Interface’.

Embedded Firmware -

- Embedded firmware refers to the control algorithm (Program instructions) and or the configuration settings that an embedded system developer dumps into the code (Program) memory of the embedded system. It is an un−avoidable part of an embedded system. There are various methods available for developing the embedded firmware. They are listed below.

1. Write the program in high level languages like Embedded C/C++ using an Integrated Development Environment (The IDE will contain an editor, compiler, linker, debugger, simulator, etc. IDEs are different for different family of processors / controllers. For example, Kiel micro vision3 IDE is used for all family members of 8051 microcontroller, since it contains the generic 8051 compiler C51).
2. Write the program in Assembly language using the instructions supported by your application’s target processor / controller.

- The instruction set for each family of processor / controller is different and the program written in either of the methods given above should be converted into a processor understandable machine code before loading it into the program memory.

Characteristics of Embedded Systems -

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

1. Application and domain specific
2. Reactive and Real Time
3. Operates in harsh environments

4. Distributed
5. Small size and weight
6. Power concerns

Quality Attributes of Embedded Systems -

- Quality attributes are the non−functional requirements that need to be documented 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.
- The various quality attributes that needs to be addressed in any embedded system development are broadly classified into two, namely ‘Operational Quality Attributes’ and ‘Non−Operational Quality Attributes’.

Operational Attributes -

- The operational quality attributes represent the relevant quality attributes related to the embedded system when it is in the operational mode or ‘online’ mode.
- The important quality attributes coming under this category are listed below:

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

Non-Operational 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 this category are listed below.

1. Testability & Debug-ability

2. Evolvability
3. Portability
4. Time to prototype and market
5. Per unit and total cost.