Prepared by **EBIN P.M C.Eng. (I), M.Tech**
ASSISTANT PROFESSOR
KOTTAYAM INSTITUTE OF TECHNOLOGY & SCIENCE (KITS)

MODULE-1
FUNDAMENTALS OF EMBEDDED SYSTEMS

SYSTEM

- A system is an arrangement in which all its unit assemble work together according to a set of rules.
- It can also be defined as a way of working, organizing or doing one or many tasks according to a fixed plan. For example, a watch is a time displaying system. Its components follow a set of rules to show time. If one of its parts fails, the watch will stop working. So we can say, in a system, all its subcomponents depend on each other.

EMBEDDED SYSTEM

- Embedded means something that is attached to another thing.
- An embedded system can be thought of as a computer hardware system having software embedded in it.
- An embedded system can be an independent system or it can be a part of a large system. An embedded system is a microcontroller or microprocessor based system which is designed to perform a specific task.
- For example, a fire alarm is an embedded system; it will sense only smoke.
- An embedded system has three components − It has hardware. It has application software. It has Real Time Operating system that supervises the application software and provide mechanism to let the processor run a process as per scheduling by following a plan to control the latencies.
- RTOS defines the way the system works. It sets the rules during the execution of application program.
- A small scale embedded system may not have RTOS.
- So we can define an embedded system as a Microcontroller based, software driven, and reliable, real-time control system.
- An embedded system is designed to do a specific job only. Example: a washing machine can only wash clothes, an air conditioner can control the temperature in the room in which it is placed.
- The hardware & mechanical components will consist all the physically visible things that are used for input, output, etc.
- An embedded system will always have a chip (either microprocessor or microcontroller) that has the code or software which drives the system.

An embedded system is a combination of 3 things:

- Hardware
- Software
- Mechanical Components

Embedded computing systems have to provide sophisticated functionality:

■ ***Complex algorithms:*** The operations performed by the microprocessor may be very sophisticated. For example, the microprocessor that controls an automobile engine must perform complicated filtering functions to optimize the performance of the car while minimizing pollution and fuel utilization.

■ ***User interface:*** Microprocessors are frequently used to control complex user interfaces that may include multiple menus and many options. The moving maps in Global Positioning System (GPS) navigation are good examples of sophisticated user interfaces.

To make things more difficult, embedded computing operations must often be performed to meet deadlines:

■ ***Real time:*** Many embedded computing systems have to perform in real time—if the data is not ready by a certain deadline, the system breaks. In some cases, failure to meet a deadline is unsafe and can even endanger lives. In other cases, missing a deadline does not create safety problems but does create unhappy customers—missed deadlines in printers ,for example, can result in scrambled pages.

■ ***Multirate:*** Not only must operations be completed by deadlines, but many embedded computing systems have several real-time activities going on at the same time. They may simultaneously control some operations that run at slow rates and others that run at high rates. Multimedia applications are prime examples of **multirate** behavior. The audio and video portions of a multimedia stream run at very different rates, but they must remain closely synchronized. Failure to meet a deadline on either the audio or video portions spoils the perception of the entire presentation.

Costs of various sorts are also very important:

■ ***Manufacturing cost:*** The total cost of building the system is very important in many cases. Manufacturing cost is determined by many factors, including the type of microprocessor used, the amount of memory required, and the types of I/O devices.

■ ***Power and energy:*** Power consumption directly affects the cost of the hardware, since a larger power supply may be necessary. Energy consumption affects battery life, which is important in many applications, as well as heat consumption, which can be important even in desktop applications.

CHARACTERISTICS OF AN EMBEDDED SYSTEM

- **Single-functioned** − an embedded system usually performs a specialized operation and does the same repeatedly. For example: A pager always functions as a pager.

- **Tightly constrained** − All computing systems have constraints on design metrics, but those on an embedded system can be especially tight. Design metrics is a measure of an implementation's features such as its cost, size, power, and performance. It must be of a size to fit on a single chip, must perform fast enough to process data in real time and consume minimum power to extend battery life.
- **Reactive and Real time** − Many embedded systems must continually react to changes in the system's environment and must compute certain results in real time without any delay. Consider an example of a car cruise controller; it continually monitors and reacts to speed and brake sensors. It must compute acceleration or de-accelerations repeatedly within a limited time; a delayed computation can result in failure to control of the car.
- **Microprocessors based** − It must be microprocessor or microcontroller based.
- **Memory** − It must have a memory, as its software usually embeds in ROM. It does not need any secondary memories in the computer.
- **Connected** − It must have connected peripherals to connect input and output devices.
- **HW-SW systems** − Software is used for more features and flexibility. Hardware is used for performance and security.

APPLICATION OF EMBEDDED SYSTEM

The application areas and the products in the embedded domain are countless.

1. Consumer Electronics: Camcorders, Cameras.

2. Household appliances: Washing machine, Refrigerator.

3. Automotive industry: Anti-lock breaking system (ABS), engine control.

4. Home automation & security systems: Air conditioners, sprinklers, fire alarms.

5. Telecom: Cellular phones, telephone switches.

6. Computer peripherals: Printers, scanners.

7. Computer networking systems: Network routers and switches.

8. Healthcare: EEG, ECG machines.

9. Banking & Retail: Automatic teller machines, point of sales.

10. Card Readers: Barcode, smart card readers.

CHALLENGES IN EMBEDDED COMPUTING SYSTEM DESIGN

- ***How much hardware do we need?***
- Control over the amount of computing power we apply to our problem.
- Select the type of microprocessor ,amount of memory and the peripheral devices
- To meet both performance deadlines and manufacturing cost constraints, the choice of hardware is important
- Too little hardware and the system fail to meet its deadlines, too much hardware and it becomes too expensive.

- ***How do we meet deadlines?***
- The brute force way of meeting a deadline is to speed up the hardware so that the program runs faster, that makes the system more expensive.
- It is also entirely possible that increasing the CPU clock rate may not make enough difference to execution time, since the program's speed may be limited by the memory system.

- ***How do we minimize power consumption?***
- In battery-powered applications, power consumption is extremely important. Even in non-battery applications, excessive power consumption can increase heat dissipation.
- One way to make a digital system consume less power is to make it run more slowly, but naively slowing down the system can obviously lead to missed deadlines.
- Careful design is required to slow down the noncritical parts of the machine for power consumption while still meeting necessary performance goals.

- ***How do we design for upgradability?***
- The hardware platform may be used over several product generations or for several different versions of a product in the same generation, with few or no changes.
- We want to be able to add features by changing software.

- ***Does it really work?***
- Reliability is always important when selling products
- Reliability is especially important in some applications, such as safety-critical systems.

Let's consider some ways in which the nature of embedded computing machines makes their design more difficult.

■ ***Complex testing:*** Exercising an embedded system is generally more difficult than typing in some data. We may have to run a real machine in order to generate the proper data. The timing of data is often important, meaning that we cannot separate the testing of an embedded computer from the machine in which it is embedded.

■ ***Limited observability and controllability:*** Embedded computing systems usually do not come with keyboards and screens. This makes it more difficult to see what is going on and to affect the system's operation. We may be forced to watch the values of electrical signals on the microprocessor bus, for example, to know what is going on inside the system. Moreover, in real-time applications we may not be able to easily stop the system to see what is going on inside.

■ ***Restricted development environments:*** The development environments for embedded systems (the tools used to develop software and hardware) are often much more limited than those available for PCs and workstations. We generally compile code on one type of machine, such as a PC, and download it onto the embedded system. To debug the code, we must usually rely on programs that run on the PC or workstation and then look inside the embedded system.

PERFORMANCE IN EMBEDDED COMPUTING

- Embedded system designers have a very clear performance goal in mind—their program must meet its ***deadline***. The heart of embedded computing is ***real-time computing***

- The program receives its input data; the deadline is the time at which a computation must be finished. If the program does not produce the required output by the deadline, then the program does not work, even if the output that it eventually produces is functionally correct. This notion of deadline-driven programming is at once simple and demanding.

- We need tools to help us analyze the real-time performance of embedded systems; we also need to adopt programming disciplines and styles that make it possible to analyze these programs.

- In order to understand the real-time behavior of an embedded computing system, we have to analyze the system at several different levels of abstraction. Those layers include:

- ***CPU:*** The CPU clearly influences the behavior of the program, particularly when the CPU is a pipelined processor with a cache.
- ***Platform:*** The platform includes the bus and I/O devices. The platform components that surround the CPU are responsible for feeding the CPU and can dramatically affect its performance.
- ***Program:*** Programs are very large and the CPU sees only a small window of the program at a time. We must consider the structure of the entire program to determine its overall behavior.
- ***Task:*** We generally run several programs simultaneously on a CPU, creating a ***multitasking system***. The tasks interact with each other in ways that have profound implications for performance.

■ **manufacturing cost**

■ **performance (both overall speed and deadlines)**

■ **power consumption.**

We must also consider the tasks we need to perform at every step in the design process. At each step in the design, we add detail:

■ We must ***analyze*** the design at each step to determine how we can meet the specifications.

■ We must then ***refine*** the design to add detail.

■ And we must verify the design to ensure that it still meets all system goals, such as cost, speed, and so on.

Specification

- ➢ The specification is more precise—it serves as the contract between the customer and the architects.
- ➢ The specification must be carefully written so that it accurately reflects the customer's requirements and does so in a way that can be clearly followed during design.
- ➢ The specification should be understandable enough so that someone can verify that it meets system requirements and overall expectations of the customer.
- ➢ It should also be unambiguous enough that designers know what they need to build.
- ➢ If global characteristics of the specification are wrong or incomplete, the overall system architecture derived from the specification may be inadequate to meet the needs of implementation.

Architecture design of embedded system

- ➢ The specification does not say how the system does things, only what the system does.
- ➢ Describing how the system implements those functions is the purpose of the architecture.
- ➢ The architecture is a plan for the overall structure of the system that will be used later to design the components that make up the architecture.
- ➢ The creation of the architecture is the first phase of what many designers think of as design.

To understand what an architectural description is, let's look at sample architecture for the moving map. The moving map is a handheld device that displays for the user a map of the terrain around the user's current position; the map display changes as the user and the map device change position. The moving map obtains its position from the GPS, a satellite-based navigation system. Following figure shows sample system architecture in the form of a ***block diagram*** that shows major operations and data flows among them.