UNIT-I

Embedded Computing & The embedded system design process

1. Define Embedded Computing System.

It is any device that includes a programmable computer but is not itself intended to be a general-purpose computer.

2. List out the uses of microprocessors in automobiles & household appliances.

Automobiles:

- Automobiles, trains and planes also use microprocessor technology.
- Consumer vehicles-buses, cars, trucks -integrate microprocessors to communicate important information throughout the vehicle. E.g., navigation systems provide information using microprocessors and global positioning system (GPS) technology.

Household appliances:

- The **programmable thermostat** allows the control of temperature at homes. In this system, a microprocessor works with the temperature sensor to determine and adjust the temperature accordingly.
- High-end coffee makers, washing machines, and radio clocks contain microprocessor technology.
- Some other home items that contain microprocessors are: microwaves, toasters, televisions, VCRs, DVD players, ovens, stoves, clothes washers, stereo systems, home computers, alarm clocks, hand-held game devices, thermostats, video game systems, bread machines, dishwashers, home lighting systems and even some refrigerators with digital temperature control.

3. Justify the use of programmable CPU rather than a hardwired unit.

A programmable CPU was used rather than a hardwired unit for two reasons: First, it made the system easier to design and debug; and second, it allowed the possibility of upgrades and using the CPU for other purposes.

4. List & explain the characteristics of Embedded Computing Applications.

- ***Complex algorithms:*** The operations performed by the microprocessor may be very sophisticated.
- ***User interface:*** Microprocessors are frequently used to control complex user interfaces that may include multiple menus and many options.
- ***Real time:*** Many embedded computing systems have to perform in real time—if the data isn't ready by a certain deadline, the system breaks.
- ***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.

- ***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, because a larger power supply may be necessary.

5. List & explain in brief the role of Microprocessors in embedded systems design.

- ***Flexible:*** Use of predesigned instruction for set-faster execution.
- ***Efficient:*** Several instructions can be executed per cycle in high-performance processors.
- ***Highly optimized:*** Microprocessors are very efficient utilizers of logic; and can be used for many different algorithms simply by changing the program it executes.
- ***Programmability:*** somewhat depend on hardware that is number of registers, capacity etc., according to that they have to write programs.
- ***Real time:*** performance is often best achieved my microprocessors.
- ***Low power and low cost:*** Personal computers are designed to satisfy a broad mix of computing requirements and to be very flexible. Those features increase the complexity and price of the components.

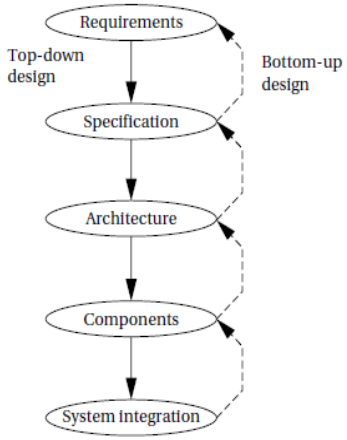
6. List & explain the challenges in Embedded computing System Design.

- ***Amount of hardware:*** We select the type of microprocessor used, we can select the amount of memory, the peripheral devices, and more.
- ***Meeting Deadlines:*** By brute force i.e., speeding the hardware, increasing CPU clock rate.
- ***Minimizing power consumption:*** Design is required to slow down the noncritical parts of the machine for power consumption while still meeting necessary performance goals.
- ***Design for upgradeability:*** We should to be able to add features by changing software.
- ***Reliability:*** Products are supposed to function properly, such as safety-critical systems.
- ***Complex Testing:*** To test embedded systems, real time data should be gathered.
- ***Limited observability and controllability:*** E.C. Systems do not come with keyboards and screens. This makes it more difficult to see what is going on inside the system.
- ***Restricted development environments:*** Compile and run can't be on the same machine. Compile code on one type of machine such as a PC, and download it into the embedded system.

7. Write a note on performance of Embedded computing Systems.

Embedded system designers, have a very clear performance goal in mind—their program must meet its deadline. At the heart of embedded computing is real-time computing, which is the science and art of programming to deadlines. The program receives its input data; the deadline is the time at which a computation must be finished. If the program doesn't produce the required output by the deadline, then the program doesn't work, even if the output that it eventually produces is functionally correct. 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.

8. With a neat diagram explain the embedded system design process.



- In this top-down view, we start with the system requirements.
- In the next step, specification, we create a more detailed description of what we want. But the specification states only how the system behaves, not how it is built.
- The details of the system's internals begin to take shape when we develop the architecture, which gives the system structure in terms of large components.
- Once we know the components we need, we can design those components, including both software modules and any specialized hardware we need.
- Based on those components, we can finally build a complete system.
- Components section will consider design from the top down—we will begin with the most abstract description of the system and conclude with concrete details.
- The alternative is a bottom-up view in which we start with components to build a system.
- Bottom-up design steps are shown in the figure as dashed-line arrows. We need bottom-up design because we do not have perfect insight into how later stages of the design process will turn out.
- Decisions at one stage of design are based upon estimates of what will happen later.

9. Distinguish between requirements & specification.

Separating out requirements analysis and specification is often necessary because of the large gap between what the customers can describe about the system they want and what the architects need to design the system. Consumers of embedded systems are usually not themselves embedded system designers or even product designers. Their understanding of the system is based on how they envision users' interactions with the system. They may have unrealistic expectations as to what can be done within their budgets, and they may also express their desires in a language very different from system architects' jargon. Capturing a consistent set of requirements from the customer and then massaging those requirements into a more formal specification is a structured way to manage the process of translating from the consumer's language to the designers. Requirements may be functional or nonfunctional. We must of course capture the basic functions of the embedded system, but functional description is often not sufficient.

10. Write a note on non-functional requirements of Embedded computing System Design.

- ***Performance:*** The speed of the system is often a major consideration both for the usability of the system and for its ultimate cost. As we have noted, performance may be a combination of soft performance metrics such as approximate time to perform a user-level function and hard deadlines by which a particular operation must be completed.
- ***Cost:*** The target cost or purchase price for the system is almost always a consideration. Cost typically has two major components: manufacturing cost includes the cost of components and

assembly; nonrecurring engineering (NRE) costs include the personnel and other costs of designing the system.

- ***Physical size and weight:*** The physical aspects of the final system can vary greatly depending upon the application. An industrial control system for an assembly line may be designed to fit into a standard-size rack with no strict limitations on weight. A handheld device typically has tight requirements on both size and weight that can ripple through the entire system design.
- ***Power consumption:*** Power, of course, is important in battery-powered systems and is often important in other applications as well. Power can be specified in the requirements stage in terms of battery life—the customer is unlikely to be able to describe the allowable wattage.

11. Explain the simple requirements form.

Name	GPS moving map
Purpose	
Inputs	
Outputs	
Functions	
Performance	
Manufacturing cost	
Power	
Physical size and weight	

Figure shows a sample requirements form that can be filled out at the start of the project. We can use the form as a checklist in considering the basic characteristics of the system. Let's consider the entries in the form:

- ***Name:*** Giving a name to the project not only simplifies talking about it to other.
- ***Purpose:*** This should be a brief one-or two-line description of what the system is supposed to do.
- ***Inputs and outputs:*** The inputs and outputs to the system encompass a wealth of detail.
- ***Performance:*** In most of these cases, the computations must be performed within a certain time frame. It is essential that the performance requirements be identified early to ensure that the system works properly.
- ***Manufacturing cost:*** This includes primarily the cost of the hardware components. Cost has a substantial influence on architecture.
- ***Power:*** Rough idea of how much power the system can consume. Whether the machine will be battery powered or plugged into the wall. Battery-powered machines must be much more careful about how they spend energy.
- ***Physical size and weight:*** Indication of the physical size of the system to help guide certain architectural decisions.

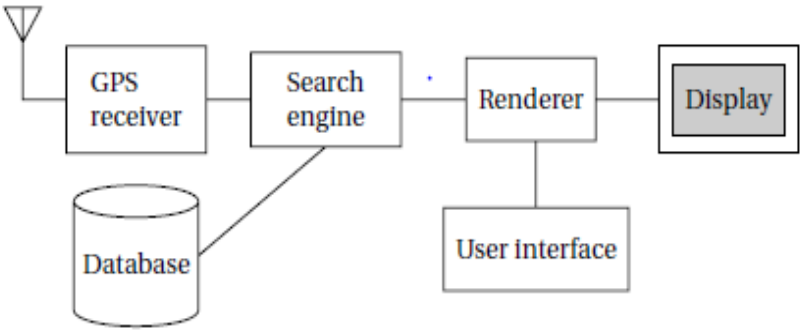
12. Explain the requirement chart for any embedded product of your choice.

GPS moving map:

- ***Functionality:*** This system is designed for highway driving and similar uses, not nautical or aviation uses that require more specialized databases and functions. The system should show major roads and other landmarks available in standard topographic databases.
- ***User interface:*** The screen should have at least 400 × 600-pixel resolution. The device should be controlled by no more than three buttons. A menu system should pop up on the screen when buttons are pressed to allow the user to make selections to control the system.
- ***Performance:*** The map should scroll smoothly. Upon power-up, a display should take no more than one second to appear, and the system should be able to verify its position and display the current map within 15 seconds.
- ***Cost:*** The selling cost (street price) of the unit should be no more than $120.
- Physical size and weight: The device should fit comfortably in the palm of the hand.
- ***Power consumption:*** The device should run for at least eight hours on four AA batteries, with at least 30 minutes of those eight hours comprising operation with the screen on.

Name	GPS moving map
Purpose	Consumer-grade moving map for driving use
Inputs	Power button, two control buttons
Outputs	Back-lit LCD display 400 × 600
Functions	Uses 5-receiver GPS system; three user-selectable resolutions; always displays current latitude and longitude
Performance	Updates screen within 0.25 seconds upon movement
Manufacturing cost	$40
Power	100 mW
Physical size and weight	No more than 2″ × 6″, 12 ounces

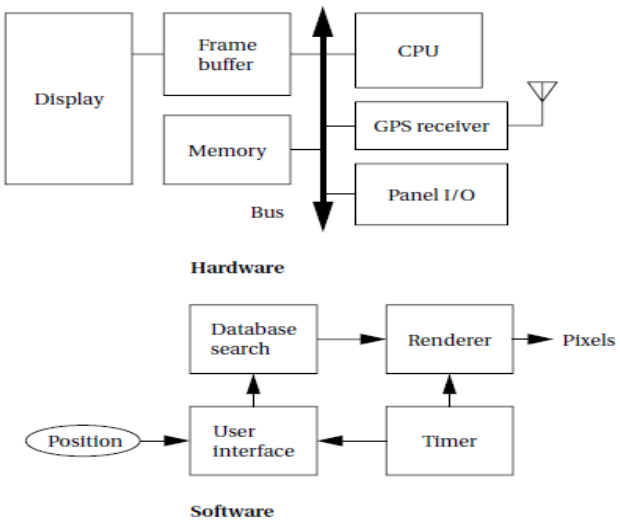
13. Explain the architecture design for GPS moving map.



Block diagram for GPS moving map

- Figure shows a sample system architecture in the form of a block diagram that shows major operations and data flows among them.
- This block diagram is still quite abstract—we haven't yet specified which operations will be performed by software running on a CPU, what will be done by special purpose hardware, and so on.
- The diagram does, however, go a long way toward describing how to implement the functions described in the specification.
- We clearly see, for example, that we need to search the topographic database and to render (i.e., draw) the results for the display.
- We have chosen to separate those functions so that we can potentially do them in parallel—performing rendering separately from searching the database may help us update the screen more fluidly

14. Hardware & software design architecture for GPS moving map.



Hardware and software architectures for the moving map

- The hardware block diagram clearly shows that we have one central CPU surrounded by memory and I/O devices.
- We have chosen to use two memories: a frame buffer for the pixels to be displayed and a separate program/data memory for general use by the CPU.
- The software block diagram fairly closely follows the system block diagram, but we have added a timer to control when we read the buttons on the user interface and render data onto the screen.
- To have a truly complete architectural description, we require more detail, such as where units in the software block diagram will be executed in the hardware block diagram and when operations will be performed in time.

15. What is system integration? Explain the system integration process.

System integration is the combination of systems to build a larger system or set of systems.

- After the components are built we have the satisfaction of putting them together and seeing a working system.
- Bugs are typically found during system integration, and good planning can help us find the bugs quickly.
- By building up the system in phases and running properly chosen tests, we can often find bugs more easily. If we debug only a few modules at a time, we are more likely to uncover the simple bugs and be able to easily recognize them.
- By fixing the simple bugs early will we be able to uncover the more complex or obscure bugs that can be identified only by giving the system a hard workout.
- We need to ensure during the architectural and component design phases that we make it as easy as possible to assemble the system in phases and test functions relatively independently.
- Determining why things don’t work correctly and how they can be fixed.