Dr.Y.NARASIMHA MURTHY Ph.D
yayavaram@yahoo.com

UNIT -I. Introduction to Embedded systems

INTRODUCTION : An embedded system is an electronic system,which includes a single chip microcomputers(Microcontrollers) like the ARM or Cortex or Stellaris LM3S1968 .It is configured to perform a specific dedicated application .Software is programmed into the on chip ROM of the single chip computer. This software is not accessible to the user , and software solves only a limited range of problems .Here the microcomputer is embedded or hidden inside the system. Every **embedded microcomputer** system , accepts inputs, performs computations, and generates outputs and runs in "real time."

For Example a typical automobile now a days contains an average of ten microcontrollers. In fact, modern houses may contain as many as 150 microcontrollers and on average a consumer now interacts with microcontrollers up to 300 times a day. General areas that employ embedded systems covers every branch of day to day science and technology, namely Communications, automotive, military, medical, consumer, machine control etc...

Ex: Cell phone , Digital camera , Microwave Oven,MP3 player, Portable digital assistant & automobile antilock brake system etc.

Characteristics of an Embedded System : The important characteristics of an embedded system are

- Speed (bytes/sec) : Should be high speed
- Power (watts) : Low power dissipation
- Size and weight : As far as possible small in size and low weight
- Accuracy (% error) : Must be very accurate
- Adaptability : High adaptability and accessibility.
- Reliability : Must be reliable over a long period of time.

So,an embedded system must perform the operations at a high speed so that it can be readily used for real time applications and its power consumption must be very low and the size of the system should be as for as possible small and the readings must be accurate with minimum error.The system must be easily adaptable for different situations.

Dr.Y.NARASIMHA MURTHY Ph.D
yayavaram@yahoo.com

CATEGORIES OF EMBEDDED SYSTEMS : Embedded systems can be classified into the following 4 categories based on their functional and performance requirements.

- Stand-alone embedded systems
- Real-time embedded systems -- Hard real-time systems & Soft real-time system
- Networked embedded systems and
- Mobile Embedded systems.

Based on the performance of the Microcontroller they are also classified into (i) Small scaled embedded system (ii) Medium scaled embedded system and (iii) Large scaled embedded system.

Stand alone Embedded systems : A stand-alone embedded system works by itself. It is a self-contained device which do not require any host system like a computer.. It takes either digital or analog inputs from its input ports, calibrates, converts, and processes the data, and outputs the resulting data to its attached output device, which either displays data, or controls and drives the attached devices. Temperature measurement systems, Video game consoles , MP3 players, digital cameras, and microwave ovens are the examples for this category.

Real-time embedded systems : An embedded system which gives the required output in a specified time or which strictly follows the time dead lines for completion of a task is known as a Real time system. i.e a Real Time system , in addition to functional correctness, also satisfies the time constraints .

There are two types of Real time systems.(i) Soft real time system and (ii) Hard real time system.

Soft Real-Time system : A Real time system in which ,the violation of time constraints will cause only the degraded quality, but the system can continue to operate is known as a Soft real time system. In soft real-time systems, the design focus is to offer a guaranteed bandwidth to each real-time task and to distribute the resources to the tasks.

Ex: A Microwave Oven , washing machine ,TV remote etc.

Hard Real-Time system : A Real time system in which ,the violation of time constraints will cause critical failure and loss of life or property damage or catastrophe is known as a Hard Real time system.

These systems usually interact directly with physical hardware instead of through a human being .The hardware and software of hard real-time systems must allow a worst case execution (WCET) analysis that guarantees the execution be completed within a strict deadline. The chip selection and RTOS selection become important factors for hard real-time system design.

Ex: Deadline in a missile control embedded system , Delayed alarm during a Gas leakage , car airbag control system , A delayed response in pacemakers ,Failure in RADAR functioning etc.

Networked embedded systems : The networked embedded systems are related to a network with network interfaces to access the resources. The connected network can be a Local Area Network (LAN) or a Wide Area Network (WAN), or the Internet. The connection can be either wired or wireless.

The networked embedded system is the fastest growing area in embedded systems applications. The embedded web server is such a system where all embedded devices are connected to a web server and can be accessed and controlled by any web browser.

Ex: A home security system is an example of a LAN networked embedded system where all sensors (e.g. motion detectors, light sensors, or smoke sensors) are wired and running on the TCP/IP protocol.

Mobile Embedded systems : The portable embedded devices like mobile and cellular phones , digital camaras,MP3 players, PDA (Personal Digital Assistants) are the example for mobile embedded systems. The basic limitation of these devices is the limitation of memory and other resources.

Small scaled embedded system : An embedded system supported by a single 8–16 bit Microcontroller with on-chip RAM and ROM designed to perform simple tasks is a Small scale embedded system.

Medium scaled embedded system : An embedded system supported by 16–32 bit Microcontroller /Microprocessor with external RAM and ROM that can perform more complex operations is a Medium scale embedded system.

Large scaled embedded system: An embedded system supported by 32-64 bit multiple chips which can perform distributed jobs is considered as a Large scale embedded system.

Application Areas of Embedded Systems: The embedded systems have a huge variety of application domains which varies from very low cost to very high cost and from daily life consumer electronics to industry automation equipments, from entertainment devices to

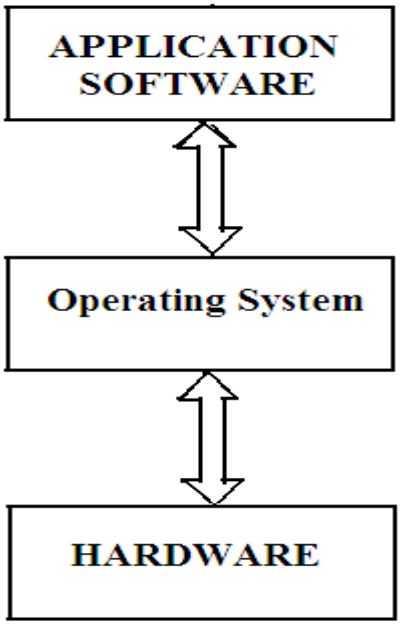
academic equipments, and from medical instruments to aerospace and weapon control systems. So,the Embedded systems span all aspects of our modern life. The following table gives the various applications of embedded systems.

S.No	Embedded System	Application
1	Home Appliances	Dishwasher, washing machine, microwave, Top-set box, security system , HVAC system, DVD, answering machine, garden sprinkler systems etc..
2	Office Automation	Fax, copy machine, smart phone system, modern, scanner, printers.
3	Security	Face recognition, finger recognition, eye recognition, building security system , airport security system, alarm system.
4	Academia	Smart board, smart room, OCR, calculator, smart cord.
5	Instrumentation	Signal generator, signal processor, power supplier,Process instrumentation,
6	Telecommunication	Router, hub, cellular phone, IP phone, web camera
7	Automobile	Fuel injection controller, anti-locking brake system, air-bag system, GPS, cruise control.
8	Entertainment	MP3, video game, Mind Storm, smart toy.
9	Aerospace	Navigation system, automatic landing system, flight attitude controller, space explorer, space robotics.
10	Industrial automation	Assembly line, data collection system, monitoring systems on pressure, voltage, current, temperature, hazard detecting system, industrial robot.
11	Personal	PDA, iPhone, palmtop, data organizer.
12	Medical	CT scanner, ECG , EEG , EMG ,MRI, Glucose monitor, blood pressure monitor, medical diagnostic device.
13	Banking & Finance	ATM, smart vendor machine, cash register ,Share market
14	Miscellaneous:	Elevators, tread mill, smart card, security door etc.

Overview of embedded systems architecture:

Every embedded system consists of customer-built hardware components supported by a Central Processing Unit (CPU), which is the heart of a microprocessor (µP) or microcontroller (µC). A microcontroller is an integrated chip which comes with built-in memory, I/O ports, timers, and other components. Most embedded systems are built on microcontrollers, which run faster than a custom-built system with a microprocessor, because all components are integrated

within a single chip. Operating system play an important role in most of the embedded systems. But all the embedded systems do not use the operating system. The systems with high end applications only use operating system. To use the operating system the embedded system should have large memory capability. So, This is not possible in low end applications like remote systems, digital cameras,MP3 players , robo toys etc.The architecture of an embedded system with OS can be denoted by layered structure as shown below. The OS will provide an interface between the hardware and application software.



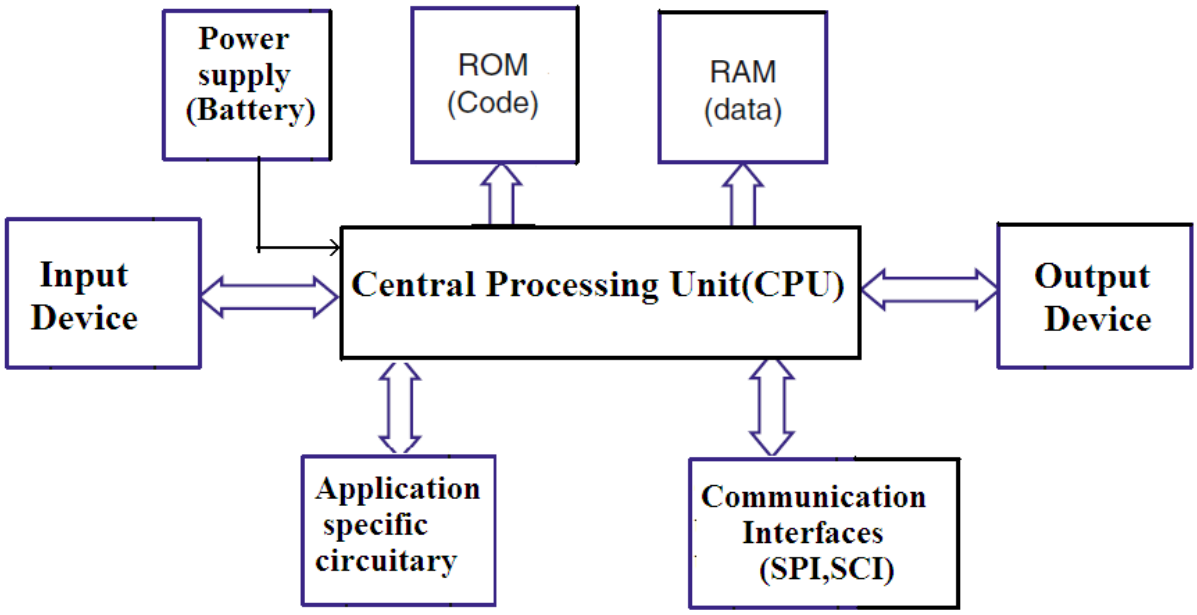
In the case of embedded systems with OS ,once the application software is loaded into memory it will run the application with out any host system.

Coming to the hardware details of the embedded system, it consists of the following important blocks.

- CPU(Central Processing Unit)
- RAM and ROM
- I/O Devices
- Communication Interfaces

- Sensors etc. (Application specific circuitary)

This hardware architecture can be shown by the following block diagram.



Central Processing Unit : A CPU is composed of an Arithmetic Logic Unit (ALU), a Control Unit (CU), and many internal registers that are connected by buses. The ALU performs all the mathematical operations (Add, Sub, Mul, Div), logical operations (AND, OR), and shifting operations within CPU .The timing and sequencing of all CPU operations are controlled by the CU, which is actually built of many selection circuits including latches and decoders .The CU is responsible for directing the flow of instruction and data within the CPU and continuously running program instructions step by step.

The CPU works in a cycle of fetching an instruction, decoding it, and executing it, known as the fetch-decode-execute cycle. The cycle begins when an instruction is fetched from a memory location pointed to by the PC to the IR via the data bus.

For embedded system design, many factors impact the CPU selection, e.g., the maximum size (number of bits) in a single operand for ALU (8, 16, 32, 64 bits), and CPU clock frequency for timing tick control, i.e. the number of ticks (clock cycles) per second in measures of MHz.

Memory : Embedded system memory can be either on-chip or off-chip. On chip memory access is much fast than off-chip memory, but the size of on-chip memory is much smaller than the size of off-chip memory. Usually, it takes at least two I/O ports as external address lines plus a few control lines such as R/W and ALE control lines to enable the extended memory. Generally the data is stored in RAM and the program is stored in ROM.

I/O Ports : The I/O ports are used to connect input and output devices. The common input devices for an embedded system include keypads, switches, buttons, knobs, and all kinds of sensors (light, temperature, pressure, etc).

The output devices include Light Emitting Diodes (LED), Liquid Crystal Displays (LCD), printers, alarms, actuators, etc. Some devices support both input and output, such as communication interfaces including Network Interface Cards (NIC), modems, and mobile phones.

Communication Interfaces : To transfer the data or to interact with other devices ,the embedded devices are provided the various communication interfaces like RS232, RS422, RS485 ,USB, SPI(Serial Peripheral Interface) ,SCI (Serial Communication Interface) ,Ethernet etc.

Application Specific Circuitry : The embedded system some times receives the input from a sensor or actuator. In such situations certain signal conditioning circuitry is needed. This hardware circuitry may contain ADC , Op-amps ,DAC etc. Such circuitry will interact with the embedded system to give correct output.

Power supply: Most of the embedded systems now days work on battery operated supplies. Because low power dissipation is always required. Hence the systems are designed to work with batteries.

Specialties of an Embedded Systems : An embedded system has certain specialties when compared to a normal computer system or a workstation or a mainframe computer system.

(i).Embedded systems are dedicated to specific tasks, whereas PCs are generic computing platforms.

(ii).Embedded systems are supported by a wide array of processors and processor architectures

(iii). Embedded systems are usually cost sensitive.

(iv). Embedded systems have real-time constraints.

(v).If an embedded system use an operating system , it is most likely using a real-time perating system (RTOS), but not Windows 9X, Windows NT, Windows 2000, Unix, Solaris, etc.

(vi). The implications of software failure is much more severe in embedded systems than in desktop systems.

(vii) Embedded systems often have power constraints.

(ix). Embedded systems must be able to operate under extreme environmental conditions.

(x). Embedded systems utilizes fewer system resources than desktop systems.

(xi). Embedded systems often store all their object code in ROM.

(xii). Embedded systems require specialized tools and methods to be efficiently designed when compared to desktop computers.

(xiii). Embedded microprocessors often have dedicated debugging circuitry.

(xiv).Embedded systems have Software Up gradation capability

(xv). Embedded systems have large User Interfaces for real time applications.

Recent trends in Embedded systems : With the fast developments in semiconductor industry and VLSI technology ,one can find tremendous changes in the embedded system design in terms of processor speed , power , communication interfaces including network capabilities and software developments like operating systems and programming languages etc.

Processor speed and Power : With the advancements in processor technology ,the embedded systems are now days designed with 16,32 bit processors which can work in real time environment. These processors are able to perform high speed signal processing activities which resulted in the development of high definition communication devices like 3G mobiles etc.Also the recent developments in VLSI technology has paved the way for low power battery operated devices which are very handy and have high longevity. Also , the present day embedded systems are provided with higher memory capabilities ,so that most of them are based on tiny operating systems like android etc.

Communication interfaces : Most of the present day embedded systems are aimed at internet based applications. So,the communication interfaces like Ethernet, USB, wireless LAN etc.have become very common resources in almost all the embedded systems. The developments in memory technologies also helped in porting the TCP/IP protocol stack and the HTTP server

software on to the embedded systems. Such embedded systems can provide a link between any two devices any where in the globe.

Operating systems : With recent software developments ,there is a considerable growth in the availability of operating systems for embedded systems. Mainly new operating systems are developed which can be used in real time applications. There are both commercial RTOSes like Vx Works , QNX,WIN-CE and open source RTOSes like RTLINUX etc. The Android OS in mobiles has revolutionized the embedded industry.

Programming Languages : There is also a remarkable development in the programming languages. Languages like C++, Java etc. are now widely used in embedded application programming. For example by having the Java virtual machine in a mobile phones ,one can download Java applets from a server and can be executed on your mobile.

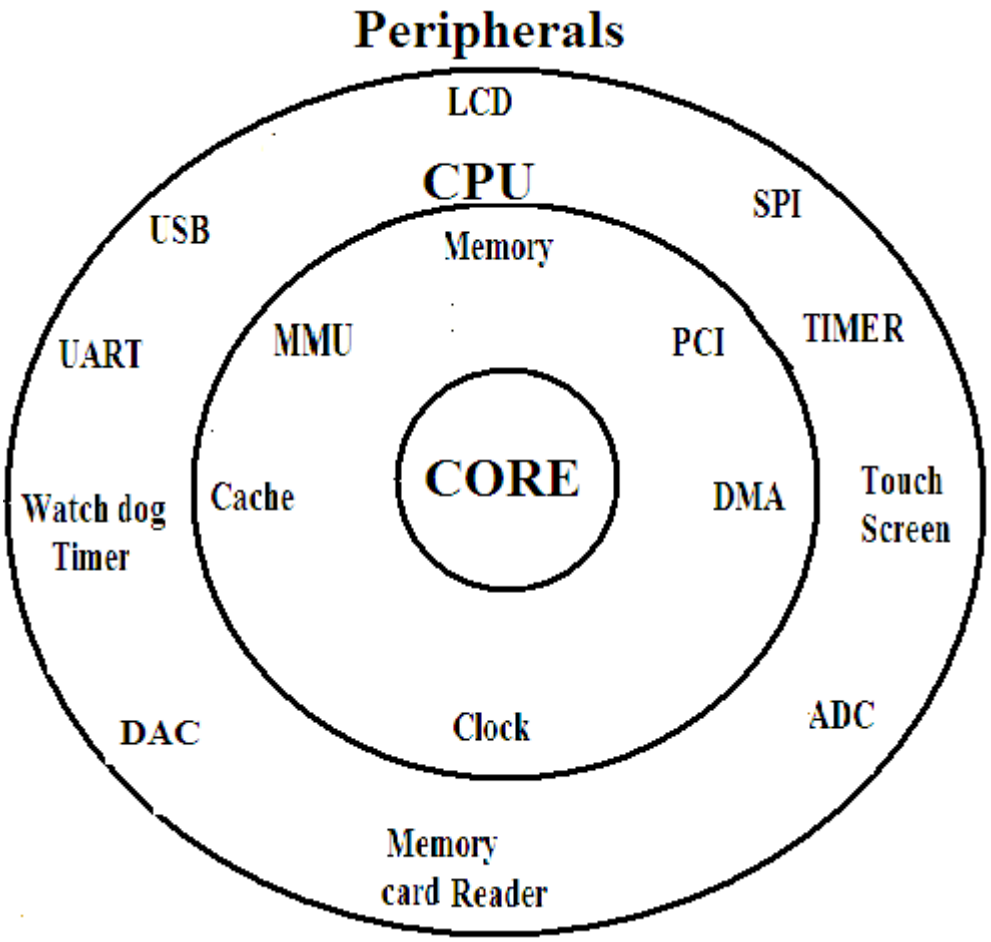
In addition to these developments, now a days we also find new devices like ASICs and FPGAs in the embedded system market. These new hardware devices are popular as programmable devices and reconfigurable devices.

Detailed Hardware architecture of an Embedded system : The hardware architecture of an embedded systems is very important, because it is one of the powerful tools that can be used to understand an embedded systems design or to resolve challenges faced while designing a new system. The hardware architecture of any embedded system consists of three sections namely : **Core, Central Processing Unit (CPU) and Peripherals.**

Core is the component which executes the instructions. CPU contains the core and the other components which support the core to execute programs. Peripherals are the components which communicate with other systems or physical world (Like ports, ADC,DAC, Watch dog Timers etc.). The core is separated from other components by the system bus.

The CPU in the embedded system may be a general purpose processor like a microcontroller or a special purpose processor like a DSP (Digital signal processor). But any CPU consists of of an Arithmetic Logic Unit (ALU), a Control Unit (CU), and many internal registers that are connected by buses. The ALU performs all the mathematical operations (Add, Sub, Mul, Div), logical operations (AND, OR), and shifting operations within CPU.

Dr.Y.NARASIMHA MURTHY Ph.D
yayavaram@yahoo.com



The timing and sequencing of all CPU operations are controlled by the CU, which is actually built of many selection circuits including latches and decoders .The CU is responsible for directing the flow of instruction and data within the CPU and continuously running program instructions step by step.

There are many internal registers in the CPU.

The accumulator (A) is a special data register that stores the result of ALU operations. It can also be used as an operand. The Program Counter (PC) stores the memory location of the next instruction to be executed. The Instruction Register (IR) stores the current machine instruction to be decoded and executed..

The Data Buffer Registers store the data received from the memory or the data to be sent to memory. The Data Buffer Registers are connected to the data bus. The Address Register stores the memory location of the data to be accessed (get or set). The Address Register is connected to the address bus.

In an embedded system, the CPU may never stop and run forever .The CPU works in a cycle of fetching an instruction, decoding it, and executing it, known as the fetch-decode-execute cycle. The cycle begins when an instruction is fetched from a memory location pointed to by the PC to the IR via the data bus.

The memory is divided into Data Memory and Code Memory. Most of data is stored in Random Access Memory (RAM) and code is stored in Read Only Memory (ROM). This is due to the RAM constraint of the embedded system and the memory organization .The RAM is readable and writable, faster access and more expensive volatile storage, which can be used to store either data or code. Once the power is turned off, all information stored in the RAM will be lost.The RAM chip can be SRAM (static) or DRAM (dynamic) depending on the manufacturer. SRAM is faster than DRAM, but is more expensive.

The ROM, EPROM, and Flash memory are all read-only type memories often used to store code in an embedded system. The embedded system code does not change after the code is loaded into memory. The ROM is programmed at the factory and can not be changed over time. The newer microcontrollers come with EPROM or Flash instead of ROM. Most microcontroller development kits come with EPROM as well. EPROM and Flash memory are easier to rewrite than ROM. EPROM is an Erasable Programmable ROM in which the contents can be field programmed bya special burner and can be erased by a UV light bulb. The size of EPROM ranges up to 32kb in most embedded systems. Flash memory is an Electrically EPROM which can be programmed from software so that the developers don't need to physically remove the EPROM from the circuit to re-program it. It is much quicker and easier to re-write Flash than other types of EPROM. When the power is on, the first instruction in ROM is loaded into the PC and then the CPU fetches the instruction from the location in the ROM pointed to by the PC and stores it in the IR to start the continuous CPU fetch and execution cycle. The PC is advanced to the address of the next instruction depending on the length of the current instruction or the destination of the Jump instruction.

The I/O ports are used to connect input and output devices. The common input devices for an embedded system include keypads, switches, buttons, knobs, and all kinds of sensors (light, temperature, pressure, etc). The output devices include Light Emitting Diodes (LED), Liquid Crystal Displays (LCD), printers, alarms, actuators , etc. Some devices support both input and

output, such as communication interfaces including Network Interface Cards (NIC), modems, and mobile phones.

Clock : The clock is used to control the clocking requirement of the CPU for executing instructions and the configuration of timers. For ex: the 8051 clock cycle is $(1/12)10^{-6}$ second (1/12μs) because the clock frequency is 12MHz. A simple 8051 instruction takes 12 cycles (1ms) to complete. Of course, some multi-cycle instructions take more clock cycles.

A timer is a real-time clock for real-time programming. Every timer comes with a counter which can be configured by programs to count the incoming pulses. When the counter overflows (resets to zero) it will fire a timeout interrupt that triggers predefined actions. Many time delays can be generated by timers. For example ,a timer counter configured to 24,000 will trigger the timeout signal in 24000x 1/12μs = 2ms.

In addition to time delay generation, the timer is also widely used in the real-time embedded system to schedule multiple tasks in multitasking programming. The watchdog timer is a special timing device that resets the system after a preset time delay in case of system anomaly. The watchdog starts up automatically after the system power up.

One need to reboot the PC now and then due to various faults caused by hardware or software. An embedded system cannot be rebooted manually, because it has been embedded into its system. That is why many microcontrollers come with an on-chip watchdog timer which can be configured just like the counter in the regular timer. After a system gets stuck (power supply voltage out of range or regular timer does not issue timeout after reaching zero count) the watchdog eventually will restart the system to bring the system back to a normal operational condition.

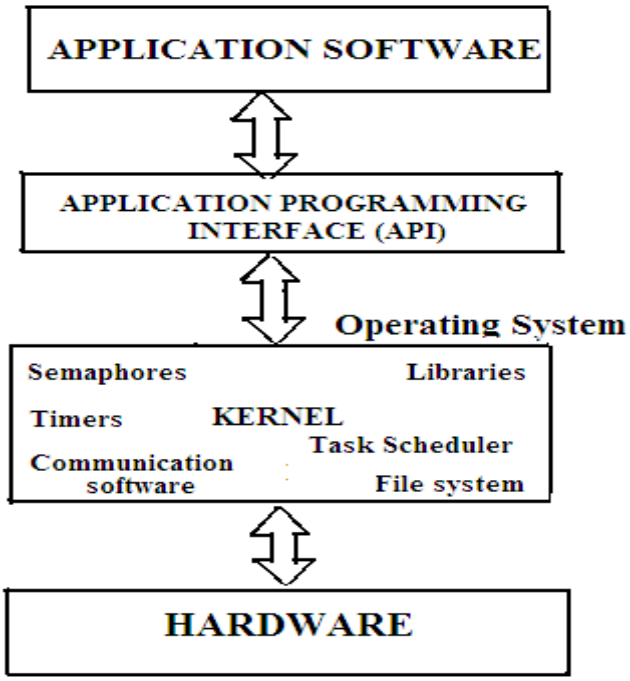
ADC & DAC :

Many embedded system application need to deal with non-digital external signals such as electronic voltage, music or voice, temperature, pressures, and many other signals in the analog form. The digital computer does not understand these data unless they are converted to digital formats. The ADC is responsible for converting analog values to binary digits. The DAC is responsible for outputting analog signals for automation controls such as DC motor or HVDC furnace control.

In addition to these peripherals, an embedded system may also have sensors, Display modules like LCD or Touch screen panels, Debug ports certain communication peripherals like I^2C, SPI,

Ethernet,CAN ,USB for high speed data transmission. Now a days various sensors are also becoming an important part in the design of real time embedded systems. Sensors like temperature sensors, light sensors , PIR sensors ,gas sensors are widely used in application specific circuitry.

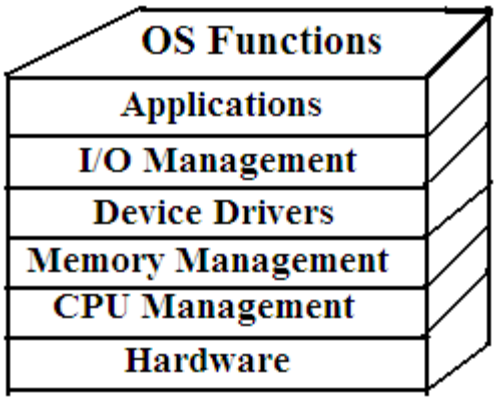
Software architecture: To design an efficient embedded system , both hardware and software aspects are equally important. The software of an embedded system is mainly aimed at accessing the hardware resources properly. The software of an embedded system means both operating system and application software. But every embedded system need not contain the operating system.For low end applications , operating system is not needed. In such cases the designer has to write the necessary software routines to access the hardware. The architecture of the software in an embedded system can be shown by the following figure.



The central part or nucleus of the operating system is the Kernel .A kernel connects the application software to the hardware of an embedded system. The other important components of the OS are Device manager, Communication software, Libraries and File system. The kernel will take care of task scheduling , priorities , memory management etc.It manages the tasks to

achieve the desired performance of the system . It schedules the tasks and provide inter process communication between different tasks.

The **device manager** manages the I/O devices through interrupts and device drivers. The device drivers provide the necessary interface between the application and the hardware. A device driver is a specific type of software developed to allow interaction with hardware devices. This constitutes an interface for communicating with the device, through the specific system bus or communications subsystem that the hardware is connected to, providing commands to receiving data from the device, and on the other end, the requisite interfaces to the operating system and software applications.



The communication software provides necessary protocols to make the embedded system network enabled. This software integrates the upper layer protocols such as TCP/IP stack with the operating system.

Application programming interface is used by the designer to write the application software.The API provides the function calls to access the operating system services.

Application Specific software : It sits above the O.S. The application software is developed according to the features of the development tools available in the OS.These development tools provide the function calls to access the services of the OS. These function calls include, creating a task ,to read the data from the port and write the data to the memory etc.

The various function calls provided by an operating system are

i.To create ,suspend and delete tasks.

ii. To do task scheduling to providing real time environment.

iii.To create inter task communication and achieve the synchronization between tasks.

iv.To access the I/O devices.

vi.To access the communication protocol stack .

The designer develops the application software based on these function calls.

Communication Software: To connect to the external world through internet ,the embedded system need a communication interface. The communication software include the Ethernet interface and the TCP/IP protocol suit .Now a days even small embedded systems like mobile phones ,PDAs are network enabled through these TCP/IP support. The TCP/IP protocol suite is shown in the diagram below.

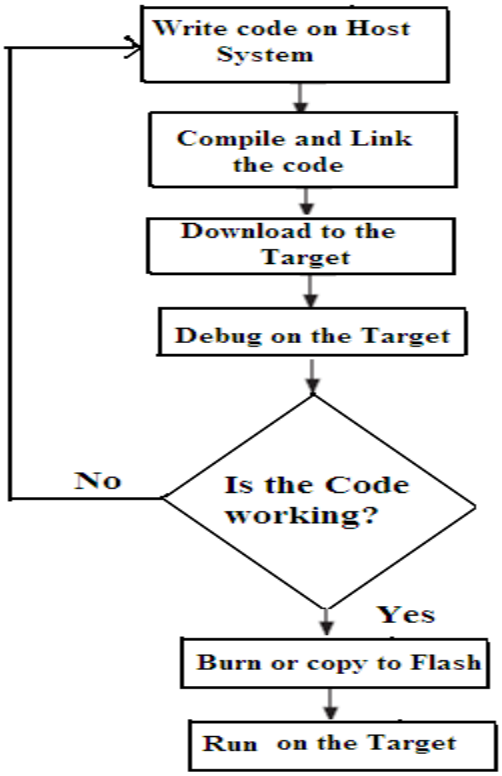
Application layer
Transport Layer TCP/UDP
IP Layer
Data Link Layer
Physical Layer

This suite consists of different layers like, application layer, Transport layer , IP layer etc.TCP means Transmission Control Protocol. It ensures that the data is delivered to the application layer without any errors. The UDP (User Datagram protocol) provides a connectionless service for error control and flow of data. This TCP/IP protocol suite helps to understand the working of communication software packages.

Cross platform development: Some times the host computer used for the development of application software may not be used to debug or compile the software.Then another system which contains all the of running development tools (editors, compilers, assemblers, debuggers etc.) may be used. This type of choosing another host sytem ,other than the original host system is known as Cross platform development. Some common differences between host and target machines are different operating system, different system boards or a different CPU.

A cross platform development environment allows you to maximize the use of all your resources. This can include everything from your workstations and servers to their disk space and cpu cycles.

Here host machine is the machine on which you write and compile programs.A target machine may be another general-purpose computer, a special-purpose device employing a single-board computer or any other intelligent device. Debugging is an important issue in cross-platform development. Since you are usually not able to execute the binary files on the host machine, they must be run on the target machine. The flow chart for the cross-platform development is shown below.



In this method first the source code is developed on the host computer system and this code is compiled and linked using the cross platform development tools..Then the code is downloaded on to the target and debugged on the target system. If the code is working properly it is burn into the EPROM or Flash ROM .Finally the code is run on the target system.If the code is not correct ,it is again sent to development stage where it is corrected.

Cross compilation tools are very important for successful product development. Selection of these tools should be made based upon the embedded system itself as well as features to test and debug software remotely.The necessary tools for the cross platform development are

- Cross- compiler
- Cross –assembler

- Cross-Linker
- Cross –debugger
- Cross-compiled libraries.

These components will enable to compile, link and debug code for the target environment through the cross-compilation environment .

Boot Sequence : Booting means starting the system. An embedded system can be booted in one of the following ways.

i).Execute from ROM using the RAM for Data.

ii).Execute from RAM after loading the image from RAM.

iii).Execute from RAM after downloading from the host.

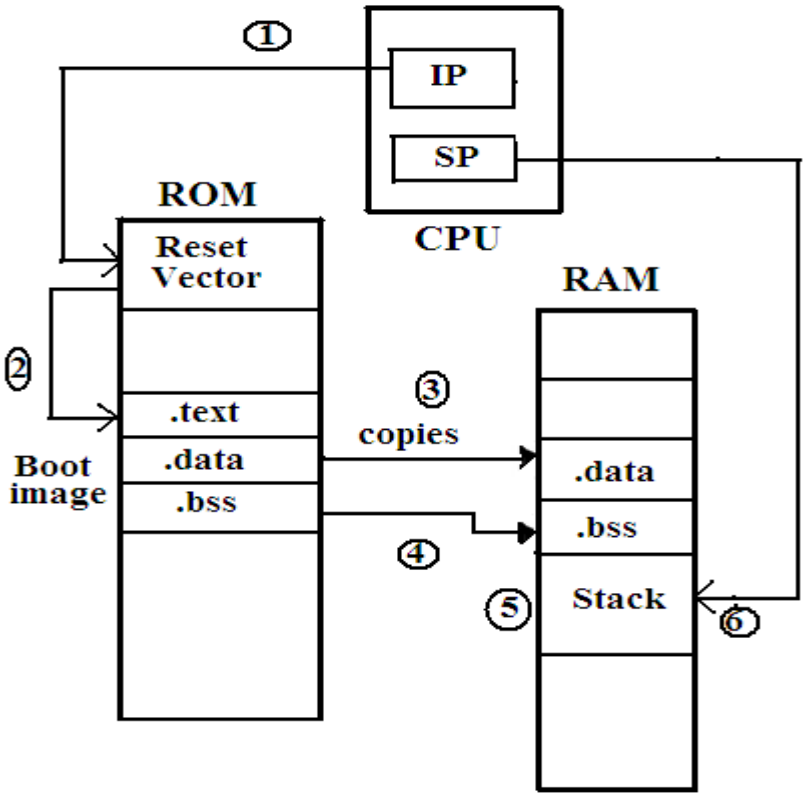
Normally booting from ROM is the fastest process.the process for executing from ROM using the RAM for data is shown in the figure below.

Executing from ROM Using RAM for Data :

Some embedded devices have limited memory resources that the program image executes directly out of the ROM. Sometimes the board vendor provides the boot ROM, and the code in the boot ROM does not copy instructions out to RAM for execution. In such cases, the data sections must still reside in RAM. Boot sequence for an image running from ROM is shown below figure.

The two registers of CPU the Instruction Pointer (IP) register and the Stack Pointer (SP) register are important. The IP points to the next instruction (code in the .text section) that the CPU must execute, while the SP points to the next free address in the stack. The stack is created from a space in RAM, and the system stack pointer registers must be set appropriately at start up.

Dr.Y.NARASIMHA MURTHY Ph.D
yayavaram@yahoo.com



The boot sequence for an image running from ROM is as follows :

i).The CPU's IP is hardwired to execute the first instruction in memory (the reset vector).

ii).The reset vector jumps to the first instruction of the .text section of the boot image. The .text section remains in ROM ; the CPU uses the IP to execute .text. This code is called boot strap code .This code initializes the memory system, including the RAM.

iii).The .data section of the boot image is copied into RAM because it is both readable and writeable.

iv).Space is reserved in RAM for the .bss section of the boot image because it is both readable and writeable. There is nothing to transfer because the content for the .bss section is empty.

v).Stack space is reserved in RAM.

vi).The CPU's SP register is set to point to the beginning of the newly created stack.

At this point, the boot completes. The CPU continues to execute the code in the .text section and initializes all the hardware and software components until it is complete or until the system is shut down.

Dr.Y.NARASIMHA MURTHY Ph.D
yayavaram@yahoo.com

Embedded system Development Tools : Basically the embedded tools are divided into two types(i).Hardware Development tools and (ii) Software Development tools.

Hardware development tools : Hardware tools for embedded development include development or evaluation boards for specific processors, like Friendly ARM's Mini2440, Pandaboard , Beagleboard and Craneboard etc..In addition to this various othes devices like Digital multimeters ,Logic Analyzers , Spectrum Analyzers and Digital CROs etc.are also required in embedded design.

The **digital multimeter** is used to measure voltages, currents and to check the continuity in the circuits in an embedded systems. Because the embedded system also contains some application specific circuitry which some times require debugging.

The **Logic analyzer** is used to check the timings of the signals ,and their correctness.

The **Spectrum analyzer** is helpful to to analyze the signals in the frequency domain.

The **digital CRO** helps to display the output waveforms and also to store a portion of the waveforms etc.

Software development tools /testing tools : The software development tools include the operating system development suite ,cross platform development tools, ROM emulator ,EPROM programming and In circuit Emulator (ICE) etc. The operating system development suite consists of API calls to access the OS services.This suite can run on either Windows or UNIX/Linux systems.

Under the cross platform tools,the compiler generates the object code for the source code developed in high level languages like C and C++ or Java etc.For LINUX systems a number of GNU tools are available.

The EPROM programmer is used to in circuit programming by burning the code in the memory of the target system.

The instruction set Simulator(ISS) software creates the virtual version of the processor on the PC.

Assembler and Compiler: The binary code obtained by translating an assembly language program using an assembler is smaller and runs faster than the binary code obtained by translating a high level language using a compiler since the assembly language gives the programmer complete control over the functioning of a processor. The advantage of using a high level language is that a program written in a high level language is easier to understand and

maintain than a program written in assembly language. Hence time critical applications are written in assembly language while complex applications are written in a high level language.

Cross compilation tools are very important for successful product development. Selection of these tools should be made based upon the embedded system itself as well as features to test and debug software remotely. The cross-platform development tools should be compatible with the host machine. Depending upon CPU family used for the target system, the toolset must be capable of generating code for the target machine. In the case of GNU development tools, we need to have a number of things to work together to generate executable code for the target. At least one of the following tools must be available on the machine.

•Cross compiler

•Cross assembler

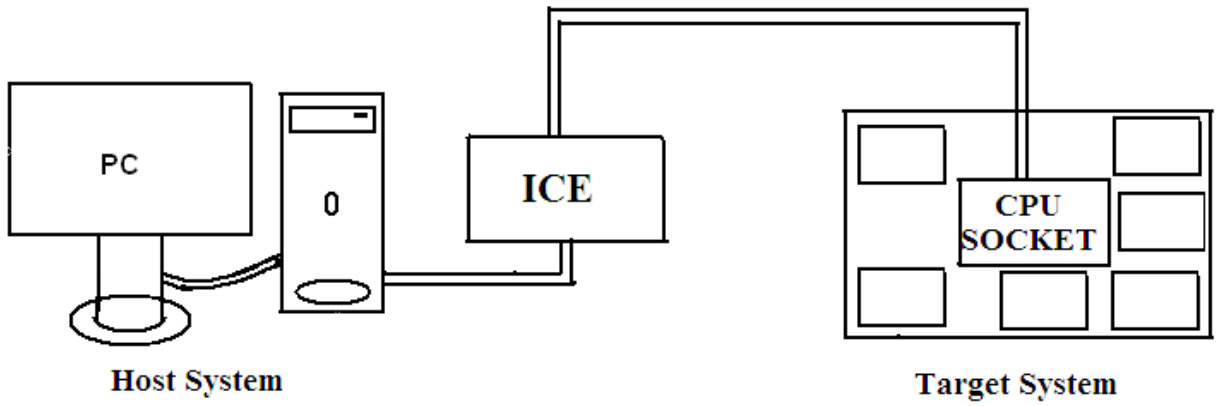
•Cross linker

•Cross debugger

•Cross-compiled libraries for the target host.

•Operating system-dependent libraries and header files for the target system

Simulator*:* A simulator is software tool that runs on the host and simulates the behavior of the target's processor and memory. The simulator knows the target processor's architecture and instruction set. The program to be tested is read by the simulator and as instructions are executed the simulator keeps track of the values of the target processor's registers and the target's memory. Simulators provide single step and breakpoint facilities to debug the program.



Emulator : Another important tool is the ICE(In-Circuit Emulator),which emulates the CPU. An emulator is a hardware tool that helps in testing and debugging the program on the target. The target's processor is removed from the circuit and the emulator is connected in its place. The emulator drives the signals in the circuit in the same way as the target's processor and hence the emulator appears to be the processor to all other components of the embedded system. Emulators also provide features such as single step and breakpoints to debug the program.

Software emulators are software tools that can emulate a particular CPU. Using a software emulator one can debug the code and find out CPU register values, stack pointers and other information without having a real CPU. Software emulators are useful when we don't have the real hardware available for testing and debugging and want to see how the CPU will behave when a program is run on it.

----------xx----------

Acknowledgment : Grateful to the following authors ,whose books helped in preparing this class notes.

References: 1.The Linux Development Platform Configuring, Using, and Maintaining a Complete Programming Environment by Rafeeq Ur Rehman

2. Embedded Software Development with C, Kai Qian •David den Haring •Li Cao,Springer.

3Embedded Real-time Systems- Dr.K.V.K.K Prasad