UNIT -I

Syllabus: Introduction to embedded systems - classification, purpose, applications, features and architecture considerations of embedded system – Memory (RAM & ROM), timers, clocks, address bus, data bus, Embedded Processor and their types, overview of design process of embedded systems, programming languages and tools for embedded design.

Introduction to embedded systems:

An embedded system is a combination of computer hardware and software, either fixed in capability or programmable, designed for a specific function or functions within a larger system. Industrial machines, agricultural and process industry devices, automobiles, medical equipment, cameras, household appliances, airplanes, vending machines and toys, as well as mobile devices, are possible locations for an embedded system.

"Embedded System is a combination of software and hardware which is used to perform specific task with often some real-time constraints"

Embedded systems are computing systems, but they can range from having no user interface (UI) -- for example, on devices in which the system is designed to perform a single task -- to complex graphical user interfaces (GUIs), such as in mobile devices. User interfaces can include buttons, LEDs, touchscreen sensing and more. Some systems use remote user interfaces as well.

History of embedded systems

Embedded systems date back to the 1960s. Charles Stark Draper developed an integrated circuit (IC) in 1961 to reduce the size and weight of the Apollo Guidance Computer, the digital system installed on the Apollo Command Module and Lunar Module. The first computer to use ICs, it helped astronauts collect real-time flight data.

In 1965, Autonetics, now a part of Boeing, developed the D-17B, the computer used in the Minuteman I missile guidance system. It is widely recognized as the first mass-produced embedded system. When the Minuteman II went into production in 1966, the D-17B was replaced with the NS-17 missile guidance system, known for its high-volume use of integrated

circuits. In 1968, the first embedded system for a vehicle was released; the Volkswagen 1600 used a microprocessor to control its electronic fuel injection system.
By the late 1960s and early 1970s, the price of integrated circuits dropped and usage surged. The first microcontroller was developed by Texas Instruments in 1971. The TMS 1000 series, which became commercially available in 1974, contained a 4-bit processor, read-only memory (ROM) and random-access memory (RAM), and cost around $2 apiece in bulk orders.

Also in 1971, Intel released what is widely recognized as the first commercially available processor, the 4004. The 4-bit microprocessor was designed for use in calculators and small electronics, though it required eternal memory and support chips. The 8-bit Intel 8008, released in 1972 had 16 KB of memory; the Intel 8080 followed in 1974 with 64 KB of memory. The 8080's successor, x86 series, was released in 1978 and is still largely in use today.

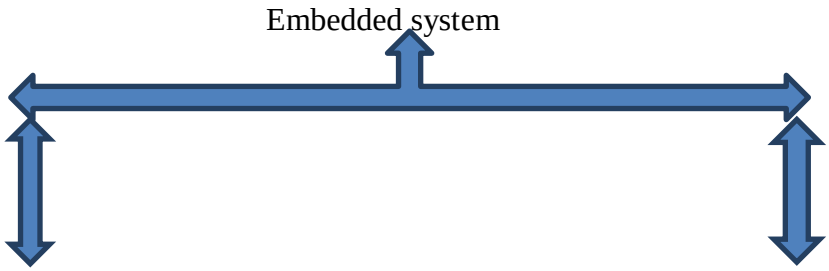
In 1987, the first embedded operating system, the real-time VxWorks, was released by Wind River, followed by Microsoft's Windows Embedded CE in 1996. By the late 1990s, the first embedded Linux products began to appear. Today, Linux is used in almost all embedded devices.

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.

Classification: Embedded systems can be classified into the following 4 categories based on their functionality and performance requirements.



Functionality	**performance**
• Stand alone embedded systems	small scale embedded system
• Real time embedded system a) Hard real time E.S b) Soft Real time E.S	medium scale embedded s/m
• Networked embedded system	large scale embedded system
• Mobile embedded system	

Standalone Embedded systems:

A stand-alone embedded system works by itself. It is a self-contained device which does 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.

EX: 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 deadlines 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 cameras, 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. 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.

Small scaled embedded system: These systems are designed with a single 8- or 16- bit microcontroller; they have little hardware and software complexities and involve board- level design. They may even be battery operated. When developing embedded software for these, an editor, assembler and cross assembler, specific to the microcontroller or processor used, are the main programming tools. Usually, C used for developing these systems. C program compilation is done into the assembly, and executable codes are then appropriately located in the system memory. The software has to fit within the memory available and keep in view the need to limit power dissipation when system is running continuously.

Medium scaled embedded system: These systems are usually designed with a single or few 16- or 32-bit microcontrollers or DSPs or Reduced Instruction Set Computers (RISCs). These have both hardware and software complexities. For complex software design, there are the following programming tools: RTOS, Source code engineering tool, Simulator, Debugger and Integrated Development Environment (IDE). Software tools also provide the solutions to the hardware complexities. An assembler is of little use as a programming tool. These systems may also employ the readily available ASSPs and IPs (explained later) for the various functions for example, for the bus interfacing, encrypting, deciphering, discrete cosine transformation and inverse transformation, TCP/IP protocol stacking and network connecting functions.

Large scaled embedded system: Sophisticated embedded systems have enormous hardware and software complexities and may need scalable processors or configurable processors and programmable logic arrays. They are used for cutting edge applications that need hardware and software co-design and integration in the final system; however, they are constrained by the processing speeds available in their hardware units. Certain software functions such as

encryption and deciphering algorithms, discrete cosine transformation and inverse transformation algorithms, TCP/IP protocol stacking and network driver functions are implemented in the hardware to obtain additional speeds by saving time. Some of the functions of the hardware resources in the system are also implemented by the software. Development tools for these systems may not be readily available at a reasonable cost or may not be available at all. In some cases, a compiler or retarget able compiler might have to be developed for these.

Purpose of Embedded Systems:

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

1. **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 term "data" refers all kinds of information, viz. text, voice, image, video, electrical signals and any other measurable quantities. Data can be either analog (continuous) or digital (discrete). Embedded systems with analog data capturing techniques collect data directly in the form of analog signals whereas 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. If the data is digital, it can be directly captured without any additional interface by digital embedded systems. The collected data may be stored directly in the system or may be transmitted to some other systems or it may be processed by the system or it may be deleted instantly after giving a meaningful representation. These actions are purely dependent on the purpose for which the embedded system is designed. Embedded systems designed for pure measurement applications without storage, used in control and instrumentation domain, collects data and gives a meaningful representation of the collected data by means of graphical representation or quantity value and deletes the collected data when new data arrives at the data collection terminal. Analog and digital CROs without storage memory arc typical examples of this. Any measuring equipment used in the medical domain for monitoring without storage functionality also comes under this category.

Some embedded systems store the collected data for processing and analysis. Such systems incorporate a built-in/plug-in storage memory for storing the captured data. Some of them give the user a meaningful representation of the collected data by visual (graphical/quantitative) or audible means using display units [Liquid Crystal Dis-play (LCD). Light Emitting Diode (LED), etc.) buzzers, alarms, etc. Examples arc: measuring instruments with storage memory and monitoring instruments with storage memory used in medical applications. Certain embedded

systems store the data and will not give a representation of the same to the user, whereas the data is used for internal processing.

Digital Camera

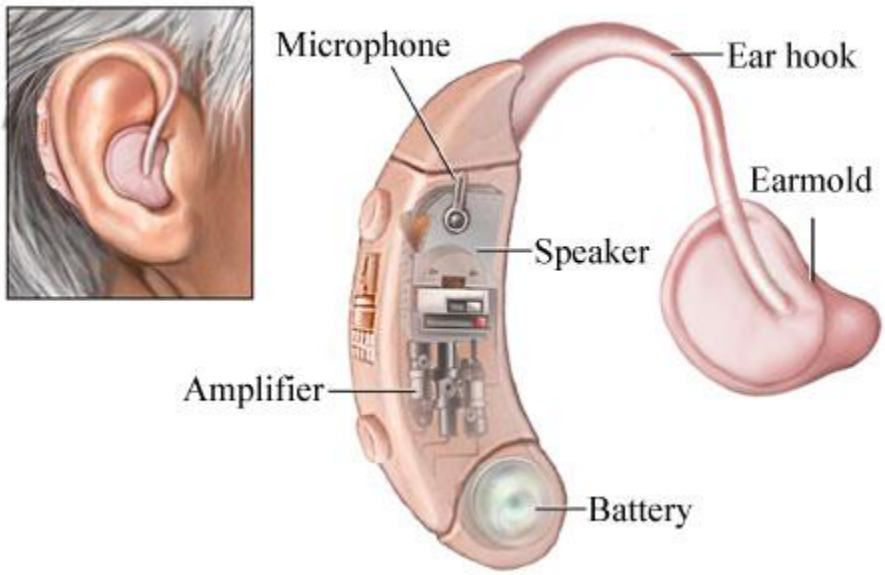
A digital camera is a typical example of an embedded system with data collection/storage/ representation of data. Images arc captured and the captured image may be stored within the memory of the camera. The captured image can also be presented to the user through a graphic LCD unit.

2 Data Communication Embedded data communication systems are deployed in applications ranging from complex satellite communication systems to simple home networking systems. The data collected by an embedded terminal may require transferring of the same to some other system located remotely. The transmission is achieved either by a wire-line medium or by a wire-less medium. Wire-line medium was the most common choice in an olden days embedded systems. As technology is changing, wireless medium is becoming the de-facto standard for data communication in embedded systems. A wireless medium offers cheaper connectivity solutions and make the communication link free from the hassle of wire bundles. Data can either be transmitted by analog means or by digital means. Modem industry trends are settling towards digital communication.

Wi-Fi network Router

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/1P, PS2, etc.). Certain embedded systems act as a dedicated transmission unit between the sending and receiving terminals, offering sophisticated functionalities like data packetizing, encrypting and decrypting. Network hubs, routers, switches, etc. arc typical examples of dedicated data transmission embedded systems. They act as mediators in data communication and provide various features like data security, monitoring etc.

3 Data (Signal) Processing As mentioned earlier, 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 arc employed in applications demanding signal processing like speech coding, synthesis, audio video codec, transmission applications, etc.

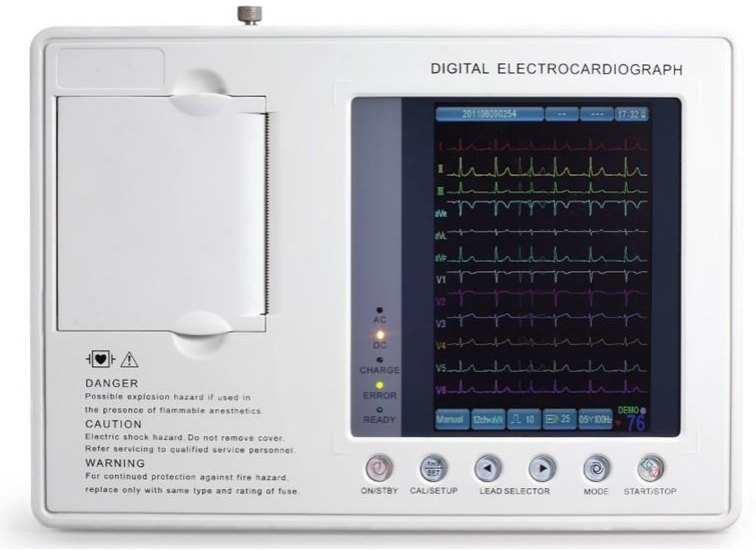


Digital hearing Aid

A digital hearing aid is a typical example of an embedded system employing data processing. Digital hearing aid improves the hearing capacity of hearing-impaired persons.

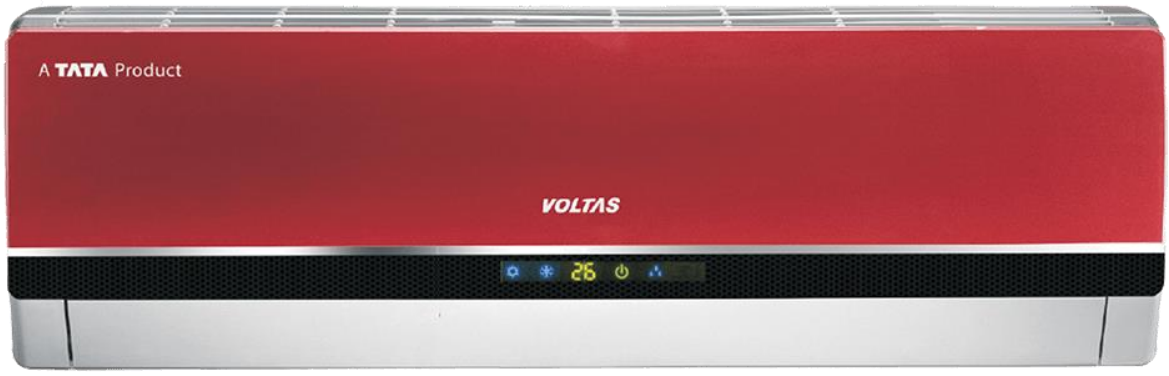
4 Monitoring Embedded systems falling under this category are specifically designed for monitoring purpose. Almost all embedded products coming under the medical domain are with monitoring functions only. They arc used for determining the state of some variables using input sensors. They cannot impose control over variables.

A very good example is the electro cardiogram (ECG) machine for monitoring the heartbeat of a patient. The machine is intended to do the monitoring of the heartbeat. It cannot impose control over the heartbeat. The sensors used in ECG are the different electrodes connected to the patient's body. Some other examples of embedded systems with monitoring function are measuring instruments like digital CRO, digital multimeters, logic analyzers, etc. used in Control & Instrumentation applications. They arc used for knowing (monitoring) the status of some variables like current, voltage, etc. They cannot control the variables in turn.



ECG

5 Control Embedded systems with control functionalities impose control over some variables according to the changes in input variables. A system with control functionality contains both sensors and actuators. Sensors are connected to the input port for capturing the changes in environmental variable or measuring variable. The actuators connected to the output port are controlled according to the changes in input variable to put an impact on the controlling variable to bring the controlled variable to the specified range.



AC

Air conditioner system used in our home to control the room temperature to a specified limit is a typical example for embedded system for control purpose. An air conditioner contains a mom temperature-sensing element (sensor) which may be a thermistor and a handheld unit for setting up (feeding) the desired temperature. The handheld unit may be connected to the central embedded unit residing inside the air-conditioner through a wireless link or through a wired link. The air compressor unit acts as the actuator. The compressor is controlled according to the current room temperature and the desired temperature set by the end user. Here the input variable is the current room temperature and the controlled variable is also the room temperature. The controlling variable is cool air flow by the compressor unit. If the controlled variable and input

variable are not at the same value, the controlling variable tries to equalise them through taking actions on the cool air flow.

6 Application Specific User Interface These are embedded systems with application-specific user interfaces like buttons, switches, keypad, lights, bells, display units,etc.

Mobile phone

Mobile phone is an example for this. In mobile phone the user interface is provided through keypad, graphic LCD module, system speaker, vibration alert etc.

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 equipment's, 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.

Main Features of Embedded Systems

a) Embedded systems are application specific & single functioned; the application is known apriori, the programs are executed repeatedly.

b) Efficiency is of paramount importance for embedded systems. They are optimized for energy, code size, execution time, weight & dimensions, and cost.

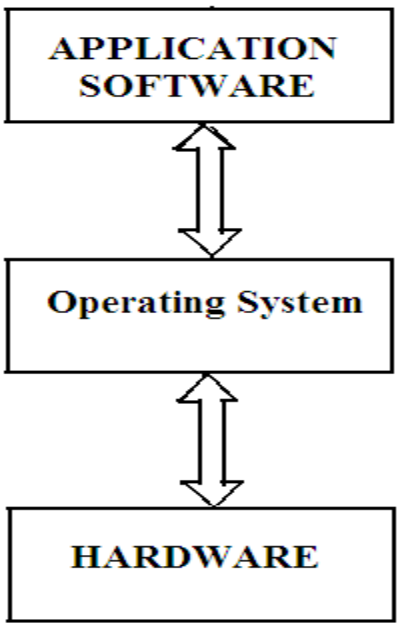
c) Embedded systems are typically designed to meet real-time constraints; a real-time system reacts to stimuli from the controlled object/ operator within the time interval dictated by the environment. For real-time systems, right answers arriving too late (or even too early) are wrong.

d) Embedded systems often interact (sense, manipulate & communicate) with the external world through sensors and actuators and hence are typically reactive systems; a reactive system is in continual interaction with the environment and executes at a pace determined by that environment.

e) They generally have minimal or no user interface.

Overview of embedded systems architecture:

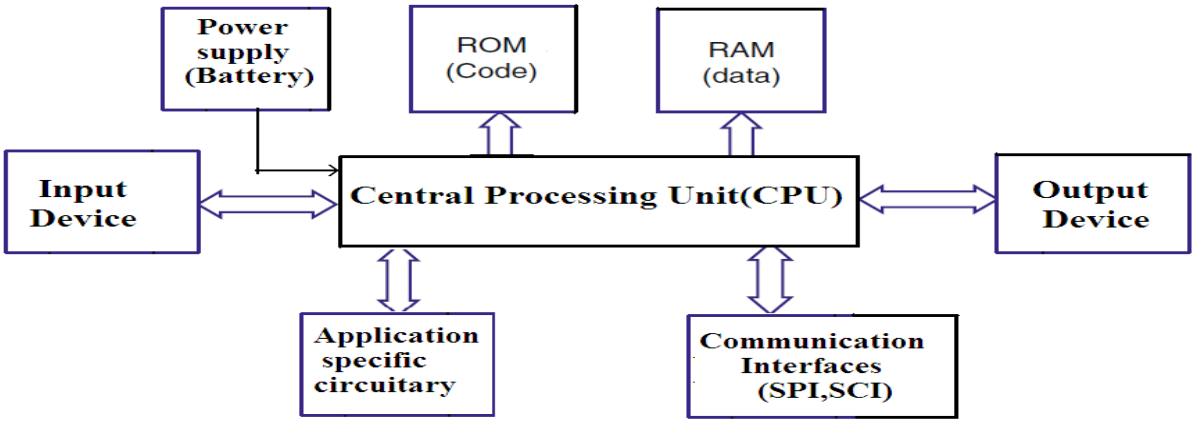
- Every embedded system consists of customer-built hardware components supported by a Central Processing Unit (CPU), which is the brain of a microprocessor (µP) or microcontroller (µC).
- A microcontroller is an integrated chip which comes with built-in memory, I/O ports, timers, and other components.
- 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 plays 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, robot 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 without 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 circuitry)



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.

- 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
- 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

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.
- 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 cannot 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 by a 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 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.

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 sometimes 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.

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.

Address bus and data bus:

- According to computer architecture, a bus is defined as a system that transfers data between hardware components of a computer or between two separate computers.
- Initially, buses were made up using electrical wires, but now the term bus is used more broadly to identify any physical subsystem that provides equal functionality as the earlier electrical buses.
- Computer buses can be parallel or serial and can be connected as multidrop, daisy chain or by switched hubs.
- System bus is a single bus that helps all major components of a computer to communicate with each other.
- It is made up of an address bus, data bus and a control bus. The data bus carries the data to be stored, while address bus carries the location to where it should be stored.

Address Bus

- Address bus is a part of the computer system bus that is dedicated for specifying a physical address.

- When the computer processor needs to read or write from or to the memory, it uses the address bus to specify the physical address of the individual memory block it needs to access (the actual data is sent along the data bus).
- More correctly, when the processor wants to write some data to the memory, it will assert the write signal, set the write address on the address bus and put the data on to the data bus.
- Similarly, when the processor wants to read some data residing in the memory, it will assert the read signal and set the read address on the address bus.
- After receiving this signal, the memory controller will get the data from the specific memory block (after checking the address bus to get the read address) and then it will place the data of the memory block on to the data bus.

The size of the memory that can be addressed by the system determines the width of the data bus and vice versa. For example, if the width of the address bus is 32 bits, the system can address 232 memory blocks (that is equal to 4GB memory space, given that one block holds 1 byte of data).

Data Bus

- A data bus simply carries data. Internal buses carry information within the processor, while external buses carry data between the processor and the memory.
- Typically, the same data bus is used for both read/write operations. When it is a write operation, the processor will put the data (to be written) on to the data bus.
- When it is the read operation, the memory controller will get the data from the specific memory block and put it in to the data bus.

What is the difference between Address Bus and Data Bus?

- Data bus is bidirectional, while address bus is unidirectional. That means data travels in both directions but the addresses will travel in only one direction.

The reason for this is that unlike the data, the address is always specified by the processor. The width of the data bus is determined by the size of the individual memory block, while the width of the address bus is determined by the size of the memory that should be addressed by the system.

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.

Embedded processor and their types

Embedded systems arc domain and application specific and arc built around a central core. Thc 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 arc processor/controller based. The processor may be a micro-processor or a microcontroller or a

digital signal processor, depending on the domain and application. Most of the imbedded systems in the industrial control and monitoring applications make use of the commonly available microprocessors or 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.

1.1 Microprocessors A Microprocessor is a silicon chip representing a central processing unit (CPU), which is capable of performing arithmetic as well as logical operations according to a pre-de-fined set of instructions, which is specific to the manufacturer. In general the CPU contains the Arithmetic and Logic Unit (ALU). control talk and working registers. A microprocessor is a dependent unit and it requites the combination of other hardware like memory, timer unit, and interrupt controller. etc. for proper functioning. Intel claims the credit for developing the first microprocessor unit Intel 4004, a 4bit processor which was released in November 1971. It featured I K data memory, a 12bit program counter and 4K program memory. sixteen Obit general purpose registers and 46 instructions. It ran at a clock speed of 740 kHz. It was designed for olden day's calculators. In 1972, 14 more instructions were added to the 4004 instruction set and the program space is upgraded to 8K. Also interrupt capabilities were added to it and it is renamed as Intel 4040. It was quickly replaced in April 1972 by Intel 8008 which was similar to Intel 4040, the only difference was that its program counter was 14 bits wide and the 8008 served as a minima controller. In April 1974 Intel launched the first 8 bit processor, the Intel 8080, with 16bit address bus and program counter and seven 8bit registers (A-E.11,1.: BC. DE. and HL pairs formed the 16bit register for this processor). lntel 8080 was the most commonly used processors for industrial control and other embedded applications in the 1975s. Since the processor required other hardware components as mentioned earlier for its proper functioning, the systems made out of it were bulky and were lacking compactness. Immediately after the release of Intel 8080. Motorola also entered the market with their processor. Motorola 6800 with a different architecture and instruction set compared to 8080. In 1976 Intel came up with the upgraded version of 8080 — Intel 8083. with two newly added instructions, three interrupt pins and serial LO. Clock generator and bus controller circuits were built-in and the power supply pan was modified to a single +5 V supply. In July 1976 Zilog entered the microprocessor market with its Z80 processor as competitor to Intel. Actually it was designed by an ex-Intel designer. Frederica Foggia and it was an improved version of Intel's 8080 processor, maintaining the original 8080 architecture and instruction set with an $bit data bus and a 16bit address bus and was capable of executing all instructions of 8080. it included 80 more new instructions and it brought out the concept of register banking by doubling the register set. Z80 also included two sets of index registers for flexible design. Technical advances in the

field of semiconductor industry brought a new dimension to the micro-processor market and twentieth century witnessed a fast growth in processor technology. 16, 32 and 64 bit processors came into the place of conventional *bit processors, The initial 2 MHz clock is now an old story. Today processors with clock speeds up to 2.4 Gllz arc available in the market. More and more competitors entered into the processor market offering high speed, high pet fonnance and low cost processors for customer design needs. Intel, AMD, Freescale, IBM. TI, Cyrix, Hitachi, NEC. LSI Logic. etc. arc the kcy players in the processor market. Intel still leads the market with cutting edge technologies in the processor industry.

If you examine any embedded system you will find that it is built around any of the core units men-tioned above. GPP core(s) or ASIP core(s) on either an Application Specific Integrated Circuit (ASIC) or a Very Large Scale Integration (VLSI) circuit.

1.2 General Purpose Processor (GPP) vs. Application-Specific Instruction Set Processor (ASIP)
A General Purpose Processor or GPI' is a processor designed for general computational tasks. The processor running inside your laptop or desktop (Pentium 4/AMDIon. ctc.) is a typical ex-ample for general purpose processor. They are produced in large voltunes and targeting the general market. Due to the high volume production, the per unit cost for a chip is low compared to ASIC or other specific ICs. A typical general purpose processor contains an Arithmetic and Logic Unit (ALU) and Control Unit (CU). On the other hand, Application Specific Instruction Set Processors (ASIPs) arc processors with architecture and instruction set optimised to specific-domain/application require-ments like network processing, automotive, telecom, media applications, digital signal processing. con-trol applications, etc. ASIPs fill the architectural spectrum between general purpose processors and Application Specific Integrated Circuits (ASICs). The need for an ASIP arises when the traditional general purpose processor are unable to meet the increasing application needs. Most of the embedded systems arc built around application specific instruction set processors. Some microcontrollers (like automotive AVR. USI3 AVR from Atmel). system on chips. digital signal processors, etc. arc examples for application specific instruction set processors (ASIPs). ASIPs incorporate a processor and on-chip peripherals. demanded by the application requirement, program and data memory.

1.3 Microcontrollers A Microcontroller is a highly integrated chip that contains a CPU, scratch pad RAM, special and general purpose register arrays, on chip ROM/FLASH memory for program stor-age. timer and interrupt control units and dedicated I/O ports. Microcontrollers can be considered as a super set of microprocessors. Since a microcontroller contains all the necessary functional blocks for independent working, they found greater place in the embedded domain in place of microprocessors. Apart from this, they are cheap, cost effective and arc

readily available in the market. Texas Instrument's TMS 1000 is considered as the world's first microcontroller. We cannot say it as a fully functional microcontroller when we compare it with modern microcontrollers. TI followed Intel's 4004/4040, 4 bit processor design and added some amount of RAM, program storage memory (ROM) and I/O support on a single chip, there by eliminated the requirement of multiple hardware chips for self-functioning. Provision to add custom instructions to the CPU was another innovative feature of TMS 1000. TMS 1000 was released in 1974. In 1977 Intel entered the microcontroller market with a family of controllers corning under one umbrella named .31CS-487m family. The processors came under this family were 80381IL, 8039HL, 8040A11L, 804811, 804911 and 8030A11. Intel 8048 is recognised as Intel's first microcontroller and it was the most prominent member in the MCS-481M" family. It was used in the original IBM PC key-board. The inspiration behind 8048 was Fairchild's F8 microprocessor and Intel's goal of developing a low cost and small size processor. The design of 8048 adopted a true Harvard architecture where pro-gram and data memory shared the same address bus and is differentiated by the related control signals.

Eventually Intel came out with its most fruitful design in the 8bit microcontroller domain-the 8051 family and its derivatives. It is the most popular and powerful 8bit microcontroller ever built. It was developed in the 1980s and was put under the family MCS-51. Almost 75% of the microcontrollcrs used in the embedded domain were 8051 family based controllers during the 1980-90s. 8051 processor cores are used in more than 100 devices by more than 20 independent manufacturers like Maxim, Philips, Atmel, etc. under the license from Intel. Due to the low cost, wide availability, memory efficient instruction set, mature development tools and Boolean processing (bit manipulation operation) capability, 8051 family derivative microcontrollers arc much used in high-volume consumer electronic devices. entertainment industry and other gadgets where cost-cutting is essential.

1.4 Digital Signal Processors Digital Signal Processors (DSPs) arc powerful special purpose 8116/32 bit microprocessors designed specifically to meet the computational demands and power constraints of today's embedded audio, video, and communications applications. Digital signal processors are 2 to 3 times faster than the general purpose microprocessors in signal processing applications. This is because of the architectural difference between the two. DSPs implement algorithms in hardware which speeds up the execution whereas general purpose processors implement the algorithm in firm-ware and the speed of execution depends primarily on the clock for the processors. In general, DSP can be viewed as a microchip designed for performing high speed computational operations for 'addition', 'subtraction', 'multiplication' and 'division', A typical digital signal processor incorporates the following key units: Program Memory- Memory for storing the program required by DSP to process the data Data Memory

Working memory for storing temporary variables and data/signal to be processed. Computational Engine Performs the signal processing in accordance with the stored program memory. Computational Engine incorporates many specialized 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 DSP is employed. Digital signal processing employs a large amount of real-time calculations. Sump of products(SOP) calculation, convolution, fast fourier transform(FFT), discrete Fourier transform(DFT), etc, are some of the operations performed by DSPs.

2 Application Specific Integrated Circuits (ASICs) Application Specific Integrated Circuit (ASIC) is a microchip designed to perform a specific or unique application. It is used as replacement to conventional general purpose logic chips. It integrates several functions into a single chip and there by reduces the system development cost. Most of the ASICs are proprietary products. As a single chip. 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. 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.

If the Non-Recurring Engineering Charges (NRE) is borne by a third party and the Application Specific Integrated Circuit (ASIC) is made openly available in the market, the ASIC is referred as Application Specific Standard Product (ASSP). The ASSP is marketed to multiple customers just as a general-purpose product is. but to a smaller number of customers since it is for a specific application. -The ADE7760 Energy Metre ASIC developed by Analog Devices for Energy metreing applications is a typical example for ASSP".

Since Application Specific Integrated Circuits (ASICs) are proprietary products, the developers of such chips may not be interested in revealing the internal details of it and hence it is very difficult to point out an example of it. Moreover it will create legal disputes if an illustration of such an ASIC prod-uct is given without getting prior permission from the manufacturer of the ASIC.

3 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-fused 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, sim-ulate, and test their designs. Then, a design can be quickly programmed into a device, and immediately tested in a live circuit The PLD that is used for this prototyping is the exact same PLD that will be used in the final production of a piece of end equipment, such as a network router, a DSL modem, a DVD player, or an automotive navigation system, There arc 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 dur-ing 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. Once the design is final, customers can go into immediate production by simply programming as many PLDs as they need with the final software design file.

3.1 CPLDs and FPGAs The two major types of programmable logic devices are Field Programmable Gate Arrays (FPGAs) and Complex Programmable Logic Devices (CPLDs). Of the two. FPGAs offer the highest amount of logic density, the most features, and the highest performance. The largest FPGA now shipping. part of the Xilinx Virtexnet line of devices, provides eight million "system gates" (the relative density of logic). These advanced devices also offer features such as built-in hardwired processors (such as the IBM power PC), substantial amounts of memory, clock management systems, and support for many of the latest, very fast device-to-device signaling technologies. FPGAs are used in a wide variety of applications ranging from data processing and storage, to instrumentation. telecommunications, and digital signal processing. CPLDs, by contrast, offer much smaller amounts of logic-up to about 10,000 gates. But CPLDs offer very predictable timing characteristics and are therefore ideal for critical control applications. CPLDs such as the Xilinx CoolRunnerTM" series also require extremely low amounts of power and are very in-expensive, making them ideal for cost-sensitive. battery-operated, portable applications such as mobile phones and digital handheld assistants.

4 Commercial Off-the-Shelf Components (COTS) A Commercial Off-the-Shelf (COTS) product is one which is used 'as-Le. 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 or an Application Specific Integrated circuit or a programmable logic device. 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, devices and components for operation at very high temperatures, clectro-optic IR imaging arrays, UV/IR detectors, etc. The major advantage of using COTS is that they arc readily available in the market, are cheap and a developer can cut down higher development time to a great extent. This in turn reduces the time to market your embedded systems.

The TCP/IP plug-in module available from various manufactures like 'AVIZnet', `Freescale, `Dynalog*, etc. arc very good examples of COTS product (Fig. 2.7). This network plug-in module gives the TCP/IP connectivity to the system you are developing. There is no need to design this module yourself and write the firmware for the TCP/ IP protocol and data transfer. Everything will be read-ily supplied by the COTS manufacturer. What you need to do is identify the COTS for your system and give the plug-in option on your board according to the hardware plug-in connections given in the specifications of the COTS. Though multiple vendors supply COTS for the same application, the major problem faced by the end-user is that there are no operational and manufacturing standards. A Commercial off-the-shelf (COTS) component manufactured by a vendor need not have hardware plug-in and firmware interface compatibility with one manufactured by a second vendor for the same application. This restricts the end-user to stick to a particular vendor for a particular COTS. This greatly affects the product design.

The major drawback of using COTS components in embedded design is that the manufacturer of the COTS component may withdraw the product or discontinue the production of the COTS at any time if a rapid change in technology occurs, and this will adversely affect a commercial manufacturer of the embedded system which makes use of the specific COTS product.

MEMORY

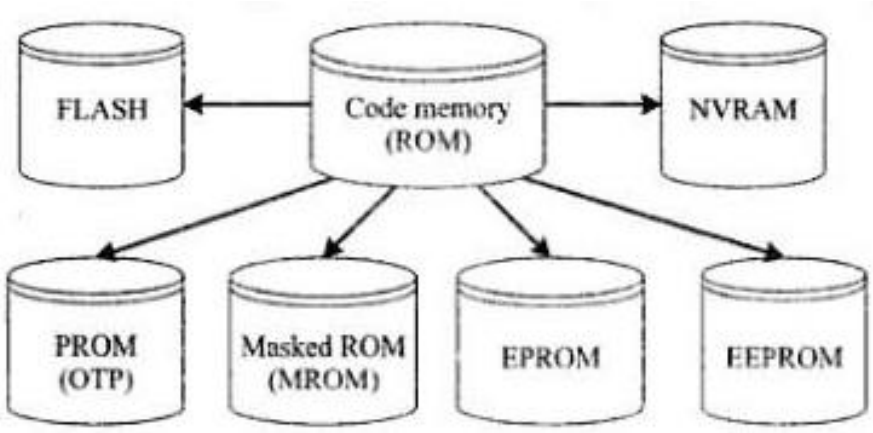
Memory is an important part of a processor/controller based embedded systems. Some of the processors/controllers contain built in memory and this memory is referred as on-chip memory. Others do not contain any memory inside the chip and requires external memory to be connected with the control-ler/processor to store the control algorithm. It is called off-chip memory. Also some working memory is required for holding data temporarily during certain operations. This section deals with the different types of memory used in embedded system applications.

Program Storage Memory (ROM):

The program memory or code memory of an embedded system stores the program instructions and it can be classified into different types as per the block diagram representation.

The code memory retains its contents even after the power to it is turned off. It is generally known as non-volatile storage memory. Depending on the fabrication, erasing and programming techniques they are classified into the following types.

Masked ROM (MROM) Masked ROM is a one-time programmable device. Masked ROM makes use of the hardwired technology for storing data. The device is factory programmed by



masking and metallisation process at the time of production itself, according to the data provided by the end user. The primary advantage of this is low cost for high volume production. They are the least expensive type of solid state memory. Different mechanisms arc used for the masking process of the ROM, like

1. Creation of an enhancement or depletion mode transistor through channel implant.
2. By creating the memory cell either using a standard transistor or a high threshold transistor. In the high threshold mode, the supply voltage required to turn ON the transistor is above the normal ROM IC operating voltage. This ensures that the transistor is always off and the memory cell stores always logic 0.

Masked ROM is a good candidate for storing the embedded firmware for low cost embedded devices. Once the design is proven and the firmware requirements are tested and frozen, the binary data (The firmware cross compiled/assembled to target processor specific machine code) corresponding to it can be given to the MROM fabricator. The limitation with MROM based firmware storage is the inability to modify the device firmware against firmware upgrades. Since the MROM it permanent in bit Morose, it is not possible to alter the bit information.

Programmable Read Only Memory (PROM) / (OTP) Unlike Masked ROM Memory. One Time Programmable Memory (OTP) or PROM is not preprogrammed by the manufacturer. The end user is responsible for programming these devices. This memory has nicinvnte or polysilicon wires arranged in a matrix. These wires can be functionally viewed as fuses. It is programmed by a PROM programmer which selectively burns the fuses according to the bit pattern to be stored. Fuses which are not blown/burned represents a logic "l" whereas fuses which arc blown/burned represents a logic -0". The default state is logic "I". OTP is widely used for commercial production of embedded systems whose proto-typed versions are proven and the code is finalized. It is a low cost solution for commercial production. OTPs cannot be reprogrammed.

Erasable Programmable Read Only Memory (EPROM) OTPs arc not useful and worth for development purpose. During the development phase the code is subject to continuous chang-es and using an OTP each time to load the code is not economical. Erasable Programmable Rcad Only Memory (EPROM) gives the flexibility to re-program the same chip. EPROM stores the bit

information by charging the floating gate of an FET. Bit information is stored by using an EPROM programmer, which applies high voltage to charge the floating gate. EPROM contains a quartz crystal window for erasing the stored information. If the window is exposed to ultraviolet rays for a fixed duration, the entire memory will be erased. 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 minutes. So it is a tedious and time-consuming process.

Electrically Erasable Programmable Read Only Memory (EEPROM) As the name indicates, the information contained in the EEPROM memory can be altered by using electrical signals at the register/Byte level. They can he erased and reprogrammed in-circuit. These chips include a chip erase mode and in this mode they can be erased in a fcw milliseconds. It provides greater flexibility for system design. The only limitation is their capacity is limited when compared with the standard ROM (A few kilobytes).

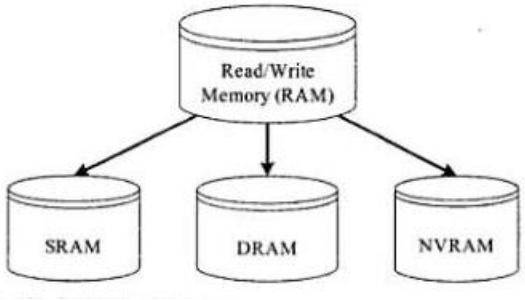
FLASH FLASH is the latest ROM technology and is the most popular ROM technology used in today's embedded designs. FLASH memory is a variation of EEPROM technology. It combines the re-programmability of EEPROM and the high capacity of standard ROMs. FLASH memory is organ-ised as sectors (blocks) or pages. FLASH memory stores information in an army of floating gate MOS-FET transistors. The erasing of memory can be done at sector level or page level without affecting the other sectors or pages. Each sector/page should be erased before ic-programming. The typical erasable capacity of FLASH is 1000 cycles. W27C512 from W1NBOND (www,winbond corn) is an example of 64KB FLASH memory.

NVRAM 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 arc packed together in a single package. The life span of NVRAM is expected to be around 10 years. DSI644 from Maxim/Dallas is an example of 32KB NVRAM.

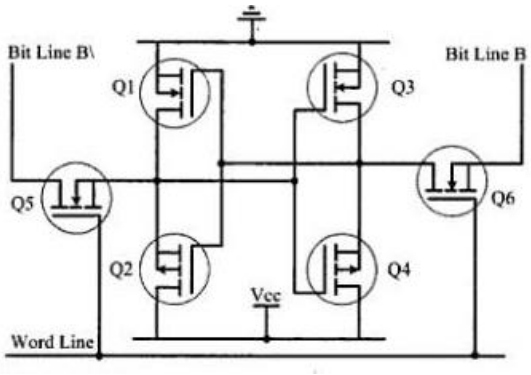
Read-Write Memory/Random Access Memory (RAM)

RAM is the data memory or working memory of the controller/processor. Controller/processor can read from it and write to it. RAM is volatile, meaning when the power is turned off, all the contents are destroyed. RAM is a direct access memory, meaning we can access the desired memory location directly without the need for traversing through the entire memory locations to reach the desired memory position (i.e. random access of memory location). This is in contrast to the Sequential Access Memory (SAM). where the desired memory location is accessed by either traversing through the entire memory or through a 'seek' method. Magnetic tapes, CD ROMs, etc. are examples of sequential access memories. RAM generally falls into three categories: Static RAM (SRAM). dynamic RAM (DRAM) and nun-volatile RAM (NVRAM).

Static RAM (SRAM) Static RAM stores data in the form of voltage. They arc made up of flip-flops. Static RAM is the fastest form of RAM available. In typical implementation, an SRAM

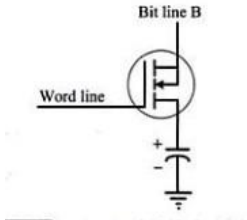


cell (bit) is realized using six transistors (or 6 MOSFETs). Four of the transistors arc used 'for building the latch(flip-flop) part of the memory cell and two for controlling the access. SRRAM is fast in operation due to its resistive networking and switching capabilities. In its simplest representation an SRAM cell can be visualized as shown in fig:



The major limitations of SRAM are low capacity and high cost. Since a minimum of six transistors are required to build a single memory cell, imagine how many memory cells we can fabricate on a silicon wafer.

Dynamic RAM (DRAM) Dynamic RAM stores data in the form of charge. They are made up of MOS transistor gates. The advantages of DRAM arc its high density and low cost compared to SRAM. The disadvantage is that since the information is stored as dune it gets leaked off with time and to prevent this they need to be refreshed periodically. Special circuits called DRAM controllers are used for the refreshing operation. The refresh operation is done periodically in milli-seconds interval. Figure illustrates the typical implementation of a DRAM cell. The MOSFET acts as the gate for the incoming and outgoing data whereas the capacitor acts as the bit storage unit. Table given below summarizes the relative merits and demerits of SRAM and DRAM technology.



SRAM cell	DRAM cell
Made up of 6 CMOS transistors (MOSFET)	Made up of a MOSFET and a capacitor
Doesn't require refreshing	Requires refreshing
Low capacity (Less dense)	High capacity (Highly dense)
More expensive	Less expensive
Fast in operation. Typical access time is 10ns	Slow in operation due to refresh requirements. Typical access time is 60ns. Write operation is faster than read operation.

NVRAM 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 arc packed together in a single package. NVRAM is used for the non-volatile storage of results of operations or for setting up of flags. ctc. The life span of NVRAM is expected to be around 10 years. DS 1744 from Maxim/Dallas is an example for 32KB NVRAM.

Design process of embedded systems

Abstraction of Steps in Design Process: A design process is called bottom-to-top design if it builds starting from the components. A design process is called top-to-down design if it first starts with abstraction of the process and then thereafter the details are created. Top-to-dowm design approach is most favored approach. Following lists, the five levels of abstraction from top level to bottom level during a design process.

1, Requirements
Definition and analysis or system requirements. it means obtaining complete clarity required purpose. Inputs, outputs, functioning, design metrics and validation for the finally developed systems specifications: There should also be consistency in the requirements. Example is a design process which requires talk time of four hours in a day on the one hand, and requires battery that needs recharging after two days on the other hand. Both requirements from the designed System are inconsistent.

2. Specifications
Clear and precise description of parameters and functions of the required systems to guide the user for, expectations from the system. It guides the design of system architecture
Example: Designer needs specifications for (1) hardware, for example, peripherals, devices processor and memory specifications, (ii) data types and processing specifications,(iii) expected system behavior specifications, (iv) constraints of design, and (v) expected life cycle specifications of the product, Process specifications are analyzed by making lists of inputs on events, outputs on events, and processes activated on each event.

3. Architecture
Design of data models, attributes in data structure. data flow graphs. program models. software architecture layers and hardware architectures are defined in the architecture.

4. Components

The fourth level is the component-level design. Software components processes. interfaces and algorithms are defined in this level. Software development process can he modeled on component-based, object-oriented model. Design of hardware components, AS1Ps, special function processors. AS1Cs and IPs are defined for the VLSI. Hardware components may be defined for circuit on board as follows: (a) Microprocessor, ASIP and single-purpose processors (b) Memory RAM ROM or internal and external flash or secondary memory (c) Peripherals (d) Internal and external devices to the system Ports and buses (e) interfacing circuit (f) Power source or battery

5. System Integration

Designed components ire integrated in the system. Components may work fine independently but when integrated may not fulfill the design metrics. Below table lists the actions at each step in the design process.

Table -Action Taken at Each Design Step

Action	Step taken
Analysis	Design is analysed
Steps for improvement	The results of analysis are used to take steps for improvement in order to meet the design specifications and metrics
Verification	System design is verified. Verification ensures that it meets the design metrics initially considered for the design.

Research by software-engineering experts hay e shown that on an average, a designer needs to spend about 50% of time for planning. analysis and design, 40% for testing, validation and debugging and 10-15 % on coding.

I. Software design can be assumed to consist of four layers: architecture design, data design. interfaces design and component level design layers.

2. The design stage steps are from abstraction level to detailed level during design. This follows the verification of the system. Continuous refinements are needed in design by the effective communication between the designers and implementers.

Programming Languages for embedded design

- High level language (C, C++, JAVA….)
- Assembly language
- Mixed C & Assembly language

Assembly-Language Programming

Assembly-language coding of an application has the following advantages:

1. Assembly codes are sensitive to the processor, memory, ports and devices. Assembly software gives a precise control of the processor internal devices. and makes full use of processor-specific features of processor instruction set and addressing modes.

2. Machine codes are compact, and are processor and memory sensitive. The system thus needs a smaller memory. No additional memory needed due to data type's selection, conditions and declarations of rules. A program is also not compiler specific and library-function specific.
3. Certain codes such as device-driver codes may need only few assembly instructions. Assembly codes for these can be compact and precise. and are conveniently written.
4. Bottom-up-design approach is easily usable. Approach to this way of designing a program is as follows: First code the basic functional modules (submodules) and then use these to build a bigger module. Submodules of the specific and distinct sets of actions are first coded. Programs for delay, counting_ finding time intervals and many applications can be written. Then the final program is designed by integrating the modules.

High-level Language Programming

High-level language coding of source .files in C. C++, C#, Visual C++, or Java provides great programming ease. and has many advantages. Therefore. most embedded programming is in the high-level language. Basic advantages are as follows:

1. Program-development cycle is much shorter in high-level language even for complex systems due to the following: use of (1) routines (procedures) [called functions in C/C++ and methods in Java], (ii) standard library functions. and (iii) modular programming, top-down design and object-oriented design approaches. Application programs in high-level language are structured. This ensures that the software is based on sound software engineering principles. Application programs are programmed using given operating system, file systems. device drivers and network drivers, and have access to devices by generic functions. Application program uses Application Program Interfaces (APIs). which makes task of development of an application simple.
2. Top-down-design programming approach, used in high-level language is as follows: Main program is designed first. then its modules, submodules, and finally, the functions.
3. A high-level language program facilitates declaration of data types. Type declarations simplify the programming. Each data type is an abstraction for the methods, which are permitted for using, manipulating. representing, and for defining a set of permissible operations on that data.
4. The program facilitates 'type checking'. This makes a program less prone to error. For example, type checking does not permit subtraction, multiplication and division on the char data types.
5. The program facilitates use of program-flow-control structures, such as loops and conditional statements.
6. The program has portability, and is not processor specific. Therefore, when hardware changes, only the modules for the device drivers and device management, initialization and program-locator modules, and initial boot-up record data need modifications, OS takes care of these functions.

Mixed C & Assembly (In-line assembly)

Additional advantages of C as a high-level language are as follows:
It is a language between low(assembly) and high-level language. Inserting the assembly language codes in between C language codes is called in-line assembly. A direct hardware control is thus also feasible by in-line assembly, and the complex part of the program can be high-level language.

Tools for embedded design

1. Editor

- The first tool you need for Embedded Systems Software Development Tools is text editor.
- This is where you write the code for your embedded system.
- The code is written in some programming language. Most commonly used language is C or C++.
- The code written in editor is also referred to source code.

2. Compiler

- The second among Embedded Systems Software Development Tools is a compiler.
- A compiler is used when you are done with the editing part and made a source code.
- The function of compiler is to convert the source code in to object code.
- Object code is understandable by computer as it in low level programming language.
- So we can say that a compiler is used to convert a high level language code in to low level programming language.

3. Assembler

- The third and an important one among Embedded Systems Software Development Tools is an assembler.
- The function of an assembler is to convert a code written in assembly language into machine language.
- All the mnemonics and data is converted in to op codes and bits by an assembler.
- We all know that our computer understands binary and it works on 0 or 1, so it is important to convert the code into machine language.

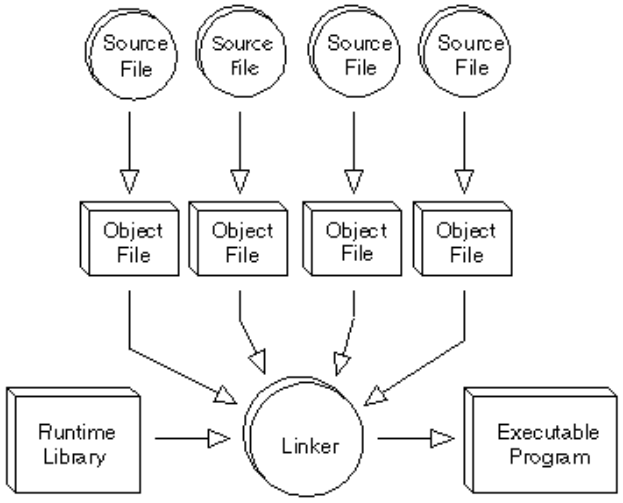
- That was the basic function of an assembler, now I am going to tell you about a debugger.

4. Debugger

- As the name suggests, a debugger is a tool used to debug your code.
- It is important to test whether the code you have written is free from errors or not. So, a debugger is used for this testing.
- Debugger goes through the whole code and tests it for errors and bugs.
- It tests your code for different types of errors for example a run time error or a syntax error and notifies you wherever it occurs.
- The line number or location of error is shown by debugger so you can go ahead and rectify it.
- So from the function, you can see how important tool a debugger is in the list of Embedded Systems Software Development Tools.

5. Linker

- The next one in basic Embedded Systems Software Development Tools is a linker.
- A linker is a computer program that combines one or more object code files and library files together in to executable program.
- It is very common practice to write larger programs in to small parts and modules to make job easy and to use libraries in your program.
- All these parts must be combined into a single file for execution, so this function requires a linker.



6. Simulator

- Among all embedded software tools, simulating software is also needed.
- A simulator helps you to see how your code will work in real time.
- You can see how sensors are interacting, you can change the input values from sensors, and you can see how the components are working and how changing certain values can change parameters.

7. RTOS:

An Operating system(OS) for multitasking, process, memory, IO, network, devices, file system and for real-time control of processes.

8. Prototyper:

For simulating source code engineering including compiling, debugging and navigating through the codes using a browser, summarizing complete status of final target system during the development phase. Tornado Prototyper from Windriver for integrated cross-development environment with a set of tools.

9. Cross-assembler:

For converting object codes or executable codes for a processor at development system to other codes for another processor for embedded system and vice versa.

10. Cross-compiler:

For compiling source codes for a another processor and vice versa

11. Locator:

Uses cross-assembler output and a memory allocation map and provides the locator-program's output.

12. Interpreter:

For expression by expression(line-by-line) translation to the machine executable codes.

13. Integrated Development Environment: Software and hardware environment that consists of simulators, editors, compilers, assemblers, RTOS, debuggers, stethoscope, tracer, emulators, logic analyzers, application codes burners for the integrated development of a system.