Tulsiramji Gaikwad-Patil College of Engineering and Technology
Wardha Road, Nagpur-441 108
NAAC Accredited

Department of Information Technology

'UNIT –I NOTES

Subject:- Embedded Systems **Semester:- VIII**

Unit – I

Syllabus

Introduction to Embedded System:
Introduction, Embedded system vs General computing system, History of embedded system, Processor embedded into a system, Embedded hardware units and devices in a system, Embedded software in a system, examples in a embedded system, Embedded SoC, Complex system design and processors, Design process in ES, Formalization of system design, Classification of Es, Skills required in Embedded system design, Characteristics and quality attributes of Embedded system.

Q:1) What is embedded system? Explain components of embedded system hardware.

Ans:- 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 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.

Embedded Systems Hardware Components

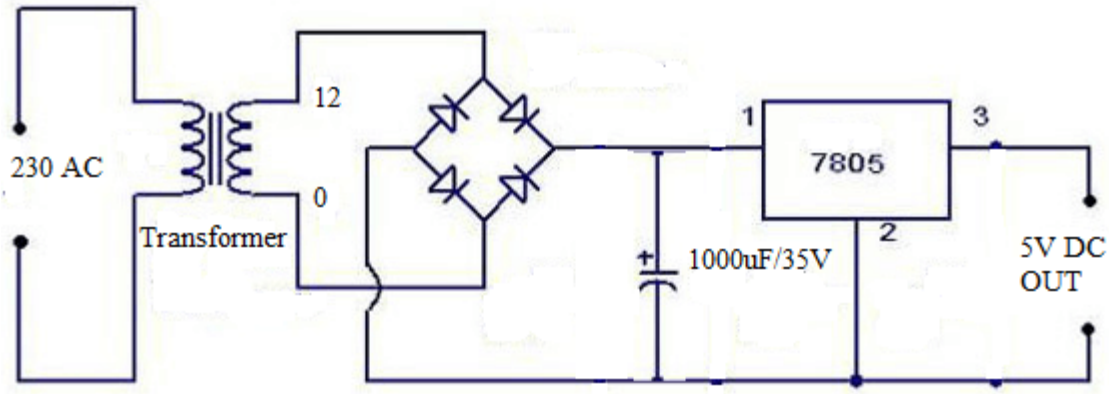
As we know embedded systems are the combination of hardware and software. There are different hardware components like power supply, processor, memory, timers and counters that make the embedded hardware.

Power Supply

The power supply is an essential part of any embedded systems circuits. An embedded system may need a supply of 5 volts or if it is low power then maybe 3.3 or 1.8v. The supply may be provided with the help of battery or we can use any wall adapter. It will depend on the application need.

The power supply circuit can be designed with the help some little knowledge of electronics. For that, we need a bridge rectifier circuit, capacitor as a filter and a voltage regulator that provides constant output supply.

MR.SARVESH WAR



Processor

A processor is the main brain inside any embedded systems. This is a major factor that affects the performance of the system. There are different processors available in the market. An embedded system may use microprocessor or microcontroller.

The processor comes in different architecture like 8-bit, 16-bit and 32-bit. The 8-bit processor is generally used in a small application where we need some basic computation like input and output no heavy processing.

Memory

If we are using a microcontroller like AT89s51, AT89s52 or ATmega. The memory is available on-chip. We generally talk about two types of memory in the embedded systems

- Read-Only memory(RAM)
- Random Access Memory(ROM)
- Electrically Erasable Programmable Read-Only Memory (EEPROM)

RAM memory is volatile memory and used for temporary storage of the data. And the selection of it depends on the user need and the application.

The ROM memory or Code Memory. This is used for the storage of the program. Once system powered, the system fetches the code from the ROM memory.

Timers-Counters

In some application, we need to generate some delay. Like for blinking an LED, we need a delay. For making square pulse we need a delay.

But there is some issue when we generate the delay from the normal coding style by making any loop running for a particular time. Definitely, this will give you some delay but the code after this loop remains in waiting for state and delayed.

So it is not the best approach to generate the delay. For such kind of application where we need a delay for a specific time interval without affecting the normal code execution, we use timer and counter.

Mr.SARVESH WARJURKAR

By setting some register for timer and counter using the programming we get the desired delay. The amount of delay depends on the system frequency and crystal oscillator.

Input and Output

To interact with the embedded systems we need input. The input may be provided by the user or by some sensor. Sometimes some systems need more input or output. So the processor selection will be based on I/O.

These input and output are generally divided into ports like P0, P1, P2 and P3 in 8051microcontrollersr. And PA, PB, PC and PD in ATmega series of the microcontroller.

The I/O need to be configured for input or output based on the provided register. And for that, we need to refer the datasheet of the manufacturer.

Application Specific Circuits

Some hardware components are common while designing the embedded systems. But some are different and depends on the application need. Like a temperature sensor need a temperature sensor for sensing the temperature. While others hand an alcohol detector has a sensor to detect the alcohol level.

But the remaining hardware components might be the same like

Power Supply

Processor

Display Device

Buzzer for Alert

Software Components

Once the hardware is completed we need to build the software for the embedded devices. There are different software tools for programming and coding. These software tools are referred to as software components.

How is software embedded into a system?

We need a program written in assembly or in embedded c language. And then we compile it. This compiled code converted into HEX code. This hex code is programmed or burned into the ROM of the system using some programmer.

These are the tools that are generally used in embedded system development

Assembler

When you program in assembly language. This assembly language program is converted into the HEX code using this utility. Then using some hardware called as a programmer we write the chip.

Emulator

MR.SARVESH WARJURKAR

An emulator is hardware or software tool that has a similar functionality to the target system or guest system. It enables the host system to execute the functionality and other components. It is a replica of the target system. And used for debugging the code and issues.

Once program or code is fixed at the host system. It is transferred to the target system.

Debugger

Sometimes we are not getting expected results or output due to errors or bug. There are certain tools that are specifically used for the debugging process. Where we can see the controls flow and register value to identify the issue.

Compiler

A compiler is a software tool that converts one programming language into target code that a machine can understand. The compiler basically used for translating the high-level language into the low-level language like machine code, assembly language or object code.

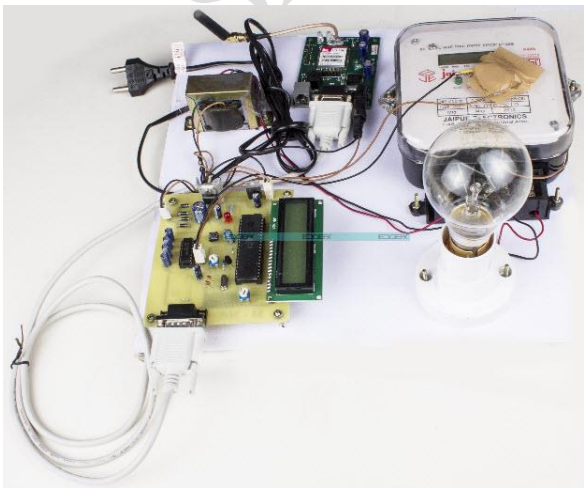
Q:2 Write examples and application of embedded system?

Ans:- Applications of Embedded Systems

Embedded systems find numerous applications in various fields such as digital electronics, telecommunications, computing network, smart cards, satellite systems, military defense system equipment, research system equipment, and so on. Let us discuss a few practical applications of embedded systems that are used in designing embedded projects as a part of engineering final year electronics projects.

IOT Based Energy Meter Reading Through Internet

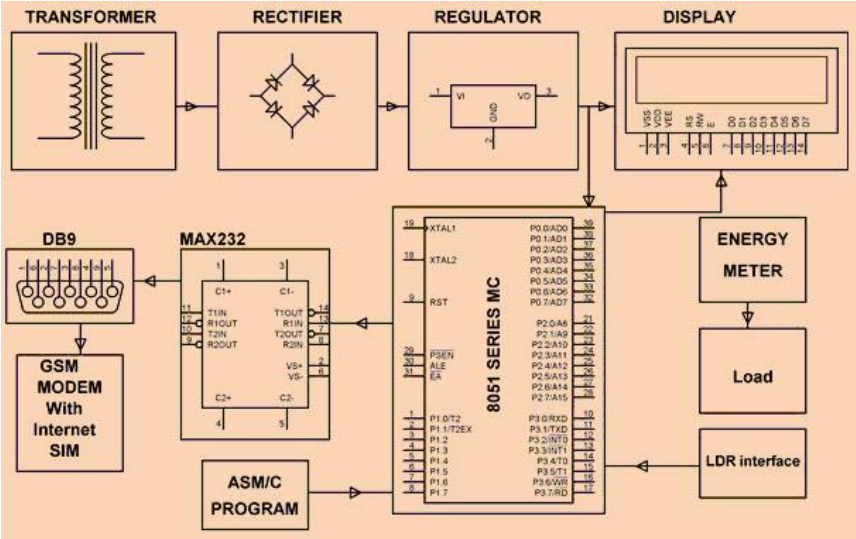
Internet of Things-IOT based energy meter reading through the internet is an innovative application of real time embedded systems. Using this project you can avail the facility of displaying (in the format of chart and gauge) units of power consumed and the cost of consumption over the internet.



IOT Based Energy Meter Reading Through Internet by Edgefxkits.com

SARVESH WARJURKAR

Digital energy meter is used for designing innovative embedded projects, this digital energy meter blinking LED will flash around 3200 times for one unit, this LED signal and microcontroller are interfaced using a light dependent resistor (LDR). Thus, whenever LED flashes, this blinking will activate the LDR sensor, that sends an interrupt signal to the microcontroller for each flash of LED. Based on the interrupts received by the microcontroller, it will display the reading of energy meter on an LCD display which is interfaced it.



IOT Based Energy Meter Reading Through Internet Block Diagram by Edgefxkits.com

This project consists of a GSM modem which is interfaced to the microcontroller using RS232 link and level shifter IC. The reading of energy meter can be sent to the GSM modem, SIM used in this GSM modem is enabled with internet facility. Thus, the energy meter can be directly transmitted to a specific web page to display it over the internet and view in the format of graphical representation from anywhere in the world.

- Internet (IOT) of Things based underground cable fault distance display using GSM
- Electronic passport system using smart card
- Patient body temperature monitoring remotely using Internet of Things (IOT)
- Power saver for street light using high sensitive LDR managed by Arduino
- GSM based prepaid energy meter
- Automatic meter reading system using Zigbee
- A notice board display system using voice commands using an Android phone
- Home automation using voice commands
- Solar based electric fencing system to deter cattle

Q:3 Explain how to get processor embedded into a system.

Mr.SARVESH WAKURKAR

Ans:- Processors in a System

A processor has two essential units −

- Program Flow Control Unit (CU)
- Execution Unit (EU)

The CU includes a fetch unit for fetching instructions from the memory. The EU has circuits that implement the instructions pertaining to data transfer operation and data conversion from one form to another.

The EU includes the Arithmetic and Logical Unit (ALU) and also the circuits that execute instructions for a program control task such as interrupt, or jump to another set of instructions.

A processor runs the cycles of fetch and executes the instructions in the same sequence as they are fetched from memory.

Types of Processors

Processors can be of the following categories −

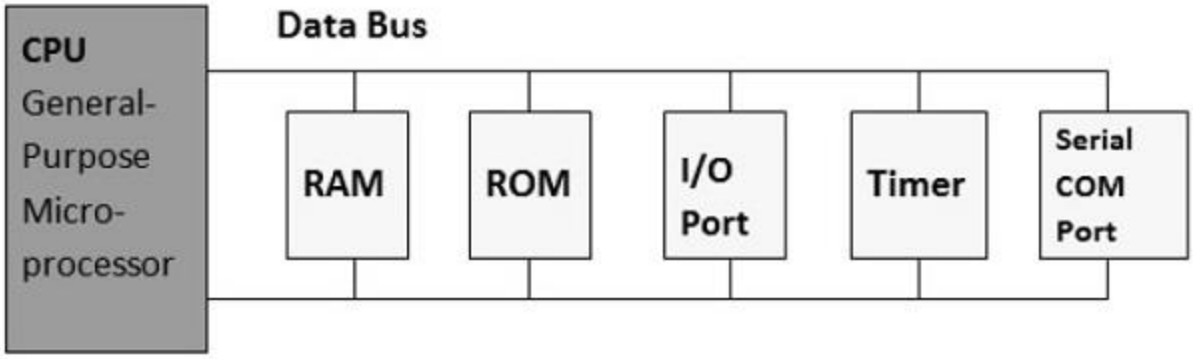
- General Purpose Processor (GPP)
    - Microprocessor
    - Microcontroller
    - Embedded Processor
    - Digital Signal Processor
    - Media Processor
- Application Specific System Processor (ASSP)
- Application Specific Instruction Processors (ASIPs)
- GPP core(s) or ASIP core(s) on either an Application Specific Integrated Circuit (ASIC) or a Very Large Scale Integration (VLSI) circuit.

Microprocessor

A microprocessor is a single VLSI chip having a CPU. In addition, it may also have other units such as coaches, floating point processing arithmetic unit, and pipelining units that help in faster processing of instructions.

Earlier generation microprocessors' fetch-and-execute cycle was guided by a clock frequency of order of ~1 MHz. Processors now operate at a clock frequency of 2GHz

Mr.SARVESH WARJURKAR



A SIMPLE BLOCK DIAGRAM OF A MICROPROCESSOR

Microcontroller

A microcontroller is a single-chip VLSI unit (also called **microcomputer**) which, although having limited computational capabilities, possesses enhanced input/output capability and a number of on-chip functional units.

CPU	RAM	ROM
I/O Port	Timer	Serial COM Port

Microcontrollers are particularly used in embedded systems for real-time control applications with on-chip program memory and devices.

Microprocessor vs Microcontroller

Let us now take a look at the most notable differences between a microprocessor and a microcontroller.

Microprocessor	**Microcontroller**
Microprocessors are multitasking in nature. Can perform multiple tasks at a time. For example, on computer we can play music while writing text in text editor.	Single task oriented. For example, a washing machine is designed for washing clothes only.
RAM, ROM, I/O Ports, and Timers can be added externally and can vary in numbers.	RAM, ROM, I/O Ports, and Timers cannot be added externally. These components are to be embedded together on a chip and are fixed in numbers.
Designers can decide the number of memory or I/O ports needed.	Fixed number for memory or I/O makes a microcontroller ideal for a limited but specific task.

MR. AVESH WARJURKAR

External support of external memory and I/O ports makes a microprocessor-based system heavier and costlier.	Microcontrollers are lightweight and cheaper than a microprocessor.
External devices require more space and their power consumption is higher.	A microcontroller-based system consumes less power and takes less space.

Q:5) Write short note on Compilers & Assemblers?

Ans:- Compiler

A compiler is a computer program (or a set of programs) that transforms the source code written in a programming language (the source language) into another computer language (normally binary format). The most common reason for conversion is to create an executable program. The name "compiler" is primarily used for programs that translate the source code from a highlevel programming language to a low-level language (e.g., assembly language or machine code).

Cross-Compiler

If the compiled program can run on a computer having different CPU or operating system than the computer on which the compiler compiled the program, then that compiler is known as a cross-compiler.

Decompiler

A program that can translate a program from a low-level language to a high-level language is called a decompiler.

Language Converter

A program that translates programs written in different high-level languages is normally called a language translator, source to source translator, or language converter.

A compiler is likely to perform the following operations −

- Preprocessing
- Parsing
- Semantic Analysis (Syntax-directed translation)
- Code generation
- Code optimization

Assemblers

An assembler is a program that takes basic computer instructions (called as assembly language) and converts them into a pattern of bits that the computer's processor can use to perform its basic

Mr.SARVESH WARJUKAR

operations. An assembler creates object code by translating assembly instruction mnemonics into opcodes, resolving symbolic names to memory locations. Assembly language uses a mnemonic to represent each low-level machine operation (opcode).

Q:6 Short note on Emulator & Simulator

Ans:- Debugging Tools in an Embedded System

Debugging is a methodical process to find and reduce the number of bugs in a computer program or a piece of electronic hardware, so that it works as expected. Debugging is difficult when subsystems are tightly coupled, because a small change in one subsystem can create bugs in another. The debugging tools used in embedded systems differ greatly in terms of their development time and debugging features. We will discuss here the following debugging tools −

- Simulators
- Microcontroller starter kits
- Emulator

Simulators

Code is tested for the MCU / system by simulating it on the host computer used for code development. Simulators try to model the behavior of the complete microcontroller in software.

Functions of Simulators

A simulator performs the following functions −

- Defines the processor or processing device family as well as its various versions for the target system.
- Monitors the detailed information of a source code part with labels and symbolic arguments as the execution goes on for each single step.
- Provides the status of RAM and simulated ports of the target system for each single step execution.
- Monitors system response and determines throughput.
- Provides trace of the output of contents of program counter versus the processor registers.
- Provides the detailed meaning of the present command.
- Monitors the detailed information of the simulator commands as these are entered from the keyboard or selected from the menu.
- Supports the conditions (up to 8 or 16 or 32 conditions) and unconditional breakpoints.
- Provides breakpoints and the trace which are together the important testing and debugging tool.
- Facilitates synchronizing the internal peripherals and delays.

MR.SARVESH WARJURKAR

Microcontroller Starter Kit

A microcontroller starter kit consists of −

- Hardware board (Evaluation board)
- In-system programmer
- Some software tools like compiler, assembler, linker, etc.
- Sometimes, an IDE and code size limited evaluation version of a compiler.

A big advantage of these kits over simulators is that they work in real-time and thus allow for easy input/output functionality verification. Starter kits, however, are completely sufficient and the cheapest option to develop simple microcontroller projects.

Emulators

An emulator is a hardware kit or a software program or can be both which emulates the functions of one computer system (the guest) in another computer system (the host), different from the first one, so that the emulated behavior closely resembles the behavior of the real system (the guest).

Emulation refers to the ability of a computer program in an electronic device to emulate (imitate) another program or device. Emulation focuses on recreating an original computer environment. Emulators have the ability to maintain a closer connection to the authenticity of the digital object. An emulator helps the user to work on any kind of application or operating system on a platform in a similar way as the software runs as in its original environment.

Q:7) Explain the Peripheral Devices in Embedded Systems

Ans:- Embedded systems communicate with the outside world via their peripherals, such as following

- Serial Communication Interfaces (SCI) like RS-232, RS-422, RS-485, etc.
- Synchronous Serial Communication Interface like I2C, SPI, SSC, and ESSI
- Universal Serial Bus (USB)
- Multi Media Cards (SD Cards, Compact Flash, etc.)
- Networks like Ethernet, LonWorks, etc.
- Fieldbuses like CAN-Bus, LIN-Bus, PROFIBUS, etc.
- imers like PLL(s), Capture/Compare and Time Processing Units.
- Discrete IO aka General Purpose Input/Output (GPIO)
- Analog to Digital/Digital to Analog (ADC/DAC)
- Debugging like JTAG, ISP, ICSP, BDM Port, BITP, and DP9 ports

Criteria for Choosing Microcontroller

MR.SARVESH WARJURKAR

While choosing a microcontroller, make sure it meets the task at hand and that it is cost effective. We must see whether an 8-bit, 16-bit or 32-bit microcontroller can best handle the computing needs of a task. In addition, the following points should be kept in mind while choosing a microcontroller −

- **Speed** − What is the highest speed the microcontroller can support?
- **Packaging** − Is it 40-pin DIP (Dual-inline-package) or QFP (Quad flat package)? This is important in terms of space, assembling, and prototyping the end-product.
- **Power Consumption** − This is an important criteria for battery-powered products.
- **Amount of RAM and ROM** on the chip.
- **Count of I/O pins and Timers** on the chip.
- **Cost per Unit** − This is important in terms of final cost of the product in which the microcontroller is to be used.

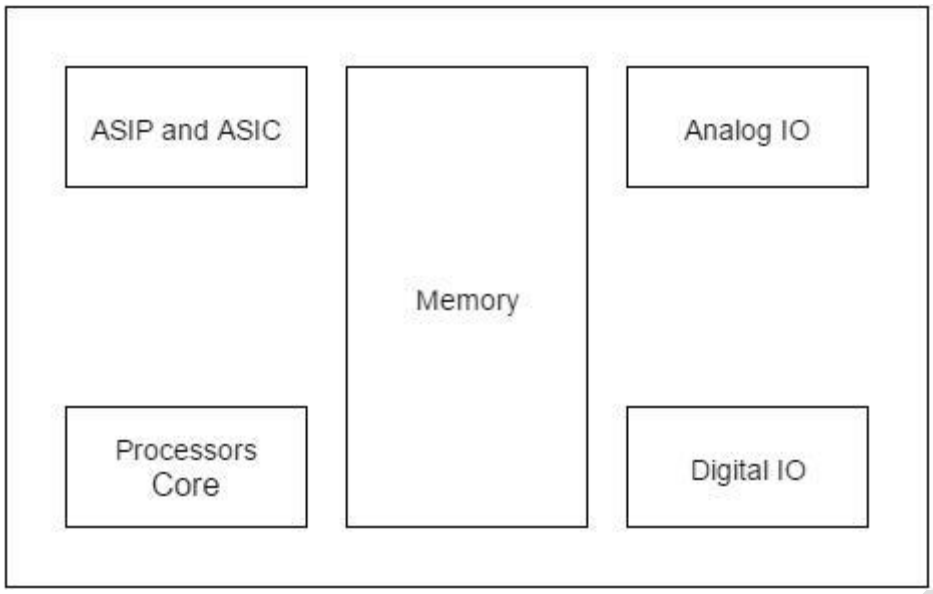
Further, make sure you have tools such as compilers, debuggers, and assemblers, available with the microcontroller. The most important of all, you should purchase a microcontroller from a reliable source.

Q:8) What are the characteristics and advantages of embedded systems

Ans:- Characteristics of an Embedded System

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

NILESH WARJURKAR



Advantages

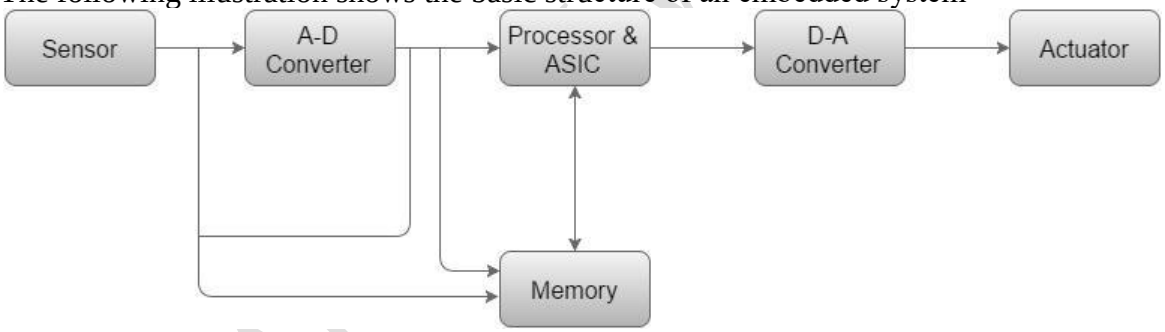
- Easily Customizable
- Low power consumption
- Low cost
- Enhanced performance

Disadvantages

- High development effort
- Larger time to market

Basic Structure of an Embedded System

The following illustration shows the basic structure of an embedded system −



- Sensor − It measures the physical quantity and converts it to an electrical signal which can be read by an observer or by any electronic instrument like an A2D converter. A sensor stores the measured quantity to the memory.
- A-D Converter − An analog-to-digital converter converts the analog signal sent by the sensor into a digital signal.
- Processor & ASICs − Processors process the data to measure the output and store it to the memory.
- D-A Converter − A digital-to-analog converter converts the digital data fed by the processor to analog data
- Actuator − An actuator compares the output given by the D-A Converter to the actual (expected) output stored in it and stores the approved output.

Q:9 Explain the various classification of Embedded systems.

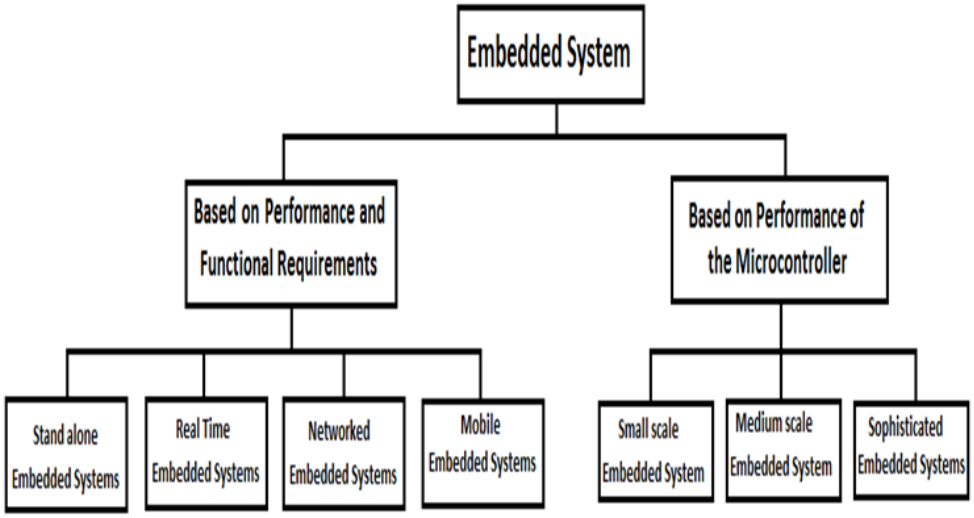
Ans:- An embedded system is an electronic system that has software and is embedded in computer hardware. It is programmable or non-programmable depends on the task specification. To be concern about the characteristics of an embedded system involved its speed, size, power,

MR. SANJAY WARJURKAR

reliability, accuracy and adaptability. Therefore, when the embedded system performs the operations at high speed, then it can be used for real-time applications.

To be concern about the characteristics of an embedded system we can classify it into two broad categories are as follows;

1. Based on performance and functional requirements
2. Based on performance of the microcontroller.



Classification of embedded systems image as flowchart

Figure: Classification of Embedded system

Based on performance and functional requirements of system embedded systems are classified into four categories as follows;

- Stand alone embedded systems
- Real time embedded systems
- Networked embedded systems
- Mobile embedded systems

Stand alone embedded system: This system don't require host system like a computer system, it works by itself. It takes the input from the input ports either analog or digital and processes, computes and transfers the data and gives the resulting data through the connected device-which controls, drives or displays the associated devices. For examples stand alone embedded systems are mp3 players, digital cameras, video game consoles, microwave ovens and temperature measurement systems.

Real time embedded systems:

A system called real time embedded system, which gives a required output in a particular time. These types of embedded systems follow the time deadlines for completion of a task. Real time

MR.SARVESH V KAR

embedded systems are classified into two types such as soft real time embedded system and hard real time embedded systems based on the time preciseness.

Networked embedded system: Networked embedded systems are related to a network to access the resources. The connected network can be LAN, WAN or the internet. The connection can be any wired or wireless. This kind of embedded system is the fastest growing technological area in embedded system applications. The embedded web server is a type of system wherein all embedded devices are connected to a web server and accessed and controlled by a web browser. For example the LAN networked embedded system is a home security system wherein all sensors are connected and run on the protected protocol TCP/IP.

Mobile Embedded Systems: Mobile embedded systems are highly preferable in portable embedded devices like cell phones, mobiles, digital cameras, wireless mp3 players and personal digital assistants, etc. The basic limitation of these devices is the other resources and limitation of memory.

Another category of embedded system based on performance of microcontroller and it is further classified into three categories as follows;

1. Small scale embedded system
2. Medium scale embedded system
3. Sophisticated Embedded Systems

Small Scale Embedded Systems: These types of embedded systems are designed with a single 8-bit or 16-bit microcontroller. They have tiny scaled hardware, software complexities and involve board-level design. They may even be battery operated. When embedded software is developing for this tiny scaled hardware, an editor, an assembler or cross assembler, specific to the microcontroller or processor used, are the main programming tools. Usually, 'C programming language' is 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 existing and keep in view the need to limit power dissipation when system is running continuously.

Medium Scale Embedded Systems: These systems are usually designed with a single or few 16-bit or 32-bit microcontrollers or Digital Signal Processor (DSPs) or Reduced Instruction Set Computers (RISCs) being used. These system have both hardware and software complexities. For complex software design of medium scale embedded system, there are the following programming tools: RTOS, Source code engineering tool, Simulator, Debugger and Integrated Development Environment (IDE). Software tools also give the clarifications to the hardware complexities. An assembler is of slight use as a programming tool. These systems may also utilize the readily available Application-Specific Standard Product (ASSPs) and IPs for the various functions. For example, for the bus interfacing, encrypting, deciphering, discrete cosine transformation and inverse transformation, TCP/IP protocol is stacking and network connecting functions.

Sophisticated Embedded Systems: Sophisticated embedded systems have massive hardware and software complexities and may require ASIPs, IPs and PLAs scalable or configurable processors

MUKESH WARJURKAR

and programmable logic arrays. They are used for cutting edge applications that require hardware and software co-design and integration in the final system. 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 retargetable (Compiler configures according to the specific target) compiler might have to be developed for these.

Q:10) Explain the embedded software development process

Ans:- Development Process of Embedded Systems

The development process of an embedded systems mainly includes hardware design process and software design process. Unlike the design process of software on a typical platform, the embedded system design implies that both hardware and software are being designed similarly Although this isn't continuously the case, it is a truth for many designs currently. The deeper implications of this concurrent design process profoundly impact how embedded systems are designed.

Embedded System Design Process

The different steps in an embedded system design process include the following.

- Determine the requirements
- Design the system architecture
- Select the OS
- Choose the processor and peripherals
- Choose the development platform
- Code the applications and optimize
- Verify the software on the host system
- Verify the software on the target system

MR.SARVESH WARTURKAR

Tulsiramji Gaikwad-Patil College of Engineering and Technology
Wardha Road, Nagpur-441 108
NAAC Accredited

Department of Information Technology

'UNIT –II NOTES

Subject:- Embedded Systems **Semester:- VIII**

Unit – I

Syllabus

Embedded System Design:
Hardware and Software design, Co-design, Embedded Software development Tools: In Circuit Emulators, Cross compilers, cross assemblers and tool chain, linker locator, Address resolution, PROM programmer, Rom Emulator. Memories: EPROM, PROM, Flash.

Q:1 Define the following terms. i)Cross compiler. ii) Cross Assemblers. ii) Host Machine. iv) Target Machine.

Ans:-

i) **Cross Compiler:-** A **cross compiler** is a compiler capable of creating executable code for a platform other than the one on which the compiler is running. For example, a compiler that runs on a Windows 7 PC but generates code that runs on Android smartphone is a cross compiler.

A cross compiler is necessary to compile for multiple platforms from one machine. A platform could be infeasible for a compiler to run on, such as for the microcontroller of an embedded system because those systems contain no operating system. In paravirtualization one machine runs many operating systems, and a cross compiler could generate an executable for each of them from one main source.

The **Canadian Cross** is a technique for building cross compilers for other machines. Given three machines A, B, and C, one uses machine A (e.g. running Windows XP on an IA-32 processor) to build a cross compiler that runs on machine B (e.g. running Mac OS X on an x86-64 processor) to create executables for machine C (e.g. running Android on an ARM processor). When using the Canadian Cross with GCC, there may be four compilers involved:

The *proprietary native Compiler for machine A (1)* (e.g. compiler from Microsoft Visual Studio) is used to build the *gcc native compiler for machine A (2)*.

The *gcc native compiler for machine A (2)* is used to build the *gcc cross*

compiler from machine A to machine B (3)
The *gcc cross compiler from machine A to machine B (3)* is used to build the *gcc cross compiler from machine B to machine C (4)*

2) Cross Assembler:-

A **cross assembler** is a program which generates machine code for a processor other than the one it is currently run on.
An **assembler** is a program that converts assembly language ("human readable" text - if you are a nerd) into the actual binary processor specific machine code (non-human readable binary code - unless you are a nerd). Normally the machine code generated is for the processor used in the machine it is run on. A **cross assembler** takes this conversion process a step further by allowing you to generate machine code for a *different* processor than the one the compiler is run on.
Cross assemblers are generally used to develop programs which are supposed to run on game consoles, appliances and other specialized small electronics systems which are not able to run a development environment. They can also be used to speed up development for low powered system, for example XAsm enables development on a PC based system for a Z80 powered MSX computer. Even though the MSX system is capable of running an assembler, having the additional memory, processor speed and storage capabilities like a harddisk significantly speeds up development efforts

3) Host Machine:-
A host virtual machine is the server component of a virtual machine (VM), the underlying hardware that provides computing resources to support a particular guest virtual machine (guest VM).
The host virtual machine and the guest virtual machine are the two components that make up a virtual machine. The guest VM is an independent instance of an operating system and associated software and information. The host VM is the hardware that provides it with computing resources such as processing power, memory, disk and network I/O (input/output), and so on.
A virtual machine monitor (VMM) or hypervisor intermediates between the host and guest VM, isolating individual guest VMs from one another and making it possible for a host to support multiple guests running different operating systems.
A guest VM can exist on a single physical machine but is usually distributed across multiple hosts for load balancing. A host VM, similarly, may exist as part of the resources of a single physical machine or as smaller parts of the resources of multiple physical machines.

Q: 2) Write in brief about in – circuit formulator.
Ans:-
An ICE is just one of the many debugging tools at your disposal. It's also among the most powerful.
Embedded systems pose unique debugging challenges. With neither terminal

MR SARVESH WARJURKAR

nor display (in most cases), there's no natural way to probe these devices, to extract the behavioral information needed to find what's wrong. This magazine is filled with ads from vendors selling a wide variety of debuggers. They let us connect an external computer to the system being debugged to enable single stepping, breakpoints, and all of the debug resources enjoyed by programmers of desktop computers.

An in-circuit emulator (ICE) is one of the oldest embedded debugging tools, and is still unmatched in power and capability. It is the only tool that substitutes its own internal processor for the one in your target system. Using one of a number of hardware tricks, the emulator can monitor everything that goes on in this on-board CPU, giving you complete visibility into the target code's operation. In a sense, the emulator is a bridge between your target and your workstation, giving you an interactive terminal peering deeply into the target and a rich set of debugging resources.

Until just a few years ago, most emulators physically replaced the target processor. Users extracted the CPU from its socket, plugging the emulator's cable in instead. Today, we're usually faced with a soldered-in surface-mounted CPU, making connection strategies more difficult. Some emulators come with an adapter that clips over the surface-mount processor, tri-stating the device's core, and replacing it with the emulator's own CPU. In other cases, the emulator vendor provides adapters that can be soldered in place of the target CPU. As chip sizes and lead pitches shrink, the range of connection approaches expands.

Beware: connecting the emulator will probably be difficult and frustrating. Physical features of the target system and CPU placement can get in the way of some adapters, so plan for ICE insertion at hardware design time, if at all possible. Add at least a few days to your schedule. Work closely with the vendors to surmount these difficulties.

Q:3) "Locating program components properly". How does locator resolute this issue in the embedded Environment? Explain.

Ans:- The **locator** uses this information to assign physical memory addresses to each of the code and data sections. It will produce an output file that contains a binary image that can be loaded into the target ROM. A commonly used linker/locater for **embedded systems** is ld (GNU).

When build tools run on the same system as the program they produce, they can make a lot of assumptions about the system. This is typically not the case in embedded software development, where the build tools run on a *host* computer that differs from the *target* hardware platform. There are a lot of things that software development tools can do automatically when the target platform is well defined. [1] This automation is possible because the tools can exploit features of the hardware and operating system on which your program will execute. For example, if all of your programs will be executed on IBM-compatible PCs running Windows, your compiler can automate—and, therefore, hide from your view—certain aspects of the software build process.

MUSARVESH WARUDKAR

Embedded software development tools, on the other hand, can rarely make assumptions about the target platform. Instead, the user must provide some of her own knowledge of the system to the tools by giving them more explicit instructions.

The process of converting the source code representation of your embedded software into an executable binary image involves three distinct steps:

1. Each of the source files must be compiled or assembled into an object file.
2. All of the object files that result from the first step must be linked together to produce a single object file, called the relocatable program.
3. Physical memory addresses must be assigned to the relative offsets within the relocatable program in a process called relocation.

The result of the final step is a file containing an executable binary image that is ready to run on the embedded system.

The embedded software development process just described is illustrated in Figure 4-1. In this figure, the three steps are shown from top to bottom, with the tools that perform the steps shown in boxes that have rounded corners. Each of these development tools takes one or more files as input and produces a single output file. More specific information about these tools and the files they produce is provided in the sections that follow.

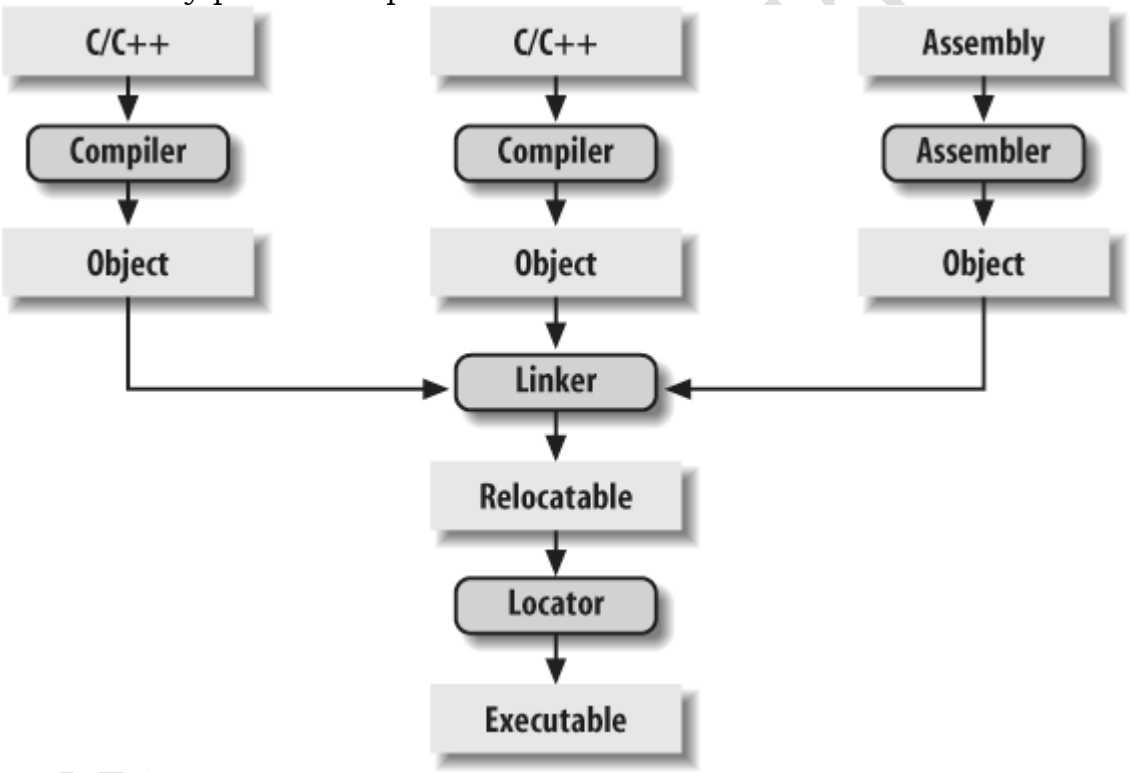


Figure 4-1. The embedded software development process

Each of the steps of the embedded software build process is a transformation performed by software running on a general-purpose computer. To distinguish this development computer (usually a PC or Unix workstation) from the target embedded system, it is referred to as the host computer. The compiler, assembler, linker, and locator run on a host computer rather than on the embedded system itself. Yet, these tools combine their efforts to produce an executable binary image that will execute properly only on the target embedded system. This split of responsibilities is shown in Figure 4-2.

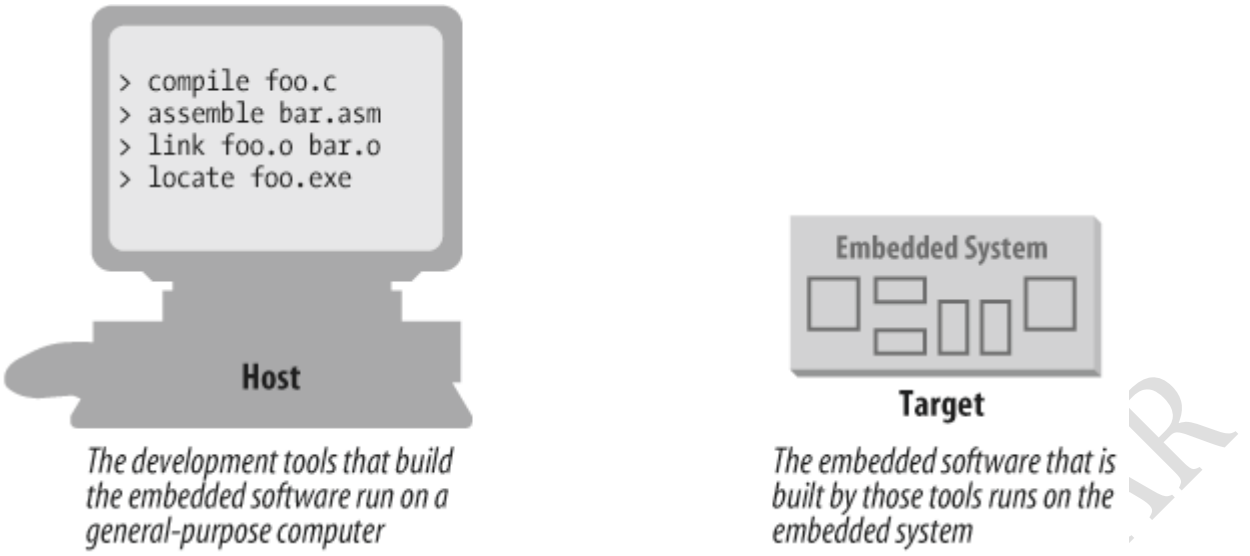


Figure 4-2. The split between host and target

In this book, we'll be using the GNU tools (compiler, assembler, linker, and debugger) for our examples. These tools are extremely popular with embedded software developers because they are freely available (even the source code is free) and support many of the most popular embedded processors. We will use features of these specific tools as illustrations for the general concepts discussed. Once understood, these same basic concepts can be applied to any equivalent development tool.

Compiling

The job of a *compiler* is mainly to translate programs written in some human-readable language into an equivalent set of opcodes for a particular processor. In that sense, an *assembler* is also a compiler (you might call it an "assembly language compiler"), but one that performs a much simpler one-to-one translation from one line of human-readable mnemonics to the equivalent opcode. Everything in this section applies equally to compilers and assemblers. Together these tools make up the first step of the embedded software build process.

Of course, each processor has its own unique machine language, so you need to choose a compiler that produces programs for your specific target processor. In the embedded systems case, this compiler almost always runs on the host computer. It simply doesn't make sense to execute the compiler on the embedded system itself. A compiler such as this—that runs on one computer platform and produces code for another—is called a *cross-compiler*. The use of a cross-compiler is one of the defining features of embedded software development.

Q:4) Explain PROM programmer.

Ans:-

PROM programmers, so called because they were initially designed to program PROMs, EPROMs and EEPROMs, are synonymous with EPROM programmers, chip programmers, IC programmers, memory programmers, flash programmers and IC device programmers. They are also referred to as universal device programmers because they can program a host of different types of chips or ICs

MRCARVESHWARUDAR

without the need for additional "personality cards" (which were used in older "non-universal" programmers.)

Bipolar PROMs

This universal device programmer supports a wide range of PROMs, which are the oldest version of programmable read-only memory (PROM) and uses technology called "bipolar". Because of its old technology, bipolar PROMs require a variety of high voltages for programming and therefore PROMs are not supportable by lower cost programmers. Bipolar PROMs are programmable only once and cannot be erased.

CMOS PROMs

After bipolar, the next major technology that emerged was CMOS (Complementary metal oxide silicon). CMOS allows the design of UV-erasable cells which gave rise to erasable PROMs or EPROMs. Memories chips that are based on the EPROM cells can be erased and reprogrammed numerous times -- provided that the IC chips have a glass (actually quartz) windows to allow UV lights to go through. Those do not have glass windows cannot be erased and therefore are called one-time-programmable, or OTP, EPROMs.

While all EPROMs are byte-wide (8-bit) or word-wide (16-bit), most PROMs are only 4-bit or 8-bit wide.

Since PROMs are very old memory devices, it is not always possible to find them these days. Therefore it helps to know which ones are compatible, in case the original part number you need is not available.

For the same reason, it is also important to buy a programmer that supports a wide variety of bipolar PROMs.

Q:5 Describe the function of embedded software development tools with their applications

Ans:- Embedded Software is the software that controls an embedded system. All embedded systems need some software for their functioning. Embedded software or program is loaded in the microcontroller which then takes care of all the operations that are running. For developing this software, a number of different tools are needed which I will discuss further. These tools include editor, compiler, assembler and debugger. Let's have a look on them.

- **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++.

MR.SARVESH WARTUPKAR

- The code written in editor is also referred to source code.

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.

- **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.

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.

Embedded Systems Software Development Tools, embedded software tools,embedded system tools, embedded software, embedded system software

5. Linker

MR.SARVESH WARJURKAR

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.

Now let's talk about libraries.

6. Libraries

A library is a pre written program that is ready to use and provides specific functionality.

For Embedded Systems Software Development Tools, libraries are very important and convenient.

Library is a file written in C or C++ and can be used by different programs and users.

For example, arduino microcontroller comes with a number of different libraries that you can download and use while developing your software.

For instance, controlling LED or reading sensor like an encoder can be done with a library.

The last one on my list is a simulator.

7. 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.

These were the basic software tools required for embedded software development.

When I am talking about embedded software tools, it is also important to give you an idea about IDE which is the next section of my article.

Integrated Development Environment (IDE) – Embedded Systems Software Development Tools

An Integrated Development Environment is software that contains all the necessary tools required for embedded software development.

For creating software for you embedded system, you need all of the above mentioned tools.

So it is very helpful to have software that can provide all of the necessary tools from writing to testing of your code, in one package.

An IDE normally consists of a code editor, compiler and a debugger.

MR SARVESH WARJURKAR

An Integrated Development Environment also provides a user interface.

An example of integrated development environment is Microsoft visual studio. It is used for developing computer programs and supports different programming languages.

Q:6) What are the different design goals? Explain the need of Co-design.

Ans:- Design Constraints

Designer of an Embedded System faces two conflicting requirements [1]; *High Performance* and *LowCost* [2].

Performance of a system refers to its Direct features. There could be Direct or Indirect features in a product (both add to product cost). Let us consider example of a Digital Still Camera. Its direct features include Zoom (how many X), Picture Quality (How many mega pixels), Optics (which lenses in particular), Storage (How much memory), Speed (How many clicks per second), and Battery life. The indirect features include free accessories and spares, image compression and image processing softwares, warranty and after-sales support. In this section we will only discuss the direct features of a product because these features are directly under control of system designer. A careful selection and design of these features, can greatly improve the final system cost.

High Reliability and High Quality are two other factors which could be decisive during the Design Process. Some (Mission critical and Life critical) systems require very high reliability (e.g. Space Shuttles, Heart Pace-maker, ABS in a Car). Though cost could still be a constraint for such systems, but it is very much relaxed. Any failure of such product could be fatal and hence the design process requires special emphasis on reliability. In some products, quality is a major criteria (apart from low cost).

Safety Norms (e.g. low emi-emc radiations for medical equipments), and Low Power Design (for battery powered handheld devices) could also act as design constraints for such systems.

Design Flexibility

Flexibility is an important design goal in most embedded systems. Under fast growing technological environment, many features of a product get outdated very fast. New features keep emerging and they soon become desirable in a product. A flexible system can adapt to these changes with very less effort (with minimum re-design) and in least time. Product flexibility is a desired characteristics in order to stay competitive (it helps to mantain low cost and low delivery times for product upgrade, because minimum redesign is needed).

MR.SARVESH WARJURKAR

Component selection

The most challenging task for a system designer is that of component selection. Designers need a variety of components (Processors, Memories, Active and Passive electrical components, Software Development Tools, and a few ready to use Software Modules) to build their systems. Overwhelming choices available in the market, make it very puzzling for the designer to choose a particular component. Designers can measure the suitability of a given component on following criteria:

* *Feature Identification:* Identify the "must features" and "optional features" of your system (to be designed). Any component which you choose should be able to meet the "must features" of your system. Any component which can meet the "optional features" with no additional cost (or minimum cost) should be given priority.
* *Component Life Time:* How long does the vendor of a chosen plan to continue the sale of this component? If vendor stops the production of this component, when you are still manufacturing your product, then you will have to probably replace this component. Finding a similar component in future may not be easy. The replacement could even involve a re-design which will add to a major cost. Make sure that the assured life time of the component matches with your product plans.

Layered Architecture

Embedded Systems contain software modules which are closely knit with the hardware. It is desired that the software modules are partitioned in to multiple layers. Each layer should be only dependent on the layers immediately above and below the given layer. It should be agnostic to the changes made to any other layer in the system.
For example consider a software module which receives video data in X format, converts it to Y format, and displays it on a LCD panel. In absence of a layered architecture, any single change made to the specification (data format X or Y, or LCD panel) will require a change to the complete software module. Changing a big software module is prone to errors and is difficult to debug. However the given software module can be partitioned in to following modules/layers:

* Video Format converter (converts data from format X to format Y)
* Low level LCD driver
* Main Application
* APIs (Application Programmer Interfaces) for the LCD driver and Video Format Converter

Operating Systems

Most embedded Systems either run without a distinct operating system or employ a very thin operating system. It is mainly because of two reason:

MR SARVESH WARBHUVAN

* Most embedded systems run small and simple application which are very easy to manage by user (hence no OS is needed).
* In most designs, the hardware underneath the embedded systems is utilized to optimum level and there is very less (or no) additional bandwidth (of Hardware resources) available. Though, an OS can help in managing the system in easier way, but OS requires a good processor bandwidth for its own functioning (which is rarely available). Hence most embedded systems to run a stripped-down version of OS.

Q:7) Explain in detail the importance of embedded software development process.

Ans:-

Mr.SARVESH WARJURKAR