**CHAPTER 05: EMBEDDED SYSTEM**

**Preview:**

In this chapterstudents will know the embedded system. Embedded system is the system in which hardware can be controlled using software program. It can be application specific system that is system designed for specific task. Also students will learn different hardware and software development tools in embedded system are RAM, ROM, timing and counter unit etc and compiler, debugger. And can understand embedded software development cycle. And a brief about System On Chip.

**5.1 Introduction to Embedded System,**

A computer is a system that has the following or more components.

1. A microprocessor
2. A large memory comprising the following two kinds:
3. Primary memory (semiconductor memories - RAM, ROM and fast accessible caches)
4. Secondary memory (magnetic memory located in hard disks, diskettes and cartridge

tapes and optical memory in CD-ROM)

1. Input units like keyboard, mouse, digitizer, scanner, etc.
2. Output units like video monitor, printer, etc.
3. Networking units like Ethernet card, front-end processor-based drivers, etc.
4. I/O units like a modem, fax cum modem, etc.

**Definition:**

Embedded system is one that has computer-hardware with software embedded in it as

one of its most important component.

* It is a dedicated computer-based system for an application(s) or product.
* It may be either an independent system or a part of a larger system.
* As its software usually embeds in ROM (Read Only Memory) it does not need secondary memories as in a computer.

**An embedded system has three main components:**

1. It has hardware.
2. It has main application software. The application software may perform concurrently the series of tasks or multiple tasks.
3. It has a real time operating system (RTOS)

**5.1.1Hardware Components of Embedded System**



Fig 5.1Different Hardware Units

Figure shows the units in the hardware of an embedded system.

* **Processor in system:**

A **processor** is the main component of the embedded system. For an embedded system designer, knowledge of microprocessors and microcontrollers is a prerequisite.

Just as a microprocessor is the most essential part of a computing system, a microcontroller is the most essential component of a control or communication circuit.

A processor has two essential units: Program Flow Control Unit (CU) and Execution Unit (EU). The CU includes a fetch unit for fetching instructions from the memory.

**For a system designer, the following are important considerations when selecting a processor:**

1. Instruction set.
2. Maximum bits in an operand (8 or 16 or 32) in a single arithmetic or logical operation.
3. Clock frequency in MHz and processing speed in Million Instructions Per Second (**MIPS**).
4. Processor ability to solve the complex algorithms used in meeting the deadlines for their processing.
5. Fast context switching and thus lower latencies of the tasks in complex real time applications.
6. Atomic ALU operations and thus no shared data problem. The latter occurs due to an incomplete ALU (non-atomic) operation when an operand of a larger number of bits is placed in two or four registers.
7. RISC core for fast, more precise and intensive calculations by the embedded software.

Calculations for real time image processing and for aerodynamics are two examples where there is a need for fast, precise and intensive calculations and fast context-switching. Important embedded processor chips for embedded systems belong to the following two streams of families.

Stream 1: ARM family ARM 7\* and ARM 9\*

Stream 2: Intel family i960.

Stream 3: AMD family 29050.

An embedded system processor chip or core can be one of the following.

1. General Purpose Processor (GPP):

1. Microprocessor.
2. Microcontroller
3. Embedded Processor.
4. Digital Signal Processor (DSP).
5. Media Processor.

2. Application Specific System Processor (ASSP) as additional processor.

Systems additionally incorporates the Application Specific System processor ASSP

chip(s) or core (s) in its design. These(ASSP) have been recently become available.

Real-time processing need in embedded systems arises for digital television, high

definition TV decoders, set-up boxes, DVD (Digital Video Disc) players, Web phones,

video- conferencing and other systems.

3. Multiprocessor system using General Purpose processors (GPPs) and Application

Specific Instruction Processors (ASIPs)

4. GPP core (s) can be integrated into either an Application Specific Integrated

Circuit. (ASIC), or a Very Large Scale Integrated Circuit (VLSI) circuit or an FPGA core

Integrated with processor unit(s) in a VLSI (ASIC) chip.

5. A DSP provides fast, discrete-time, signal-processing instructions.

* **Power Supply:** Most systems have a power supply of their own. The supply has a specific operation range or a range of voltages.
* **System clock:** is the next important unit of a system. A processor needs a **clock oscillator** circuit. The clock controls the various clocking requirements of the CPU, of the system timers and the CPU machine cycles.
* **RTC:** A timer circuit suitably configured is the **system-clock**, also called real-time clock (RTC). An RTC is used by the schedulers and for real-time programming. An RTC is designed as follows: Assume a processor generates a clock output every 0.5 ms.
* **Reset:** means that the processor starts the processing of instructions from a starting address. That address is one that is set by default in the processor program counter (or instruction pointer and code segment registers in x86 processors) on a power-up.

A Program that runs on power up or reset can be

- System Program that executes from beginning

- System boot up program

- System Initialization program

* **Memories:** In a system, there are various types of memories.



Fig 5.2 Hierarchy of memory

**5.1.2 Software embedded into system,**

* The softwareis the most important aspect, the brain of the embedded system.

An embedded system processor and the system need software that is specific to a

given application of that system.

* The processor of the system processes the instruction codes and data. In the final stage, these are placed in the memory (ROM) for all the tasks that have to be executed.
* Embedded software is also a unique placement and arrangement of bytes for instructions and data.
* A program or a small specific part can be coded in the assembly language or in ‘C’ language.

**5.1.3 RTOS**

* RTOS is system that gives response in predictable time, RTOS also supervises the application software and provides a mechanism of scheduling processes (tasks) and do the context-switch between the various processes (tasks).
* RTOS defines the way the system works.
* It organizes access to a resource in sequence to the number of tasks of the system.
* It schedules their working and execution by following a plan to control the latencies and to meet the deadlines.

Note: Latency refers to the waiting period between running the codes of a task and the instance at which the need for the task arises.

A small-scale embedded system may not need an RTOS.

**5.2 System-On-Chip:**

A typical SoC consists of:

1. A microcontroller, microprocessor or DSP core(s). Some SoCs — called multi-processor system on chip (MPSoC) — include more than one processor core.
2. Memory blocks including a selection of ROM, RAM, EEPROM and flash memory.
3. Timing sources including oscillators and phase-locked loops.
4. Peripherals including counter-timers, real-time timers and power-on reset generators.
5. External interfaces including industry standards such as USB, FireWire, Ethernet, USART, SPI.
6. Analog interfaces including ADCs and DACs.
7. Voltage regulators and power management circuits.

****

Fig 5.3: Microcontroller based system on chip

**5.3 Concept of Device Driver**

* Device driver is a software routine, allowing a higher level computer program to interact with a hardware devices.
* A device driver is typically specific to a given operating system. Device drivers is a software routine that allow application software to attach to, read and write data from, and change the behavior of the peripheral device.
* Programming an embedded device driver requires a different skill set than computer software programming.
* The coding must be precise to meet memory constraints and specialized needs of the product that the embedded system resides in.
* **For example** coffee vending machine. The device is at input port which collects the coin. The system now awakens and activates an interrupt to execute a service routine. This is the device driver routine for that input port. Thus the system which has devices will need ddevice drivers. Therefore embedded system must design codes for

1. Device Initialization
2. Device Activation
3. Device driving using interrupt service routine
4. Resetting or device deactivation.

We can study device driver either with ISR or without ISR

**5.3.1 Device Servicing with ISR**

* In response to the interrupt, the routine or program, which is running presently interrupts and an interrupt service routine (ISR) executes.
* ISR is also called device driver in case of the devices and called exception or signal or trap handler in case of software interrupts
* Processor executes the program, called interrupt service routine or signal handler or trap handler or exception handler or device driver, related to input or output from the port or device or related to a device function on an interrupt and does not wait and look for the input ready or output completion or device-status ready or set.

**How does call to ISR differ from a function (routine) call?**

* On a function call, the instructions are as function in the ‘C’ or in a method in Java.
* In ISR also the instructions are as function in the ‘C’ or in a method in Java
* A function call is after executing present instruction in any program and is a planned (user programmed) diversion from the present instruction in sequence of instructions to another sequence of instructions; this sequence of instructions executes till the return from that.
* An ISR call is after executing present instruction in any program and is interrupt related diversion from the current sequence of instructions to another sequence of instructions; this sequence of instructions executes till the return from that or till another interrupt of higher priority

**5.3.2 Device Drivers:**

A system has a number of physical devices and each device may have multiple functions.

Hence each device will need a driver. For example a timer device performs timing as well as counting operation. A transceiver device transmits as well as well as receives data. Therefore common driver for each device in the system may be required.

Driver controls three functions

(i) Initializing that is activated by placing appropriate bits at the control register or word.

(ii) Calling an ISR on interrupt or on setting a status flag in the status register and run (drive)

the ISR (also called Interrupt Handler Routine).

(iii) Resetting the status flag after interrupt service.

Device driver are important part of system because

* They provide software layer between the application and actual device. While running an application, the system device has to be used. Driver provide a routine that facilitate use of function in an application.
* Driver routines are written in such a way that they can be used by application developer.

**5.3.3 Writing Physical Device Driving ISR’s in a System**

The following points must be considered before writing device driver.

* The device has three resistors data resistor, control resistor and status resistor.
* The device address may have more than one resistor.
* The device may be initialized setting control bits.
* The device closes by resetting the control resistor bits.
* All actions of device are controlled by control resistor bits.
* The control bit controls which address corresponds to which data resistor.

**5.3.4 Device Types:-**

For each type of device, there is a set of the generic commands. For example, for char

device one set of commands and for block device there can be another set.

**Virtual device driver**:-

**Definition:-**

* An embedded system is designed to execute multiple functions and to control multiple physical as well as virtual devices.
* Virtual device is a concept of system software design. An example is “file”. Reading a file gives input to processor. Similarly processor writes to output file. Drivers for virtual devices also have to be written. Types of Physical and Virtual devices in a system may be as follows: char, block, loop back device, file, pipe, socket, RAM disk, sound, video and media.
* Device activation is similar to opening a file. Resetting a device is like closing a file
* Virtual device driver controls the flow of data
* Allows more than one application to access the same memory or physical device without conflict.

Examples of virtual device driver

1. Char Device: For example, a device to which one character is sent at one time or is read from it at one time. For example, mouse, keyboard, keypad, timer
2. Block Device: For example, a device to which one block of characters is sent at one time or is read from it at one time. For example, printer, disk.
3. Pipe device: A device from to which the blocks of characters are send from one end and accessed from another ends in FIFO mode (first-in first-out) after a connect function is executed to connect two ends.

**5.4 Software & Hardware development tools**

Development of hardware and software for embedded systems involves a range of tools that include Simulators, editors, compilers, assemblers, OTP (one time programmers ) etc. That is we require IDE (Integrated Development Environment) for development of embedded system.

**5.4.1 IDE**

Consists of

* Simulators
* editors,
* compilers,
* assemblers,
* emulators
* logic analyzers
* EPROM/EEPROM application codes burner.
* Provides an integrated development environment

1. **Editor:-**

A source code editor is a text editor program designed specifically for editing source

code to control embedded systems. It may be a standalone application or it may be

built into an integrated development environment (e.g. IDE).

1. **Compiler:-**

A compiler is a computer program that translates the source code into computer

language (object code). The compiler is called a cross compiler if the source code is

compiled to run on a platform other than the one on which the cross compiler is run.

For embedded systems the compiler always runs on another platform, so a cross

compiler is needed.

1. **Debugger:**

A debugger is a computer program that is used to test and debug other programs. The

code to be examined might alternatively be running on an instruction set simulator

(ISS) or Emulator, a technique that allows great power in its ability to halt when

specific conditions are encountered but which will typically be much slower than

executing the code directly on the appropriate processor.

1. **Simulator:-**

Code tested for the MCU/ system by simulating it on host computer used for code

development Uses knowledge of target processor or microcontroller, and target

system architecture on the host processor.

* First does cross compilation of the codes and place these into host system RAM.
* Behavior of target system processor registers simulated
* Simulates hardware units like emulator, peripherals, network and input-output

devices on a host (PC (or workstation or laptop).

* A simulator remains independent of a particular targeted system
* Results expected from codes at target system RAM, peripherals, network and

input-output devices obtained at the host system RAM

**Simulator Features**

* Defines the processor or processing device family as well as its various versions for the target system.
* Provides the detailed information of the status of peripheral devices (simulated, assumed to be attached) with the defined system.

1. **Emulator:-**

In [computing](http://en.wikipedia.org/wiki/Computing), an emulator is hardware or software or both that duplicates (or 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 target system).

Emulation refers to the ability of a [computer program](http://en.wikipedia.org/wiki/Computer_program) in an electronic device to emulate (imitate) another program or device.

A hardware emulator is an emulator which takes the form of a hardware device.

The terms "burning", "flashing", "installing", or "downloading" all mean the same thing

-- the (semi-)automated process of putting the executable image into the non-volatile memory of the embedded system

1. **Linker:-**

A linker or link editor is a program that takes one or more objects generated by compilers and assembles them into a single executable program or a library that can later be linked to in itself.

All of the object files resulting from compiling must be combined in a special way before the program can be executed.

The object files themselves are individually incomplete, most notably is that some of the internal variable and function references have not yet been resolved.

The job of the linker is to combine these object files and, in the process, to resolve all of the unresolved symbols. Linkers can take objects from a collection called a library.

1. **In circuit Emulator(ICE),**

An in-circuit emulator (ICE) is a [hardware](http://searchcio-midmarket.techtarget.com/definition/hardware) [interface](http://searchcio-midmarket.techtarget.com/definition/interface) that allows a programmer to change or [debug](http://searchsoftwarequality.techtarget.com/definition/debugging) the [software](http://searchsoa.techtarget.com/definition/software) in an [embedded system](http://searchenterpriselinux.techtarget.com/definition/embedded-system). The ICE is temporarily installed between the embedded system and an external [terminal](http://searchnetworking.techtarget.com/definition/terminal) or [personal computer](http://whatis.techtarget.com/definition/personal-computer-PC) so that the programmer can observe and alter what takes place in the embedded system, which has no display or keyboard of its own.

An ICE serves as a "surrogate" [CPU](http://searchcio-midmarket.techtarget.com/definition/CPU) (central processing unit) for the [microcomputer](http://whatis.techtarget.com/definition/microcomputer) in an embedded system. The ICE usually has a connector that fits the CPU socket in the system. If the connector provided with the ICE does not match the socket in the system, a suitable adapter can usually be found. An ICE can assist design engineers in [product development](http://searchcio-midmarket.techtarget.com/definition/product-development), and also assist programmers or [end user](http://whatis.techtarget.com/definition/end-user)s in product upgrading, modification, or maintenance.

1. **Target Board**

During development process, a host system is used

* Then locating and burning the codes in the target board.
* Target board hardware and software later copied to get the final embedded system
* Final system function exactly as the one tested and debugged and finalized during the development process

****

Fig 5.4: target system

**Target system differs from a final system**

* Target system interfaces with the computer as well works as a standalone system
* In target system might be repeated downloading of the codes during the development phase.
* Target system copy made that later on functions as embedded system
* Designer later on simply copies it into final system or product.
* Final system may employs ROM in place of flash, EEPROM or EPROM in

embedded system.

**Target System Examples**

Phillips LPC 21xx development board, ARM Powered STR710

ARM MCU System Development Board

**5.4.2 Device Programmer:-**

In the field of [computer hardware](http://en.wikipedia.org/wiki/Computer_hardware), the term programmer, chip programmer or device programmer refers to advice that configures programmable non-volatile [digital circuits](http://en.wikipedia.org/wiki/Digital_circuits) suchas [EPROMs](http://en.wikipedia.org/wiki/EPROM), [EEPROMs](http://en.wikipedia.org/wiki/EEPROM), [Flashes](http://en.wikipedia.org/wiki/Flash_Memory), [PALs](http://en.wikipedia.org/wiki/Programmable_Array_Logic), [FPGAs](http://en.wikipedia.org/wiki/Field_Programmable_Gate_Array) or [programmable logic](http://en.wikipedia.org/wiki/Programmable_logic) circuits.

For programming a circuit, it is either inserted into a socket or the programmer is directly connected by an adapter to the circuit board ([In-System Programming](http://en.wikipedia.org/wiki/In-System_Programming)). Afterwards the data is transferred into the circuit by applying signals to the connecting pins.

**5.5 Embedded software development cycle**

In this we will study the standard software development process for embedded system.

Fig shows software development process.

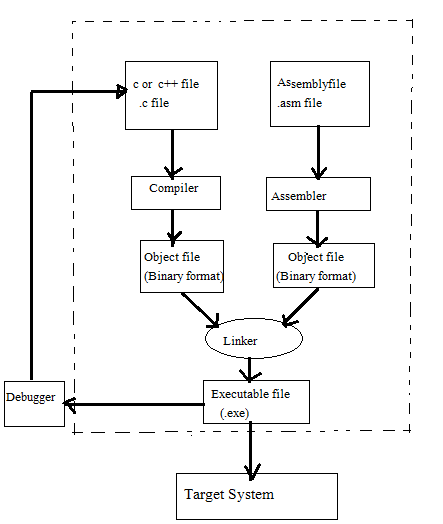


Fig 5.5 Software Development Cycle

Generally designing application start with writing code using editor like C or C++. Then the code is compiled or assembled using compiler or assembler into binary files.

These binary files are combined into final executable file using linkers. These tasks collectively can be called as implementation phase.

The next phase is to test the program by running the executable file under the command of a debugger.

Debugging can be done by either simulation or emulation. If we discover errors, then we return back implementation phase, make necessary corrections and repeat the same process. Once the program is error free w can now load it into the target system.

Testing and debugging phase of developing programs major part of the design process.

**QUESTION BANK**

**2M questions**

1. Give practical example of each embedded system?
2. What do you mean by device driver?
3. What are the advantages of an embedded system?

**4M questions**

1. Explain various debugging tools used in the development of an embedded system.
2. With suitable flow chart list & describe the steps involved in the embedded software development cycle.
3. State the functions of followings
   * 1. i.Compiler
     2. Debugger
     3. Simulator
     4. Emulator
4. Write short note on design challenges (optimizing design metrics) of an

embedded system.

1. What do you mean by system on chip? [summer 12]
2. Enlist the processors available in the embedded system.
3. State the various types of memories available in embedded system. Also state the example of each.
4. Give any eight examples of embedded systems. 4M [summer 12]
5. Give any four applications of embedded system with their examples. [Summer 13]

**6M Questions**

1. Describe the software tools for designing an embedded system.