EMBEDDED SYSTEM CURRICULUM **Duration : 300 Hours**

http://www.edusoftindia.com

Embedded systems, which are electronic systems that include a microcomputer to perform a specific dedicated application, are ubiquitous. Every week millions of tiny computer chips come pouring out of factories like Atmen, Microchip, Motorola and Mitsubishi and find their way into our everyday products. Our global economy, our production of food, our transportation systems, our military defense, our communication systems, and even our quality of life depend on the efficiency and effectiveness of these embedded systems. As electronics, electrical and computer engineers we need to play a major role in all phases of this effort: planning, design, analysis, manufacturing, and marketing. Simply put, embedded technology is software or hardware that is hidden embedded in a large device or system. It typically refers to a fixed function device, as compared with a PC, which runs general-purpose applications.

Embedded technology is nothing new. It's all around us and has been for years. An early example of embedded technology is the engine control unit in a car , which measures what settings to give the engine. Your coffeemaker has embedded technology in the form of a micro controller , which is what tells it to make the coffee at 6 a.m. The

vending machine you bought your Diet Coke from at lunch has it too. Overall, billions of devices woven into everyday life use embedded technology.

So if this technology has been around all this time, why should I care now? In the past, embedded technology existed in standalone devicesvending machines and copiersthat did their jobs with little regard for what went on around them. But as technology has learned to connect devices (mobile phones, PDAs and so on) to the Internet and to each other , embedded technology's potential has grown. Suddenly it's not so much about what devices do on their own, but about what they're connected to and what actions those connections let them perform. Cell phone companies figured that out a long time ago, which is why cell phones are cheap and the service, plans are expensive. It's not the phone itself that matters, but the connectivity to a vast network of other phones, other people and the Internet. Similarly, your PDA is just a PDA; until you download software that lets you find a local restaurant or manage your finances.

To cite a few examples, we can divide our embedded products in various categories depending upon their areas of usage.

Consumer Electronics: Microwave ovens, digital cameras, CD players, DVD players, washing machines, set top box etc.

Automobile: remote keyless entry system, engine management system, biometrics application for vehicle.

Telecommunications: Telephone switches, cellular phones etc.

Medical: Biologically interfaced systems, Medical control systems, Pacemakers, Automotive Engine controllers, anti-lock brake controllers etc.

Plant control: Robots, plant monitors, industrial process systems, airplane control systems, missile guidance systems etc.

MODULE 1:C PROGRAMMING

This module provides the fundamental concepts and terminologies related to C (A procedural, imperative Computer Programming Language). The language has designed to encourage machine independent programming for many applications that had traditionally been implemented in assembly language. The language has become available on a very wide range of platform from embedded micro controllers to supercomputers.

YOU WILL BE ABLE TO

- Explain why we use C language for Embedded System.
- You will be able to define memory places where the variables are defined.
- Explain how is C language is beneficial over assembly language.
- Perform manipulations directly at computer memory where data is stored using pointers.
- Write programs using data structures. OS internally maintain all the information using data structures. Studying DS is necessary to understand Linux internals and device driver programming.

MAIN TOPICS COVERED

- Features of 'C' programming Language
- Program structure Data Types 0Constants
- Variables, Operators Assignment, Selection
- Looping Nested Looping Structures
- Functions, Recursion, Macros
- Pointers & Arrays Strings
- Sorting Techniques-Merge Sort, Heap Sort, Radix Sort, Quick Sort
- Searching Linear & Binary Search
- Concept of Hashing
- Dynamic Data Structures
- Linked List Stack Queue
- Priority Queue Circular List Doubly Linked List
- Tree Binary Tree Traversal
- Search Tree General Purpose Tree
- Representation of General Purpose Tree
- Binary Tree AVL Tree B- Tree

Module 2: 8 BIT INTEL 8051 MICRO CONTROLLER

In this module we discuss the need for microcontrollers and contrast them with general purpose microprocessors such as Pentium and other x86 microprocessors as well as the use of microcontroller in the embedded market. In this section we discuss only 8051 microcontroller and interfacing with peripherals.8051 is a basic 8 bit microcontroller . It is based on CISC architecture. This is the starting stage to enter in controller world.

Edusoft

Development | Training | Consultancy

An ISO : 9001 - 2008 Company

YOU WILL BE ABLE TO:

- Discuss the difference between microcontroller & microprocessor .
- Define the microcontroller internal architecture.
- Use the tool 'Keil'.
- Write programs in 8051 assembly.
- Write programs in Embedded C.
- Do the interfacing with various peripherals.
- Will be able to test your programs directly on testing boards.--
- You will be able to design and implement your own project

MAIN TOPICS COVERED:

- Discussion on Micro Processor Vs. Micro controller , CISC Vs. RISC
- Von Neumann Vs. Harvard Architectures, memory mapped I/O vs. I/O mapped I/O
- Intel 8031/8051 Architecture, pin details , memory in detail
- SFR Accessing &Addressing , Accessing built in Timers
- Interrupts and ISR, Serial communication
- Addressing modes , instruction set
- Interfacing with Micro controller-Practical
- Logic Controller Interface, Traffic Light Interface, 7 segment Display Interface
- Stepper motor , LCD Interface, Keyboard, Analog to Digital Converter

EMBEDDED C PROGRAMMING KEIL SOFTWARE

- Advantages of C Over Assembly, Importance of Assembly, C & Embedded C

Edusoft Research & Education

- Discussion on Embedded software development
- Embedded software development, Features of KEIL software
- Data types & extensions provided by C51, Applications of extensions provided by C51
- Discussion on different memory modules
- Pointers & extensions in C51, Absolute variable declarations
- Storage classes & its importance of global variables
- Usage of bit wise operators in embedded systems, Discussion on optimization techniques
- Interfacing with Micro controller-Practical
- Logic Controller Interface, Traffic Light Interface, 7 segment Display Interface
- Stepper motor , LCD Interface, Keyboard, Analog to Digital Converter

MODULE 3: PIC MICRO CONTROLLER

Manufactured by Microchip, the PIC ("Peripheral Interface Controller") micro controller is popular among engineers and hobbyists alike. PIC microcontrollers come in a variety of "flavors", each with different components and capabilities.

Many types of electronic projects can be constructed easily with the PIC family of microprocessors, among them clocks, very simple video games, robots, servo controllers, and many more.

The PIC is a very general purpose micro controller that can come with many different options, for very reasonable prices.

The PIC Microcontrollers architecture is based on a modified Harward RISC instruction set that provides an easy migration path from 6 to 80m pins and from 384 bytes to 128kbytes of program memory.

YOU WILL BE ABLE TO:

- Understand internal architecture of PIC

- Discuss the difference between RISC and CISC architecture
- Write the Logic oriented programs using 'MPLAB IDE'.
- Do the interfacing with various peripherals.
- Will be able to test your programs directly on testing boards.
- You will be able to design and implement industry based project

Main Topic Covered:

INTRODUCTION TO RISC ARCHITECTURE

- Difference between MP and MC, CISC v/s RISC
- Definition of MP , MC, RISC, CISC, DSP , ES, RTOS
- Code memory (ROM) OTP , Flash, windowed
- Data memory, Register File concept, Registers, Stack Organization
- I2O Lines, Peripherals, Timers
- ADC, E2PROM, UART , PWM, IIC, SPI
- Glue Logic Reset control, Brown out Detectors
- Oscillators RC, LP , XT , HS
- Watch Dog Timer and Prescalar

PIC PROGRAM/SOFTWARE DEVELOPMENT TOOLS FOR EMBEDDED CONTROL PIC 16F877: DETAILED TREATMENT / DESCRIPTION

- General Description, PIC 16F8X Device Variation, Architectural Overview
- Memory Organization, I/O Ports, Timer Modules, Data E2PROM Memory
- Special Features of the CPU, Addressing Modes, Instruction Format, Instruction set summary
- I2C, Watch dog timers and some advanced features of PIC 16F877

LOGIC ORIENTED PROGRAMS (MPLAB)

- Programs to demonstrate the usage of Instruction sets, Addition of 8/16/32-bit numbers Subtraction of 8/16/32-bit numbers, Incrementing/Decrementing of 8/16/32-bit numbers 7-segment look up tables, ASCII to HEX/HEX to ASCII conversion 8/16 bit BCD to Binary, 8/16 bit Binary to BCD, 4 digit BCD UP/DOWN

INTERFACING PROGRAMS

- Interfacing Programs to use, Logic controller , Traffic light controller , ADC/PAC card,
- Temperature control, Stepper motor/DC motor , LCD modules, keypad interface, I2C interface

LINUX INTERNALS

The primary responsibilities of any computer operating system are to facilitate Input/Output (I/O) between the system hardware and applications, and to manage the execution environment of the applications. As a System programmer you will develop a working knowledge of Linux internals in order to write kernel extensions such as device drivers. Application programmers write more efficient code after learning what takes place in the operating system when they use the kernel Application

Programming Interface (API). By learning how the Linux kernel allocates hardware and software resources to processes and files, system administrators and support individuals are better equipped to manage system performance and troubleshoot

Linux

YOU WILL BE ABLE TO:

- Describe how Linux compares to other UNIX-based operating systems
- Explain how Linux views processes and threads tasks
- Explain how the fork( ) system call creates a new process
- Explain how Linux schedules tasks for execution
- Define the term virtual memory and describe how it is implemented in Linux

Main Topics Covered

Introduction to Linux, Features of Linux OS

OVERVIEW OF SUBSYSTEMS

File Management Systems 2. Process Management Systems 3. Memory Management Systems OS services, Interrupts and Exceptions, Architecture of Kernel

FILE MANAGEMENT SYSTEMS

Structure of FMS, Super Block, in ode Block, Data Block, Boot block, inodes, Types of Data Blocks, Conversion of Pathname to inode, Structure of Regular files, Directories and Special Files, Allocation of Disk Blocks, types of files, System Calls

PROCESS MANAGEMENT SYSTEMS

Structure of Processes, User Area Fields, Process Information Table Paging, Page Sizes, Offset, Region Tables, Layout of Kernel, Context of a Process, Register Triples Allocating, Attaching, detaching, duplicating, freeing region Process Scheduling and types and priorities of Processes System Boot and Init process and Kill Signals

MEMORY MANAGEMENT SYSTEMS

Virtual and Physical Memory, Swapping and Paging Demand Paging, Page Faults, Copy on Write bits

INTER PROCESS COMMUNICATION

- Client- Server Model, Pipes, FIFO's, Drawbacks of Pipes and FIFO's
- Message Queues, Shared Memory, Semaphores-binary and array of semaphores, Socket

Programming.

LINUX DEVICE DRIVER

It is the utility to provide interface to the external device with system or kernel. This is basically the modular program into the kernel space which interacts with the application program into the user space

YOU WILL BE ABLE TO:

- Learn, in detail, how the Linux operating system kernel functions.
- Learn how to determine the activities of the kernel by examining source code and

header files, tracing the relationship of kernel structures, such as linked lists and tables, and following the flow of various algorithms.

- Understand the modular programming.
- Write the device driver code for any character , block and network type of device.
- Insert and remove your module into the kernel.
- Know the interrupt handling mechanism of OS.
- Know how to interact with the protocols.
- Describe how Kernal interact with the device driver .

Main Topics Covered

INTRODUCTION

HARDWARE BASICS

- The CPU, Memory, Buses, Controllers and Peripherals, Addresses, Timers, Intro to kernel
- Directory structure of Linux OS.
- Kernel Source Code
- Configuring & Compiling the kernel

ROLE OF DEVICE DRIVER

- Types of Device driver
- Loadable modules and its benefits
- Functions used to load and unload modules
- Passing parameters to a loadable module

WRITING A DEVICE DRIVER PROGRAM

- Important Header Files
- Writing a simple module
- Compiling and loading moduleshttp://www.ducatindia.com
- Exporting symbols from loadable module
- Creating stacked loadable module

- Printk for debugging
- Device information in /proc
- Strace command
- Ksyms and ksymoops
- Debuggers like gdb, kgdb

CLASSES OF DEVICE FILES

- Major and minor numbers
- Creating device files with mk nod
- Registering a character device driver
- IOCTL commands to interact with a device
- Implementing IOCTL in driver
- Functions for accessing user and kernel space

MEMORY & INTERRUPTS HANDLING

- Memory allocation with kmalloc and kfree
- Page oriented memory allocation
- Memory allocation in the virtual address space
- Uses of I/O ports and IRQ's
- Platform dependency issues
- Functions used for reading/writing I/O ports
- Interrupt handler functions
- Timer Interrupts, Delay execution techniques
- Task queues, Task Queue operations
- Obtaining the current time

- PCI Interface
- ISA interface
- Other PC Busses
- Requirements for Network / Block drivers
- Various network and block driver methods

VxWorks 5.4

A real-time operating system (RTOS) is a multitasking operating system intended for real-time applications. Such applications include embedded systems, spacecraft, industrial control and scientific research equipment.

An RTOS provides facilities which, if used properly, guarantee deadlines can be met generally (soft real-time) or deterministically (hard real-time). An RTOS will typically use specialized scheduling algorithms in order to provide the real-time developer with the tools necessary to produce deterministic behavior in the final system. An RTOS is valued more for how quickly and/or predictably it can respond to a particular

event than for the given amount of work it can perform over time. Key factors in an RTOS are therefore a minimal interrupt latency and a minimal thread switching latency.

YOU WILL BE ABLE TO:

- Explain Difference b/w soft RTOS & hard RTOS.
- Explain what is Interrupt handlers and the scheduler
- Explain Preemptive and Non preemptive
- Explain Priority scheduling, Priority inversion
- Explain critical section, Interrupt latency

Main Topics Covered:

INTRODUCTION TO TORNADO IDE

- Host and Target connecting them
- Tornado development environment

- Creating Boot ROM image.
- Target Server and WDB Agent

INTRODUCTION TO THE SHELL AND SHELL COMMANDS

- Debugging and debugging techniques by using the Debugger
- Error Numbers.
- Real Time multitasking and multiprocessing

WHAT IS REAL TIME?

- Requirements of Real time System
- What is a task creation
- Unitask approach Vs multitask approach
- Task states
- Multitasking Kernel
- Context switch
- Priority based schedulinghttp://www.ducatindia.com
- Round Robin scheduling
- Task states in details suspend, delay, pended
- Task hooks

INTER TASK COMMUNICATIONS

- Shared memory linked lists, ring buffers
- Message queues
- Pipes similar to pipes in Unix
- Semaphores Binary, Mutual exclusive, Counting Semaphores, signals

MEMORY MANAGEMENT

- Partitions Regions.
- Exceptions, Interrupts, and Timers
- Interrupts, Handling Interrupts priorities,
- Interrupt Stack, ISR restrictions, ISR guidelines

TIMERS

- System clock, watchdog timers, polling

CREATING I/O SYSTEM

- Basic I/O Open(), Close(),Read(), Write(), Ioctl (), Select()

FILE SYSTEM

- Dos File System
- VxWorks Device Drivers Programming
- Character Device and Block Device

NETWORKING

- Network basics, Network protocols
- Configuring the network, Remote Login
- Remote File access, NFS Network File System

NETWORK PROGRAMMING

- TCP Socket Programming
- UDP socket Programming

32 Bit ARM Micro Controller

ARM: Originally Acorn RISC Machine. A series of low-cost, power-efficient 32-bit RISC microprocessors for embedded control, computing, digital signal processing, games, consumer multimedia and portable applications. It was the first commercial RISC microprocessor

ARM (The Architecture for the Digital World) stands for "Advance RISC Machine" designs the technology that lies at the heart of advanced digital products, from wireless, networking and consumer entertainment solutions to imaging, automotive, security and storage devices. ARM's comprehensive product offering includes 16/32-bit RISC microprocessors, data engines, 3D processors, digital libraries, embedded memories, peripherals, software and development tools, as well as analog functions and

high-speed connectivity products. Combined with the company's broad Partner community, they provide a total system solution that offers a fast, reliable path to market for leading electronics companies. We use ARM7TDMI Processor

YOU WILL BE ABLE TO:

- Define the thumb Instruction, how to change into thumb state
- Discuss the Difference Between ARM and Thumb Instruction
- Discuss the Assembly Coding for ARM
- Discuss on ARM system-on-chip-Architecture thumb Assembly Code and How RISC Processor works
- Code Warier

Main Topics Covered:

PHILIPS LPC 2129: DETAILED TREATMENT/DESCRIPTION

- General Description, architectural Overview
- Memory organization I/O Ports, Timer Modules
- Special Features of the CPU, Addressing modes, Instruction Format
- ISP and IPA
- CAN modules,UARTs,12C,ADC
- Vectored interrupt and interrupt Controllers
- Watch Dog Timer

ARM PROGRAM/SOFTWARE DEVELOPMENT TOOLS

- Introduction to Keil u Vision 3 IDE

- Introduction to MCB 2100 Development KIT

LOGIC ORIENTED PROGRAM

- Programs to demonstrate the usage of Instruction set

INTERFACING PROGRAM

MATERIALS PROVIDED

- Complete course material and software's.
- 100% placement assistance.
- Individual Training Kit of 8051 Embedded Trainer for every candidate separately.
- Having own R&D team always looking for great challenges.
- Complete solution lab for microcontroller's application.
- Special seminar session for technologies updates.

(Projects all done along with classes)

Ideas from students for project are highly appreciable

Edusoft Research & Education