Embedded Systems Using C

32 Hours

Course Description

This course provides a comprehensive introduction to programming embedded systems using the C programming language. Participants will gain practical insights into developing embedded software for the 8051-microcontroller family, exploring topics such as I/O pins, timers, interrupts, serial interfaces, and power consumption. Through hands-on examples and projects, participants will learn to configure the Keil software, simulate programs, and build the associated hardware.

Audience

This course is designed for professionals, students, and enthusiasts interested in acquiring practical skills in embedded systems programming using C. It is suitable for individuals with a basic understanding of programming concepts and a keen interest in developing software for microcontrollers.

Pre-requisite Knowledge/Skills

- Basic programming knowledge
- Understanding of fundamental electronic concepts

Course Objectives

Upon completion of this course, participants will be able to:

- Understand the fundamentals of embedded systems and the selection of processors.
- Effectively program the 8051-microcontroller family, considering external interfaces, memory issues, and power consumption.
- Develop, configure, and simulate embedded software using the Keil software.
- Implement techniques for reading and managing input from switches, including addressing switch bounce.
- Structure code using object-oriented programming principles in C.
- Meet real-time constraints through hardware delays, timeouts, and reliable switch interfaces.
- Create a simple embedded operating system (sEOS) and apply it to real-world scenarios.
- Implement multi-state systems and function sequences for varied applications.
- Utilize the serial interface (RS-232) for communication, demonstrating applications such as data acquisition and remote-control systems.
- Apply the acquired skills in a practical case study: developing an intruder alarm system.

Course Outline

Module 1: Introduction to Embedded Systems

- Introduction
- What is an embedded system?
- Which processor should you use?
- Which programming language should you use?
- Which operating system should you use?
- How do you develop embedded software?
- Conclusions

Module 2: 8051 Microcontroller Family

- Introduction
- What's in a name?
- The external interface of the Standard 8051
- Reset requirements
- Clock frequency and performance
- Memory issues
- I/O pins
- Timers
- Interrupts
- Serial interface
- Power consumption
- Conclusions

Module 3: Hello, Embedded World

- Introduction
- Installing the Keil software and loading the project
- Configuring the simulator
- Building the target
- Running the simulation
- Dissecting the program
- Aside: Building the hardware
- Conclusions

Module 4: Reading Switches

- Introduction
- Basic techniques for reading from port pins
- Example: Reading and writing bytes
- Example: Reading and writing bits (simple version)
- Example: Reading and writing bits (generic version)
- The need for pull-up resistors
- Dealing with switch bounce
- Example: Reading switch inputs (basic code)
- Example: Counting goats
- Conclusions

Module 5: Adding Structure to Your Code

- Introduction
- Object-oriented programming with C
- The Project Header (MAIN.H)
- The Port Header (PORT.H)
- Example: Restructuring the 'Hello Embedded World' example
- Example: Restructuring the goat-counting example
- Further examples
- Conclusions

Module 6: Meeting Real-time Constraints

- Introduction
- Creating 'hardware delays' using Timer 0 and Timer 1
- Example: Generating a precise 50 ms delay
- Example: Creating a portable hardware delay
- Why not use Timer 2?
- The need for 'timeout' mechanisms
- Creating loop timeouts
- Example: Testing loop timeouts
- Example: A more reliable switch interface
- Creating hardware timeouts
- Example: Testing a hardware timeout
- Conclusions

Module 7: Creating an Embedded Operating System

- Introduction

- The basis of a simple embedded OS
- Introducing sEOS
- Using Timer 0 or Timer 1
- Is this approach portable?
- Alternative system architectures
- Important design considerations when using sEOS
- Example: Milk pasteurization
- Conclusions

Module 8: Multi-state Systems and Function Sequences

- Introduction
- Implementing a Multi-State (Timed) system
- Example: Traffic light sequencing
- Example: Animatronic dinosaur
- Implementing a Multi-State (Input/Timed) system
- Example: Controller for a washing machine
- Conclusions

Module 9: Using the Serial Interface

- Introduction
- What is RS-232?
- Does RS-232 still matter?
- The basic RS-232 protocol
- Asynchronous data transmission and baud rates
- Flow control
- The software architecture
- Using the on-chip UART for RS-232 communications
- Memory requirements
- Example: Displaying elapsed time on a PC
- The Serial-Menu architecture
- Example: Data acquisition
- Example: Remote-control robot
- Conclusions

Module 10: Case Study: Intruder Alarm System

- Introduction
- The software architecture