



Embedded C

SYNOPSIS

This course discusses C programming in the context of implementing C applications for 8 bit and 16 bit micro-controller architectures. A distinction is made between pure ANSI C programming and use of pragmas and extensions as found with various embedded C compilers targeted at specific microcontrollers. General ANSI C programming is taught using a PC oriented IDE such as Microsoft's Visual Studio or the GCC compiler under Eclipse.

COURSE OBJECTIVES

Students successfully completing this course will have a sound understanding of C programming. You will understand the basic C data types, arrays and pointers. You will have a good appreciation of data structures and their uses, and the use of pointers to data structures and arrays of data structures. You'll learn how to use arrays to implement circular buffers and how to use them, and how to use arrays to implement stacks, and the uses of software stacks.

You will also be introduced to basic techniques of memory management and programming with dynamic data structures - being able to implement circular buffers and stacks using singly linked and doubly linked lists. You'll also learn techniques for implementing interrupt handler code in C, multi-module programming including applications containing a mixture of C and assembly language modules, and techniques for manipulating hardware registers and special function registers.

SUITABLE FOR

Programmers and engineers who already have some understanding of programming and who now wish to gain a solid understanding of the use of C for embedded systems software development.

PREREQUISITES

Attendees should have a basic knowledge of C programming.

DELIVERY

This is instructor led C training. Each section of the material covered by the tutor is followed by hands-on practical exercises for which worked examples of the solutions are typically provided.

COURSE CONTENTS

Introduction to Embedded System and language

Generic embedded features

- Language used under Application
- Example Applications
- The Development Environment
- Application development strategies
- Wrapper routines and portability
 - a.) Desktop Application
 - b.) Embedded Application different than Desktop?

A brief on C past, present, and future

Features Highlighted

- Advantages under C
- Problems under C

- Yet Why C?

C basic program structure and tools

- A simple C program dissection
- 155 Discuss the image lifecycle

Hands on GNU tool chain for C build process

- Tools to run on RH Linux 9.0
- Tracing the image

Making the most out of C [coding w.r.t processor and compiler]

Using Bit wise Expressions, Controls, loops effectively

- A close look at const and volatile
- Cast and weakly typed language.

Portability and problems Endean's

- BI- Indian, big Indian, small Indian
- Arrays and effective usage

Function, Structures and Union

- Packed vs. Unpacked
- Bit fields

Gotcha under C: Writing portable codes that are not compiler dependent

- Unspecified behavior
- Undefined behavior

Pointers and effective use w.r.t Architecture

- Memory corruption
- Memory leaks

Do's and don't under C's Memory Handling

Hands on to GNU - X tool-chain

Using tools to build image

- Using Makefile for multiple/separately translated programs
- Creating static library [Embedded Application's backbone]
- Debugging a traditional code with character pointer

Hands on building PowerPC toolchain image

- No board for downloading: Focus Cross tool build sequence
- Image and placement using Idscript

Hands on Configuring the DJGPP toolchain

Installing the software

- Steps in building the Standalone C code
- Running C on X86 target without os



- Using the toolchain for target image with RTOS (uC-II OS)
- Creating an binary image file and running it stand alone

Understanding the X86 hardware support needed for running C

- Ten major steps involved in running a C port of DJGPP