



# C++ for Embedded Developers

## Course Description:

This course introduces the C++ language for use on realtime and embedded applications. The first part of the course focuses on the language itself, highlighting areas of concern for real-time and embedded development. The latter part covers the application of C++ to real-time systems including interrupt handling and concurrency issues. If a C++ course does not cover these fundamental issues you may find you still have a lot to learn after the training.

Attendees perform hands on embedded programming, on PC during course practicals. Target hardware may be used on need basis. Approximately 50% of the course is given over to practical work.

## Overview:

A 5 day course covering C++ in general on the first three days and real-time issues on the last two **days!!**. Fifty percent of the course is spent on practical work!!

## Course Objectives:

- . • To provide an understanding of the essentials of the C++ programming language.
- . • To give you practical experience of writing C++ for realtime & embedded systems.
- . • To give you the confidence to apply these new concepts to your next real-time project.

## Delegates will learn:

- . • The core C++ syntax and semantics
- . • How to access hardware in the language
- . • How to program interrupt handlers in C++
- . • About memory and performance issues associated with C++
- . • How real-time operating systems affect the use of the language

## Pre-requisites:

- . • A working knowledge of C

## Who Should Attend:

The course is designed for real-time engineers who are embarking on a project using C++ for the first time. It is also targeted at developers currently reluctant to move to C++ from C as they believe it poses too great an overhead. This course will clearly demonstrate both the strengths and weaknesses of C++ versus C.

## Duration:

Five days.

## Course Materials:

- . • Training Handbook

## Our other related Courses:

- . • Real-Time Systems Design with UML 2.0
- . • Embedded C Workshop
- . • Fundamentals of Real-Time Operating systems

## Course Workshop:

The course makes use of target hardware during the realtime practical exercises. We use GNU or Visual C++ compiler for labs and ARM9 hardware for any specific HW related issue! (based on need).

## Course Outline:

### Introduction to real-time systems

– What is a real-time and embedded computer system – The need for a rigorous development procedure

### From C to C++

– Non object-oriented C++ enhancements to basic C –

Conveniences of C++ over and above C

### **Introduction to Object Oriented (OO) Principles**

– Key characteristics of OO development – OO techniques and the real-time software development process

### **Introduction to Classes**

– Classes & class instances – Methods – Constructors & destructors

### **More on Classes**

– Inlining member functions – const member functions – static class members and functions – arrays of classes – implementing object relationships

### **Inheritance**

– Building class hierarchies – Dynamic binding for class methods, virtual functions – Polymorphism

### **Multiple inheritance (MI)**

– MI and interfaces

### **Functions and Operators**

– Class defined conversions – Overloading and function selection – Friend functions – Overloading operators – Dynamic memory allocation revisited

### **Exception Handling**

– What are exceptions? – Throwing an exception – The try block – Catching an exception – Rethrowing exceptions – Catch all handlers – Exception specifications

### **Templates**

– Introduce parameterised types and functions – Function templates – Class templates

### **The Standard Library**

– Introducing the Standard Template Library

### **Software Structuring**

– Structuring large scale software systems – Separate implementation from interface header files – Dealing with name conflicts – Linking with other languages

### **Embedded C++**

– A summary of Embedded C++ – Embedded C++ features

### **Real-Time Specifics**

– Low level facilities of C++ including: – Accessing hardware – Manipulating information at the bit level – Synchronising I/O with CPU via – Polling – Interrupts

### **Interrupt Programming**

– Interrupt Service Routines in C++ – functional approach – class approach

### **Target Specific Considerations:**

– Data types;  
– Language features affecting portability;  
– Non-standard C++ language features;  
– Assembly language interfacing;  
– Designing ROMable objects.

### **Concurrency:**

– Concurrency;  
– Scheduling strategies;  
– Sharing resources in multi-tasking systems;  
– Synchronizing tasks;



- Transferring data between tasks.

\* \* \* \* \*