As of 29.05.2026

Embedded C++: Object-Oriented Programming for Microcontrollers with C++/EC++ and UML - Face-to-Face Training

Master the challenges in software development for hard real-time systems: Modern embedded systems with complex microcontroller and processor architectures incorporate more and more software that has to be planned and implemented faster than ever. In many cases, requirements need to be met which are subject to standards and safety-critical aspects. Real-time capability, reusability, adjustability to changing basic conditions and readability are key factors in software.

Objectives

The EC++ workshop shows you how an object-oriented approach helps you develop higher-quality software in an embedded environment in a shorter time.

You get a compact overview of the entire development process of embedded real-time systems - software analysis, software design, software implementation and unit test - as a basis for you to implement projects with EC++.

You know how to design of software systems by using UML and implementation in the programming language C++.

You thus have the required knowledge base for meeting specific software quality requirements for embedded systems, e.g. regarding runtime and code efficiency.

Programming guidelines, such as the MISRA-C++ standard, moreover help you avoid programming errors at an early stage.

You can use state of the art object-oriented techniques to develop high-quality, complex software systems and, at the same time, save time and money.

Participants

The EC++ training addresses programmers, software developers, software designers and software architects who use C++ for embedded software applications based on object-oriented concepts.

Requirements

A very good knowledge of C-programming is required; basic microcontroller knowledge is an advantage.

Embedded C++: Object-Oriented Programming for Microcontrollers with C++/EC++ and UML - Face-to-Face Training

Content

Introduction: Object Oriented Programming Basics

- Class and object
- Attribute, operation, method
- Encapsulation
- Relation between classes
- Benefits of the object-oriented approach using a hardware driver as an example

Object-oriented Programming in C

- Classes, attributes, operations and objects
- Encapsulation in C
- Association, aggregation, composition and inheritance

- Practical tips with proven implementation approaches

Introduction: The Programming Language C++ (EC++)

- Migration from object-oriented C-programming to C++/EC++
- Classes, attributes, operations and objects
- Encapsulation in C
- Constructors, destructor, overload
- Modifier for data, operations and objects
- Memory management for objects
- Object initialization
- Namespaces
- Templates
- C++ new style casts
- Further C++ specific mechanisms and constructs
- Assembler analysis of C++ constructs
- Memory and runtime oriented assessment and optimization
- Comparison with C
- Enhancing quality features like maintainability and reusability
- Identifying risks and avoiding pitfalls
- Practical exercise: Implementing, instantiating and testing a class

Relations between Classes / Programming Classes in C++

- Association
- Aggregation
- Composition
- Inheritance
- Dependency
- Selecting the suitable relation in your design
- Implementation variants in C++
- Assembler analysis of C++ constructs
- Memory and runtime oriented assessment and optimization
- Enhancing quality features like maintainability and reusability
- Practical exercise: Implementing the relations association, inheritance and composition in the application and when using the builder pattern

Virtual Methods and Interfaces in C++

- Overwriting functions / operations
- Dynamic (late) binding
- Strictly virtual functions/ operations
- Polymorphism
- From the abstract class to the interface class
- Software architecture with interfaces
- Benefit, advantages and disadvantages of interface classes
- Implementation with C++
- Assembler analysis of C++ constructs
- Memory and runtime oriented assessment and optimization
- Enhancing quality features like maintainability and reusability
- Practical exercise: Programming an interface class with strictly virtual functions, implementing and accessing this class

C++ Class Libraries

- Statistical versus dynamic class libraries
- Generation and inclusion of class libraries
- Advantages and disadvantages of using class libraries in embedded software
- Demonstration of an example

State Machines in C++

- Basic implementation approaches
- Introduction: Table-based state machine processing
- State pattern and object-oriented state pattern
- Demonstration of an example

Callback Structures

- Typical embedded SW architectures and their modeling (layer model, interfaces)

- Architecture guidelines
- Communication between architecture elements
- Object-oriented callback structures with callback objects
- Demonstration of an example
- Practical exercise: Programming a callback structure with callback object registration in the hands-on application

Hardware Drivers and Interrupts in C++

- Driver and interrupt guidelines
- Hardware drivers with peripheral register access object-oriented in C++
- Interrupt concepts and interrupt service routines object-oriented in C++
- Enhancing quality features like maintainability and reusability
- Practical exercise: Applying an object-oriented timer driver along with a timer interrupt in your exercise application

Overview: Embedded Software Test

- Software implementation and test
- Test process
- Checklist for testing object oriented software
- Demonstration of an example with GoogleTest™

Basic Notations with UML (Unified Modeling Language)

- Modeling classes, objects and relations with UML
- Use case, sequence, activity and state diagrams
- Demonstration of the exercise in the UML model, with a focus on diagrams

From Project Planning to Implementation

- Software development process
- Quality of embedded software systems
- Efficient modeling of an embedded application with UML
- Demonstration of the exercise in the UML model, with a focus on model generation and setup

Outlook

- History and further development of C++
- Overview of advanced C++ mechanisms
- Embedded and generic C++ coding guidelines
- Interesting web links
- C++ idioms
- Clean code development

Practical Exercises in the Workshop

- Throughout the exercise (watch application), you will use the STM Arm microcontroller development kit along with a simulated hardware, based on an Arm Cortex®-M3 or M7 microcontroller respectively a suitable simulator. The exercise comprises both the program code and the related UML model.

MicroConsult Plus:

- All participants have the following options to further use their exercises and the solutions developed by MicroConsult from this workshop:
- You e-mail the files to your account, or …
- You get access to file download on request.
- You get all examples for C and C++ in electronic format and can easily adjust them to your development environment.
- You moreover get a tool and software component overview for developing embedded software.
- You get helpful notation overviews for UML (Unified Modeling Language) and SysML (Systems Modeling Language).

FACE-TO-FACE TRAINING

Date	Price *	Duration
22.06.2026 – 25.06.2026	2.600,00 €	4 days

* Price per attendee, in Euro plus VAT

Training code: E-EC++

As of 29.05.2026

Live Online - English

Duration
4 days

Face-To-Face - German

Date	Duration
07.12. – 10.12.2026	4 days

Live Online - German

Date	Duration
07.09. – 10.09.2026	4 days

Coaching

Our coaching services offer a major advantage: our specialists introduce their expertise and experience directly in your solution process, thus contributing to the success of your projects.

We will be happy to provide you with further information or submit a quotation tailored to your requirements.