

Embedded Software Design and Patterns with C - Live Online Training

Modern embedded systems with complex microcontroller and processor architectures incorporate more and more software that has to be planned and implemented faster than ever. System functions are increasingly shifted to software. Consequently, software has to be based on architecture and design. Today, you will use efficient software design methods with C-programming. 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 enhanced readability are key factors in software.

Ziele - Ihr Nutzen

You get to know the programming principles and design patterns that are essential for embedded software development. You learn how to program them in C and use them in your projects.

You learn all about object-oriented programming and implementation, including state machines in C, and get familiar with the mechanisms of an embedded/real-time operating system.

This way, you can implement hardware drivers, interrupt concepts and callback structures in C.

Teilnehmer

The training "Embedded Software Design and Patterns with C" addresses programmers, software developers, software designers and software architects using C for embedded software applications

Voraussetzungen

Knowledge of C programming is required; a basic knowledge of microcontrollers is of advantage.

Live Online Training

09.02. – 12.02.2026 2.600,00 € 4 Tage

* Preis je Teilnehmer, in Euro zzgl. USt.

Anmeldecode: LE-ESD-C

Präsenz-Training - Englisch

Termin	Dauer
24.11. – 27.11.2025	4 Tage

Live-Online - Deutsch

Termin	Dauer
09.02. – 12.02.2026	4 Tage

Präsenz-Training - Deutsch

Termin	Dauer
24.11. – 27.11.2025	4 Tage

Embedded Software Design and Patterns with C - Live Online Training

Inhalt

Embedded Software Design - Introduction

- Definition of software design
- Integration in the development process
- The role of a software designer

Object Orientation in C

- Classes and objects
- Relations: dependency, association, aggregation, composition, inheritance
- Interfaces and virtual functions
- Practical exercise: You design and implement classes and objects with different relations, execute them on an embedded target and test them. Each exercise will be part of the set-up of a measurement device application.

Select Design Principles Implemented in C

- DRY, KISS, avoid premature optimization
- SLA, SRP, dependency inversion
- Principle of least surprise
- Open/closed principle, Law of Demeter, YAGNI
- Source code conventions, MISRA

Select Design Patterns

- Software architecture patterns: layer, blackboard, pipes and filters, client server, model view controller (MVC), microkernel
- Software design patterns: builder/manager, facade, strategy
- Hardware access patterns: HW proxy, HW adapter, mediator, observer, debouncing, interrupt, polling
- Safety and reliability, one's complement, CRC, smart data, channel, protected single channel, multi-channel (dual, triple), sanity check, monitor-actuator
- Practical exercise: You use some of the above-mentioned patterns in the measurement device application

State Machines

- Design
- Implementation variants: switch-case, table, state pattern
- Practical exercise: You design and implement the object-oriented state machine in the measurement device application

Operating System

- Overview of mechanisms: task management, scheduler, synchronization, communication, resource management, time management, interrupt management, memory management
- Practical example: Applying the mechanisms in the measurement device application

Callback Structures

- Communication between architecture elements
- Design rules
- Synchronous, asynchronous
- Callback structure procedural and object-oriented
- Callback structure with and without operating system
- Variation options
- High-quality software architecture with callback structures
- Practical exercise: You design and implement an object-oriented callback structure in the measurement device application

Hardware Driver Concepts and Interrupts

- Architecture guidelines
- Software layer patterns
- Practical examples of software layer architectures
- Object-oriented driver concepts
- Interrupt handling
- Practical tips: Standards and sources for driver concepts
- Practical exercise: You design and implement a driver and the related driver access in the measurement device application

Select Refactorings in C

- Preconditions for successful refactoring
- Small steps, big steps
- Smells
- Refactoring patterns

Practical exercise

- For the underlying practical exercise (measurement device application), you will use the Arm Keil MDK (microcontroller development kit) with real hardware based on an Arm Cortex™-M7 microcontroller.

MicroConsult PLUS:

- We will provide you with a download link for your exercises and the solutions from this workshop.
- You get all examples for C and UML models 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 a helpful notation overview for UML.