MICRO CONSULT

Stand 11.01.2026

Embedded-Software-Design und Patterns mit C - Präsenz-Training

Die heutigen Embedded-Systeme mit komplexen Mikrocontroller- und Prozessorarchitekturen enthalten immer mehr Software, die aber in immer kürzerer Zeit geplant und realisiert werden muss. Immer mehr Systemfunktionalität verlagert sich in die Software. In der Konsequenz muss die Software aus Architektur und Design entstehen. Die VHIT- (vom Hirn ins Terminal) Methode ist längst nicht mehr anwendbar. Stattdessen wenden Sie erfolgreiche Methoden für das Software-Design mit C-Programmierung an. Häufig sind Vorgaben zu erfüllen, in denen Normen und sicherheitskritische Aspekte berücksichtigt werden müssen. Echtzeitfähigkeit, Wiederverwendbarkeit, Anpassbarkeit an veränderte Rahmenbedingungen und leichte Lesbarkeit der Software spielen eine immer größere Rolle.

**Ziele -
Ihr Nutzen**

Sie lernen die für die Embedded-Softwareentwicklung wichtigen Programmierprinzipien und Design Patterns kennen und können diese in C programmieren und in Ihren Projekten anwenden.

Beherrschen Sie die objektorientierte Programmierung und Implementierung, auch von Zustandsautomaten in C, und machen Sie sich mit den Mechanismen eines Embedded-/Echtzeitbetriebssystems vertraut.

So sind Sie in der Lage, Hardware-Treiber, Interrupt-Konzepte und Callback-Strukturen in C umzusetzen.

Teilnehmer

Der Kurs Embedded-Software-Design und Patterns in C richtet sich an Programmierer, Software-Entwickler, Software-Designer und Software-Architekten, die C für Embedded-Software-Applikationen einsetzen.

Voraussetzungen

Sie sollten über C-Programmierkenntnisse verfügen; Mikrocontroller-Grundkenntnisse sind von Vorteil

Embedded-Software-Design und Patterns mit C - Präsenz-Training

Inhalt

Themeneinleitung Embedded-Software-Design

- Definition Software-Design
- Einordnung in den gesamten Entwicklungsprozess
- Die Rolle des Software-Designers

Objektorientierung in C

- Klassen und Objekte
- Relationen: Dependency, Assoziation, Aggregation, Komposition, Vererbung
- Interfaces und virtuelle Funktionen
- Übung: Sie designen und implementieren Klassen sowie Objekte mit verschiedenen Relationen, führen diese auf einem Embedded-Target aus und testen sie. Mit jeder Übung wächst Schritt für Schritt eine Messgeräte-Applikation.

Ausgewählte Design-Prinzipien mit C umgesetzt

- DRY, KISS, Vorsicht vor Optimierungen
- SLA, SRP, Dependency Inversion
- Prinzip der geringsten Überraschungen
- Open/Closed-Prinzip, Law of Demeter, YAGNI
- Source-Code-Konventionen, MISRA

Ausgewählte Design Patterns

- Softwarearchitektur-Pattern: Layer, Blackboard, Pipes and Filters, Client Server, Model View Controller (MVC), Microkernel
- Softwaredesign-Pattern: Builder/Manager, Facade, Strategy
- Hardwarezugriff-Pattern: 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
- Übung: Sie wenden in der Messgeräte-Applikation die o.a. Patterns teilweise an

Zustandsautomaten

- Entwurf
- Implementierungsvarianten: Switch-Case, Tabelle, State-Pattern
- Übung: Sie designen und implementieren den objektorientierten Zustandsautomaten der Messgeräte-Applikation

Betriebssystem

- Mechanismen in der Übersicht: Task-Management, Scheduler, Synchronisation, Kommunikation, Ressourcen-Managament, Zeit-Management, Interrupt-Management, Speichermanagement
- Praxisbeispiel: Anwendungen der Mechanismen in der Messgeräte-Applikation

Callback-Strukturen

- Kommunikation zwischen Architekturelementen
- Designregeln
- Synchron, asynchron
- Callback-Struktur prozedural und objektorientiert
- Callback-Struktur mit und ohne Betriebssystem
- Variationsmöglichkeiten
- Qualitativ hochwertige Software-Architektur mit Callback-Strukturen
- Übung: Sie designen und implementieren eine objektorientierte Callback-Struktur in der Messgeräte-Applikation.

Hardware-Treiberkonzepte und Interrupts

- Architekturrichtlinien
- Software Layer Pattern
- Praxisbeispiele von Softwareschichten-Architekturen
- Objektorientierte Treiberkonzepte
- Interrupt-Handling
- Praxistipps: Standards und Quellen zu Treiberkonzepten
- Übung: Sie designen und implementieren einen Treiber und den dazugehörigen Treiberzugriff in der Messgeräte-Applikation.

Ausgewählte Refactorings in C

- Randbedingungen für erfolgreiches Refactoring
- Kleine und große Schritte
- Smells
- Refactoring Patterns

Praktische Übungen

- Für die durchgängige Übung (Messgeräte-Applikation) verwenden Sie das Arm Keil MDK (Microcontroller Development Kit) zusammen mit einer realen Hardware, basierend auf einem Arm Cortex™-M7 Mikrocontroller.

MicroConsult Plus

- Sie erhalten von uns Ihre Übungsverzeichnisse und Lösungsbeispiele für alle Übungsaufgaben.
- Sie erhalten alle C-Beispiele und UML-Modelle in elektronischer Form und können diese sehr einfach für Ihr Entwicklungsenvironment anpassen.
- Ferner erhalten Sie eine Tool- und Software-Komponentenübersicht für die Entwicklung von Embedded-Software.
- Sie bekommen zudem die Notationsübersicht für UML (Unified Modeling Language).

Präsenz-Training

Termin	Preis *	Dauer
20.04.2026 – 23.04.2026	2.600,00 €	4 Tage
12.10.2026 – 15.10.2026	2.600,00 €	4 Tage

Stand 11.01.2026

* Preis je Teilnehmer, in Euro zzgl. USt.

Anmeldecode: ESD-C

Live-Online - Deutsch

Termin	Dauer
17.02. – 20.02.2026	4 Tage
06.07. – 09.07.2026	4 Tage

Präsenz-Training - Englisch

Termin	Dauer
20.04. – 23.04.2026	4 Tage

Live-Online - Englisch

Termin	Dauer
17.02. – 20.02.2026	4 Tage
06.07. – 09.07.2026	4 Tage

Coaching

Unsere Coaching-Angebote bieten den großen Vorteil, dass unsere Experten ihr Wissen und ihre Erfahrungen direkt in Ihren Lösungsprozess einbringen und damit unmittelbar zu Ihrem Projekterfolg beitragen.

Für Ihre Anfrage oder weiterführende Informationen stehen wir Ihnen gern zur Verfügung.