MICRO CONSULT

Stand 14.06.2026

Cortex®-M23, M33: Armv8-M Architektur Training mit Security Extension - Präsenz-Training

Sie lernen die neue ARMv8-M Architektur (Cortex®-M23 und -M33) kennen und können Programme in Assembler und C erstellen. Der Schwerpunkt dieses Workshops liegt auf Software und deckt zahlreiche Themen ab, wie z.B. TrustZone, Prozessorarchitektur, erweiterter Befehlssatz, Interruptverhalten uvm. Nach dem Training können Sie die Programme in gemischter Secure- und Non-Secure-Konfiguration im Speicher platzieren und testen. Sie sind perfekt vorbereitet für die Entwicklung von Cortex®-M23/M33-basierenden Systemen.

Ziele -
Ihr Nutzen

Sie lernen die neue ARMv8-M Architektur (Cortex®-M23 und -M33) kennen und können Programme in Assembler und C erstellen.

Der Schwerpunkt dieses Workshops liegt auf Software und deckt zahlreiche Themen ab, wie z.B. TrustZone, Prozessorarchitektur, erweiterter Befehlssatz, Interruptverhalten uvm.

Nach dem Training können Sie die Programme in gemischter Secure- und Non-Secure-Konfiguration im Speicher platzieren und testen.

Sie sind perfekt vorbereitet für die Entwicklung von Cortex®-M23/M33-basierenden Systemen.

Teilnehmer

Hardware- und Software-Entwickler

Voraussetzungen

ANSI-C und Mikrocontroller-Grundkenntnisse.

Cortex®-M23, M33: Armv8-M Architektur Training mit Security Extension - Präsenz-Training

Inhalt

TrustZone for Armv8-M

- Secure State Transitions
- Function Calls from Secure State to Non-secure State
- Function Returns from Non-secure State
- Praktische Übungen zum Entwickeln und Aufsetzen von gemischt Secure/Non-Secure Projekten für den Cortex-M33

Cortex®-M (Armv8-M) Prozessor-Architektur

- Register-Organisation, Special Purpose Register
- Operation Modes (Handler/Thread, privileged/unprivileged)
- Main Stack, Process Stack, Stack Limit Register
- Cortex®-M Pipelinekonzept
- Cortex®-M Memory Map, System Control Block
- Praktische Übungen zu den neuen Stack Limit Registern

Unterschied zur Armv6-M und Armv7-M Prozessor-Architektur

Cortex®-M33, M23, M7, M4, M3, M1, M0+, M0 Instruction Set

- Thumb-2 Instruction Set
- Data Processing Instructions
- Branch and Control Flow Instructions, Subroutines

- Branch Table, If ... then Conditional Blocks
- Data Access Instructions
- Security Instructions
- Assembler-Direktiven
- Praktische Übungen zur Erstellung kleiner Assembler-Routinen, zum Assembler-Debuggen und zur Code-Optimierung

Exception und Interrupt Handling

- Exception Model
- Reset, NMI, Faults, SysTick, Debug, Supervisor Calls, External Interrupts
- Tail Chaining, Late Arriving, Tail Chaining with Security Transitions
- Nested Vector Interrupt Controller (NVIC)
- Interrupt Configuration and Status
- Interrupt Prioritization, Priority Grouping
- Security Targeting
- Banked Exceptions
- Secure Faults
- Praktische Übungen zum SystemTick, Supervisor Call und PendSV im Kontext von RTOS-Anwendungen
- Praktische Übungen zu den Fault Handlern und Ausgabe von Status-Informationen

Memory Protection Unit MPU für Embedded-Systeme

- Armv6-M und Armv7-M MPU
- Neue Armv8-M MPU
- Praktische Übungen zur Programmierung der MPU und zum dynamischen Umprogrammieren im Scheduler

Security Attribution Unit (SAU und IDAU)

- Überblick zur Security und Implementation Defined Attribution Unit
- Attribution Attributes Secure, Non-secure, Non-secure Callable
- Praktische Übung zur Programmierung der Security Attribution Unit

Embedded Core Debugging

- Core und System Debugging
- JTAG Debug Port
- 2-Pin Single Wire Debug Port
- Trace Port Interface Unit
- Embedded Trace Macro Cell
- Praktische Übungen zum Debuggen von C-Code mit dem µVision Debugger und Print-Ausgaben auf die Debug-Konsole

Embedded Software Development

- Bibliotheksroutinen an die Hardware anpassen (Retargeting)
- Code und Daten im Speicher platzieren (Scatter Loading)
- Linker Description File
- Processor Startup, Startup File
- Praktische Übung zur Platzierung von Code und Daten an vordefinierten Adressen

Effiziente C-Programmierung für die Cortex-Architektur

- Compiler-Optimierung, Compiler-Optionen
- Schnittstelle C - Assembler
- Programmierrichtlinien für Cortex-Compiler
- Lokale und globale Daten optimal verwenden

Hardwarenahe C-Programmierung nach CMSIS

- Cortex Mikrocontroller Software Interface Standard (CMSIS)
- Softwarearchitektur für Embedded-Systeme
- Strukturierte Beschreibung von Peripherie
- Zugriff auf Peripherie in C
- C-Statements und deren Ausführung in Assembler
- CMSIS-Erweiterungen für Armv8-M

Übungen mit Keil µVision in Assembler und C

- Armv6-M und Armv7-M Programmewerden auf einem Cortex-M-basierenden Evaluierungsboard entwickelt und getestet
- Die Übungen für Armv8-M werden mit einem STM32H563 Nucleo-Board durchgeführt.
- Die Übungen werden mit Keil Studio (Visual Studio Code) durchgeführt. Keil uVision wird manchmal als

Debugger verwendet

MicroConsult Plus:

- Sie erhalten die Übungen zum Download
- Darüber hinaus können Sie dank der Installationsanweisungen und Download-Links für die Tool-Umgebung die Übungen nach der Schulung wiederholen

Präsenz-Training

Preis *	Dauer
2.800,00 €	4 Tage

Anmeldecode: ARMV8MS
* Preis je Teilnehmer, in Euro zzgl. USt.

Live-Online - Deutsch

Dauer
4 Tage

Präsenz-Training - Englisch

Dauer
4 Tage

Live-Online - Englisch

Dauer
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.