As of 19.06.2026

Cortex®-M23, M33: Armv8-M Architecture Training with Security Extension - Live Online Training

Get familiar with the new Armv8-M architecture (Cortex®-M23 and -M33) and learn how to write software in C and Assembler. This workshop focuses on software and covers a variety of topics, such as the TrustZone, processor architecture, extended instruction set, exception behavior, and many more. After the training, you can locate programs in memory in secure and non-secure configuration and test them - the perfect start for designing Cortex®-M23/M33 based systems.

Objectives

Get familiar with the new Armv8-M architecture (Cortex®-M23 and -M33) and learn how to write software in C and Assembler.

This workshop focuses on software and covers a variety of topics, such as the TrustZone, processor architecture, extended instruction set, exception behavior, and many more.

After the training, you can locate programs in memory in secure and non-secure configuration and test them - the perfect start for designing Cortex®-M23/M33 based systems.

Participants

Hardware and software developers

Requirements

A basic understanding of ANSI-C and microcontrollers.

Live-Online-Training

* Price per attendee, in Euro plus VAT

Training code: LE-ARMV8MS

Face-To-Face - English

Duration
4 days

Live Online - German

Duration
4 days

Face-To-Face - German

Duration
4 days

MICRO CONSULT

As of 19.06.2026

Cortex®-M23, M33: Armv8-M Architecture Training with Security Extension - Live Online Training

Content

TrustZone for Armv8-M

- Secure state transitions
- Function calls from secure state to non-secure state
- Function returns from non-secure state
- Practical exercises: Developing and setting up mixed secure state/non-secure state projects for Cortex-M33

Cortex®-M (Armv8-M) Processor Architecture

- Register organization, special purpose register
- Operation modes (handler/thread, privileged/unprivileged)
- Main stack, process stack, stack limit register
- Cortex®-M pipeline concept
- Cortex®-M memory map, system control block
- Practical exercises with the new stack limit registers
- Differences to the Armv6-M and Armv7-M processor architecture

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 directives
- Practical exercises: Assembler routine development, assembler debugging, code optimization

Exception and 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
- Practical exercises with system tick, supervisor call and PendSV in the context of

RTOS applications

- Practical exercises with fault handlers and output of status information

Memory Protection Unit MPU for Embedded Systems

- Armv6-M and Armv7-M MPU
- New Armv8-M MPU
- Practical exercises: MPU programming and dynamic reprogramming in the scheduler

Security Attribution Unit (SAU and IDAU)

- Overview: Security and implementation defined attribution unit
- Attribution attributes secure, non-secure, non-secure callable
- Practical exercise: Programming the security attribution unit

Embedded Core Debugging

- Core and system debugging
- JTAG debug port
- 2-pin single wire debug port
- Trace port interface unit
- Embedded trace macrocell
- Practical exercise: Debugging C code with the µVision debugger and print output to the debug console

Embedded Software Development

- Adjustment of library routines to hardware (retargeting)

- Placing code and data in memory (scatter loading)
- Linker description files
- Processor start-up, start-up file
- Practical exercise: Placing code and data at predefined addresses

Efficient C-Programming for Cortex Architectures

- Compiler optimization, compiler options
- Interface C - Assembler
- Programming guidelines for Cortex compilers
- Optimized utilization of local and global data
- Tools: Arm, IAR, GNU

Hardware-near C-Programming According to CMSIS

- Cortex Microcontroller Software Interface Standard (CMSIS)
- Software architecture for embedded systems
- Structured description of peripherals
- Access to peripherals in C
- C statements and their execution in Assembler
- CMSIS extensions for Armv8-M

Practical Exercises with Keil µVision in Assembler and C

- Armv6-M and Armv7-M programs are developed and tested on a Cortex-M based evaluation board
- Exercises for Armv8-M are performed using a STM32H563 Nucleo board
- The exercises are done using Keil Studio (Visual Studio Code). Keil uVision is sometimes used as a debugger.

MicroConsult PLUS:

- Download of exercises
- In addition, installation instructions and download links for the tool environment will allow you to repeat the exercises after the training.