Stand 03.05.2026

Cortex®-M23, M33: Armv8-M Architecture Training for Trainees with Knowledge of the Previous Version - 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.

**Ziele -
Ihr Nutzen**

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.

Teilnehmer

The training addresses software and hardware developers with a basic knowledge of the previous architecture (Armv6-M/ Armv7-M architecture of the Cortex®-M0/M0+/M3/M4 or -M7).

Voraussetzungen

Basic knowledge of the Armv6-M/ Armv7-M architecture of the Cortex®-M0/M0+/M3/M4 or -M7 as well as basic knowledge of ANSI-C and microcontrollers is required. The training focuses on the new features offered by the Cortex®-M23, M33 and the Armv8-M architecture.

Live Online Training

* Preis je Teilnehmer, in Euro zzgl. USt.

Anmeldecode: LE-ARMV8MU

Präsenz-Training - Englisch

Dauer
2 Tage

Live-Online - Deutsch

Dauer
2 Tage

Präsenz-Training - Deutsch

Stand 03.05.2026

Dauer
2 Tage

Cortex®-M23, M33: Armv8-M Architecture Training for Trainees with Knowledge of the Previous Version - Live Online Training

Inhalt

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/non-secure projects for Cortex™-M33

Cortex®-M (Armv8-M) Processor Architecture

- Stack limit register
- Secure state, security transitions
- Banked registers
- Cortex®-M memory map, system control block
- Practical exercises: New stack limit registers

Differences to the Armv6-M and Armv7-M Processor Architecture

Cortex®-M33, M23 Extended Instruction Set

- Branch and control flow instructions with security transitions
- Security instructions
- Assembler directives
- Pracitcal exercises: Generating Assembler routines, Assembler debugging, code optimization

Exception and Interrupt Handling

- Security targeting
- Banked exceptions
- Banked vector tables
- Tail chaining with security transitions
- Interrupt configuration and status
- Secure exception priority boosting
- Secure faults
- Pracitcal exercises: System tick, supervisor call and PendSV in the context of RTOS applications
- Practical exercises: Fault handlers and status information output

Memory Protection Unit MPU for Embedded Systems

- Armv8-M MPU
- Comparison to previous Armv7-M MPU
- Practical exercises: Programming the MPU
- Practical exercises: Dynamic reprogramming in the scheduler

Security Attribution Unit (SAU, IDAU)

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

Hardware-near C Programming based on CMSIS

- CMSIS extensions for Armv8-M

Exercises with Keil µVision in Assembler and C

- The exercises are done using Keil Studio (Visual Studio Code). Keil uVision is sometimes used as a debugger.