





Cortex-M23 Software Development

Course Description

Cortex-M23 software development is a 4 days ARM official course. The course goes into great depth and provides all necessary know-how to develop software for systems based on Cortex-M23 processor.

The course covers the Cortex-M23 architecture, programmer's model, development tools, instruction set, CMSIS, exception handling, memory model, memory protection unit (MPU), synchronization, efficient C programming, compiler optimizations, linker optimizations, debug, DSP instructions, and security extension.

At the end of the course the participant will receive a certificate from ARM.

Course Duration

4 days





Goals

- 1. Become familiar with ARMv8-M architecture
- 2. Become familiar with Cortex-M23 architecture
- 3. Become familiar with ARMv8-M instruction set
- 4. Become familiar with the development tools for Cortex-M
- 5. Be able to handle interrupts and various exceptions
- 6. Be able to configure and use the MPU
- 7. Understand the memory model in v8-M architecture
- 8. Write an efficient C code for Cortex-M processor
- 9. Be able to debug your design
- 10. Become familiar with DSP instructions
- 11. Optimize software for Cortex-M microcontrollers with the compiler and linker
- 12. Design a secured system with TrustZone for ARMv8-M

Target Audience

Software engineers that would like developing software and Firmware for platforms based on Cortex-M23 microcontroller.

Prerequisites

- Computer architecture background
- C and Assembler
- Experience in developing embedded systems

Course Material

- ARM official course book
- Labs handbook
- Keil MDK-ARM





Agenda

Main Topics:

- Introduction to the ARM Architecture
- Cortex-M23 Overview
- ARMv8-M Baseline Programmer's Model
- Tools Overview for ARM Microcontrollers
- Cortex-M23 Processor Core
- CMSIS Overview
- ARMv8-M Baseline Assembly Programming
- ARMv8-M Baseline Exception Handling
- ARMv8-M Baseline Memory Model
- ArmV8-M Mainline Memory Protection
- ARMv8-M Synchronization
- ARMv8-M Baseline Compiler Hints & Tips
- ARMv8-M Baseline Linker Hints & Tips
- ARMv8-M Embedded Software Development
- ARMv8-M Baseline Debug
- ARMv8-M Mainline DSP Extension
- ARMV8-M Mainline Security Extension

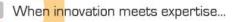
Day #1

> The ARM Architecture

- o ARM Ltd
- ARM connected community
- o ARM Classic and Cortex advanced processors
- Example ARM based system
- Development of the ARM architecture
- o Which architecture is my processor?
- ARM architecture profiles

Cortex-M23 Overview

- o Cortex-M23 processor block diagram
- Architectural features and programmer's model
- o ARMv8-M programmer's model register view
- o Modes of operation and execution
- o Memory map
- o Bus master interfaces
- o Interrupts and exceptions
- Memory protection





- Security attribution
- o Power management
- Low power features
- o System timer extension
- Floating point unit
- o Debug
- Trace
- o Configuration: synthesis and fusible
- o RTL configuration
- Integration example

ARMv8-M Baseline Programmer's Model

- o ARMv8-M profile overview
- o Data types
- o Core registers
- o Modes, privilege and stacks
- o Exceptions
- Instruction set overview

Tools Overview for ARM Microcontrollers

- o ARM compilation tools
- o Introduction to Keil MDK μVision IDE
- o ULINK debug adapters
- Development boards
- o DS5 and DSTREAM
- Fast Models from ARM
- o ARM tools licensing

> Cortex-M23 Processor Core

- o Cortex-M23 processor
- Core overview
- o Prefetch buffer
- Hardware multiplier
- o Hardware divider
- o Integer core pipeline
- o Pipeline execution
- o I/O port
- o Execution determinism
- Instruction cycle timing
- o System timer SysTick

CMSIS Overview

- o Beneficial for the ARM Cortex-M ecosystem
- o CMSIS partners
- CMSIS structure
- CMSIS bundle and documentation
- CMSIS-Core



- CMSIS-DSP
- o CMSIS-Driver
- CMSIS-RTOS
- o CMSIS-SVD
- o CMSIS-Pack
- CMSIS-DAP

Day #2

ARMv8-M Baseline Assembly Programming

- o Why do you need to know assembler?
- o Instruction set basics
- Unified Assembler Language (UAL)
- Data processing instructions
- Load/Store instructions
- Flow control
- o Miscellaneous instructions

ARMv8-M Baseline Exception Handling

- o Exception architecture overview
- o Micro-coded interrupt mechanism
- Interrupt overheads
- o Security extension TrustZone for ARMv8-M
- o Exceptions model
 - Exception entry and exit behavior
 - Prioritization and control
 - External Interrupt sensitivity
- Writing the vector table and interrupts handlers in C/C++ or Assembly
- o Internal exceptions and RTOS support
- o Fault exceptions

ARMv8-M Baseline Memory Model

- o Introduction to ARMv8-M memory model
- Memory address space and memory segments
- o Memory types and attributes
- Endianness
- o Barriers





> ARMV8-M Mainline Memory Protection

- o Motivation: memory protection
- Memory protection & security attribution
- Default memory map
- Memory Protection Unit (MPU)
- Memory regions overview
- Memory protection regions
- Memory protection unit specification
- MPU registers
- Configuring the MPU
- o Region programming
- MemManage faults

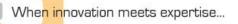
Day #3

> ARMV8-M Synchronization

- The need for atomicity
- The race for atomicity
- o Critical sections
- Effective atomicity
- LDREX, STREX and CLREX instructions
- Example of lock() and unlock() functions
- o Programs still have to be smart
- o Example: multi-thread Mutex
- Non-coherent multiprocessor
- Memory attributes
- o Configuring sharable memory
- Context switching
- Exclusives Reservation Granule (ERG)
- o Example: Multiprocessor Mutex
- Weakly ordered memory and mutual exclusion
- Ordering with DMB
- Exclusive access with LDAEX/STLEX

> ARMv8-M Baseline Compiler Hints & Tips

- Compiler support for ARMv8-M
- Language and procedure call standards
- Compiler optimizations
 - Optimization levels
 - Selecting a target
 - Automatic optimizations
 - Using volatile to limit compiler optimizations
 - Tail-call optimization





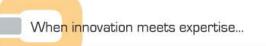
- Instruction scheduling
- Idiom recognition
- Inlining of functions
- Loop transformation
- Branch target optimization
- Link time optimization
- Coding considerations
 - Loop termination
 - Division by compile-time constants
 - Modulo arithmetic
 - Floating point
- Mixing C/C++ and assembler
 - Inline assembler
 - Intrinsics, libraries and extensions
 - CMSIS
- o Local and global data issues
 - Variable types
 - Size of local variables
 - Global RW/ZI data
 - Global data layout
 - Unaligned accesses
 - Packing of structures
 - Alignment of structures
 - Alignment of pointers
 - Optimization of memcpy()
 - Base pointer optimization

> ARMv8-M Baseline Linker Hints & Tips

- Linking basics
- System and user libraries
- Veneers
- Stack issues
- Linker optimizations and diagnostics
- ARM supplied libraries

> ARMv8-M Embedded Software Development

- Default compilation tool behavior
- System startup
 - CMSIS-CORE startup and system initialization code
 - C library initialization
- Tailoring the image memory map to a device
 - Scatter-loading
 - Linker placement rules
 - Stack and heap management
 - Further memory map consierations
- Post setup initialization
- Tailoring the C library to a device





Building and debugging an image

Day #4

> ARMv8-M Baseline Debug

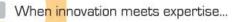
- Introduction to Debug
- Debug modes and security
- o Debug events and reset
- Flash patch and breakpoint unit (FPB)
- Data watchpoint and trace unit (DWT)
- Micro Trace Buffer (MTB)
- Embedded Trace Macrocell (ETM)
- Trace Port Interface Unit (TPIU)

ARMv8-M Baseline DSP Extension

- o Extensions overview
- DSP extension overview
- SIMD instructions
- Saturating arithmetic
- SIMD multiplies
- SIMD comparisons
- o ARMv8-M DSP instruction set

> ARMv8-M Baseline Security Extension

- Introduction to TrustZone for ARMv8-M
- Secure and non-secure states
- o Calling between security states
- o General purpose register banking
- Special purpose register banking
- Memory security
- Secure memory rules
- Memory security determination
- Memory protection unit
- Secure view of SCS
- o Non-secure view of SCS
- SAU registers
- Boot security map
- o Runtime security map
- o SAU region configuration
- o Enabling the SAU
- Configuring the SAU with CMSIS
- Branching between secure and non-secure states
- Function calls using branch instructions





- ARM C Language Extensions (ACLE)
- o Calling non-secure code from secure code
- o Calling secure code from non-secure code
- Creating an import library in ARM compiler 6
- Using the import library
- Secure gateway veneers
- o NSC veneers in ARM compiler 6
- o TT instruction
- o Security state changes using software
- Interrupts and exceptions
- o Exception priorities overview
- System handler priority
- o Secure exception prioritization
- o Configuring the NVIC
- o EXC RETURN
- o Taking an exception
- Secure -> non-secure exceptions
- o Chaining secure and non-secure exceptions
- Stack frame layout
- o Register values after context stacking
- Integrity signature