



Cortex-M55/M85 Software Development

Course Description

Cortex-M55/M85 software development is a 5 days ARM official course. The course goes into great depth and provides all necessary know-how to develop software for systems based on Cortex-M55/85 processor and how to use the Ethos-U55.

The course covers the Cortex-M55/M85 architecture (Armv8.1-M), processor core, programmer's model, instruction set, CMSIS, exception handling, memory model, caches management, memory protection unit (MPU), MVE, synchronization, efficient C programming, compiler optimizations, linker optimizations, debug & trace, floating point and DSP instructions, security extension and safety features. In addition, the Ethos-U55 is also covered.

The course has extensive hands-on labs on Arm fast model virtual platform with Keil MDK tool.

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

Course Duration

5 days with labs



When innovation meets expertise...

Goals

1. Become familiar with ARMv8.1-M architecture
2. Become familiar with Cortex-M55 and Cortex-M85 architectures
3. Become familiar with ARMv8-M instruction set
4. Be able to handle interrupts and various exceptions
5. Be able to configure and use the MPU
6. Understand the memory model in v8-M architecture
7. Manage caches
8. Extend debug capabilities with the new debug features
9. Become familiar with the new DSP and Half/single/double Precision FP instructions
10. Optimize software for Cortex-M microcontrollers with the compiler and linker
11. Design a secured system with TrustZone for ARMv8-M
12. Use the MVE engine to enhance AI performance with vectorization
13. Accelerate AI performance with the Ethos-U IP

Target Audience

Software engineers that would like developing software and Firmware for platforms based on Cortex-M55/85 as well as use AI models with the Ethos-U55.

Prerequisites

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

Course Material

- ARM official course book
- Labs handbook (via Jupiter Notebooks)
- Virtual Machine with all necessary tools for the labs

Agenda

Main Topics:

- Armv8.1-M Architecture
- Cortex-M55&M85 Architecture
- ARMv8-M.1 Mainline Programmer's Model
- CMSIS Overview
- ARMv8.1-M Mainline Assembly Programming
- ARMv8.1-M Mainline Exception Handling
- ARMv8.1-M Mainline Memory Model
- ArmV8.1-M Mainline Memory Protection
- ARMv8.1-M Synchronization
- ARMv8.1-M Mainline Compiler & Linker Hints & Tips
- ARMv8-M Mainline Debug & Trace
- ARMv8-M Mainline DSP Extension
- ARMv8-M Mainline Floating-Point Extension
- ARMV8-M Mainline Security Extension
- ARMv8.1-M MVE (M-Profile Vector Extension)
- Ethos-U Overview (U55 as a use case)

Day #1

- **ARMv8.1-M Overview**
 - ARMv8.1-M extensions
 - What does Cortex-M55 implement?
 - New loops
 - New security features
 - RAS and FuSa features
 - New debug features
 - Half-precision floating point
- **Cortex-M55/M85 Overview**
 - Cortex-M55/M85 processor block diagram
 - Architectural features and programmer's model
 - Processor core
 - ARMv8-M programmer's model register view
 - Modes of operation and execution
 - Memory map
 - Bus interfaces
 - Coprocessor interface
 - Reliability, availability, and serviceability (RAS) extension

- Interrupts and exceptions
- Memory protection
- Security attribution
- Security gating
- Power management
- Low power features
- System timer
- Extension processing unit (EPU)
- Debug
- Trace

- **ARMv8.1-M Mainline Programmer's Model**
 - ARMv8.1-M profile overview
 - Data types
 - Core registers
 - Modes, privilege and stacks
 - Exceptions
 - Instruction set overview

- **CMSIS Overview**
 - Beneficial for the ARM Cortex-M ecosystem
 - CMSIS partners
 - CMSIS structure
 - CMSIS bundle and documentation
 - CMSIS-Core
 - CMSIS-DSP
 - CMSIS-Driver
 - CMSIS-RTOS
 - CMSIS-SVD
 - CMSIS-Pack
 - CMSIS-DAP

- **ARMv8.1-M Mainline Assembly Programming**
 - Why do you need to know assembler?
 - Instruction set basics
 - Unified Assembler Language (UAL)
 - Condition codes and flags
 - Thumb instruction encoding choice
 - Data processing instructions
 - Load/Store instructions
 - Flow control
 - Miscellaneous instructions

Day #2

➤ **Cortex-M55/M85 L1 Sub-Systems**

- Tightly Coupled Memory (TCM) introduction
- TCM interfaces
- TCM configuration
- Control registers
- TCM transactions
- Accessing TCM via S-AHB
- Booting from TCM
- Instruction & Data cache
- Cache maintenance
- Automatic cache invalidation
- Cache coherency
- Direct cache access registers
- System cache support

➤ **ARMv8.1-M Mainline Exception Handling**

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

➤ **ARMv8.1-M Mainline Memory Model**

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

➤ **ARMv8.1-M Mainline Memory Protection**

- 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
- Region programming
- MemManage faults

Day #3

➤ **ARMv8.1-M Mainline Compiler & Linker 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
- 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
- Veneers
- Stack issues
- Linker optimizations and diagnostics
- ARM supplied libraries

➤ **ARMv8-M Mainline DSP Extension**

- Extensions overview
- DSP extension overview
- SIMD instructions

- Saturating arithmetic
- SIMD multiplies
- SIMD comparisons
- ARMv8-M DSP instruction set
- **ARMv8-M Mainline Floating-Point Extension**
 - Floating point extension overview
 - Registers
 - Enabling the FPU
 - Floating point instructions
 - Exceptions
 - Basic versus extended frame
 - Lazy context save
 - Interaction with the security extension
 - VLSTM and VLLDM
 - Floating point conversion instructions
- **ARMv8-M Mainline Debug**
 - Introduction to Debug
 - Debug modes and security
 - Debug events and reset
 - Flash patch and breakpoint unit (FPB)
 - Data watchpoint and trace unit (DWT)
 - Instrumentation trace Macrocell (ITM)
 - Micro Trace Buffer (MTB)
 - Embedded Trace Macrocell (ETM)
 - Trace Port Interface Unit (TPIU)
- **ARMv8.1-M PACBTI Extension**
 - Introduction to PACBTI extension
 - ROP/JOP
 - Pointer Authentication Code
 - PAC generation
 - Cryptographic keys
 - Signing and authenticate
 - Things to consider
 - Branch Target Identification
 - BTI across security states
 - BTI with PAC
 - Debugging PAC and BTI

Day #4

➤ MVE Introduction

- What is MVE?
- Benefits of MVE
- Vector Extension operation
- Vector register file
- VPR register
- Lanes
- Beats
- Vector chaining
- Exception state
- Loop-tail predication
- VPT predication
- Interleaving and de-interleaving load and store
- Vector gather load and scatter store
- How to use MVE

➤ MVE Instruction Set Architecture (ISA)

- Introduction set overview
- Data processing instructions
- Data move instructions
- Memory access instructions

➤ MVE Compiler Support

- Introduction
- Writing MVE code using intrinsics
- Automatic vectorization and tuning
- Case study: matrix multiplication

➤ MVE Code Examples

- Which use case is benefit from vectorization?
- Matrix multiply operation
- Generalize implementation
- Data type and precision

Day #5

➤ Introduction to ML & Ethos-U

- Different types of AI/ML
- Ethos-U55 vs Ethos-U65
- Operations supported by Ethos-U
- Network support in Ethos-U
- The Arm AI platform and code generation

- Ethos-U55 architecture
- Typical Ethos-U55 data flow
- **The Vela Compiler**
 - Overview and setup of Vela
 - Stages of compilation
 - Vela configuration and memory modes
 - Example usage of Vela
- **TensorFlow Lite and CMSIS-NN**
 - What is TensorFlow Lite for Microcontrollers (TFLM)?
 - What is CMSIS-NN?
 - CMSIS-NN with TFLM
 - CMSIS-NN and Ethos with TFLM
 - TF Lite Micro developer flow
 - Translation of NN operators into CMSIS-NN functions
 - What kernels are optimized?
 - Design principles in CMSIS-NN
- **Running ML Models on an Ethos-U NPU**
 - ML-embedded-evaluation-kit overview
 - Running a NN with the ML-embedded-evaluation-kit