RM5 - Cortex-M33 Implementation

This course covers the Cortex-M33 ARMv8 core

Course Environment

- Theoretical course
  - PDF course material (in English) supplemented by a printed version for face-to-face courses.
  - Online courses are dispensed using the Teams video-conferencing system.
  - The trainer answers trainees' questions during the training and provide technical and pedagogical assistance.
- Practical activities
  - Practical activities represent from 40% to 50% of course duration.
  - Code examples, exercises and solutions
  - For remote trainings:
    - One Online Linux PC per trainee for the practical activities.
    - The trainer has access to trainees' Online PCs for technical and pedagogical assistance.
    - QEMU Emulated board or physical board connected to the online PC (depending on the course).
    - Some Labs may be completed between sessions and are checked by the trainer on the next session.
  - For face-to-face trainings:
    - One PC (Linux ou Windows) for the practical activities with, if appropriate, a target board.
    - One PC for two trainees when there are more than 6 trainees.
  - For onsite trainings:
    - An installation and test manual is provided to allow preinstallation of the needed software.
    - The trainer come with target boards if needed during the practical activities (and bring them back at the end of the course).
- Downloadable preconfigured virtual machine for post-course practical activities
- At the start of each session the trainer will interact with the trainees to ensure the course fits their expectations and correct if needed

Prerequisites

- Basic understanding of microprocessors and microcontrollers

Course Environment

- Labs will be executed on an ARMv8 simulator.
- Printed training material is given to attendees during training.
- Precise and easy to use, it can be used as a reference afterwards.

Target Audience

- Any embedded systems engineer or technician with the above prerequisites.

Evaluation modalities

- The prerequisites indicated above are assessed before the training by the technical supervision of the traineein his company, or by the trainee himself in the exceptional case of an individual trainee.
- Trainee progress is assessed in two different ways, depending on the course:
  - For courses lending themselves to practical exercises, the results of the exercises are checked by the trainer while, if necessary, helping trainees to carry them out by providing additional details.

  - Quizzes are offered at the end of sections that do not include practical exercises to verifythat the trainees have assimilated the points presented
- At the end of the training, each trainee receives a certificate attesting that they have successfully completed the course.
  - In the event of a problem, discovered during the course, due to a lack of prerequisites by the trainee a different or additional training is offered to them, generally to reinforce their prerequisites,in agreement with their company manager if applicable.

Plan

First Day

Introduction to ARMv8-M Architecture

- ARM Cortex-M33 processor macrocell
- ARMv8-M Programmer's model
- Instruction pipeline
- Fixed memory map
- Privilege, modes and stacks
- Memory Protection Unit
- Security extensions
- Interrupt handling
- Nested Vectored Interrupt Controller [NVIC]
- Power management
- Debug

ARM Cortex-M33 core

- Special purpose registers
- Datapath and pipeline
- Write buffer
- Bit-banding
- System timer
- State, privilege and stacks
- System control block

Architecture of a SoC based on Cortex-M33

- Internal bus matrix
- External bus matrix to support DMA masters
- Connecting peripherals
- Sharing resources between Cortex-M4 and other CPUs
- Connection to Power Manager Controller

Second day

Embedded Software Development with Cortex-M33

- Application startup
- Placing code, data, stack and heap in the memory map, scatterloading
- Reset and initialisation
- Placing a minimal vector table
- Further memory map considerations, 8-byte stack alignment in handlers

Exercise: *Create a standalone C application displaying data on a serial line*

The T32 Instruction Set variant supported on ARMv8M

- General points on syntax

- Data processing instructions
- Branch and control flow instructions
- Memory access instructions
- Exception generating instructions
- If…then conditional blocks
- Stack in operation
- Stack limit registers
- Exclusive load and store instructions, implementing atomic sequences
- Memory barriers and synchronization

Exercise: *Create assembly-level functions to implement simple algorithms*

Synchronization and Semaphores

- Exclusive access instructions
- The Local, Global and External monitors
- Interaction with exclusive access instructions
- Load Exclusive and Store Exclusive usage and constraints

Exercise: *Implement atomic variable manipulation using exclusive access instructions*
Exercise: *Implement spinlocks*

Cortex-M DSP Instruction Set

- Multiply instructions
- Packing / unpacking instructions
- V6 ARM SIMD packed add / sub instructions
- SIMD combined add/sub instructions, implementing canonical complex operations
- Multiply and multiply accumulate instructions
- SIMD sum absolute difference instructions
- SIMD select instruction
- Saturation instructions

Exercise: *Code assembly-language optimized data-processing algorithms*

Third day

CMSIS DSP support

- The CMSIS library framework
- The CMSIS DSP intrinsic functions
- The optimized CMSIS data-processing functions

Exercise: *Recode data-processing functions in C using intrinsics*
Exercise: *Recode the same using CMSIS high-level data processing functions*
Exercise: *Compare performance of the various implementations*

Floating point Unit

- Introduction to IEEE754
- Floating point arithmetic
- Cortex-M4F single precision FPU
- Register bank
- Enabling the FPU
- FPU performance, fused MAC
- Improving the performance by selection flush-to-zero mode and default NaN mode
- Extension of AAPCS to include FP registers

Exercise: *Enable the FPU and use it for simple floating point algorithms*

C/C++ Compiler hints and Tips for Cortex-M33

- Mixing C/C++ and assembly
- Coding with GCC compiler

- Measuring stack usage
- Unaligned accesses
- Local and global data issues, alignment of structures
- Further optimisations, linker feedback

Exercise: *Measure stack usage of a program*
Exercise: *Place a user-defined data structure at a fixed address*

Exceptions

- Exception behavior, exception return
- Non-maskable exceptions
- Privilege, modes and stacks
- Fault escalation
- Priority boosting
- Vector table

Exercise: *Manage synchronous exceptions to simplify FPU usage*
Exercise: *Manage the SVC exception to switch between user and privileged modes*

Fourth day

Interrupts

- Basic interrupt operation, micro-coded interrupt mechanism
  - Interrupt entry / exit, timing diagrams
  - Interrupt stack
  - Tail chaining
- Interrupt response, pre-emption
- Interrupt prioritization
- Interrupt handlers
- The Nested Vectored Interrupt Controller (NVIC)

Exercise: *Handle a timer interrupt in C or assembly language*
Exercise: *Manage interrupt masking and nesting between two interrupts*

The Security Extension

- Security states
- Register banking between security states
- Stacks and security states
- Security Extension and exceptions
- Secure state address protection
- Secure and Non-Secure states interactions
  - Secure sate transitions
  - Function calls from Non-Secure to Secure state
  - Returning from Secure state
- Exceptions and the Security Extension
  - Handling Secure Exceptions
  - Handling Non-Secure Exceptions while in the Secure state
  - Returning from a Non-Secure exception to the Secure state
- The Security Attribution Unit
- The Implementation Defined Attribution Unit

Exercise: *Implement a minimal secure monitor*

Memory Protection Unit

- Memory types
- Access order
- Memory barriers, self-modifying code
- Memory protection overview, ARM v8 PMSA
- Cortex-M33 MPU and bus faults

- Fault status and address registers
- Region overview, memory type and access control
- Setting up the MPU

Exercise: *Use the MPU to protect an area of memory against unintended access*

Debugging features

- Invasive Debug
  - Coresight debug infrastructure
  - Halt mode
  - Vector catching
  - Debug event sources
  - Flash patch and breakpoint features
  - Data watchpoint and trace
  - ARM debug interface specification
  - Coresight components
  - AHB-Access Port
  - Possible DP implementations: Serial Wire JTAG Debug Port [SWJ-DP] or SW-DP
- Non-Invasive debug
  - Basic ETM operation
  - Instruction trace principles
  - Instrumentation trace macrocell
  - ITM stimulus port registers
  - DWT trace packets
  - Hardware event types
  - Instruction tracing
  - Synchronization packets
  - Interface between on-chip trace data from ETM and Instrumentation Trace Macrocell [ITM]
  - TPIU components
  - Serial Wire connection

Renseignements pratiques

Duration : 4 days
Cost : 2860 € HT